







**Note**

Before using this information and the product that it supports, be sure to read the general information under “Notices and trademarks” on page 277.

---

# Contents

## Chapter 1. Overview of IBM WebSphere Classic Data Event Publisher for z/OS. . . . . 1

## Chapter 2. Configuring event publishing 5

Basic configurations for Classic event publishing . . . . .	5
Basic configurations for Classic event publishing from Adabas databases . . . . .	5
Basic configurations for Classic event publishing from CA-IDMS databases . . . . .	8
Basic configurations for Classic event publishing from IMS databases . . . . .	12
Basic configurations for Classic event publishing from CICS VSAM files by using file control agents . . . . .	15
Basic configurations for Classic event publishing from CICS VSAM files by using auto-journal agents . . . . .	19
Basic configurations for Classic event publishing from CICS VSAM files by using log-reading agents . . . . .	22
Basic configurations for Classic event publishing from VSAM files . . . . .	26
Configuring logging for data servers . . . . .	29
Using the CACLOG DD for storing log messages . . . . .	29
Defining log streams for storing log messages . . . . .	30
Defining logger services . . . . .	34
Configuring TCP/IP connection handlers . . . . .	35
Configuring name services . . . . .	36
Name services . . . . .	37
Creating list structures for control tables and filter tables . . . . .	37
Configuring and starting an independent data server to run the name service . . . . .	38
Mapping tables and creating publications for Classic event publishing. . . . .	39
Configuring Classic Data Architect . . . . .	40
Mapping data for change capture . . . . .	46
Creating views on existing tables . . . . .	59
Viewing and modifying objects for Classic event publishing. . . . .	62
Populating metadata catalogs . . . . .	73
Specifying the WebSphere MQ objects to use for change capture . . . . .	74
Creating and modifying publications . . . . .	77
Configuring WebSphere MQ for Classic event publishing. . . . .	78
WebSphere MQ objects that are required for publishing to local destinations. . . . .	78
WebSphere MQ objects that are required for publishing to remote destinations . . . . .	80
WebSphere MQ authorizations that are required for Classic event publishing . . . . .	83
Defining distribution services . . . . .	84
Defining correlation services. . . . .	85
Record selection exit for multiple record layouts . . . . .	89

Configuring change capture from Adabas databases . . . . .	91
Change-capture agents for Adabas databases . . . . .	92
Configuring access to Adabas databases . . . . .	92
Running more than one correlation service when capturing from Adabas databases . . . . .	93
Installing change-capture agents for Adabas databases . . . . .	94
Configuring change capture from CA-IDMS databases . . . . .	95
Change-capture agents for CA-IDMS databases . . . . .	95
Configuring connectivity to CA-IDMS database systems. . . . .	95
Configuring data servers to access multiple CA-IDMS central versions . . . . .	97
Running more than one correlation service when capturing from multiple CA-IDMS central versions on the same z/OS LPAR . . . . .	98
Local mode versus central version mode for CA-IDMS databases . . . . .	99
Ensuring that the minimum number of journals is available for recovery . . . . .	99
Configuring automatic recovery of change data from CA-IDMS databases . . . . .	100
Starting the process of capturing changes from CA-IDMS databases . . . . .	101
Configuring change capture from IMS databases . . . . .	103
IMS environments and supported database types . . . . .	103
Change-capture agents for IMS databases . . . . .	103
Augmenting DBDs . . . . .	104
Running more than one correlation service when capturing from IMS . . . . .	107
Creating recovery data sets before activating change-capture agents for IMS data sources . . . . .	108
Installing change-capture agents for IMS databases. . . . .	109
Configuring the tracking of log files . . . . .	110
Restart points for IMS databases . . . . .	113
Configuring change capture from CICS VSAM files . . . . .	116
Three types of change-capture agent for CICS VSAM files . . . . .	116
Configuring change capture from CICS VSAM files by using file control exits . . . . .	117
Configuring and running auto-journal agents . . . . .	125
Configuring change capture from CICS VSAM files by using log-reading change-capture agents . . . . .	128
Configuring access to CICS VSAM files for change capture . . . . .	131
Configuring change capture from VSAM files . . . . .	134
NVA instances for capturing changes to VSAM files . . . . .	134
Customizing NVA instances . . . . .	135

## Chapter 3. Administering event publishing. . . . . 139

Administering data servers for Classic event publishing or Classic replication . . . . .	139	Recovery process for IMS databases . . . . .	178
Starting data servers . . . . .	139	Recovery agents for IMS databases . . . . .	182
Displaying information about data servers . . . . .	139	Sequence checking for log records for IMS databases . . . . .	184
Displaying the values of single configuration parameters . . . . .	141	Situations in which data cannot be recovered from IMS databases . . . . .	186
Modifying configurations while data servers are running . . . . .	141	How Cross Memory queue overruns affect the recovery of change data from IMS databases . . . . .	186
Saving changes to configuration files . . . . .	142	Control files for recovery agents for IMS databases . . . . .	188
Stopping data servers . . . . .	143	Determining the status of change-capture agents for IMS databases . . . . .	189
Displaying log messages that are written to SYSTEM DD . . . . .	143	Examples of determining the status of change-capture agents for IMS databases . . . . .	191
Viewing log messages with the log print utility (CACPRTL) . . . . .	143	Determining whether change data needs to be recovered from IMS databases . . . . .	193
Administering correlation services . . . . .	146	Putting change-capture agents that are unknown to correlation services into recovery mode . . . . .	194
Starting correlation services . . . . .	146	Example of putting into recovery mode a change-capture agent that is unknown to a correlation service . . . . .	195
Generating reports on the activity of correlation services . . . . .	146	Using DBRC to identify log files with change data that needs to be recovered from IMS databases . . . . .	196
CSA reporting and maintenance utility . . . . .	148	Creating recovery data sets for IMS databases when change-capture agents are in recovery mode . . . . .	199
Stopping correlation services . . . . .	151	Recovering change data from IMS log files . . . . .	201
Considerations for recycling correlation services . . . . .	152	Returning change-capture agents for IMS databases to active mode . . . . .	205
Reinitializing correlation services . . . . .	152	Recovering change data from CICS VSAM files . . . . .	206
Activating publications and Q subscriptions . . . . .	153	Recovering change data when capturing changes with file control agents . . . . .	207
Deactivating publications and Q subscriptions . . . . .	153	Starting the auto-journal agent to recover change data . . . . .	208
Deactivating all of the publications or Q subscriptions that use a send queue . . . . .	154	Administering NVA instances with the NVA SVC Manager . . . . .	209
Reinitializing publications and Q subscriptions . . . . .	154	Filters to restrict the change data that NVA instances capture . . . . .	209
Monitoring change capture . . . . .	155	Loading NVA load modules into DLPA and installing them . . . . .	209
Starting distribution services . . . . .	155	Uninstalling NVA instances from SVC chains and removing those instances from DLPA . . . . .	211
Displaying information about the configuration of publications and Q subscriptions . . . . .	156	Installing NVA instances into SVC chains when not using DLPA . . . . .	212
Displaying information about send queues . . . . .	157	Uninstalling NVA instances from SVC chains when not using DLPA . . . . .	213
Generating reports on the activity of distribution services and publication services . . . . .	158	Enabling NVA instances to capture changes . . . . .	214
Displaying the WebSphere MQ configuration for change capture . . . . .	160	Disabling NVA instances from capturing changes . . . . .	216
Displaying metrics about the correlation services that are connected to a distribution service . . . . .	160	Setting filters to restrict the changes that NVA instances capture . . . . .	217
Displaying metrics about publications in Classic event publishing . . . . .	161	Querying the NVA instances that are in SVC chains . . . . .	218
Displaying information about publication services . . . . .	162	Setting trace levels for NVA instances . . . . .	219
Stopping distribution services . . . . .	163	Displaying information about NVA instances and associated correlation services . . . . .	220
Recovering change data from Adabas databases . . . . .	164	Generating reports when capturing changes with NVA instances . . . . .	220
Recovering change data automatically from Adabas databases . . . . .	164	Stopping the capture of changes that are made to single VSAM files . . . . .	224
Recovering change data manually from Adabas databases . . . . .	165		
Options for controlling recovery agents for Adabas databases . . . . .	165		
Moving from recovery mode to active mode for Adabas databases . . . . .	168		
Recovering data from CA-IDMS databases . . . . .	168		
Recovery mode for CA-IDMS databases . . . . .	168		
Recovery agents for CA-IDMS . . . . .	169		
Recovery point files for CA-IDMS databases . . . . .	170		
Optional keywords for recovery agents that run in CV mode . . . . .	170		
Recovering change data manually from CA-IDMS databases . . . . .	171		
Recovering change data from IMS databases . . . . .	177		

Resetting change capture for single VSAM files	225	Transaction message	236
<b>Chapter 4. Reference for event publishing.</b>	<b>227</b>	Row operation message	243
Configuration parameters for change capture.	227	Schema for XML messages	245
Format of configuration parameters	227	The catalog initialization and maintenance utility (CACCATUT)	262
CLIENT CODEPAGE configuration parameter	227	Creating and initializing sequential metadata catalogs	262
COLUMN DELIMITER configuration parameter	228	Creating and initializing linear metadata catalogs	263
COMMON FILTER TABLE NAME configuration parameter	228	Upgrading metadata catalogs	263
CONTROL TABLE NAME configuration parameter	228	Copying metadata catalogs	264
DATASOURCE configuration parameter	230	Reorganizing metadata catalogs	265
DEFLOC configuration parameter	230	Generating reports about metadata catalogs	266
LD TEMP SPACE configuration parameter	230	<b>Accessing information about IBM</b>	<b>275</b>
MESSAGE POOL SIZE configuration parameter	232	Contacting IBM	275
NL configuration parameter	232	Accessible documentation	276
NL CAT configuration parameter	232	Providing comments on the documentation	276
ROW DELIMITER configuration parameter	233	<b>Notices and trademarks</b>	<b>277</b>
SERVER CODEPAGE configuration parameter	233	Notices	277
SERVICE INFO ENTRY configuration parameter	233	Trademarks	279
STRING DELIMITER configuration parameter	234	<b>Index</b>	<b>281</b>
XML messages for Classic event publishing	234		
How XML delimiters are handled in character data	234		
msg: Root element for XML messages	235		



---

# Chapter 1. Overview of IBM WebSphere Classic Data Event Publisher for z/OS

With IBM® WebSphere® Classic Data Event Publisher for z/OS®, you can link data events with both business processes and data synchronization solutions.

Classic event publishing provides efficient change-data publishing environments for the following scenarios:

- Provide application-to-application integration that makes it possible to push operational customer data to a packaged customer relationship management (CRM) application
- Initiate business processes, for example the addition of a customer record that could initiate a welcome e-mail, credit verification, and updates to the CRM system
- Monitor critical data events, such as low inventory levels that drive a workflow for restocking a product
- Feed a data warehouse, data mart, or operational data store by pushing change data to an ETL product that then populates the data store

IBM WebSphere Classic Data Event Publisher for z/OS captures data changes in non-relational databases, packages the changes into a consistent relational format, and publishes the changes as self-describing messages on WebSphere MQ message queues.

The data "events" can then be used by WebSphere applications and process integration middleware or by a JMS-aware application, tool, or message broker to drive subsequent processing. This loosely coupled integration helps to ensure that each application can be changed independently of every other application.

In Classic event publishing, you work with the following components:

## Data servers

A data server is an address space, defined within a z/OS LPAR, in which a number of services run to support Classic event publishing. These services are defined by service information entries in a configuration file. A *service information entry* contains a number of fields in which you set values to define a service and the behavior of that service.

The logger service (CACLOG) accepts log messages from the services that are running in a data server and writes those messages either to a data set or to a z/OS log stream. You configure the logger service with a service information entry.

## Metadata catalogs

A metadata catalog is a set of tables that contain information about how to convert data from non-relational to relational formats for your publications.

A metadata catalog is created when you install IBM WebSphere Classic Data Event Publisher for z/OS. You can use the catalog initialization and maintenance utility (CIMU) to maintain that metadata catalog, reorganize the metadata catalog to reclaim wasted space, copy the catalog to another catalog, or create another metadata catalog.

## Classic Data Architect

This client application provides a feature-rich graphical user interface that you can use to map relational tables to your non-relational data sources.

- Import files that describe your non-relational data sources. If your data source is an SAG Adabas or CA-IDMS database, you can connect directly to the database to discover the data structures that you want to map to.
- Map relational tables, create views, and generate the DDL that defines the tables and views.
- Promote relational tables and views to metadata catalogs by running the generated DDL.
- Run test queries against relational tables and views that are defined in metadata catalogs.
- Configure and administer publications.

Classic Data Architect can run on Windows® or Linux® workstations.

## Metadata utility

The metadata utility is a z/OS CLI-based application that connects to a data server and updates the metadata catalogs with the contents of DDL statements that are read from a SYSIN input stream.

## Publications

Publications direct data events to specific WebSphere MQ message queues. Publications consist of two components:

- A *source table or view* is a table or view that you create with Classic Data Architect and promoted to a metadata catalog. The source table or view maps to the non-relational data source that you want to capture changes to.
- Messages contain descriptions of the changes that happen to your source data. The changes are described as changes to values in columns and rows of relational tables or views. Each message can describe a row-level change or all of the changes in a single transaction. Messages can be formatted according to an XML schema or they can be in a delimited format.

Publications are ongoing processes, not single events. You can activate and deactivate publications. When you activate a publication, messages are generated for it as long as changes occur to the source table or view. When you deactivate a publication, the process of generating messages for that publication stops.

The messages for a publication are put on a WebSphere MQ message queue. You direct the messages to this message queue by associating the publication with a *publishing queue map*, which is itself associated with a message queue.

## Change-capture agents

Change-capture agents monitor source databases or VSAM files for changes to the data that you map to tables and views in Classic Data Architect. The type of change-capture agent that you use depends on the location of your source data. When a change-capture agent detects that source data is modified, the agent sends a description of the change to a correlation service. Change-capture agents can also send messages that indicate the beginning or end of a transaction.

When a change-capture agent encounters an error, the agent can go into recovery mode and stop running. After you fix the error but before you

can start the change-capture agent, you need to capture that changes that occurred to your source data while the change-capture agent was not running. To capture, or *recover*, these changes, you must run a *recovery agent*. Not all change-capture agents have a corresponding recovery agent. For more details, see the documentation for the change-capture agent that you are working with.

### **Correlation services**

Correlation services receive from one or more change-capture agents messages that describe changes to source data. Correlation services correlate the descriptions with the tables and views that you create in Classic Data Architect and convert the descriptions into a relational format. Finally, correlation services stage these modified descriptions until the change-capture agent sends notice of either a COMMIT or a ROLLBACK for a transaction. After receiving a COMMIT, the correlation service sends to a distribution service messages that describe the changes involved in the transaction.

You configure a correlation service with a service information entry. Only one correlation service can run within a single data server, but multiple correlation services can in on the same z/OS LPAR.

### **Distribution services**

Distribution services receive messages from one or more correlation services. The messages describe in a relational format the changes to source non-relational data. The distribution service sequences the changes and passes them to a publication service.

You configure a distribution service with a service information entry.

### **Publication services**

Publication services receive messages from distribution services and format the messages either in XML or in a delimited format. Publication services publish the messages to WebSphere MQ message queues. You specify the message queues to use when you create the publications that the messages belong to.

### **Name service**

Name services provide a common interface for all correlation services and change-capture agents on a single z/OS LPAR to access the control tables and filter tables.

*Control tables*, which correlation services access, record recovery information for each change-capture agent. These control tables are different from the control tables that store information about publications.

*Filter tables*, which change-capture agents access, contain information that lets change-capture agents determine whether changes are for database objects or files that you want to capture changes from.



---

## Chapter 2. Configuring event publishing

Configuring Classic event publishing involves configuring data servers and the services that run in them, mapping tables and creating publications, configuring WebSphere MQ, and configuring change-capture agents.

---

### Basic configurations for Classic event publishing

After you install IBM WebSphere Classic Data Event Publisher for z/OS, you can use the following procedures for creating basic configurations. You can follow the steps in these procedures or use the steps as aids for creating more complex configurations.

#### Basic configurations for Classic event publishing from Adabas databases

You can use these two procedures to create basic configurations for Classic event publishing from Adabas databases or as aids for creating more complex configurations.

##### Configuring Classic event publishing with one data server from Adabas databases

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service and a distribution service run in the same data server. You can use this type of configuration to learn how Classic event publishing works.

##### About this task

The steps in this procedure take place in three different locations:

##### z/OS LPAR

The LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS.

##### Linux or Windows server

The server where you installed Classic Data Architect.

##### Adabas database system

The location of the Adabas database where your source data is located.

##### Procedure

To configure Classic event publishing from Adabas databases:

1. **z/OS LPAR:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.
2. **z/OS LPAR:** Configure connectivity between the data server and your Adabas database. See “Configuring access to Adabas databases” on page 92.

3. **z/OS LPAR:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
4. **z/OS LPAR:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.
  - c. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
  - d. Define the distribution service. See “Defining distribution services” on page 84.
  - e. Define the correlation service. See “Defining correlation services” on page 85.
5. **z/OS LPAR:** Configure a name service. See “Configuring name services” on page 36.
6. **z/OS LPAR:** Start the data server. See “Starting data servers” on page 139.
7. **z/OS LPAR:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
8. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to an Adabas database, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
9. **Adabas database system:** Install the change-capture agent. See “Installing change-capture agents for Adabas databases” on page 94.

### **Configuring Classic event publishing with two data servers from Adabas databases**

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service runs in a data server on one z/OS LPAR, and a distribution service runs in another data server on another LPAR. You can use configurations with two or more data servers in production environments.

#### **About this task**

Running correlation services and distribution services in separate data servers has three effects:

- Resource contention with the DBMS and the applications that are updating data is reduced.
- The performance of IBM WebSphere Classic Data Event Publisher for z/OS significantly improves.
- Because WebSphere MQ runs only on the z/OS LPAR where the distribution service runs, the number of licenses needed for WebSphere MQ is reduced.

The steps in this procedure take place in four different locations:

### **z/OS LPAR 1**

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the distribution service will run.

### **z/OS LPAR 2**

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the correlation service will run.

### **Linux or Windows server**

The server where you installed Classic Data Architect.

### **Adabas database system**

The location of the Adabas database where your source data is located.

## **Procedure**

To configure Classic event publishing from Adabas databases:

1. **z/OS LPAR 1:** Copy the data server JCL, which is in the CACPS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACPSCF in the SCACCONF data set.
2. **z/OS LPAR 1:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
3. **z/OS LPAR 1:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
  - c. Define the distribution service. See “Defining distribution services” on page 84.
4. **z/OS LPAR 1:** Start the data server. See “Starting data servers” on page 139.
5. **z/OS LPAR 1:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
6. **z/OS LPAR 2:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.

7. **z/OS LPAR 2:** Configure connectivity between the data server and your Adabas database. See “Configuring access to Adabas databases” on page 92.
8. **z/OS LPAR 2:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
9. **z/OS LPAR 2:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.
  - c. Define the correlation service. See “Defining correlation services” on page 85.
  - d. Comment out the service information entry for CACDSRD.
10. **z/OS LPAR 2:** Configure a name service. See “Configuring name services” on page 36.
11. **z/OS LPAR 2:** Start the data server. See “Starting data servers” on page 139.
12. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to an Adabas database, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
13. **Adabas database system:** Install the change-capture agent. See “Installing change-capture agents for Adabas databases” on page 94.

## Basic configurations for Classic event publishing from CA-IDMS databases

You can use these two procedures to create basic configurations for Classic event publishing from CA-IDMS databases or as aids for creating more complex configurations.

### Configuring Classic event publishing with one data server from CA-IDMS databases

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service and a distribution service run in the same data server. You can use this type of configuration to learn how Classic event publishing works.

#### About this task

The steps in this procedure take place in three different locations:

#### **z/OS LPAR**

The LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS.

#### **Linux or Windows server**

The server where you installed Classic Data Architect.

#### **CA-IDMS database system**

The location of the CA-IDMS database where your source data is located.

#### Procedure

To configure Classic event publishing from CA-IDMS databases:

1. **z/OS LPAR:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.
2. **z/OS LPAR:** Configure connectivity to your CA-IDMS database from the data server. See “Configuring connectivity to CA-IDMS database systems” on page 95.
3. **z/OS LPAR:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
4. **z/OS LPAR:** Open the configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.
  - c. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
  - d. Define the distribution service. See “Defining distribution services” on page 84.
  - e. Define the correlation service. See “Defining correlation services” on page 85.
5. **z/OS LPAR:** Configure a name service. See “Configuring name services” on page 36.
6. **z/OS LPAR:** Start the data server. See “Starting data servers” on page 139.
7. **z/OS LPAR:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
8. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to a CA-IDMS database, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
9. **CA-IDMS database system:** Configure automatic procedures to help recover change data if the change-capture agent goes into recovery mode, and start the process of capturing changes.
  - a. **Optional:** Prevent the archiving of journals that the recovery agent might need to recover change data from. See “Ensuring that the minimum number of journals is available for recovery” on page 99.
  - b. **Optional:** Configure automatic recovery of change data. You can set up automatic procedures to check whether the change-capture agent goes into recovery mode at any time after it starts and to recover change data. See “Configuring automatic recovery of change data from CA-IDMS databases” on page 100.
  - c. Start the process of capturing changes. See “Starting the process of capturing changes from CA-IDMS databases” on page 101.

## Configuring Classic event publishing with two data servers from CA-IDMS databases

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service runs in a data server on one z/OS LPAR, and a distribution service runs in another data server on another LPAR. You can use configurations with two or more data servers in production environments.

### About this task

Running correlation services and distribution services in separate data servers has three effects:

- Resource contention with the DBMS and the applications that are updating data is reduced.
- The performance of IBM WebSphere Classic Data Event Publisher for z/OS significantly improves.
- Because WebSphere MQ runs only on the z/OS LPAR where the distribution service runs, the number of licenses needed for WebSphere MQ is reduced.

The steps in this procedure take place in four different locations:

#### z/OS LPAR 1

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the distribution service will run.

#### z/OS LPAR 2

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the correlation service will run.

#### Linux or Windows server

The server where you installed Classic Data Architect.

#### CA-IDMS database system

The location of the CA-IDMS database where your source data is located.

### Procedure

To configure Classic event publishing from CA-IDMS databases:

1. **z/OS LPAR 1:** Copy the data server JCL, which is in the CACPS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACPSCF in the SCACCONF data set.
2. **z/OS LPAR 1:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
3. **z/OS LPAR 1:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.

- b. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
  - c. Define the distribution service. See “Defining distribution services” on page 84.
- 4. **z/OS LPAR 1:** Start the data server. See “Starting data servers” on page 139.
- 5. **z/OS LPAR 1:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
- 6. **z/OS LPAR 2:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.
- 7. **z/OS LPAR 2:** Configure connectivity to your CA-IDMS database from the data server. See “Configuring connectivity to CA-IDMS database systems” on page 95.
- 8. **z/OS LPAR 2:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
- 9. **z/OS LPAR 2:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.
  - c. Define the correlation service. See “Defining correlation services” on page 85.
  - d. Comment out the service information entry for CACDSRD.
- 10. **z/OS LPAR 2:** Configure a name service. See “Configuring name services” on page 36.
- 11. **z/OS LPAR 2:** Start the data server. See “Starting data servers” on page 139.
- 12. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to a CA-IDMS database, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
- 13. **CA-IDMS database system:** Configure automatic procedures to help recover change data if the change-capture agent goes into recovery mode, and start the process of capturing changes.
  - a. **Optional:** Prevent the archiving of journals that the recovery agent might need to recover change data from. See “Ensuring that the minimum number of journals is available for recovery” on page 99.
  - b. **Optional:** Configure automatic recovery of change data. You can set up automatic procedures to check whether the change-capture agent goes into

recovery mode at any time after it starts and to recover change data. See “Configuring automatic recovery of change data from CA-IDMS databases” on page 100.

- c. Start the process of capturing changes. See “Starting the process of capturing changes from CA-IDMS databases” on page 101.

## Basic configurations for Classic event publishing from IMS databases

You can use these two procedures to create basic configurations for Classic event publishing from IMS™ databases or as aids for creating more complex configurations.

### Configuring Classic event publishing with one data server from IMS databases

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service and a distribution service run in the same data server. You can use this type of configuration to learn how Classic event publishing works.

#### About this task

The steps in this procedure take place in three different locations:

##### z/OS LPAR

The LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS.

##### Linux or Windows server

The server where you installed Classic Data Architect.

##### IMS database system

The location of the IMS database where your source data is located.

#### Procedure

To configure Classic event publishing from IMS databases:

1. **z/OS LPAR:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.
2. **z/OS LPAR:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
3. **z/OS LPAR:** Open the configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.
  - c. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER

- CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
- d. Define the distribution service. See “Defining distribution services” on page 84.
  - e. Define the correlation service. See “Defining correlation services” on page 85.
4. **z/OS LPAR:** Configure a name service. See “Configuring name services” on page 36.
  5. **z/OS LPAR:** Start the data server. See “Starting data servers” on page 139.
  6. **z/OS LPAR:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
  7. **IMS database system:** Augment the DBD files that describe the data that you want to capture changes from. See “Augmenting DBDs” on page 104.
  8. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to an IMS database, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
  9. **IMS database system:** Optionally perform steps to help you recover change data if the change-capture agent goes into recovery mode, and then install the change-capture agent.
    - a. **Optional:** Configure the tracking of IMS log files. See “Configuring the tracking of log files” on page 110.
    - b. **Optional:** Create recovery data sets. See “Creating recovery data sets before activating change-capture agents for IMS data sources” on page 108.
    - c. Install and activate the change-capture agent. See “Installing change-capture agents for IMS databases” on page 109.

## Configuring Classic event publishing with two data servers from IMS databases

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service runs in a data server on one z/OS LPAR, and a distribution service runs in another data server on another LPAR. You can use configurations with two or more data servers in production environments.

### About this task

Running correlation services and distribution services in separate data servers has three effects:

- Resource contention with the DBMS and the applications that are updating data is reduced.
- The performance of IBM WebSphere Classic Data Event Publisher for z/OS significantly improves.
- Because WebSphere MQ runs only on the z/OS LPAR where the distribution service runs, the number of licenses needed for WebSphere MQ is reduced.

The steps in this procedure take place in four different locations:

#### z/OS LPAR 1

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the distribution service will run.

## **z/OS LPAR 2**

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the correlation service will run.

## **Linux or Windows server**

The server where you installed Classic Data Architect.

## **IMS database system**

The location of the IMS database where your source data is located.

## **Procedure**

To configure Classic event publishing from IMS databases:

1. **z/OS LPAR 1:** Copy the data server JCL, which is in the CACPS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACPSCF in the SCACCONF data set.
2. **z/OS LPAR 1:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
3. **z/OS LPAR 1:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
  - c. Define the distribution service. See “Defining distribution services” on page 84.
4. **z/OS LPAR 1:** Start the data server. See “Starting data servers” on page 139.
5. **z/OS LPAR 1:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
6. **z/OS LPAR 2:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.
7. **z/OS LPAR 2:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.

8. **z/OS LPAR 2:** Open the data server's configuration file for editing and perform the following tasks:
  - a. Define the logger service. See "Defining logger services" on page 34.
  - b. Configure a TCP/IP connection handler. See "Configuring TCP/IP connection handlers" on page 35.
  - c. Define the correlation service. See "Defining correlation services" on page 85.
  - d. Comment out the service information entry for CACDSRD.
9. **z/OS LPAR 2:** Configure a name service. See "Configuring name services" on page 36.
10. **z/OS LPAR 2:** Start the data server. See "Starting data servers" on page 139.
11. **IMS database system:** Augment the DBD files that describe the data that you want to capture changes from. See "Augmenting DBDs" on page 104.
12. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to an IMS database, and create publications. See "Mapping tables and creating publications for Classic event publishing" on page 39.
13. **IMS database system:** Optionally perform steps to help you recover change data if the change-capture agent goes into recovery mode, and then install the change-capture agent.
  - a. **Optional:** Configure the tracking of IMS log files. See "Configuring the tracking of log files" on page 110.
  - b. **Optional:** Create recovery data sets. See "Creating recovery data sets before activating change-capture agents for IMS data sources" on page 108.
  - c. Install and activate the change-capture agent. See "Installing change-capture agents for IMS databases" on page 109.

## Basic configurations for Classic event publishing from CICS VSAM files by using file control agents

After you install IBM WebSphere Classic Data Event Publisher for z/OS, you can use these two procedures to create basic configurations for Classic event publishing from CICS® VSAM files by using file control agents or as aids for creating more complex configurations.

### Configuring Classic event publishing from CICS VSAM files by using file control agents and one data server

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service and a distribution service run in the same data server. You can use this type of configuration to learn how Classic event publishing works.

#### About this task

The steps in this procedure take place in three different locations:

#### z/OS LPAR

The LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS.

#### Linux or Windows server

The server where you installed Classic Data Architect.

#### CICS system

The location of your CICS installation.

## Procedure

To configure Classic event publishing with a file control exit:

1. **CICS system:** Define the file control agent. See Defining file control agents to CICS.
2. **CICS system:** Define the programs that enable and disable the file control agent. See Defining in CICS the programs that enable and disable file control agents.
3. **CICS system:** Define the transactions to enable or disable the file control agent. See Defining CICS transactions that run the programs that enable or disable file control agents.
4. **CICS system:** If you plan to use the auto-journal agent to recover change data, activate auto-journal logging for the VSAM files that you want to capture changes from. See “Activating auto-journal logging for VSAM files” on page 121.
5. **z/OS LPAR:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.
6. **z/OS LPAR:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
7. **z/OS LPAR:** Open the configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.
  - c. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
  - d. Define the distribution service. See “Defining distribution services” on page 84.
  - e. Define the correlation service. See “Defining correlation services” on page 85.
  - f. **Optional:** Name the correlation service. See “Configuring named correlation services for the file control agent for CICS VSAM files” on page 123.
  - g. If you plan to use the auto-journal agent to recover change data, create the service information entry for the auto-journal agent. See “Creating service information entries for auto-journal agents” on page 125.
8. **z/OS LPAR:** Configure a name service. See “Configuring name services” on page 36.
9. **z/OS LPAR:** Start the data server. See “Starting data servers” on page 139..

10. **z/OS LPAR:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
11. **z/OS LPAR:** Configure connectivity between the data server and your CICS VSAM files. See “Configuring access to CICS VSAM files for change capture” on page 131.
12. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to CICS VSAM files, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
13. **CICS system:** Enable the file control agent. See “Enabling file control agents” on page 124.

### **Configuring Classic event publishing from CICS VSAM files by using file control agents and two data servers**

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service runs in a data server on one z/OS LPAR, and a distribution service runs in another data server on another LPAR. You can use configurations with two or more data servers in production environments.

#### **About this task**

Running correlation services and distribution services in separate data servers has three effects:

- Resource contention with the DBMS and the applications that are updating data is reduced.
- The performance of IBM WebSphere Classic Data Event Publisher for z/OS significantly improves.
- Because WebSphere MQ runs only on the z/OS LPAR where the distribution service runs, the number of licenses needed for WebSphere MQ is reduced.

The steps in this procedure take place in four different locations:

#### **z/OS LPAR 1**

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the distribution service will run.

#### **z/OS LPAR 2**

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the correlation service will run.

#### **Linux or Windows server**

The server where you installed Classic Data Architect.

#### **CICS system**

The location of your CICS installation.

#### **Procedure**

To configure Classic event publishing with a file control exit:

1. **CICS system:** Define the file control agent. See Defining file control agents to CICS.
2. **CICS system:** Define the programs that enable and disable the file control agent. See Defining in CICS the programs that enable and disable file control agents.

3. **CICS system:** Define the transactions to enable or disable the file control agent. See Defining CICS transactions that run the programs that enable or disable file control agents.
4. **CICS system:** If you plan to use the auto-journal agent to recover change data, activate auto-journal logging for the VSAM files that you want to capture changes from. See “Activating auto-journal logging for VSAM files” on page 121.
5. **z/OS LPAR 1:** Copy the data server JCL, which is in the CACPS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACPSCF in the SCACCONF data set.
6. **z/OS LPAR 1:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
7. **z/OS LPAR 1:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
  - c. Define the distribution service. See “Defining distribution services” on page 84.
8. **z/OS LPAR 1:** Start the data server. See “Starting data servers” on page 139.
9. **z/OS LPAR 1:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
10. **z/OS LPAR 2:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.
11. **z/OS LPAR 2:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
12. **z/OS LPAR 2:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.

- c. Define the correlation service. See “Defining correlation services” on page 85.
  - d. Comment out the service information entry for CACDSRD.
  - e. **Optional:** Name the correlation service. See “Configuring named correlation services for the file control agent for CICS VSAM files” on page 123.
  - f. If you plan to use the auto-journal agent to recover change data, create the service information entry for the auto-journal agent. See “Creating service information entries for auto-journal agents” on page 125.
13. **z/OS LPAR 2:** Configure a name service. See “Configuring name services” on page 36.
  14. **z/OS LPAR 2:** Start the data server. See “Starting data servers” on page 139.
  15. **z/OS LPAR 2:** Configure connectivity between the data server and your CICS VSAM files. See “Configuring access to CICS VSAM files for change capture” on page 131.
  16. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to CICS VSAM files, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
  17. **CICS system:** Enable the file control agent. See “Enabling file control agents” on page 124.

## Basic configurations for Classic event publishing from CICS VSAM files by using auto-journal agents

After you install IBM WebSphere Classic Data Event Publisher for z/OS, you can use these two procedures to create basic configurations for Classic event publishing from CICS VSAM files by using auto-journal agents or as aids for creating more complex configurations.

### Configuring Classic event publishing from CICS VSAM files by using auto-journal agents and one data server

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service and a distribution service run in the same data server. You can use this type of configuration to learn how Classic event publishing works.

#### About this task

The steps in this procedure take place in three different locations:

#### z/OS LPAR

The LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where your VSAM files are located.

#### Linux or Windows server

The server where you installed Classic Data Architect.

#### CICS system

The location of your CICS installation.

#### Procedure

To configure Classic event publishing with an auto-journal agent:

1. **CICS system:** Activate auto-journal logging for the VSAM files that you want to capture changes from. See “Activating auto-journal logging for VSAM files” on page 121.
2. **z/OS LPAR:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.
3. **z/OS LPAR:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
4. **z/OS LPAR:** Open the configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.
  - c. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
5. **z/OS LPAR:** Configure a name service. See “Configuring name services” on page 36.
6. **z/OS LPAR:** Start the data server. See “Starting data servers” on page 139.
7. **z/OS LPAR:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
8. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to CICS VSAM files, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
9. **z/OS LPAR:** Stop the data server. See “Stopping data servers” on page 143.
10. **z/OS LPAR:** Open the configuration file for editing and perform the following tasks:
  - a. Define the distribution service. See “Defining distribution services” on page 84.
  - b. Define the correlation service. See “Defining correlation services” on page 85.
  - c. Create the service information entry for the auto-journal agent. See “Creating service information entries for auto-journal agents” on page 125.
11. **z/OS LPAR:** Start the correlation service, the auto-journal agent, and the distribution service. See Starting change capture from CICS VSAM files by using auto-journal agents.

### **Configuring Classic event publishing from CICS VSAM files by using auto-journal agents and two data servers**

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service runs in a data server on one z/OS LPAR, and a

distribution service runs in another data server on another LPAR. You can use configurations with two or more data servers in production environments.

### **About this task**

Running correlation services and distribution services in separate data servers has three effects:

- Resource contention with the DBMS and the applications that are updating data is reduced.
- The performance of IBM WebSphere Classic Data Event Publisher for z/OS significantly improves.
- Because WebSphere MQ runs only on the z/OS LPAR where the distribution service runs, the number of licenses needed for WebSphere MQ is reduced.

The steps in this procedure take place in four different locations:

#### **z/OS LPAR 1**

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the distribution service will run.

#### **z/OS LPAR 2**

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the correlation service will run. Your VSAM files are also on this LPAR.

#### **Linux or Windows server**

The server where you installed Classic Data Architect.

#### **CICS system**

The location of your CICS installation.

### **Procedure**

To configure Classic event publishing with an auto-journal agent:

1. **z/OS LPAR 1:** Copy the data server JCL, which is in the CACPS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACPSCF in the SCACCONF data set.
2. **z/OS LPAR 1:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
3. **z/OS LPAR 1:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE

- configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
- c. Define the distribution service. See “Defining distribution services” on page 84.
  4. **z/OS LPAR 1:** Start the data server. See “Starting data servers” on page 139.
  5. **z/OS LPAR 1:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
  6. **CICS system:** Activate auto-journal logging for the VSAM files that you want to capture changes from. See “Activating auto-journal logging for VSAM files” on page 121.
  7. **z/OS LPAR 2:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
    - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
    - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.
  8. **z/OS LPAR 2:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
  9. **z/OS LPAR 2:** Open the data server’s configuration file for editing and perform the following tasks:
    - a. Define the logger service. See “Defining logger services” on page 34.
    - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.
  10. **z/OS LPAR 2:** Configure a name service. See “Configuring name services” on page 36.
  11. **z/OS LPAR 2:** Start the data server. See “Starting data servers” on page 139.
  12. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to CICS VSAM files, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
  13. **z/OS LPAR 2:** Open the data server’s configuration file for editing and perform the following tasks:
    - a. Define the correlation service. See “Defining correlation services” on page 85.
    - b. Comment out the service information entry for CACDSRD.
    - c. Create the service information entry for the auto-journal agent. See “Creating service information entries for auto-journal agents” on page 125.
  14. **z/OS LPAR 2:** Start the correlation service and the auto-journal agent. See Starting change capture from CICS VSAM files by using auto-journal agents.

## Basic configurations for Classic event publishing from CICS VSAM files by using log-reading agents

After you install IBM WebSphere Classic Data Event Publisher for z/OS, you can use these two procedures to create basic configurations for Classic event publishing from CICS VSAM files by using log-reading agents or as aids for creating more complex configurations.

## Configuring Classic event publishing from CICS VSAM files by using log-reading agents and one data server

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service and a distribution service run in the same data server. You can use this type of configuration to learn how Classic event publishing works.

### About this task

The steps in this procedure take place in three different locations:

#### **z/OS LPAR**

The LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS.

#### **Linux or Windows server**

The server where you installed Classic Data Architect.

#### **CICS system**

The location of your CICS installation.

### Procedure

To configure Classic event publishing with a log-reading change-capture agent:

1. **z/OS LPAR:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.
2. **z/OS LPAR:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
3. **z/OS LPAR:** Open the configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.
  - c. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
4. **z/OS LPAR:** Configure a name service. See “Configuring name services” on page 36.
5. **z/OS LPAR:** Start the data server. See “Starting data servers” on page 139.
6. **CICS system:** Configure connectivity between the data server and your CICS VSAM file. See “Configuring access to CICS VSAM files for change capture” on page 131.

7. **CICS system:** Set the retention period and AUTODELETE specifications of the CICS system log, user log, and log of log streams. See “Retention period and AUTODELETE of log data” on page 128.
8. **z/OS LPAR:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
9. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to CICS VSAM files, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
10. **z/OS LPAR:** Stop the data server. See “Stopping data servers” on page 143.
11. **z/OS LPAR:** Open the configuration file for editing and perform the following tasks:
  - a. Define the distribution service. See “Defining distribution services” on page 84.
  - b. Define the correlation service. See “Defining correlation services” on page 85.
  - c. Define the log-reading agent. See “Defining log-reading agents for CICS VSAM files” on page 128.
12. **z/OS LPAR:** Start the correlation service, the log-reading change-capture agent, and the distribution service. See “Starting change capture from CICS VSAM files with log-reading agents” on page 130.

### **Configuring Classic event publishing from CICS VSAM files by using log-reading agents and two data servers**

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service runs in a data server on one z/OS LPAR, and a distribution service runs in another data server on another LPAR. You can use configurations with two or more data servers in production environments.

#### **About this task**

Running correlation services and distribution services in separate data servers has three effects:

- Resource contention with the DBMS and the applications that are updating data is reduced.
- The performance of IBM WebSphere Classic Data Event Publisher for z/OS significantly improves.
- Because WebSphere MQ runs only on the z/OS LPAR where the distribution service runs, the number of licenses needed for WebSphere MQ is reduced.

The steps in this procedure take place in four different locations:

#### **z/OS LPAR 1**

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the distribution service will run.

#### **z/OS LPAR 2**

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the correlation service will run.

#### **Linux or Windows server**

The server where you installed Classic Data Architect.

## CICS system

The location of your CICS installation.

## Procedure

To configure Classic event publishing with a log-reading change-capture agent:

1. **z/OS LPAR 1:** Copy the data server JCL, which is in the CACPS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACPSCF in the SCACCONF data set.
2. **z/OS LPAR 1:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
3. **z/OS LPAR 1:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
  - c. Define the distribution service. See “Defining distribution services” on page 84.
4. **z/OS LPAR 1:** Start the data server. See “Starting data servers” on page 139.
5. **z/OS LPAR 1:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
6. **z/OS LPAR 2:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.
7. **z/OS LPAR 2:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
8. **z/OS LPAR 2:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.
9. **z/OS LPAR 2:** Configure a name service. See “Configuring name services” on page 36.

10. **z/OS LPAR 2:** Start the data server. See “Starting data servers” on page 139.
11. **CICS system:** Configure connectivity between the data server and your CICS VSAM file. See “Configuring access to CICS VSAM files for change capture” on page 131..
12. **CICS system:** Set the retention period and AUTODELETE specifications of the CICS system log, user log, and log of log streams. See “Retention period and AUTODELETE of log data” on page 128.
13. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to CICS VSAM files, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
14. **z/OS LPAR 2:** Stop the data server. See “Stopping data servers” on page 143.
15. **z/OS LPAR 2:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the correlation service. See “Defining correlation services” on page 85.
  - b. Comment out the service information entry for CACDSRD.
  - c. Define the log-reading agent. See “Defining log-reading agents for CICS VSAM files” on page 128.
16. **z/OS LPAR 2:** Start the correlation service and the log-reading change-capture agent. See “Starting change capture from CICS VSAM files with log-reading agents” on page 130.

## Basic configurations for Classic event publishing from VSAM files

You can use these two procedures to create basic configurations for Classic event publishing from VSAM files or as aids for creating more complex configurations.

### Configuring Classic event publishing with one data server from VSAM files

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service and a distribution service run in the same data server. You can use this type of configuration to learn how Classic event publishing works.

#### About this task

The steps in this procedure take place in two different locations:

##### z/OS LPAR

The LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where your VSAM files are located.

##### Linux or Windows server

The server where you installed Classic Data Architect.

#### Procedure

To configure Classic event publishing from VSAM files:

1. **z/OS LPAR:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.

- b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.
2. **z/OS LPAR:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
3. **z/OS LPAR:** Open the configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.
  - c. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
  - d. Define the distribution service. See “Defining distribution services” on page 84.
  - e. Define the correlation service. See “Defining correlation services” on page 85.
4. **z/OS LPAR:** Configure a name service. See “Configuring name services” on page 36.
5. **z/OS LPAR:** Start the data server. See “Starting data servers” on page 139.
6. **z/OS LPAR:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
7. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to native VSAM files, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
8. **z/OS LPAR:** Configure and start an NVA instance. See “Configuring change capture from VSAM files” on page 134.

### **Configuring Classic event publishing with two data servers from VSAM files**

These steps explain how to create a basic configuration for Classic event publishing in which a correlation service runs in a data server on one z/OS LPAR, and a distribution service runs in another data server on another LPAR. You can use configurations with two or more data servers in production environments.

#### **About this task**

Running correlation services and distribution services in separate data servers has three effects:

- Resource contention with the DBMS and the applications that are updating data is reduced.
- The performance of IBM WebSphere Classic Data Event Publisher for z/OS significantly improves.
- Because WebSphere MQ runs only on the z/OS LPAR where the distribution service runs, the number of licenses needed for WebSphere MQ is reduced.

The steps in this procedure take place in three different locations:

### **z/OS LPAR 1**

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the distribution service will run.

### **z/OS LPAR 2**

An LPAR where you installed IBM WebSphere Classic Data Event Publisher for z/OS and where you will configure the data server in which the correlation service will run and where your VSAM files are located.

### **Linux or Windows server**

The server where you installed Classic Data Architect.

## **Procedure**

To configure Classic event publishing from VSAM files:

1. **z/OS LPAR 1:** Copy the data server JCL, which is in the CACPS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACPSCF in the SCACCONF data set.
2. **z/OS LPAR 1:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
3. **z/OS LPAR 1:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. If you want to publish messages in Unicode (UTF-8) rather than in EBCDIC, which is the default code page, set the CLIENT CODEPAGE configuration parameter to a value that is different from the value of the SERVER CODEPAGE configuration parameter. See “CLIENT CODEPAGE configuration parameter” on page 227 and “SERVER CODEPAGE configuration parameter” on page 233.
  - c. Define the distribution service. See “Defining distribution services” on page 84.
4. **z/OS LPAR 1:** Start the data server. See “Starting data servers” on page 139.
5. **z/OS LPAR 1:** Configure WebSphere MQ for Classic event publishing. See “Configuring WebSphere MQ for Classic event publishing” on page 78.
6. **z/OS LPAR 2:** Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.
  - b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCSCF in the SCACCONF data set.

7. **z/OS LPAR 2:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” or “Defining log streams for storing log messages” on page 30.
8. **z/OS LPAR 2:** Open the data server’s configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure a TCP/IP connection handler. See “Configuring TCP/IP connection handlers” on page 35.
  - c. Define the correlation service. See “Defining correlation services” on page 85.
  - d. Comment out the service information entry for CACDSRD.
9. **z/OS LPAR 2:** Configure a name service. See “Configuring name services” on page 36.
10. **z/OS LPAR 2:** Start the data server. See “Starting data servers” on page 139.
11. **Linux or Windows server:** In Classic Data Architect, create control tables, set capture parameters, create relational tables that map to VSAM files, and create publications. See “Mapping tables and creating publications for Classic event publishing” on page 39.
12. **z/OS LPAR 2:** Configure and start an NVA instance. See “Configuring change capture from VSAM files” on page 134.

---

## Configuring logging for data servers

The logger service (CACLOG) accepts log messages from the services that are running in a data server and writes those messages either to a temporary or permanent data set or to an MVS™ log stream. You configure the logger service with a service information entry in the configuration file for the data server.

### Using the CACLOG DD for storing log messages

The logger service (CACLOG) can write log messages to a permanent or temporary data set, depending on how you define the CACLOG DD statement in the JCL for the data server in which you run the service.

#### Restrictions

If you want to use the log print utility to view the log messages, you must stop the data server. However, you can view log messages by using the DISPLAY keyword in field 10 of the service information entry for the logger service. The messages are written to the data set that is specified in the SYSTERM DD statement, which is by default the SYSOUT data set.

IBM recommends configuring logger services to write to log streams rather than to data sets. You can read the log without stopping the data server. Using log streams also limits the CPU usage and storage costs that are required for formatting and displaying the log messages.

#### Procedure

To use the CACLOG DD statement of the CACCNTL job step in the data server JCL to create storage for log messages, perform either of the following steps:

- Create a temporary data set to store log messages from the data server by specifying a temporary name (for example, &&L0G).

- Create a physical data set by allocating a binary, sequential file with these attributes:  
RECFM=FBS  
LRECL=1  
BLKSIZE=4080

## Defining log streams for storing log messages

Use the IXCMIAPU Administration Data Utility to define a log stream for storing log messages from a data server. This utility is a part of z/OS System Logger.

### Before you begin

You must have authorization to use the IXCMIAPU Administrative Data Utility if you want to add or change information about a log stream and to update the LOGR policy.

If you plan to define a CF-structure log stream, System Logger requires a coupling facility.

For information about setting up and managing coupling facilities, see z/OS V1R6.0 MVS Setting Up a Sysplex.

### About this task

You can find sample JCL to define DASD-only and CF-structure log streams in SCACSAMP member CACDEFSL.

It is recommended to use a separate log stream for each data server so that you are not confused when reading log output. If multiple data servers send log records to the same log stream, it might not be possible to determine which data server generated a particular log record.

### Procedure

To define an MVS log stream for storing log messages:

1. Decide whether to define the log stream as a DASD-only log stream or as a CF-structure log stream.
2. In the Administrative Data Utility (IXCMIAPU), define the log stream. The following parameters are relevant to defining log streams for log records from a data server:

#### **NAME( )**

Specify a unique name for the log stream.

The logger service and the log print utility (CACPRTL) both require this name.

#### **DASDONLY( )**

Possible values:

**YES** Defines a DASD-only log stream.

**NO** Defines a CF-structure log stream. NO is the default value for this parameter.

**STRUCTNAME( )**

If you are defining a CF-structure log stream, specify the CF structure that you want to use to store log data. The structure must be specified under the CFRM policy.

**STG\_DUPLEX( )**

If you are defining a CF-structure log stream, specify either of these values:

**YES** The z/OS System Logger will duplex log records to a staging data set. If you use a staging data set, the initial write to the log stream causes the z/OS System Logger to format the staging data set.

If you do not use a staging data set and a system outage occurs, you will lose the log records that are stored in the coupling facility.

**NO** The z/OS System Logger will not duplex log records to a staging data set.

For more information about duplexing, see section 9.2.5 "Duplexing Log Data" in *z/OS V1R6.0 MVS Setting Up a Sysplex*.

**HIGHOFFLOAD( )**

You can accept the default value of 80. The logger service writes data to the log stream, but does not retrieve written records. Only the log print utility retrieves records.

**LOWOFFLOAD( )**

You can accept the default value of 0.

**MAXBUFFSIZE( )**

Specifies the largest possible size for blocks of data that are written to a DASD-only log stream. The recommended value is 4096.

If you plan to switch from a DASD-only log stream to a CF-structure log stream at a later time, this value must be less than or equal to the MAXBUFFSIZE value of the CF structure.

**LS\_SIZE( )**

Specifies the size of the offload data sets for the log stream. 1000 4KB blocks is the recommended size.

If you set a low trace level, you might need a larger size because the volume of log messages increases.

**STG\_SIZE( )**

Specifies the size of the staging data set for the log stream. 1000 4KB blocks is the recommended size.

The larger the size of the staging data set, the more time is required to format the data set when the logger service first writes to the log stream.

**AUTODELETE( ) and RETPD( )**

Set AUTODELETE to YES to allow the z/OS System Logger to delete log messages.

For RETPD, specify the number of days before data in the log stream is eligible for deletion. Set a value that is long enough to maintain enough records for diagnosing problems but short enough to prevent the log stream from growing too large.

## Logstreams for storing log records from data servers

You can configure the logger service to write log records to a log stream that is defined in the z/OS System Logger.

Storing your log records in a log stream allows you to extract and view those records without stopping the data server. You also do not need to use the DISPLAY option for the logger service to mirror the log records to SYSOUT. The DISPLAY option can lead to increasing processing overhead and space for spooling the output.

For more information about the System Logger, see the IBM Redbook *Systems Programmer's Guide to: z/OS System Logger*.

## Components of log streams

### System Logger

z/OS System Logger serves as a repository to store log records written by the data server. The logger service within the data server is the application program producing the log records.

You define z/OS System Logger resources through the administrative data utility IXCMIAPU. You must have authorization to use IXCMIAPU if you want to add or change information about a log stream and to update the LOGR policy.

For information about setting up and managing coupling facilities, see z/OS V1R6.0 MVS Setting Up a Sysplex.

### Log stream

A log stream represents a logical collection of information that is stored and managed by z/OS System Logger on behalf of an application.

### DASD-only log stream

A DASD-only log stream uses storage (a dataspace) within the System Logger address space (IXGLOGR) and supports concurrent access from only a single image.

The following figure shows sample JCL:

```
//IXCMIAPU JOB (IICF), 'CR8 DASD LOG STREAM', NOTIFY=&SYSUID
/*
//LOGST EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
  DATA TYPE(LOGR) REPORT(NO)
  DEFINE LOGSTREAM NAME(IICF.DASD.LOG)
    DASDONLY(YES)
    MAXBUFSIZE(4096)
    LS_SIZE(1000)
    STG_DATACLAS(DCVSAMLS) STG_SIZE(1000)
    RETPD(7) AUTODELETE(YES)
/*
```

Figure 1. Sample JCL to define a DASD-only log stream

### CF-structure log stream

A CF-structure log stream uses storage in the coupling facility (CF). Applications on multiple systems can write to one log at the same time. This log stream type requires a CF.

For information about setting up and managing coupling facilities, see *z/OS V1R6.0 MVS Setting Up a Sysplex*.

The following figure shows sample JCL:

```
//IXCMIAPU JOB (IICF), 'CR8 CF LOG STREAM', NOTIFY=&SYSUID
/*
//LOGST EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
  DATA TYPE(LOGR) REPORT(NO)
  DEFINE LOGSTREAM NAME(IICF.CF.LOG) STRUCTNAME(IICF1)
    LS_SIZE(1000)
    STG_DATACLAS(DCVSAMPLS)
    STG_SIZE(1000) STG_DUPLEX(YES)
    RETPD(7) AUTODELETE(YES)
/*
```

Figure 2. Sample JCL to define a CF-structure log stream

### Staging data set

A data set that is used to duplex log stream data that was not yet offloaded. If you use a DASD-only log stream, the z/OS System Logger automatically provides staging, which is also known as duplexing. A staging data set is optional with a CF-structure log stream.

### Coupling facility

On z/OS, a coupling facility (CF) is a special logical partition that provides high-speed caching, list processing, and locking functions in a parallel sysplex. The CF has responsibility for management of shared resources, like CF-structures for log streams.

### Performance tips

The best performing log stream will likely be a CF-structure type log stream without duplexing. A DASD-only log stream automatically duplexes, but a CF-structure log stream duplexes only on request. The overhead of writing log records to a staging data set might be significant with high-volume logging, most notably in the System Logger's address space. However, if you run without a staging data set you risk losing log records if the CF fails, because newly written data is in the CF only.

Under normal production operations, few log records are written to the log stream and duplexing does not incur a high overhead. In an environment that generates higher volumes of log information, running without a staging data set might improve logging performance, though you would need to tolerate loss of log data if the CF failed.

### Options for deleting log records from log streams

The AUTODELETE and RETPD parameters that you set when you define log streams work in conjunction with the PURGE and PURGEALL options of the log print utility to delete log message from log streams.

The following table lists the combinations that are possible for the values of AUTODELETE and RETPD. The Result column explains when log messages become eligible for deletion and when you must use the PURGE option of the log print utility to mark the log messages for deletion.

The phrase "eligible for deletion" means that z/OS System Logger can delete the log message during the next period when z/OS System Logger checks for messages to delete. The phrase does not guarantee that z/OS System Logger will delete all eligible log messages during the next check. For additional information about when messages that are eligible for deletion are physically removed, refer to z/OS V1R6.0 MVS Setting Up a Sysplex.

Table 1. Possible combinations and results of AUTODELETE and RETPD values

Value of AUTODELETE	Value of RETPD	Result
YES	0	Log messages are immediately eligible for deletion by the z/OS System Logger as soon as they are written. <b>Recommendation:</b> IBM does not recommend this combination of values. z/OS System Logger could remove the messages before you have a chance to review them in the event of a problem.
NO	0	Log messages are immediately eligible for deletion. However, you must mark the log messages for deletion by a utility because the z/OS System Logger deletes the log records only if they are so marked. You must periodically empty the log stream by using either the PURGE or PURGEALL option of the log print utility.
YES	> 0	Log messages are eligible for deletion at either of these times: <ul style="list-style-type: none"> <li>• After the maximum age that is specified in RETPD</li> <li>• After the log messages are marked for deletion by either the PURGE or PURGEALL option of the log print utility</li> </ul>
NO	> 0	Log messages are eligible for deletion only after the maximum age that is specified by RETPD, even if the log messages are marked for deletion. You must run the log print utility with the PURGE option to allow log records that reach the maximum age to be deleted.

## Defining logger services

You define the logger service (CACLOG) by modifying its service information entry in the configuration file for the data server.

### Before you begin

Stop the data server if it is running. See "Stopping data servers" on page 143.

If you want the logger service to write log messages to an z/OS log stream, you must know the name of the z/OS log stream that you want to use.

### About this task

When you modify the service information entry, you can accept the default values for all of the fields except for fields 7 and 10.

### Procedure

To define a logger service:

1. Open for editing the configuration file of the data server in which you plan to run the logger service.
2. In field 10 of the service information entry for the logger service, specify one of the following options:

**NO\_DATA**

Writes log records to the CACLOG DD and sets the log buffer size to 16 KB.

**nK** Changes the default size of the log buffer. Values can be between 4K and 1000K.

**DISPLAY**

Writes log records to the CACLOG DD, sets the log buffer size to 16 KB, and mirrors the log records to the data set that is specified in the SYSTERM DD statement. The default data set is SYSOUT.

**STREAM=*name of log stream***

Writes the log records to the specified z/OS log stream and sets the log buffer size to 16 KB.

**DISPLAY,STREAM=*name of log stream***

Performs both of the actions that are described for the DISPLAY and STREAM keywords.

3. **Optional:** Change the default size of the log buffer. After the keyword that you selected in step 3, type a comma and then type the size in kilobytes for the log buffer. Values can be between 4K and 1000K.

---

## Configuring TCP/IP connection handlers

Because Classic Data Architect communicates by using a TCP/IP connection with data server where a correlation service is configured, you must configure a TCP/IP connection handler in the data server.

Before configuring a TCP/IP connection handler service, contact your network administrator to obtain the host name or IP address where you will be running the data server. Also, ask your network administrator for a unique TCP port number that is not being used by any other applications on that z/OS system.

- The port number or service name must not be in use by any other application. The port number should be greater than 1024 because numbers 1 through 1024 are often reserved.
- If your z/OS TCP/IP system is using off-load gateways, ensure that the IP address that is specified reflects the IP address of the z/OS TCP/IP stack, not the address of the IP stack of an off-load gateway.
- The sample TCP/IP connection handler definitions use the default IP address of 0.0.0.0, which normally resolves to the IP address of the host that the data server is operating on. In most cases, you will not have to provide the actual IP address.

**Note:** In some configurations, the default 000.000.000.000 might resolve to the internal loop back address of 0.0.0.127. In these instances, you need to replace 0.0.0.0 with the actual IP address or host name. You can detect this condition by starting the data server and issuing a NETSTAT command to see the IP address that the data server address space name is listening for connections on. If the IP address displayed does not correspond to the

correct IP address, the default 0.0.0.0 IP address resolution technique is not working properly on your system.

To configure the TCP/IP connection handler:

1. Open the data server's configuration file for editing. A sample configuration file is member CACCSCF in the SCACCONF data set.
2. Look for the comment TCP/IP CONNECTION HANDLER.
3. If your network administrator has assigned the data server an IP address or a port number that is different from the default value, modify the IP address and port number. The sample TCP/IP connection handler is configured to use the 0.0.0.0 IP address. This address resolves to the IP address of the host where the data server is executing and is assigned the default port number 5002.
4. If you are not using the default high-level qualifier for the TCP/IP system data set:
  - a. Uncomment the TASK PARAMETERS configuration parameter.
  - b. Modify the TCPIP\_PREFIX and TCPIP\_MACH subparameters.
    - The TCPIP\_PREFIX variable sets the high-level qualifier for finding the TCP/IP system data sets. It can be set to use the installation defined data sets, or a user-defined data set.
    - The TCPIP\_MACH variable sets the address space name/subsystem name of the TCPIP stack. For IBM's TCP/IP system utilizing the Berkeley Socket interface, this parameter can also be specified in the hlq.TCPIP.DATA file under the parameter TCPIPUSERID.

The default for both variables is TCPIP:

```
TASK PARAMETERS= =TCPIP_PREFIX=TCPIP =TCPIP_MACH=TCPIP
```

5. Keep the rest of the values in the service information entry as they are.
6. Save the changes to the configuration file.

The service information entry should now look like the following example:

```
SERVICE INFO ENTRY = CACINIT TCPIP 2 1 1 100 4 5M 5M \  
TCP/IP_address_or_hostname/port_number_or_service
```

---

## Configuring name services

Configuring a name service involves deciding which control tables and filter tables you need, as well as whether to run the name service in its own data server.

### Restrictions

Only one name service can run within a single z/OS LPAR. To find out whether a name service is already running on an LPAR, issue the D GRS system command:

```
D GRS,RES=(SYSCAC,CACCNS*)
```

If the output of the D GRS command does not identify any generic resources under the minor CACCNS component name, no name services are running on the LPAR.

### Procedure

To configure a name service:

1. If you want to use list structures in a coupling facility to store the control tables and filter tables that are maintained by the name service, create one list structure for every control table and every filter table. See "Creating list

structures for control tables and filter tables.” IBM recommends using list structures for control tables in production environments.

2. In the configuration file for the every data server where you configured a correlation service, specify a value for the COMMON FILTER TABLE NAME parameter, identifying the filter table that the correlation service will maintain.
3. If you want to run the name service in a separate data server:
  - a. Comment out the CONTROL TABLE NAME and FILTER TABLE NAME configuration parameters.
  - b. configure and start that data server. See “Configuring and starting an independent data server to run the name service” on page 38.

IBM recommends running name services in a separate data server in production environments.

4. If you do not want to run the name service in a separate data server, in the configuration file for the data server specify values for the CONTROL TABLE NAME and FILTER TABLE NAME configuration parameters.

## Name services

Name services provide a common interface for all correlation services and change-capture agents on a single z/OS LPAR to access the control tables and filter tables.

*Control tables*, which correlation services access, record recovery information for each change-capture agent. *Filter tables*, which change-capture agents access, contain information that lets change-capture agents determine whether changes are for database objects or files that you want to capture changes from.

Control tables and filter tables can be local tables (on z/OS LPAR on which the correlation services and change-capture agents are running) or list structures that are defined in a coupling facility.

You can run a name service in either of two configurations:

- In a data server where one or more correlation services are configured.
- In a separate data server that is used only for running the name service.

Only one name service can run on a single z/OS LPAR.

You cannot shut down the data server that is running a name service until you first shut down all of the data servers on the z/OS LPAR that are running correlation services that are using the name service.

## Creating list structures for control tables and filter tables

You can create control tables and filter tables as list structures in a coupling facility.

### Before you begin

Before creating list structures for control tables, you must know the names of the correlation services that you plan to run on the z/OS LPAR where the name service will run. If you plan to run only one correlation service, you can use the name (NONAME)

Before creating list structures for filter tables, you must know the number of metadata catalogs that your correlation services will access. For each metadata catalog, there must be one unique filter table.

### About this task

Use list structures rather than local tables when it is important for the information in the control tables and filter tables to persist if the data server that is hosting the name service shuts down or terminates due to an error.

### Procedure

To create list structures for control tables and filter tables:

1. For each control table and filter table that you need to create, create list structure definition that can be input into the IXCMIAPU utility. See member in the SCACSAMP library for a sample definition of a list structure.
2. Run the IXCMIAPU utility and supply your list structure definitions.

## Configuring and starting an independent data server to run the name service

You can run the name service in a data server that is independent of the data servers where you plan to run your correlation services.

### Before you begin

- You must create the service information entries for the correlation services that you plan to have in your configuration.
- If you want to use list structures in a coupling facility to store your control tables and filter tables, those list structures must exist before you start a data server that is configured to run a name service.
- You must set a value for the COMMON FILTER NAME parameter in the configuration file for each data server where you plan to run correlation services.
- Before you start a data server that is configured to run a name service, make sure that none of the data servers where correlation services are configured are running.

### Restrictions

Only one data server in a z/OS LPAR can host a name service.

After you start this data server, you must not shut it down with an operator command until you shut down all of the other data servers that are hosting correlation services that are connected to the name service.

### About this task

Running the name service by itself in a data server, rather than in a data server where one or more correlation services are configured, is recommended for production environments.

### Procedure

To configure and start an independent data server to run the name service:

1. **z/OS LPAR:** Copy the data server JCL, which is in the CACCNS member of the SCACSAMP data set, to your PROCLIB and make the following changes to it:
  - a. Customize the JCL for your environment. See the instructions that are included as comments in the JCL.

- b. In the VHSCONF DD statement, point to the configuration file. The sample configuration file is member CACCNSCF in the SCACSAMP data set.
2. **z/OS LPAR:** Create storage for log messages. You can use data sets or log streams for storing log messages. See “Using the CACLOG DD for storing log messages” on page 29 or “Defining log streams for storing log messages” on page 30.
3. Open the configuration file for editing and perform the following tasks:
  - a. Define the logger service. See “Defining logger services” on page 34.
  - b. Configure the size of the memory region in which memory is allocated for services. See “MESSAGE POOL SIZE configuration parameter” on page 232.
  - c. Specify the control tables that you want to use. See “CONTROL TABLE NAME configuration parameter” on page 228.
  - d. Specify the filter tables that you want to use. See “COMMON FILTER TABLE NAME configuration parameter” on page 228.
4. Start the data server. See “Starting data servers” on page 139.

---

## Mapping tables and creating publications for Classic event publishing

Use Classic Data Architect to create relational tables and views that map to data sources in supported non-relational database management systems. You use these tables and views as sources for your publications in IBM WebSphere Classic Data Event Publisher for z/OS.

### Before you begin

You must perform the following tasks on the data server where the correlation service will run:

- Create and initialize a metadata catalog. Unless a metadata catalog exists in the data server where the correlation service will run, you cannot promote the tables and views that you create to that data server.
- Set up the configuration file.
- Customize and run the CACCTRL JCL in the SCACSAMP data set.
- If your configuration uses one data server, start that data server. If your configuration uses two data servers, start the data server where the correlation service is configured.
- Create WebSphere MQ objects to support Classic event publishing. You need to create a WebSphere MQ queue manager, a message queue to use as a restart queue (also called an in-doubt resolution queue), and at least one send queue.

### About this task

You create the relational tables and views in a project in Classic Data Architect. Then, you promote these objects to a data server.

### Procedure

To create tables and views that you can use as sources in Classic event publishing:

1. Configure Classic Data Architect by creating prerequisite objects, creating connections to data servers, setting preferences, importing reference files, and granting privileges. See “Configuring Classic Data Architect” on page 40.

2. Create tables and views that you can use as sources in publications. See “Mapping data for change capture” on page 46 and “Creating views on existing tables” on page 59.
3. **Optional:** Modify your tables or views. See “Viewing and modifying objects for Classic event publishing” on page 62.
4. Generate and run DDL to promote your tables and views to a data server. See “Generating DDL” on page 73.
5. **Optional:** If you choose not to run the DDL from Classic Data Architect but from the metadata utility, export the DDL to a remote z/OS host. See “Exporting SQL to remote z/OS hosts” on page 74.
6. On the data server where you plan to run the correlation service, run the DDL that is in member CACREPSP in the SCACSAMP data set. This DDL creates the structures that store the definitions of the publishing queue maps and publications that you create.
  - a. Use FTP to copy the DDL in a file with an SQL extension into the directory where your data design project is located. For example, if the name of your project is MyProject, the directory for the project might be (on Windows) C:\workspace\MyProject.
  - b. In Classic Data Architect, right-click the folder for the project that you copied the file into. Select **Refresh**. The SQL file appears in the SQL Scripts folder in the data design project that you selected.
  - c. Right-click the SQL file and select **Run**.
  - d. In the **Connection Selection** window, select the connection to the data server where you want to run the DDL and click **Finish**. You can create a new connection to a data server if a connection does not yet exist.
  - e. Look in the Data Output view to verify that the DDL ran correctly.
7. Specify the WebSphere MQ manager that you created, as well as the message queue that you created to use as the restart queue. See “Specifying the WebSphere MQ objects to use for change capture” on page 74.
8. Create at least one publishing queue map. Publishing queue maps specify send queues that you want to use for Classic event publishing. See “Creating and modifying publishing queue maps for Classic event publishing” on page 76.
9. Create at least one publication. See “Creating and modifying publications” on page 77

## Configuring Classic Data Architect

Before you or other users can use Classic Data Architect to create objects and promote them to data servers, you need to perform several configuration tasks.

### Before you begin

If you are using Classic Data Architect for Classic event publishing or Classic replication, you must perform the following tasks on the data server where the correlation service will run:

- Create and initialize a metadata catalog. Unless a metadata catalog exists in the data server where the correlation service will run, you cannot promote the tables and views that you create to that data server.
- Set up the configuration file.
- Customize and run the CACCTRL JCL in the SCACSAMP data set.
- Start the data server.

If you are using Classic Data Architect for Classic federation, you must perform the following tasks on the data server where the query processor will run:

- Create and initialize a metadata catalog. Unless a metadata catalog exists in the data server where the correlation service will run, you cannot promote the tables and views that you create to that data server.
- Set up the configuration file.
- Start the data server.

## Procedure

To configure Classic Data Architect:

1. Create objects for organizing your work in Classic Data Architect. See “Creating objects to organize your work.”
2. **Optional:** Create a connection to at least one data server. If you are using a DB2® for z/OS database for Classic federation, create a connection to that database. See “Creating connections to data servers and to DB2 for z/OS” on page 43. Connections to data servers are required if you plan to:
  - Run DDL on a data server directly from Classic Data Architect. If you do not want to run the DDL from Classic Data Architect, you can export the DDL in an SQL file for batch processing.
  - Use the discovery process in Classic Data Architect to locate in Adabas or CA-IDMS databases the data structures that you want to map to.
3. Set various preferences. See “Setting preferences” on page 44.
4. Import Classic reference files into a data design project. See “Importing COBOL copybooks, schema and subschema reports, and DBDs into projects” on page 42.
5. Grant privileges to users to create objects or to run commands on a data server. See “Granting privileges and privileges for performing actions on data servers” on page 44.

## Creating objects to organize your work

You organize the tables, views, indexes, and stored procedures that you create in different containers within Classic Data Architect.

### About this task

#### Workspaces

When you first open Classic Data Architect, you create a workspace that will contain your work. All users who work in Classic Data Architect have their own workspaces and their own projects within those workspaces. Within a workspace, there are two types of objects that you must create: data design projects and physical data models.

#### Data design projects

Within a project, you design the tables and views that you eventually will create in the metadata catalog on the data server. The type of project that you create in Classic Data Architect is called a *data design project*. You can create any number of data design projects, using them to organize your physical data models, or you can put all of your physical data models into one project.

#### Physical data models

A physical data model is a collection of schemas that contain the tables that you create to map to the data in your data sources. Schemas can also contain views on those tables and stored procedures that you might want

to use to perform operations on the result sets for queries. Instead of creating your tables directly in the metadata catalog on the data server and modifying them there, you create and modify your tables in the model and then promote them to a metadata catalog.

For example, you might have one data server for a test environment and another for a production environment. In a model, you can create a table and then run the DDL to create the table in the test data server to test the table. If you need to modify the table, you can drop the table from the metadata catalog in the test data server, modify the table in Classic Data Architect, then re-create the table in the metadata catalog and test it again. When you have a version of the table that you want to put into production, you can create the table in the production data server.

## Procedure

To set up a work environment:

1. Create a data design project. Open the New Data Design Project wizard by selecting **File > New > Data Design Project** from the menu bar at the top of Classic Data Architect. The data design project appears in the Data Project Explorer.
2. Create a physical data model with the database type "Classic Integration." Open the New Physical Data Model wizard by right-clicking the data design project folder and selecting **New > Physical Data Model**. A new physical data model appears in the Data Models folder of your project. Close the Diagram1 tab because diagramming is a function in Eclipse but not a function in Classic Data Architect.

## Importing COBOL copybooks, schema and subschema reports, and DBDs into projects

Before you can create tables with Classic Data Architect, you must import Classic reference files into a data design project. For CA-Datcom, CICS VSAM, sequential, native VSAM, and IMS, you import COBOL copybooks and DBDs; For CA-IDMS, you import schema and subschema reports.

### About this task

If you want to create tables that map to Adabas databases, you do not need to import files that describe the data structures that you want to map to. When you create an Adabas table, you provide information that tells Classic Data Architect how to locate the data structures in the Adabas database that you want to use.

If you want to create tables that map to CA-IDMS databases, you can either import schema and subschema reports into Classic Data Architect or you can tell Classic Data Architect how to locate the records and sets that you want to use in the CA-IDMS database.

When you import files into a project from a local or remote server, the files are organized into subfolders for that project according to the file types.

- COBOL copybooks are placed into a subfolder that is called **COBOL Copybooks**.
- CA-IDMS schema and subschema reports are placed into a subfolder that is called **CA-IDMS Schemas/Subschemas**.
  - You can also tell Classic Data Architect to discover subschema information directly from CA-IDMS.

- DBD files are placed into a subfolder called **IMS DBDs**.

You can then use the files that you import into a project to create one or more tables in one or more schemas in any project.

To select the files to import, you can browse the local file system for files or connect to a z/OS server and browse data sets for members to download using FTP.

When you import a file from your local file system, the file extension is replaced. For example, a COBOL copybook `mycopybook.txt` is imported into the project as `mycopybook.cpy`. Imported files are given the following extensions:

*Table 2. File types and extensions given when files are imported*

File type	Extension
COBOL copybook	cpy
Schema	sch
Subschema	sub
DBD	dbd

When you import subschemas and schemas, the wizard checks that the subschema belongs to the specified schema and that the file or data set member exists. No other validation is done.

### Procedure

To import reference files into a project:

1. In the Data Project Explorer, right-click the project folder and select **Import**.
2. Select **Classic Data Architect References** and click **Next**.
3. Use the wizard to select the location and verify the contents of the reference files that you want to import, select the references, and specify the folder that you want to import the files into.
4. **Optional:** If you imported one or more copybooks, you can validate the copybooks one at a time. In the **COBOL Copybooks** folder, right-click a copybook and select **Validate COBOL copybook**. If Classic Data Architect detects errors in the copybook, the copybook will open in an editor. The **Problem** page of the **Properties** view will describe the errors.

### Creating connections to data servers and to DB2 for z/OS

Several of the tasks that you perform in Classic Data Architect require connections to data servers or to DB2 for z/OS databases.

#### About this task

You must connect to a data server to run SQL DDL to promote your tables, views, and stored procedures to a metadata catalog.

If you plan to use Classic federation to query or update a DB2 for z/OS database, you must create a connection to that database from Classic Data Architect.

### Procedure

To create connections to data servers and to DB2 for z/OS:

- Create connections to one or more data servers. Open the New Connection wizard by right-clicking the Connections folder in the Database Explorer and selecting **New Connection**.
- If you are using a DB2 for z/OS database, create a connection to that database. Open the New Connection wizard by right-clicking the Connections folder in the Database Explorer and selecting **New Connection**.

## Setting preferences

You can set various preferences in Classic Data Architect.

### Procedure

To set preferences:

1. Set various global values that Classic Data Architect can use as default values in its wizards. Open the Preferences window by selecting **Window > Preferences**.
2. Under Classic Integration Mapping, you can:
  - Set preferences for discovering Adabas files and mapping Adabas fields to columns in relational tables..
  - Set preferences for access to CICS VSAM.
  - Set an FTP subcommand for Classic Data Architect to use for DBCS data when importing Classic Data Architect References or exporting files that contain SQL statements.
  - Specify whether to map PIC9(n) USAGE DISPLAY data to the character or decimal SQL data type.
3. Specify the locale that the COBOL parser will use when validating COBOL copybooks that you want to base tables on. Select **COBOL** in the Preferences window, and then select the **More COBOL options** tab. Set the locale in the **Compile time locale name** field.

## Granting privileges and privileges for performing actions on data servers

If you want users of Classic Data Architect to perform actions in a metadata catalog on a data server, such as run SQL scripts or view objects, you must grant privileges to those users. You must also grant system-level privileges to any users who need to run commands on a data server.

### About this task

The Privileges page of the Properties view for a database in a data design project only lists privileges. Adding privileges to the list does not automatically grant those privileges on a data server.

To grant a privilege, you must create the privilege in the Privileges page, generate the GRANT statement for that privilege, and then run the GRANT statement on a data server.

### Procedure

To grant privileges for performing actions in a metadata catalog on a data server:

1. Select the database in your data design project.
2. In the Properties view, click the **Privileges** tab. The table on the Privileges page lists the users who have one or more privileges.

3. Create a new privilege. Click the yellow icon (  ) In the Grant System or Database Privilege window and specify these values:

**Grantee**

Type the ID of the user or group that you want to grant the privilege to, or select PUBLIC to grant the privilege to all users.

**Type** Select either SYSTEM or one of the following types:

**\$ADABAS**

Allows grantees to create, drop, and view Adabas tables and views on a data server.

**\$CFI** Allows grantees to create, drop, and view a metadata catalog on a data server.

**\$DATACOM**

Allows grantees to create, drop, and view CA-Datcom tables and views on a data server.

**\$IDMS**

Allows grantees to create, drop, and view CA-IDMS tables and views on a data server.

**\$IMS** Allows grantees to create, drop, and view IMS tables and views on a data server.

**\$SEQUENT**

Allows grantees to create, drop, and view sequential tables and views on a data server.

**\$SP** Allows grantees to create, drop, and view stored procedures on a data server.

**\$VSAM**

Allows grantees to create, drop, and view CICS VSAM and native VSAM tables and views on a data server.

**Privilege**

If you selected SYSTEM in the **Type** field, select one of the following privileges:

**SYSADM**

Grants all privileges on all objects that are in a metadata catalog. Users with this privilege can grant privileges and privileges to other users.

**SYSOPR**

Grants remote operator privileges to display reports for and manage a data server.

**DISPLAY**

Grants remote operator privileges for displaying reports for a data server.

4. When you want to promote the privileges to a data server, right-click the database and select **Generate DDL**. In the Generate DDL wizard, follow these steps:
- On the **Options** page of the wizard, deselect all check boxes except **Fully qualified names**, **Quoted identifiers**, and **GRANT statements**.
  - On the **Objects** page, deselect all of the check boxes.

- c. On the **Save and Run DDL** page, specify the name of the SQL file that the wizard will create. Verify that the GRANT statements are correct. Select the **Run DDL on server** check box.
- d. On the **Select Connection** page, select the connection to the data server, or create a new connection to a data server.
- e. On the **Summary** page, verify the actions that the wizard will perform and click **Finish**.
- f. In the Data Output view (which is by the Properties view by default), verify that the SQL statements ran successfully on the data server.

### Revoking privileges for performing actions on data servers:

Deleting privileges removes them only from a database in a data design project. Revoking privileges removes them from a data server.

### About this task

The Privileges page of the Properties view for a database in a data design project only lists privileges. Deleting a privilege from that list does not automatically revoke that privilege from a data server.

### Procedure

To revoke a privilege that exists on a data server:

1. In the Data Project Explorer, right-click the database in which the privilege is listed.
2. On the **Privileges** page of the Properties view, select the **Revoke** check box for the privilege that you want to revoke.
3. Right-click the database and select **Generate DDL**. In the Generate DDL wizard, follow these steps:
  - a. On the **Options** page of the wizard, deselect all check boxes except **Fully qualified names**, **Quoted identifiers**, and **GRANT statements**.
  - b. On the **Objects** page, deselect all of the check boxes.
  - c. On the **Save and Run DDL** page, specify the name of the SQL file that the wizard will create. Verify that the REVOKE statements are correct. Select the **Run DDL on server** check box.
  - d. On the **Select Connection** page, select the connection to the data server.
  - e. On the **Summary** page, verify the actions that the wizard will perform and click **Finish**.
  - f. In the Data Output view (which is by the Properties view by default), verify that the SQL statements ran successfully on the data server.

## Mapping data for change capture

Change capture takes place when a change-capture agent sends messages to a correlation service to notify that correlation service of changes to the data in a data source. Change capture is part of Classic event publishing and Classic replication.

The following list shows all of the tasks that you can perform for mapping data for change capture.

## Creating Adabas tables and views for change capture

To capture changes from an Adabas database, you must create a relational table that maps to that database. You can also create a view on the table. Use the New Adabas Table wizard to create a table and a view.

### Before you begin

- Configure the data server where you plan to run the correlation service that will process change data from your Adabas database.
- Create a metadata catalog.
- Decide which data structures to map in your database.
- Configure a connection between the data server and your Adabas database.
- If you want to use Predict, you must know the Adabas file number of the Predict dictionary and the name of the view that you want to map to. If you are not using Predict, you must know the number of the Adabas file that you want to map to.

### Restrictions

- Each column in the table that you create must be associated with a field in the file, a superdescriptor, or a subdescriptor.
- If Predict formatting is available, the following Predict formats are supported:
  - character (A,AL,AV)
  - binary (B) with length of 2 or 4
  - date (D, DS, DT)
  - floating point (F)
  - integer (I)
  - logical (L)
  - numeric packed and unpacked (N, NS, P, PS, U, US)
  - time (T, TS)

If only Adabas field formatting is available, the following formats are supported:

- alphanumeric (A)
- binary (B) with length of 2 or 4
- fixed point (F)
- floating point (G)
- packed decimal (P) and unpacked decimal (U)

### About this task

For more information about creating tables and views that map to Adabas databases, see the related links for Adabas syntax diagrams and for views.

### Procedure

To create an Adabas table and optionally a view for change capture:

1. **Optional:** Use the Adabas page of the Preferences window to set these default values:
  - The name of the Predict dictionary that you want to use.
  - The date format that Classic Data Architect should convert dates to.
  - The time format that Classic Data Architect should convert times to.
  - The maximum length for VARCHAR data.

- The maximum length for LVARCHAR data.
  - The maximum number of occurrences for all fields that occur multiple times in an Adabas file.
  - Whether to use the **User Synonym** field from the Predict Dictionary when this field is defined.
2. Map your Adabas database to a relational table by using the New Adabas Table wizard.
    - a. Open this wizard by right-clicking either the database in your data design project or one of the schemas within the database. Select **Add Classic Object > Adabas table**.
    - b. Select the model and schema in which to create the table.
    - c. Choose whether or not to create a view on the table.
    - d. Choose either to connect to your Adabas database through an existing connection to a data server, or to create a new connection to a data server. Either data server must be configured to access the Adabas database.
    - e. Specify the format of dates and times, the lengths of VARCHAR and LVARCHAR data types, and the maximum number of occurs. The default values that appear are either the global defaults that are set for the Adabas database or the defaults that are set in the **Adabas** page of the Preferences window.
    - f. Specify that you want to use the table (and view, if you are creating one) for change capture.
    - g. Provide either the Predict or Adabas information that is necessary for the discovery process.
    - h. Select the Adabas fields to map to columns in your relational table.
    - i. If you are creating a view, specify the criteria for the WHERE clause.
    - j. Modify the names of columns and provide null values.

When you finish the wizard, the new table appears under the selected schema. If you created a view, the view also appears under the selected schema.

3. **Optional:** Modify the table properties or add privileges. Select the table and make any changes in the Properties view.
4. **Optional:** Generate the DDL for the table. You can generate the DDL later, if you do not want to generate it now. You can also generate the DDL for all of the objects within the same schema. See “Generating DDL” on page 73.
  - a. Right-click the table and select **Generate DDL**.
  - b. In the Generate DDL wizard, follow these steps:
    - 1) Choose to generate CREATE statements.
    - 2) Choose to generate DDL for tables.
    - 3) Name the file in which to save the DDL within your project.
    - 4) Choose whether to run the DDL on a data server. After you run the DDL, check the Data Output view to determine whether the DDL ran successfully.
    - 5) Choose whether to open the DDL for editing.
5. **Optional:** If you ran the DDL successfully on a data server, validate the table by running a test query against your Adabas database. Be sure that the data server is connected to that database.
  - a. In the Database Explorer, search your data server for the schema that you created the table in. Expand the schema and expand the **Tables** folder.
  - b. Right-click the table and select **Data > Sample Contents**.

- c. Check the Data Output view to determine whether the test query ran successfully.
6. **Optional:** If you created a view,, generate the DDL for the view. You can generate the DDL later. You can also generate the DDL for all of the objects within the same schema. See “Generating DDL” on page 73.
  - a. Right-click the view and select **Generate DDL**.
  - b. In the Generate DDL wizard, follow these steps:
    - 1) Choose to generate CREATE and ALTER statements.
    - 2) Choose to generate DDL for views.
    - 3) Name the file in which to save the DDL within your project.
    - 4) Choose whether to run the DDL on a data server. After you run the DDL, check the Data Output view to determine whether the DDL ran successfully.
    - 5) Choose whether you want to open the DDL for editing.
7. **Optional:** If you ran the DDL successfully on a data server, validate the view by running a test query against your Adabas database. Be sure that the data server is connected to that database.
  - a. In the Database Explorer, search your data server for the schema that you created the view in. Expand the schema and expand the **Views** folder.
  - b. Right-click the view and select **Data > Sample Contents**.
  - c. Check the Data Output view to determine whether the test query ran successfully.

## Creating CA-IDMS tables and views for change capture

To capture changes from a CA-IDMS database, you must create a relational table that maps to that database. You can also create a view on the table to filter record types or to filter rows and columns. Use the New CA-IDMS Table wizard to create the table and optionally the view.

### Before you begin

- Configure the data server where you plan to run the correlation service that will process change data from your CA-IDMS database.
- Create a metadata catalog.
- Decide which records to map to and the best path through the database to access the records. The sets that are defined in the subschema for the records determine the path.
- Configure a connection between the data server and your CA-IDMS database. The data server must be able to access the CA-IDMS central version that contains the subschema definitions, schema definitions, and data of the records that are being mapped.

### Restrictions

- The record path in the mapping must be from set owner to set member only.
- The owner DBKEY must be part of each member record that is included in the path.

### About this task

In the New CA-IDMS Table wizard, you can map a single record or a specific path to as many as 10 records. You define a path by starting with a single record, and then navigating sets to additional records defined in the subschema. The subschema information that you use for mapping determines which records and

sets are available. You can import the subschema information from a combination of CA-IDMS schema and subschema report files or directly from the CA-IDMS database by using Classic Data Architect's discovery process.

CA-IDMS schema and subschema reports are produced by running the CA-IDMS schema and subschema compilers and capturing the punched output into a z/OS data set. JCL to punch these reports is in the SCACSAMP library with the member name CACIDPCH.

For more information about creating tables and views that map to CA-IDMS databases, see the related links for CA-IDMS syntax diagrams and for views.

## Procedure

To create a CA-IDMS table and optionally a view for change capture:

1. Map your CA-IDMS database to a relational table and optionally create a view by using the New CA-IDMS Table wizard.
  - a. Open this wizard by right-clicking either the database in your data design project or one of the schemas within the database. Select **Add Classic Object > CA-IDMS table**.
  - b. Select the CA-IDMS schema and subschema to base the table on.
  - c. Specify that the table is for change capture.
  - d. Choose whether to create a view on the table.
  - e. Provide information that specifies how to access the CA-IDMS database.
  - f. For each record in the path, specify a COBOL copybook, select an 01 level if there is more than one 01 level, and then select the elements to map as columns in your relational table. See "Mapping CA-IDMS paths for change capture" on page 51.
  - g. Select the elements to map to columns in your relational table.
  - h. If you are creating a view, specify the criteria for the WHERE clause.

When you finish the wizard, the new table appears under the selected schema. If you created a view, it also appears under the selected schema.

2. **Optional:** Modify the table properties or add privileges. Select the table and make any changes in the Properties view.
3. **Optional:** Generate the DDL for the table. You can generate the DDL later, if you do not want to generate it now. You can also generate the DDL for all of the objects within the same schema. See "Generating DDL" on page 73.
  - a. Right-click the table and select **Generate DDL**.
  - b. In the Generate DDL wizard, follow these steps:
    - 1) Choose to generate CREATE statements.
    - 2) Choose to generate DDL for tables.
    - 3) Name the file in which you want to save the DDL within your project.
    - 4) Choose whether to run the DDL on a data server. After you run the DDL, check the Data Output view to determine whether the DDL ran successfully.
    - 5) Choose whether you want to open the DDL for editing.
4. **Optional:** If you ran the DDL successfully on a data server, validate the table by running a test query against your CA-IDMS database. Be sure that the data server is connected to that database.

- a. In the Database Explorer, search your data server for the schema that you created the table in. Expand the schema and expand the **Tables** folder.
  - b. Right-click the table and select **Data > Sample Contents**.
  - c. Check the Data Output view to determine whether the test query ran successfully.
5. **Optional:** If you created a view, generate the DDL for the view. You can generate the DDL later. You can also generate the DDL for all of the objects within the same schema. See “Generating DDL” on page 73.
- a. Right-click the view and select **Generate DDL**.
  - b. In the Generate DDL wizard, follow these steps:
    - 1) Choose to generate CREATE and ALTER statements.
    - 2) Choose to generate DDL for views.
    - 3) Name the file in which to save the DDL within your project.
    - 4) Choose whether to run the DDL on a data server. After you run the DDL, check the Data Output view to determine whether the DDL ran successfully.
    - 5) Choose whether you want to open the DDL for editing.
6. **Optional:** If you ran the DDL successfully on a data server, validate the view by running a test query against your CA-IDMS database. Ensure that the data server is connected to that database.
- a. In the Database Explorer, search your data server for the schema that you created the view in. Expand the schema and expand the **Views** folder.
  - b. Right-click the view and select **Data > Sample Contents**.
  - c. Check the Data Output view to determine whether the test query ran successfully.

### Mapping CA-IDMS paths for change capture:

When you create tables for change capture, you map paths by selecting a starting record in the subschema and navigating set relationships to the final target record for change capture.

The set navigation path in the mapping must be from owner to member records only. The correlation service is limited to navigating up owner chains when gathering related information.

Only changes to the final record in the path are captured. Changes to any other record in the path are ignored unless you map a separate table to capture changes to that record. However, mapping related records is required when fields in those records provide context to changed records in the database schema. For example, if a CA-IDMS schema contains salary records that are related to employee records by a set, to insert an instance of the SALARY record for a specific employee would require the employee ID in the EMPLOYEE record to be included in the message sent by the distribution service.

Change data in records other than the final record is gathered outside of the updating transaction. The latency of collecting this information is based on the amount of time that it takes the change-capture agent to send the change to the correlation service plus the time required to package the change and forward the change to the distribution service. Related change data itself might have been updated or deleted between the transaction and the time that the information was

forwarded to the distribution service. For this reason, map only identifying fields (for example, primary or foreign keys) in related records that are not subject to change.

If a related record is deleted between the time that an update transaction takes place and the time that the change is forwarded to the distribution service, related fields are passed to the distribution service as SQL NULL to indicate that the information is not available.

You might need to modify the application schema to ensure that related data information can be gathered in the case of record deletes. In some cases, CA-IDMS does not keep an owner pointer in the record prefix, and therefore owner information is not available if the record is deleted. Change capture is not supported in these cases.

If you require owner information for delete purposes, then you must change one or more set definitions in the path to include the LINKED TO OWNER statement so that owner pointers are maintained in the record prefix information. For more information about updating the schema, see your CA/CA-IDMS reference manuals.

In some cases, you might need to gather information from multiple record owners using different paths in the schema. Because change capture is supported for only a single path in the schema, gathering identifying fields from multiple paths is not possible in a single table mapping. However, because change messages sent to the distribution service are based on a single unit-of-work in the database, you can define multiple tables for change capture.

Using the CA-IDMS employee demo schema as an example, assume that you are interested in capturing changes to the employee record but you require both the office code and department ID that are associated with an employee when the employee is changed. You can map the path DEPARTMENT->EMPLOYEE as one table and OFFICE->EMPLOYEE as a second table to gather both the department ID and office ID when an employee is changed. The application that receives change data can combine the information from both mapped tables as a change to the employee record, and automatically trigger a change message for both table mappings.

### **Creating CICS VSAM tables and views for change capture**

To capture changes from a CICS VSAM file, you must create a relational table that maps to that file. You can also create a view on the table to filter record types and to filter rows and columns.

#### **Before you begin**

- Configure the data server where you plan to run the correlation service that will process change data from your CICS VSAM file.
- Create a metadata catalog.
- Decide which record elements to map in the CICS VSAM file.
- Configure a connection between the data server and your CICS VSAM file.
- Ensure that the **COBOL Copybooks** folder in your project contains a copybook that lists the records that you want to map to columns.

#### **Restrictions**

The VSAM file that you map to must correspond to a cluster for an ESDS, KSDS, or RRDS data set.

### About this task

For more information about creating tables and views that map to CICS VSAM files, see the related links for VSAM syntax diagrams and for views.

### Procedure

To create a CICS VSAM table and optionally a view for change capture:

1. Map your CICS VSAM file to a relational table and optionally create a view by using the New CICS VSAM Table wizard.
  - a. Open the wizard by right-clicking either the database in your data design project or one of the schemas within the database. Select **Add Classic Object > CICS VSAM table**.
  - b. Select the COBOL copybook to base the table on.
  - c. Specify that the table is for change capture.
  - d. Choose whether or not to create a view on the table.
  - e. Provide information about which CICS file control table to use and how to access the CICS VSAM file.
  - f. Select the elements to map to columns in your relational table.
  - g. If you are creating a view, specify the criteria for the WHERE clause.

When you finish the wizard, the new table appears under the selected schema. If you created a view, the view also appears under the selected schema.

2. **Optional:** Modify the table properties or add privileges. Select the table and make any changes in the Properties view.
3. **Optional:** Generate the DDL for the table. You can generate the DDL later. You can also generate the DDL for all of the objects within the same schema. See “Generating DDL” on page 73.
  - a. Right-click the table and select **Generate DDL**.
  - b. In the Generate DDL wizard, follow these steps:
    - 1) Choose to generate CREATE statements.
    - 2) Choose to generate DDL for tables.
    - 3) Name the file in which to save the DDL within your project.
    - 4) Choose whether to run the DDL on a data server. After you run the DDL, check the Data Output view to determine whether the DDL ran successfully.
    - 5) Choose whether to open the DDL for editing.
4. **Optional:** If you ran the DDL successfully on a data server, validate the table by running a test query against your CICS VSAM file. Be sure that the data server is connected to the system where the file is located.
  - a. In the Database Explorer, search your data server for the schema that you created the table in. Expand the schema and expand the **Tables** folder.
  - b. Right-click the table and select **Data > Sample Contents**.
  - c. Check the Data Output view to determine whether the test query ran successfully.
5. **Optional:** If you created a view, generate the DDL for the view. You can generate the DDL later. You can also generate the DDL for all of the objects within the same schema. See “Generating DDL” on page 73.

- a. Right-click the view and select **Generate DDL**.
- b. In the Generate DDL wizard, follow these steps:
  - 1) Choose to generate CREATE and ALTER statements.
  - 2) Choose to generate DDL for views.
  - 3) Name the file in which to save the DDL within your project.
  - 4) Choose whether to run the DDL on a data server. After running the DDL, check the Data Output view to find out whether the DDL ran successfully.
  - 5) Choose whether you want to open the DDL for editing.
6. **Optional:** If you ran the DDL successfully on a data server, you can validate the view by running a test query against your CICS VSAM file. Ensure that the data server is connected to the system where the file is located.
  - a. In the Database Explorer, search your data server for the schema that you created the view in. Expand the schema and expand the **Views** folder.
  - b. Right-click the view and select **Data > Sample Contents**.
  - c. Check the Data Output view to determine whether the test query ran successfully.

## Creating IMS tables and views for change capture

To capture changes from an IMS database, you must create a relational table that maps to that database. You can also create a view on the table to filter record types or to filter rows and columns.

### Before you begin

- Configure the data server where you plan to run the correlation service that will process change data from your IMS database.
- Create a metadata catalog.
- Decide which segment to map and the required path for navigating to the segment from a physical root or index.
- Configure a connection between the data server and your IMS database, if you plan to generate and run the DDL to create the table and optionally the view in the metadata catalog.
- Ensure that the IMS DBDs folder in your project contains a database definition file (DBD) that lists the segments from which you want to select the fields to map to columns.
- Ensure the COBOL Copybooks folder in your project contains a copybook for each of the IMS segments that you want to map to.

### Restrictions

- You must augment DBDs if you want to use them for change capture. See <http://publib.boulder.ibm.com/infocenter/iiclzos/v9r1/index.jsp?topic=/com.ibm.websphere.ii.eventpub.imscc.doc/configuring/iiyeccimoaugdbd.dita>.
- To reduce the amount of information that is sent to the publication service, define columns only for the data that changes in the segments for which you want to capture changes.
- If you have redefinitions in your segments, define a table for each redefinition. Ensure that each table contains the control column that allows you to identify a record's type.

### About this task

For more information about creating tables and views that map to IMS databases, see the related links for IMS syntax diagrams and for views.

## Procedure

To create an IMS table and optionally a view for change capture:

1. Map your IMS database to a relational table and optionally create a view by using the New IMS Table wizard.
  - a. Open this wizard by right-clicking either the database in your data design project or one of the schemas within the database. Select **Add Classic Object > IMS table**.
  - b. Select the DBD file to base the table on.
  - c. Specify that the table is for change capture.
  - d. Choose whether to create a view on the table.
  - e. Provide information about how to access the IMS database.
  - f. For the each segment that is in the path, specify a COBOL copybook, select the desired 01 level if there is more than one, and select the elements that you want to map as columns.
  - g. If you are creating a view, specify the criteria for the WHERE clause.

When you finish the wizard, the new table appears under the selected schema. If you created a view, it also appears under the selected schema.

2. **Optional:** Modify the table properties or add privileges. Select the table and make any changes in the Properties view.
3. **Optional:** Generate the DDL for the table. You can generate the DDL later. You can also generate the DDL for all of the objects within the same schema. See “Generating DDL” on page 73.
  - a. Right-click the table and select **Generate DDL**.
  - b. In the Generate DDL wizard, follow these steps:
    - 1) Choose to generate CREATE statements.
    - 2) Choose to generate DDL for tables.
    - 3) Name the file in which to save the DDL within your project.
    - 4) Choose whether to run the DDL on a data server. After you run the DDL, check the Data Output view to determine whether the DDL ran successfully.
    - 5) Choose whether to open the DDL for editing.
4. **Optional:** If you ran the DDL successfully on a data server, validate the table by running a test query against your IMS database. Be sure that the data server is connected to that database.
  - a. In the Database Explorer, search your data server for the schema that you created the table in. Expand the schema and expand the **Tables** folder.
  - b. Right-click the table and select **Data > Sample Contents**.
  - c. Check the Data Output view to determine whether the test query ran successfully.
5. **Optional:** If you created a view, generate the DDL for the view. You can generate the DDL later. You can also generate the DDL for all of the objects within the same schema. See “Generating DDL” on page 73.
  - a. Right-click the view and select **Generate DDL**.
  - b. In the Generate DDL wizard, follow these steps:
    - 1) Choose to generate CREATE and ALTER statements.

- 2) Choose to generate DDL for views.
  - 3) Name the file in which to save the DDL within your project.
  - 4) Choose whether to run the DDL on a data server. After running the DDL, check the Data Output view to determine whether the DDL ran successfully.
  - 5) Choose whether to open the DDL for editing.
6. **Optional:** If you ran the DDL successfully on a data server, validate the view by running a test query against your IMS database. Ensure that the data server is connected to that database.
- a. In the Database Explorer, search your data server for the schema that you created the view in. Expand the schema and expand the **Views** folder.
  - b. Right-click the view and select **Data > Sample Contents**.
  - c. Check the Data Output view to determine whether the test query ran successfully.

### Creating native VSAM tables and views for change capture

To capture changes that are made to a native VSAM file, you must create a relational table that maps to that file. You can also create a view on the table to filter record types or to filter rows and columns.

#### Before you begin

- Configure the data server where you plan to run the correlation service.
- Create a metadata catalog.
- Ensure that a COBOL copybook that references the file exists in the **COBOL Copybooks** folder in your data design project.

#### Restrictions

The VSAM files that you map to must correspond to a cluster for an ESDS, KSDS, or RRDS data set.

The VSAM files that you map to must not be under RLS (record-level sharing) or TVS control.

#### Procedure

To create a VSAM table and optionally a view for change capture:

1. Map your VSAM file to a relational table and optionally a view by using the New VSAM Table wizard.
  - a. Open this wizard by right-clicking either the database in your data design project or one of the schemas within the database. Select **Add Classic Object > VSAM table**.
  - b. Select the copybook to base the table on.
  - c. Choose to use the table for change capture.
  - d. Choose whether to create a view on the table.
  - e. Provide information about how to access the VSAM file.
  - f. Provide the URL of a Cross Memory (XM) queue. The URL consists of these values:
    - The first four characters of the name of the data space for the queue.

- The first four characters of the name of the queue. The change-capture agent that is capturing changes to the native VSAM table writes change data to this XM queue.

The format of the XM URL is `XM1/name_of_data_space/name_of_queue`

- Select the elements to map to columns in your relational table.
- If you are creating a view, specify the criteria for the WHERE clause.

When you finish the wizard, the new table appears under the selected schema. If you created a view, it also appears under the selected schema.

- Optional:** Modify the table properties or add privileges. Select the table and make any changes in the Properties view.
- Optional:** Generate the DDL for the table. You can generate the DDL later, if you do not want to generate it now. You can also generate the DDL for all of the objects within the same schema. See “Generating DDL” on page 73.
  - Right-click the table and select **Generate DDL**.
  - In the Generate DDL wizard, follow these steps:
    - Choose to generate CREATE statements.
    - Choose to generate DDL for tables. You can also choose to generate DDL for indexes.
    - Name the file in which to save the DDL within your project.
    - Choose whether to run the DDL on a data server. After you run the DDL, check the Data Output view to determine whether the DDL ran successfully.
    - Choose whether to open the DDL for editing.
- Optional:** If you ran the DDL successfully on a data server, validate the table by running a test query against your VSAM file. Ensure that the data server is connected to the system where the file is located.
  - In the Database Explorer, search your data server for the schema that you created the table in. Expand the schema and expand the **Tables** folder.
  - Right-click the table and select **Data > Sample Contents**.
  - Check the Data Output view to determine whether the test query ran successfully.
- Optional:** If you created a view, generate the DDL for the view. You can generate the DDL later. You can also generate the DDL for all of the objects within the same schema. See “Generating DDL” on page 73.
  - Right-click the view and select **Generate DDL**.
  - In the Generate DDL wizard, follow these steps:
    - Choose to generate CREATE and ALTER statements.
    - Choose to generate DDL for views.
    - Name the file in which to save the DDL within your project.
    - Choose whether to run the DDL on a data server. After you run the DDL, check the Data Output view to find out whether the DDL ran successfully.
    - Choose whether to open the DDL for editing.
- Optional:** If you ran the DDL successfully on a data server, validate the view by running a test query against your VSAM file. Be sure that the data server is connected to the system where the file is located.
  - In the Database Explorer, search your data server for the schema that you created the view in. Expand the schema and expand the **Views** folder.
  - Right-click the view and select **Data > Sample Contents**.

- c. Check the Data Output view to determine whether the test query ran successfully.

## Altering tables to support change capture

If you did not choose to use a table for change capture when you created it, you can alter the table so that you can use it for change capture.

### Restrictions

- You cannot alter a table for change capture if the table contains record arrays.
- You cannot capture changes from tables for CA-Datcom, DB2, and sequential data sources.
- For CA-IDMS tables, all set relationships in the path must be owner-to-member relationships.
- For IMS tables, the following restrictions apply:
  - The DBD cannot be defined as a Logical or INDEX database.
  - The DBD must have IMS data capture turned on.
  - A non-sequence field cannot be included in the table definition if the DBD change-capture option of KEY is specified. IMS captures changes only to key data. Therefore, your IMS table can include only columns that map to that key data.
  - All columns must map to the leaf segment if the IMS DBD change-capture option of DATA only is specified.
  - All columns must map to non-leaf segments if the IMS DBD change-capture option of PATH only is specified.
- For CICS VSAM and native VSAM tables, the following restrictions apply:
  - The VSAM file cannot reference an alternate index.
  - If you want to capture changes from native VSAM, the VSAM file must be defined as a DS data set, not as a DD data set.
- For native VSAM tables, you must define a Cross Memory URL.

### Procedure

To alter a table for change capture:

- To alter a table in a data design project:
  1. In the Data Project Explorer, select the table.
  2. On the General page of the Properties view, set the DATA CAPTURE flag to CHANGES.
  3. Generate the DDL for the table and run the DDL on a data server.
- To alter a table that already exists in a metadata catalog:
  1. In the Database Explorer, expand the connection to your data server until you locate the table.
  2. Right-click the table and select **Enable Change Capture**.

The change to the table takes effect immediately.

## Altering views to support change capture

You can alter a view so that you can use the view as a source for change capture.

### Restrictions

You can alter a view for change capture only if the view meets all three of these criteria:

- The view references only one table that is altered for change capture.
- The view references all of the columns in the base table and renames none of the columns.
- The WHERE clause does not contain references to other tables.
- The view references an Adabas, CA-IDMS, CICS VSAM, IMS, or native VSAM table.

### Procedure

To alter a view for change capture:

- To alter a view in a data design project:
  1. In the Data Project Explorer, select the view.
  2. On the General page of the Properties view, set the DATA CAPTURE flag to CHANGES.
  3. Generate the DDL for the view and run the DDL on a data server.
- To alter a view that already exists in a metadata catalog:
  1. In the Database Explorer, expand the connection to your data server until you locate the view.
  2. Right-click the view and select **Enable Change Capture**.

The change to the view takes effect immediately.

## Creating views on existing tables

You can create a view on a table that already exists in a data design project, with the SQL builder, with the SQL editor, or with the Properties view.

If you create a view that you want to use for a publication or subscription, remember to alter the view to support change capture.

### Creating views on existing tables with the SQL builder

To create a view on a table that already exists either only in your project or also in a metadata catalog, you can use the SQL builder, a graphical utility that builds the SELECT statement for the view for you. After you create the SELECT statement, you can add the view to a schema in your project.

### Restrictions

If you are creating a view for change capture, these restrictions apply:

- The view cannot reference more than one table. This includes tables in the FROM clause or the WHERE clause as in the case of sub-selects.
- The view cannot reference another view.
- The view must reference all of the columns in the base table.
- The base table must not map to record arrays.

### Procedure

To create a view with the SQL builder:

1. In the Data Project Explorer, expand the physical data model in which you are working. Expand the database in which you are working. Right-click the **SQL Statements** folder and select **New SQL Statement**.
2. In the New SQL Statement window, follow these steps:
  - a. Ensure that SELECT is selected in the **Statement template** field.

- b. Give the statement a descriptive name.
- c. Ensure that the **SQL builder** radio button is selected.
- d. Click **OK** to open the SQL builder.

The title that appears for the SQL builder is the name that you gave to the SELECT statement. For example, if your SELECT statement is named TEST, the title for the SQL builder is TEST.

3. In the SQL builder, add the tables on which to base the SELECT statement for your view. You can add a table in one of two ways:
  - Right-click the middle section of the SQL builder and select **Add Table**. In the Add Table window, select a table to add to the SQL builder and click **OK**.
  - Left-click one of the tables in the schema in which you are creating the view and drag the table to the middle section of the SQL builder.
4. Build the SELECT statement for the view. For help building a SELECT statement, press F1 while in the SQL builder and follow the link to the online help for the SQL builder.
5. **Optional:** Test the SELECT statement. The tables that are referenced by the view must already exist on the data server. To test the SELECT statement, right-click the statement and select **Run SQL**. Look in the Data Output view to see whether the statement ran successfully.
6. In the Data Explorer view, generate and name the view::
  - a. In the **SQL Statements** folder of your physical data model, right-click the SELECT statement and select **Generate > View**. The view appears in the same schema as the tables that it references.
  - b. Click the name of the view once, pause, then click it again to highlight the name. Give the view the name that you want.
7. Select the view and use the Privileges page of the Properties view to grant privileges on the view.
8. **Optional:** Generate the DDL for the view. Right-click the view and select **Generate DDL** to open the Generate DDL wizard. With this wizard, you can generate the SQL DDL to define the view, and you can choose to run the DDL on a data server so that the view is created in the metadata catalog for that data server. You can also edit the generated DDL before you run it.
 

After you run the DDL, the view appears on the data server in the Database Explorer. To see the view, expand the data server and then expand **Schemas > the schema of the view > Views**.

If you want to generate and run the DDL for more than one object at a time, you can right-click a schema and select **Generate DDL**. The Generate DDL wizard will generate the DDL for all of the objects in the schema.
9. **Optional:** If you created the view on the data server, run a test query on the view.
  - a. In the Database Explorer, right-click the view and select **Data > Sample Contents**.
  - b. Look in the Data Output view to see the results of the test query.

### Creating views on existing tables with the SQL editor

To create a view on a table that already exists either only in your project or also on in a metadata catalog, you can use the SQL editor, a text editor that lets you write the SELECT statement for the view. After you create the SELECT statement, you can add the view to a schema in your project.

### Restrictions

If you are creating a view for change capture, these restrictions apply:

- The view cannot reference more than one table. This includes tables in the FROM clause or the WHERE clause as in the case of sub-selects.
- The view cannot reference another view.
- The view must reference all of the columns in the base table.
- The base table must not map to record arrays.

## Procedure

To create a view with the SQL editor:

1. In the Data Project Explorer, expand the physical data model in which you are working. Expand the database in which you are working. Right-click the **SQL Statements** folder and select **New SQL Statement**.
2. In the New SQL Statement window, follow these steps:
  - a. Ensure that **SELECT** is selected in the **Statement template** field.
  - b. Give the statement a descriptive name.
  - c. Ensure that the **SQL editor** radio button is selected.
  - d. Click **OK** to open the SQL editor.
3. Write the **SELECT** statement for the view.
4. **Optional:** Test the **SELECT** statement. The tables that are referenced by the view must already exist on the data server. To test the **SELECT** statement, right-click the statement and select **Run SQL**. Look in the Data Output view to see whether the statement ran successfully.
5. In the Data Explorer view generate and name the view:
  - a. In the **SQL Statements** folder of your physical data model, right-click the **SELECT** statement and select **Generate > View**. The view appears in the same schema as the tables that it references.
  - b. Click the name of the view, pause, then click again to highlight the name. Give the view the name that you want.
6. Select the view and use the Privileges page of the Properties view to grant privileges on the view.
7. **Optional:** Generate the DDL for the view. Right-click the view and select **Generate DDL** to open the Generate DDL wizard. With this wizard, you can generate the SQL DDL to define the view, and you can choose to run the DDL on a data server so that the view is created in the metadata catalog for that data server. You can also edit the generated DDL before you run it.

After you run the DDL, the view appears on the data server in the Database Explorer. To see the view, expand the data server and then expand **Schemas > the schema of the view > Views**.

If you want to generate and run the DDL for more than one object at a time, you can right-click a schema and select **Generate DDL**. The Generate DDL wizard will generate the DDL for all of the objects in the schema.
8. **Optional:** If you created the view on the data server, run a test query on the view.
  - a. In the Database Explorer, right-click the view and select **Data > Sample Contents**.
  - b. Look in the Data Output view to see the results of the test query.

## Creating views on existing tables with the Properties view

To create a view on a table that already exists either only in your project or also on in a metadata catalog, you can create an empty view in your project and then use the Properties view to create the SELECT statement.

### Restrictions

If you are creating a view for change capture, these restrictions apply:

- The view cannot reference more than one table. This includes tables in the FROM clause or the WHERE clause as in the case of sub-selects.
- The view cannot reference another view.
- The view must reference all of the columns in the base table.
- The base table must not map to record arrays.

### Procedure

To create a view with the Properties view:

1. In the Data Project Explorer, expand the physical data model in which you are working. Expand the database in which you are working. Right-click the schema in which you want to create the view and select **Add Classic Object > View**. In the Data Project Explorer, a view is created under the schema.
2. Name the view.
3. Select the view, and on the SQL page of the Properties view, type the SELECT statement.
4. On the Privileges page of the Properties view, grant privileges on the view.
5. **Optional:** Generate the DDL for the view. Right-click the view and select **Generate DDL** to open the Generate DDL wizard. With this wizard, you can generate the SQL DDL to define the view, and you can choose to run the DDL on a data server so that the view is created in the metadata catalog for that data server. You can also edit the generated DDL before you run it.

After you run the DDL, the view appears on the data server in the Database Explorer. To see the view, expand the data server and then expand **Schemas > the schema of the view > Views**.

If you want to generate and run the DDL for more than one object at a time, you can right-click a schema and select **Generate DDL**. The Generate DDL wizard will generate the DDL for all of the objects in the schema.

6. **Optional:** If you created the view on the data server, run a test query on the view.
  - a. In the Database Explorer, right-click the view and select **Data > Sample Contents**.
  - b. Look in the Data Output view to see the results of the test query.

## Viewing and modifying objects for Classic event publishing

You can view and modify the properties of the different objects that you create in Classic Data Architect for Classic event publishing. You can also change the selection of columns in tables and change the path of records for CA-IDMS tables.

### View and modify the properties of tables, columns, and views

When you click on an object in the Data Project Explorer, pages that describe the attributes of the object appear in the Properties view.

### **Change the selection of columns in tables**

For all tables, use the Change Column Selection wizard to replace columns that exist in a table or to append new columns to a table.

To open this wizard, right-click a table and select **Modify Table > Update Columns**.

### **Change the path of records and sets in a CA-IDMS table and change the name of the table**

Open the Modify CA-IDMS Table wizard by right-clicking a CA-IDMS table and selecting **Modify Table > Modify Table**.

### **View the properties of publishing queue maps and publications that you are using for Classic event publishing**

Select one of these objects in the Database Explorer and open the Properties view:

- “Publishing queue map properties” on page 64
- “Publication properties” on page 64

If you want to modify the properties of a publishing queue map or publication, right-click the publishing queue map or publication and select **Update**.

## **Column properties**

Column properties are shown in the Properties view. You can use the Properties view only to view the properties of a column. You cannot modify any of the properties.

The Properties view for a column contains the following information.

### **General page**

Displays the name of the column.

### **Type page**

Displays the SQL data type that is assigned to the column.

### **Classic Column Info**

Displays the SQL data type that is assigned to the column and the position and length of the column.

### **Classic Array Info**

If the column participates in an array, this page displays information about the OCCURS clause that defines the array.

### **Documentation**

Lets you add comments to a column.

## **Database properties**

Use the Properties view to modify the properties of a database in a data design project.

### **General page**

This page displays the name of the database, as well as the type of the database and its version. The type is Classic Integration and the version is V9.

## Privileges page

Lists the privileges that

## Documentation page

## Publication properties

Use the properties view to view the properties of a selected publication.

## General page

Use this page to view the following attributes of a publication.

**Name** The name of the publication.

### Source DBMS

The DBMS that the source table or view maps to. The DBMS can be Adabas, CA-IDMS, IMS, CICS VSAM, or native VSAM.

### Send queue

The WebSphere MQ name for the message queue that transports messages for the publication.

**State** The state of the publication. The possible states are:

- Active
- Inactive
- New

## Options page

Use this page to see the settings for these options.

### Publish before values

Specifies that, when a non-key column is updated, the corresponding message contains the value that is in the column before the change occurs and the value that is in the column after the change occurs.

### Publish changed columns only

Specifies that messages will contain values for only the columns in the source in which the values changed. By default, messages contain the values for all of the columns in a source.

## Topic page

If the publication has a Java™ Messaging Service topic, this page displays that topic.

## Documentation page

Use this page to view the optional description of the publication.

## Publishing queue map properties

Use the Properties view to display the properties of a selected publishing queue map.

## General page

This page displays the following values for the selected publishing queue map.

**Send queue**

The WebSphere MQ name of the send queue.

**State** Indicates that the send queue is active.

**Message format**

Indicates whether the publishing queue map is for publications with messages in XML, XML with a Java Messaging Service (JMS) topic, delimited format, or delimited format with a JMS topic.

**Message content**

The type of message content that the send queue transports. Message queues that are used for XML publications can transport either row-level changes or transactions.

**Maximum message size**

The maximum size for messages that are sent on the send queue. Messages that are larger than this maximum are split into two or more messages.

**Publications page**

This page lists the publications that use the send queue to transport messages.

**Documentation page**

This page displays the optional description of the publishing queue map.

**Table properties**

Table properties are shown in the Properties view. You can use the Properties view to modify the properties of a table.

If the table already exists in a metadata catalog on a data server and you want any changes that you make to the table to be reflected in the metadata catalog, you must follow these steps:

1. Drop the table from the metadata catalog. You can generate the DDL to drop the table by right-clicking the table and selecting **Generate DDL**. In the Generate DDL wizard, select the **DROP statements** check box.
2. Run the generated DDL on the data server.
3. Make your changes to the table.
4. Generate the DDL to create the table. You can generate this DDL by opening the Generate DDL wizard and selecting the **CREATE statements** check box.
5. Run the DDL on the data server.

The Properties view for a table contains the following information:

- General page
- Columns page
- Source information page
- Source columns page
- Path information page
- Source elements page
- Source fields page
- Segments page
- PCB selection page
- Privileges page

- Documentation page

### General page

Property	Description
<b>Name</b>	Type the name of the table.
<b>Schema</b>	Displays the schema in the two-part name for the table.
<b>Source DBMS</b>	Displays the type of DBMS in which the source data is located.
<b>Change capture</b>	(For all data sources except CA-Datcom and DB2 for z/OS)  <b>Changes</b> Select this value if you want to use the table for change capture.  <b>None</b> Select this value if you do not want to use the table for change capture.
<b>XM URL</b>	For a native VSAM table that is being used for change capture, the name of the data space and the name of the Cross Memory (XM) queue to use. The change-capture agent that is capturing changes to the native VSAM table writes change data to this XM queue.  The format of the XM URL is <code>XM1/name_of_data_space/name_of_queue</code>

### Columns page

Lists the columns of the table.

### Source information page

The Source information page contains source information for:

- Adabas, Table 3
- CA-Datcom, Table 4 on page 67
- CA-IDMS, Table 5 on page 67
- CICS VSAM, Table 6 on page 68
- DB2 for z/OS, Table 7 on page 69
- IMS, Table 8 on page 69
- sequential files and native VSAM files, Table 9 on page 70

*Table 3. Source information for Adabas*

Property	Description
<b>File DBID</b>	Optional: Type the identifier of the database in which the Adabas file is stored. This Adabas file is either the file that is identified in the <b>File number</b> field or the file that is referenced by the Predict view. The default value is 0. The identifier must be between 1 and 65535.
<b>View name</b>	Type the name of the Predict view that describes the contents of an Adabas file that has fields that you want to map to columns. Classic Data Architect retrieves the Adabas Field Description Table (FDT) information for the Adabas file that is referenced by the view. If you want Classic Data Architect to access an Adabas file's FDT directly, do not provide a view name. Instead, provide the number of the Adabas file in the <b>File number</b> field.

Table 4. Source information for CA-Datacom

Property	Description
<b>Table name</b>	Type an identifier of 1 to 32 characters for the CA-Datacom table that the Classic table definition references. The name follows CA-Datacom/DB entity naming conventions.
<b>Status/Version</b>	Select or type the status and version of the CA-Datacom table that contains the elements that you want to map to. The status and version can contain explicit values of TEST, PROD, HIST, or a value that begins with a T or H followed by a three digit number.
<b>URT name</b>	Type the name of the User Requirements Table (URT) that is used to access the CA-Datacom table that contains the elements that you want to map to. The name must exist in a data set that is referenced by the servers STEPLIB DD statement or reside in the link pack area. The name follows z/OS load module naming conventions.  A URT must be provided on every request for service sent to CA-Datacom. Every service request is validated against an open User Requirements Table. This technique provides security (by restricting access) and efficient allocation of CA-Datacom resources. When you define your User Requirements Tables, consider the security implications. You must decide whether you want to have one User Requirements Table per CA-Datacom table that you map into the metadata catalog or have only a few User Requirements Tables for all CA-Datacom tables that you map into the metadata catalog. If you define only a few User Requirements Tables, you will have more relaxed security.

Table 5. Source information for CA-IDMS

Property	Description
<b>Subschema name</b>	Displays the name of the subschema that was obtained through a remote connection to the CA-IDMS database or from the local subschema file that you specified.
<b>Schema name</b>	Displays the name of the schema that was obtained through a remote connection to the CA-IDMS database or from the local schema file that you specified.
<b>Schema version</b>	Type a valid 4-digit integer between 0 and 9999 that, together with the schema name, uniquely identifies a CA-IDMS schema. The schema version follows CA-IDMS schema version naming conventions.
<b>Data dictionary</b>	Type an identifier of 1 to 8 characters for the CA-IDMS database for the dictionary that contains the schema and subschema definitions. The data server binds to this dictionary to gather information from the schema and subschema when the data server creates the logical table. The identifier follows CA-IDMS database naming conventions.
<b>Data database</b>	Type an identifier of 1 to 8 characters for the CA-IDMS database that contains the user data that the data server will access at runtime.
<b>Access load module</b>	Type an identifier of 1 to 8 characters for the CA-IDMS batch access module to be used to communicate with the CA-IDMS central version that hosts the user data. The CA-IDMS identifier follows z/OS load module naming conventions.

Table 5. Source information for CA-IDMS (continued)

Property	Description
<b>VSAM information</b>	<b>RRDS</b> Specifies that a record in the subschema has a mode of VSAM and is not a member of a VSAM index set.
	<b>KSDS</b> Specifies that a record in the subschema either has a mode of VSAM and is a member of a VSAM key-sequenced data set, or has a mode of VSAM CALC.
	<b>ESDS</b> Specifies that a record in the subschema either has a mode of VSAM and is a member of a VSAM entry-sequenced data set, or has a mode of VSAM CALC.

Table 6. Source information for CICS VSAM

Property	Description
<b>FCT name</b>	Type the name of the CICS table that contains the information that is used by CICS file control for accessing the VSAM file.
<b>Local APPLID</b>	Type a short identifier of 1 to 8 characters that designates the VTAM® LU 6.2 definition that a CICS region is listening on for connection requests. The CICS LUNAME corresponds to the value of the APPLID parameter that is specified in the system initialization definition (DFHSIT macro) of the target CICS subsystem where the VSAM file is located. The CICS LUNAME follows VTAM naming conventions.
<b>CICS APPLID</b>	Type an identifier of 1 to 8 characters for the VTAM LU 6.2 definition that a CICS region is listening on for connection requests. This identifier corresponds to the value of the APPLID parameter that is specified in the system initialization definition (DFHSIT macro) of the target CICS subsystem where the VSAM file is located. This identifier follows VTAM naming conventions.
<b>Logmode</b>	Type a short identifier of 1 to 8 characters for the name of the VTAM logon mode table that is used to control the session parameters for the conversation that is established between the local LU and the CICS LU. The logon mode table name corresponds to a z/OS load module that is accessible to VTAM. The definition for a Classic supplied logon mode table is supplied in SASCSAMP member CACCMODE.
<b>Transaction ID</b>	Type a short identifier of 1 to 4 characters for the name of the supplied CICS transaction that is used for data access and validation purposes. The CICS transaction ID corresponds to the CICS TRANSACTION definition. Sample CICS transaction, connection, program, and session definitions are supplied in SCACSAMP member CACCDEF. The sample CICS transaction ID is EXV1 and can be modified.
<b>Record exit</b>	Optional: Type a name for a record processing exit that is invoked to decompress sequential records when the file is accessed. The exit name given must exist in a data set that is referenced by the server's STEPLIB DD statement or reside in the link pack area. The exit name follows z/OS load module naming conventions.
<b>Maximum length</b>	Type the maximum length (in bytes) of the buffer that is needed by the record exit to decompress a record.
<b>Network name</b>	Type a short identifier of 1 to 8 characters for the name of the network where the CICS LUNAME resides, which corresponds to the CICS subsystem that is used to access a VSAM file. The NETWORK VTAM macro definition on the local image identifies the remote SNA network where the CICS subsystem resides. The network name follows VTAM naming conventions.

Table 7. Source information for DB2 for z/OS

Property	Description
<b>Creator</b>	Displays the schema of the table.
<b>Table</b>	Displays the name of the table.
<b>Subsystem ID</b>	Displays the ID of the DB2 subsystem in which the table is located.
<b>Plan</b>	Displays the name of the DB2 application plan.  Accessing DB2 data requires binding an application plan for use by the DB2 Call Attach Facility (CAF) service. You can give the plan whatever name you want based on site-installation standards.
<b>Type</b>	Displays the type of object that the new table is mapped to.  When you import a DB2 table into Classic Data Architect, you are creating a table that you can create in a metadata catalog. This field shows that the table is mapped to a DB2 table.

Table 8. Source information for IMS

Property	Description
<b>DBD name</b>	Displays the name of the IMS DBD (database definition) that the table references.
<b>DBD type</b>	Displays the name of the IMS DBD (database definition) that the table references.
<b>Leaf segment</b>	Displays the name of the leaf segment.
<b>Index root</b>	Optional: Type a name for either of these two objects: <ul style="list-style-type: none"> <li>• The physical or logical root segment of the IMS database that is identified by the DBD.</li> <li>• The perceived root segment of the IMS database of a secondary data structure that is created by a secondary index definition that exists in the DBD.</li> </ul> <p>The default index root is the root segment of the physical or logical database that is referenced by the DBD.</p>
<b>IMS subsystem</b>	Optional: Type the 4-character name for the IMS subsystem that is used by the ODBA interface to access the IMS database that is identified by the DBD. The IMS subsystem ID is used only when the server is operating in an RRS two-phase commit environment. The IMS subsystem ID follows IMS naming conventions for subsystem identifiers.  The IMS subsystem ID must correspond to the value that is specified on the IMSID parameter on the IMSCTRL macro in the system definition of the target online IMS subsystem that is used to access or update the IMS data.  The IMS subsystem ID value is ignored for other forms of IMS data access (DRA or BMP/DBB/DLI) and when the table mapping is used for change capture.
<b>PSB name</b>	Optional: Type the name of the PSB that is scheduled to access the IMS database that is identified by the DBD. This name is used if you are using a DRA or ODBA interface to access IMS data. The standard PSB corresponds to a PSB definition that is defined to the IMS online system that is being accessed. The PSB also corresponds to a PDS member under the same name in the active ACB library of the source IMS subsystem. The standard PSB name follows z/OS load module naming conventions.

Table 8. Source information for IMS (continued)

Property	Description
<b>Join PSB name</b>	Optional: Type the name of the PSB that is scheduled to access the IMS database that is identified by the DBD. The name is used if you are using a DRA or ODBA interface to access IMS data. The JOIN PSB corresponds to a PSB definition that is defined to the IMS online system that is being accessed. The PSB also corresponds to a PDS member under the same name in the active ACB library of the target IMS subsystem. The JOIN PSB name follows z/OS load module naming conventions. The JOIN PSB is scheduled when an SQL SELECT statement is executed that contains a JOIN predicate that references multiple IMS tables and this is the first table referenced in the JOIN.

Table 9. Source information for sequential files and native VSAM files

Property	Description
<b>DS</b>	Specifies that the information from which to create the table is contained in a data set.
<b>DD</b>	Specifies that the information from which to create the table is contained in a data set with a DD name.
<b>Name</b>	Type the name of the data set or DD card in which the information for the table is located.
<b>Record exit</b>	Type the name of a record processing exit that is invoked to decompress sequential records when the file is accessed. The exit must exist in a data set that is referenced by the servers STEPLIB DD statement or reside in the link pack area. The exit name follows z/OS load module naming conventions.
<b>Maximum length</b>	Type the maximum length (in bytes) of the buffer that is needed by the record exit to decompress a record.

#### Source columns page

Lists the columns in the table.

#### Path information page (for CA-IDMS only)

Lists the records and sets with elements that are mapped to columns in the table.

#### Source elements page (for CA-IDMS only)

Lists the elements that are mapped to columns in the table.

#### Source fields page (for IMS only)

Lists the fields that are mapped to columns in the table.

#### Segments page (for IMS only)

Lists the segments that contain fields that are mapped to columns in the table.

#### PCB selection page (for IMS only)

Displays the method that Classic federation will use to select PCBs for accessing the table.

#### Privileges page

Lists that privileges that are granted on the table. Click the add button (



) to add privileges. Click the delete button (  ) to remove privileges.

### Documentation page

Lets you add comments to the table.

### View properties

View properties are shown in the Properties view. You can use the Properties view to display and modify the properties of a view.

If the view already exists in a metadata catalog on a data server and you want any changes that you make to the view to be reflected in the metadata catalog, you must follow these steps:

1. Drop the view from the metadata catalog. You can generate the DDL to drop the view by right-clicking the view and selecting **Generate DDL**. In the Generate DDL wizard, select the **DROP statements** check box.
2. Run the generated DDL on the data server.
3. Make your changes to the view.
4. Generate the DDL to create the view. You can generate this DDL by opening the Generate DDL wizard and selecting the **CREATE statements** check box.
5. Run the DDL on the data server.

The Properties view for a view contains the following information:

#### General page

**Name** Displays the name of the view in an editable field.

#### Schema

Displays the schema that contains the view.

#### Change capture

Sets the DATA CAPTURE flag on the view. This field is available only if the view meets all three of these criteria:

- The view references only one table.
- The view references all of the columns in the base table.
- The view references an Adabas, CA-IDMS, CICS VSAM, IMS, or native VSAM table.

#### CHANGES

Specifies to capture changes that are made to the data that the view references.

#### NONE

Specifies not to capture changes that are made to the data that the view references.

#### Columns page

Lists the columns that are referenced in the view.

#### SQL page

Displays the SELECT statement for the view in an editable field. Click the **Validate** button to check the statement for syntax errors.

#### Privileges page

Lists that privileges that are granted on the view. Click the add button (



) to add privileges. Click the delete button (  ) to remove privileges.

#### Documentation page

Lets you add comments to the view.

## Adding or replacing columns in tables based on copybooks

Use the Append Column wizard to add or replace columns in logical tables that are based on copybooks.

### Before you begin

The copybook that contains the columns that you want to use must be listed in the **COBOL Copybooks** folder in your project.

### About this task

You can use columns from the copybook on which the table is based or you can use columns from a different copybook.

### Procedure

To add or replace columns in a table that is based on a copybook:

1. Right-click the table and select **Modify Table > Update Columns**.
2. On page one of the wizard, select the copybook that contains the columns that you want to use.  
If the table is for an IMS data source, select the segment for the columns that you either want to add columns to or that you want to replace with different columns.
3. On page two of the wizard, select the data that you want to map to new columns.
4. On the summary page, verify that the table contains the columns that you want. Click **Finish** to generate the updated model for the table.

## Modifying the selection of records in tables for CA-IDMS databases

Use the Modify CA-IDMS Table wizard to change the selection of records in an existing table before the DDL for the table is run on a data server.

### Before you begin

The subschema and schema reports that you select must be identical to the reports that you used when you created the table. However, the names of the files that contain those reports can be different from the names of the files that you originally used.

You can provide the information on which to base the logical table in one of two ways:

- You can import schema and subschema files that were punched from the CA-IDMS dictionary and transferred via FTP to your workstation. These files must be located in the **CA-IDMS References** folder of your data project.
- You can tell Classic Data Architect to obtain the schema information that is associated with all records, sets, and areas that are listed in the required subschema directly from the CA-IDMS dictionary.

You produce CA-IDMS schema and subschema reports by running the CA-IDMS schema and subschema compilers and capturing the punched output into a z/OS data set. Sample JCL that you can use to punch these reports is in member CACIDPCH of the SAMPLIB data set.

## Procedure

To modify the selection of records in an existing table for a CA-IDMS database:

1. Open the Modify CA-IDMS Table wizard by right-clicking the logical table that you want to modify and selecting **Modify CA-IDMS table**.
2. In the wizard, modify the selection of records:
  - a. On the first page, verify that Classic Data Architect is getting the schema and subschema information from the correct location. If you are using local files that contain the subschema and schema reports and you want to use different files, browse your **CA-IDMS References** folder for the new files. The subschema and schema reports in those files must be identical to the reports that you used when you created the table.
  - b. On the second page, you can modify the information that helps the data server locate the data structures in your database, and you can change the way that the table will be used.
  - c. On the third page, you can rename the table. You can also modify the path of up to ten records and sets from which you want to choose the elements that will constitute the columns in your table.
  - d. Complete a separate wizard page for every record and set that you include in the path to select the elements that you want to map to columns in the table.
  - e. On the summary page, verify that the table contains the columns that you want. Click **Finish** to generate the model for the table.

## Populating metadata catalogs

Classic Data Architect can generate the SQL DDL statements that describe the tables, views, stored procedures, and other objects that you create.

After the DDL statements are generated, you can run them from Classic Data Architect, or you can export them to the z/OS system where your data server is located and run the DDL using the metadata utility.

### Generating DDL

When you finish designing your objects, you generate the DDL that you use to promote those objects to a metadata catalog on a data server.

#### Before you begin

If you choose to run the DDL after it is generated, you must first do the following tasks:

- Open a connection to a data server.
- Create a set of metadata catalogs on the data server.
- Set up connectivity from the data server to your data sources.

#### About this task

When the DDL is generated, you can choose to run it on a data server. You can also choose to open the DDL in an editor.

If you do not choose to run the DDL immediately after it is generated, you can run it later by opening the **SQL Scripts** folder, right-clicking the file with the DDL, and selecting **Run SQL**.

## Procedure

To generate DDL for objects in your project:

1. Open the Generate DDL wizard in either of these two ways:
  - Right-click the schema in which the objects are located and select **Generate DDL**. You can choose which objects in the schema you want to generate DDL for.
  - Right-click the object that you want to generate DDL for and select **Generate DDL**.
2. Follow the pages of the wizard to make these selections:

### Which DDL statements to generate

You can generate ALTER, COMMENT ON, CREATE, DROP, and GRANT statements. You can also choose whether to use fully qualified names and quoted identifiers.

### Which objects to generate DDL for

The available objects depend on which object you right-clicked to open the Generate DDL wizard.

### Where to create the file, which statement terminator to use, whether to run the DDL on a data server, and whether to open the DDL file for editing

The page on which you make these choices displays the DDL that will be generated.

3. After reviewing your settings, click **Finish**.

## Exporting SQL to remote z/OS hosts

The DDL that you generate for your tables, views, and stored procedures is saved to the SQL Scripts folder of your project. You can export the files that contain those scripts to the z/OS host where your data server is located.

To export the files that contain your DDL scripts:

1. Open the Export SQL window by right-clicking the file that you want to export and selecting **Export SQL**.
2. Provide the connectivity information for connecting to the remote z/OS host, and provide the location in which you want to save the DDL scripts.

Then, you can use the metadata utility to run the scripts and populate a metadata catalog.

## Specifying the WebSphere MQ objects to use for change capture

Change capture requires the use of a WebSphere MQ queue manager, a message queue to hold restart information, and (for Classic replication) a message queue for communicating with the Q Apply program. Event publishing requires you to specify which message queues to use for transporting messages for publications.

### Specifying the WebSphere MQ objects to use for change capture

WebSphere MQ is the means of transporting messages in both Classic event publishing and Classic replication. One message queue, called the *administration queue*, is used for administrative tasks for change capture. Another message queue, called the *restart queue*, is used for data recovery tasks.

### About this task

You must specify the name of the WebSphere MQ queue manager that will manage all of the send queues, the name of the administration queue, and the name of the restart queue in your configuration.

## Procedure

To specify the WebSphere MQ objects that you want to use for change capture:

1. Open the Configure WebSphere MQ Values wizard. To open the wizard, in the Database Explorer right-click the data server where you plan to create your publications or subscriptions. Select **Configure WebSphere MQ Values**.
2. In the wizard, specify the names of the following objects:

### Queue manager

The WebSphere MQ program that manages the queues that you will use as the restart queue, administration queue, and send queues.

### Restart queue

The queue that holds restart information that is required if change capture goes into recovery mode. This queue is also called the in-doubt resolution queue.

### Administration queue (for Classic replication only)

A queue that receives control messages from a Q Apply program or ASNCLP.

3. Specify a value in milliseconds for the commit interval. The default is 500 milliseconds.

The commit interval specifies how often, in milliseconds, a publication service commits transactions to WebSphere MQ. At each interval, the publication service issues an MQCMIT call. This call signals the WebSphere MQ queue manager to make messages that were placed on send queues available to the Q Apply program (in Classic replication) or other user applications (in Classic event publishing).

All of the transactions that are grouped within an MQCMIT call are considered to be a WebSphere MQ unit of work, or transaction. Typically, each WebSphere MQ transaction contains several transactions. If a transaction is large, the publication service will not issue an MQCMIT call even if the commit interval is reached. The publication service will commit only after the entire large transaction is put on the send queue.

Finding the best commit interval is a compromise between latency (the delay between the time that transactions are committed at the source and target databases) and CPU overhead that is associated with the commit process:

- To reduce latency, shorten the commit interval.  
Transactions will be pushed through with less delay. Reduced latency is especially important if changes are used to trigger events (in Classic event publishing).
- To reduce CPU overhead, lengthen the commit interval.

A longer commit interval lets you send as many transactions as possible for each WebSphere MQ transaction. If you lengthen the commit interval, you might be limited by the maximum depth (number of messages) for send queues and by the queue manager's maximum uncommitted messages (MAXUMSGS) attribute. If a publication service waits longer between commits, the publication of some transactions might be delayed, which could increase latency.

## Creating and modifying publishing queue maps for Classic event publishing

A publishing queue map identifies a WebSphere MQ queue, called a *send queue* in Classic event publishing, that is used to transport messages to applications. When you create a publishing queue map, you specify the send queue and the values of parameters for the send queue.

### Before you begin

You must have \$VSAM authority or SYSADM authority on the data server where you will store the information about the publication.

### About this task

After you create at least one publishing queue map, you can create one or more publications. The send queue in a publishing queue map can transport messages for multiple publications if the format of the messages matches the format that you specified for the publishing queue map.

For example, if you specify the XML format when you create a publishing queue map, only publications whose messages are in the XML format can use that publishing queue map.

### Procedure

To create a publishing queue map:

1. Open the New Publishing Queue Map wizard. In the Database Explorer, right-click the **Publishing Queue Maps** folder for a data server and select **New Publishing Queue Map**.
2. Specify these attributes of the publishing queue map:
  - The name of the publishing queue map.
  - The WebSphere MQ name of the send queue.
  - Whether to use the publishing queue map for publishing messages in XML, XML with a Java Messaging Service (JMS) topic, delimited format, or delimited format with a JMS topic.
  - Whether to use the publishing queue map to publish one message for each row-level change or one message for each transaction.
  - If each message that is transported by the send queue will contain an entire transaction: The maximum size for any message that is transported on the send queue for the publishing queue map. Any message that is larger than the maximum size is split into two or more smaller messages.  
If you are publishing messages that are in a delimited format and not in XML, the messages that follow the first message have no header information but always follow in order.
  - A description of the publishing queue map.

The publishing queue map appears in the **Publishing Queue Maps** folder and is active.

If you want to modify the properties of the publishing queue map, right-click it and select **Update**.

## Creating and modifying publications

In Classic event publishing, a *publication* associates a source table or view with a WebSphere MQ message queue. Messages that describe changes to source data are put on that message queue. Applications that need to be aware of the changes can read the messages from the message queue.

### Before you begin

- The source table or view for your publication must exist in the metadata catalog of the data server where the correlation service will run.
- The table or view that you want to use as the source must be altered to support change capture.
- You must know the name of the publication queue map with the send queue that you want to use for the publication.
- You must have \$VSAM authority or SYSADM authority on the data server where you will store the information about the publication.

### About this task

The source table or view maps to a data source such as Adabas, CA-IDMS, CICS VSAM, IMS, or native VSAM. When a change occurs to the data that is mapped to the source table or view, Classic event publishing captures the change, converts it to a message, and puts the message on the message queue, called a *send queue*.

Publications are used only in Classic event publishing. They are not used in Classic replication.

### Procedure

To create a publication:

1. In Classic Data Architect, open the New Publication wizard. In the Database Explorer, right-click the **Publications** folder for the data server that you want to create the publication in. Select **New Publication**.
2. In the New Publication wizard, provide the following information:
  - The name of the publication
  - The schema and name of the table or view that you want to use as the source for the publication
  - The WebSphere MQ name of the message queue to use as the send queue to transport messages for the publication. The publication queue map that you select determines whether the messages for the publication will be in XML, XML with a Java Message Service (JMS) topic, or a delimited format.
  - If you are publishing to JMS applications, the topic that you want to send in the messages for the publication
  - Whether to publish the before values in addition to the after values for a data event
  - For every row-level change, whether to publish values in changed columns only or publish the values for all columns in the row
3. Click **Finish** to create the publication.

The publication appears in the **Publications** folder. If the corresponding correlation service is running, right-click the publication and select **Activate** to start publishing messages.

If you want to change the properties of the publication, right-click the publication and select **Update**.

### **Activating, deactivating, and reinitializing publications**

Activate a publication if the publication is new or inactive but you want to start publishing messages for it. Deactivate a publication if you want to stop publishing messages for the publication. Reinitialize a publication if you changed the publication while it was active or changed the publication outside of Classic Data Architect

#### **Before you begin**

If a publication was deactivated because the publication service encountered an error when the publication service tried to write a message to a send queue, you must resolve the error before you try to activate the publication again.

The send queue of the publishing queue map that transports messages for the publication must be active.

#### **About this task**

If you want to check the state for a publication before you activate, deactivate, or reinitialize it, you can select the publication and look on the General page of the Properties view. You can also select the publishing queue map that is used by the publication and look on the Publications page of the Properties view.

#### **Procedure**

To activate, deactivate, or reinitialize one or more publications:

In the Database Explorer, select a single publication, or hold down the CTRL key and select two or more publications. Right-click your selection and in the menu select one of these actions: **Activate**, **Deactivate**, or **Reinitialize**.

---

## **Configuring WebSphere MQ for Classic event publishing**

Different sets of WebSphere MQ objects are required depending on whether you are publishing to destinations that are local to the z/OS LPAR where a publication service, which puts messages on WebSphere MQ message queues, is running or publishing to remote destinations. Also, you must ensure that the publication service has the necessary authorities.

### **WebSphere MQ objects that are required for publishing to local destinations**

You can route messages for publications to local applications or to local instances of WebSphere MQ Java Messaging Service.

#### **Configuration one: WebSphere MQ server and a local receiving application**

In this configuration, the publication service and the application that receives the messages share the same queue manager. The receiving application can get the messages from the same message queue that the publication service uses.

You need the following WebSphere MQ objects:

- One queue manager that is used by both the publication service and the receiving application.
- At least one local queue for the publication service to put messages on. In Classic event publishing, this queue is known as a *send queue*. The number of send queues that you create depends on whether and how you want to group your publications. You group publications by assigning them to publishing queue maps. You need one send queue for each publishing queue map that you create.

For example, messages for publications A and B might go to Application one, and messages for publications C and D might go to Application two. You could have Application one read from Send queue one and Application two read from Send queue two. Messages for publications A and B would go on Send queue one, and messages for publications C and D would go on Send queue two.

Send queues require the following values:

**DEFPSIST(YES)**

Specifies that messages are logged and survive a restart of the queue manager.

**GET(ENABLED)**

Allows the receiving application to get messages from the queue.

**INDXTYPE(GROUPID)**

Allows the queue manager to maintain an index of group identifiers.

**MSGDLVSQ(PRIORITY)**

Specifies that messages on the queue are delivered in first-in, first-out order within priority. It is recommended that you accept this default, even though the messages are not prioritized.

**PUT(ENABLED)**

Allows the publication service to put messages on the queue.

**SHARE**

Allows multiple applications to get messages from the queue.

- One local queue to serve as the restart queue

This local queue requires the following values:

**DEFPSIST(YES)**

Specifies that messages are logged and survive a restart of the queue manager.

**GET(ENABLED)**

Allows the publication service to get messages from the queue.

**INDXTYPE(GROUPID)**

Allows the queue manager to maintain an index of group identifiers.

**PUT(ENABLED)**

Allows the publication service to put messages on the queue.

For more information, see *WebSphere MQ Intercommunication*.

## **Configuration two: WebSphere MQ server and local WebSphere MQ Java Messaging Service**

In this configuration, you can run WebSphere MQ clients on separate machines and connect those clients to the WebSphere MQ Java Messaging Service that is running locally to the WebSphere MQ server.

See *WebSphere MQ Using Java*.

## WebSphere MQ objects that are required for publishing to remote destinations

You can route messages for data event publications to destinations that are remote from IBM WebSphere Classic Event Publisher for z/OS.

### Configuration one: WebSphere MQ server and a remote WebSphere MQ server

- You need the following objects at the WebSphere MQ server that is used by the publication service:
  - One queue manager.
  - At least one remote definition of a queue that is local to the WebSphere MQ server where the receiving application is running. In Classic event publishing, this queue is known as a *send queue*. The number of send queues that you create depends on whether and how you want to group your publications. You group publications by assigning them to publishing queue maps. You need one send queue for each publishing queue map that you create.

For example, messages for publications A and B might go to Application one, and messages for publications C and D might go to Application two. You could have Application one read from Send queue one and Application two read from Send queue two. Messages for publications A and B would go on Send queue one, and messages for publications C and D would go on Send queue two.

Send queues require the following values:

#### **PUT(ENABLED)**

Allows the publication service to put messages on the queue.

#### **DEFPSIST(YES)**

Specifies that messages are logged and survive a restart of the queue manager.

- One transmission queue, which must have the following values:

#### **USAGE(XMITQ)**

Specifies that the queue is used as a transmission queue.

#### **MAXDEPTH**

If you plan to use a single transmission queue for multiple send queues, set the maximum number of messages allowed on the transmission queue to be at least as large as the combined maximum depths for all send queues.

#### **DEFPSIST(YES)**

Specifies that messages are logged and survive a restart of the queue manager.

- One local queue to use as the restart queue

This queue requires the following values:

#### **PUT(ENABLED)**

Allows the publication service to put messages on the queue.

#### **GET(ENABLED)**

Allows the publication service to get messages from the queue.

**DEFPSIST(YES)**

Specifies that messages are logged and survive a restart of the queue manager.

- SYSTEM.CHANNEL.INITQ
- One sender channel.

This channel requires the following values:

**CHLTYPE(SDR)**

Specifies that the channel is a sender channel.

**XMITQ**(*name\_of\_transmission\_queue*)

Specifies the transmission channel to use.

**CONNNAME**(*string*)

Specifies the connection name.

**If you are using LU6.2:** Specify the string as either a logical unit name or a symbolic name:

**Logical unit name**

The logical unit information for the queue manager, comprising the logical unit name, TP name, and optional mode name. You can specify this information in one of 3 ways:

*Table 10. Forms in which you can specify the logical unit name*

Form	Example
luname	IGY12345
luname/TPname	IGY12345/APING
luname/TPname/modename	IGY12345/APING/#INTER

If you use the first form, you must specify the TP name and the mode name with the TPNAME and MODENAME attributes for the channel.

**Symbolic name**

The symbolic destination name for the logical unit information for the queue manager, as defined in the side information data set. The TPNAME and MODENAME attributes for the channel must be blank.

**If you are using TCP:** The connection string can be either a hostname or a network IP address. You can add the port number in parentheses. The connection name can include the IP name of a z/OS dynamic DNS group or a network dispatcher input port.

**TRPTYPE**(*string*)

Use LU62 or TCP, depending on whether you use LU 6.2 or TCP for the connection with the client.

- You need the following objects at the WebSphere MQ server used at the destination of the messages:
  - One queue manager.
  - At least one local queue. Each local queue requires a remote definition at the queue manager of the WebSphere MQ server used by the publication service
 This queue requires the following values:

**GET(ENABLED)**

Allows your receiving application to get messages from the queue.

**DEFPSIST(YES)**

Specifies that messages are logged and survive a restart of the queue manager.

**SHARE**

Allows multiple applications to get messages from the queue.

**MSGDLVSQ(PRIORITY)**

Specifies that messages on the queue are delivered in first-in, first-out order within priority. It is recommended that you accept this default, even though the messages are not prioritized.

- One receiver channel.

This channel requires the following values:

**CHLTYPE(RCVR)**

Specifies that the channel is a receiver channel.

**CONNNAME(string)**

There are many possible values for this attribute. See WebSphere MQ Script (MQSC) Command Reference

**TRPTYPE(string)**

Use LU62 or TCP, depending on whether you use LU 6.2 or TCP for the connection with the server.

For more information, see *WebSphere MQ Intercommunication* and *WebSphere MQ Script (MQSC) Command Reference*.

## Configuration two: WebSphere MQ server and remote WebSphere MQ client

- You need the following objects at the WebSphere MQ server:

- One queue manager.
- At least one local queue for the publication service to put messages on. In Classic event publishing, this queue is known as a *send queue*. The number of send queues that you create depends on whether and how you want to group your publications. You group publications by assigning them to publishing queue maps. You need one send queue for each publishing queue map that you create.

For example, messages for publications A and B might go to Application one, and messages for publications C and D might go to Application two. You could have Application one read from Send queue one and Application two read from Send queue two. Messages for publications A and B would go on Send queue one, and messages for publications C and D would go on Send queue two.

Send queues require the following values:

**PUT(ENABLED)**

Allows the publication service to put messages on the queue.

**DEFPSIST(YES)**

Specifies that messages are logged and survive a restart of the queue manager.

- One local queue to use as the restart queue.

This queue requires the following values:

**PUT(ENABLED)**

Allows the publication service to put messages on the queue.

**GET(ENABLED)**

Allows the publication service to get messages from the queue.

**DEFPSIST(YES)**

Specifies that messages are logged and survive a restart of the queue manager.

- An MQI server-connection channel. You need this channel if you are configuring a test environment or one-to-one connections, or if you are configuring multiple WebSphere MQ clients.

The MQI server-connection channel requires the following values:

**CHLTYPE(SVRCONN)**

Specifies that the channel is a server-connection channel.

**TRPTYPE(string)**

Use LU62 or TCP, depending on whether you use LU 6.2 or TCP for the connection with the client.

- An MQI server-connection channel and an MQI client-connection channel for each client, if you are configuring multiple clients.

The MQI client-connection channel requires these values:

**CHLTYPE(CLNTCOMM)**

Specifies that the MQI channel is a client-connection channel.

**CONNNAME(string)**

There are many possible values for this attribute. See *WebSphere MQ Script (MQSC) Command Reference*

- You need the following objects at the WebSphere MQ client:
  - An MQI client-connection channel, if you are configuring a test environment or one-to-one connection.

This channel requires the following values:

**CHLTYPE(CLNTCOMM)**

Specifies that the MQI channel is a client-connection channel.

**CONNNAME(string)**

There are many possible values for this attribute. See *WebSphere MQ Script (MQSC) Command Reference*

- If you are configuring multiple WebSphere MQ clients, you should create client-connection channels on the WebSphere MQ server. You do not need to create an MQI client-connection channel at each client.

For more information, see *WebSphere MQ Clients*.

### **Configuration three: WebSphere MQ server and a remote WebSphere MQ Java Messaging Service acting as a WebSphere MQ client**

See the guide *WebSphere MQ Using Java*

## **WebSphere MQ authorizations that are required for Classic event publishing**

The users who start distribution services, which start publication services, require various authorities in WebSphere MQ.

User IDs that start a distribution service must have authority to:

- Connect to the queue manager (MQCONN or MQCONNX) on the system where the distribution service runs.
- Perform the following actions on the send queue (whether this queue is a remote definition of a message queue on a target system or a local queue):
  - Open (MQOPEN)
  - Inquire about attributes (MQINQ)
  - Put messages (MQPUT)
  - Commit messages (MQCMIT)
  - Roll back messages (MQBACK)
- Perform the following actions on the restart queue:
  - Open (MQOPEN)
  - Inquire about attributes (MQINQ)
  - Put messages (MQPUT)
  - Get messages (MQGET)

---

## Defining distribution services

You define a distribution service by using a service information entry in the configuration file for the data server where the distribution service will run.

In most cases, you can use the default service information entry for the distribution service. The only situations in which you might need to modify the default values are if you have a very large or very small deployment.

If the distribution service that you are configuring will run in the same data server as a correlation service, the service information entry for the distribution service must come before the service information entry for the correlation service. Member CACCSCF in the SCACCONF data set contains a sample configuration file.

If the distribution service that you are configuring will run in a data server in which no correlation services are configured, you can view the sample configuration file in member CACPSCF in the SCACCONF data set.

To define a distribution service:

1. In the service information entry for the distribution service, specify the following information

**Field two: Service name**

Name the service. You can type a string up to 16 characters long. You can keep the default name DIST1.

**Field ten: this field contains at least three parameters, each separated by a comma**

**TCP/IP connection string**

Define a TCP/IP connection string for communication with correlation services. The distribution service will listen at the specified address and port number for connection requests from correlation services.

The format of the TCP/IP connection string is:

*SKT/ip\_address\_or\_hostname/port\_number*

SKT/0.0.0.0/5001

The 0.0.0.0 IP address should resolve to the host name where the data server is running. Port number 5001 is the default port for communications between correlation services and distribution services. Note that port 5002 is assigned by default for use by Classic Data Architect when connecting to a query processor.

Unless IP address resolution fails or your network administrator assigns a different port number for communications between correlation services and distribution services, you should not have to change the default values.

**Note:** When the distribution service and the correlation services that communicate with it run on the same z/OS LPAR, you can use a Cross Memory (XM) queue rather than TCP/IP. Using XM rather than TCP/IP might improve performance when large units of work are transferred between the two different types of services. Replace the TCP/IP connection string with a string that describes the Cross Memory queue, as described for the next parameter. You must assign a unique data space to the queue.

### Cross-memory (XM) queues for communication between the distribution service and publication services

There are two queues. You must assign a different data space name to each.

The format of the strings is:

```
XM1/<name_of_data_space>/<name_of_queue>/\  
<size_of_data_space_in_MB>
```

**INT=*n*** This parameter specifies the importance of processing changes that are sent by correlation services versus sending changes to the publication service. The greater the value, the greater the importance of processing changes from correlation services.

2. Make sure that the LD TEMP SPACE configuration parameter in the configuration file is uncommented. You can leave the default value.

```
SERVICE INFO ENTRY = CACDSRD DIST1 2 1 1 1 4 5M 5M \  
SKT/0.0.0.0/5001,XM1/PSD1/PSQ1/128,XM1/PSC1/PSC1/4
```

Figure 3. The sample service information entry for the distribution service

---

## Defining correlation services

You define a correlation service by using a service information entry in the configuration file for the data server in which the correlation service will run. You also need to set the size of the data sets used by correlation services for temporary storage.

### Restrictions

- Correlation services must run on the z/OS LPAR where your source DBMS is located.

- Only one correlation service can run within a single data server. To configure additional data servers for additional correlation services, for each data server follow these steps:
  1. Customize and run the CACCTRL JCL in the SCACSAMP data set.
  2. Create or upgrade a metadata catalog. See “The catalog initialization and maintenance utility (CACCATUT)” on page 262.
  3. Follow the steps that are outlined for the data server on z/OS LPAR 2 in the procedure that is specific to your data source. See the section Related Tasks at the bottom of this page.
- If multiple correlation services run on a single LPAR, all but one require unique names. The exception can use the default value of (NONAME).

### About this task

Correlation services convert the change data that they receive from change-capture agents into a relational format. They correlate the data that is in the format of the source database or VSAM files with the tables in the metadata catalog that are mapped to your data sources.

If you are capturing from more than one Adabas, CA-IDMS, or IMS database, you must configure a separate change-capture agent for each database. Each change-capture agent must send change data to a separate correlation service.

If you have multiple NVA instances capturing data from VSAM files, you might want to run more than one correlation service. Because a change-capture agent can communicate with only one correlation service, you can have separate groups of change-capture agents communicate with different correlation services. The primary advantage to such parallelism is to speed up the processing of your change data.

Member CACCSCF in the SCACCONF data set contains a sample configuration file that contains the service information entry for a correlation service.

### Procedure

To define a correlation service:

1. In the configuration file, specify the following information in the service information entry for the correlation service:

#### Field two

Define the protocol, data space, queue name, and data space queue size for receiving raw data changes from change-capture and recovery agents by using Cross Memory services. The data space or queue name must be unique on the z/OS LPAR where the correlation service is running.

The protocol name is XM1. The subtoken XQM/CSQ1 identifies the Cross Memory data space and queue name and can be modified to suit any particular site standards if applicable. Remember that the queue names and data space are valid for an entire LPAR, so avoid naming conflicts with Cross Memory objects that are used by other services running in the same LPAR.

The final subtoken 16 identifies the size (in megabytes) of the Cross Memory queue that uses the data space. This value can range from 1 to 2048 and defaults to 8 if not specified. The size of the queue depends

on the number of expected active agents and the burst volume of changes from each agent. A value of 512 is recommended to help ensure queue space during peak periods.

**Restriction:** You must define a unique data space or queue name for each correlation service and distribution service. If you use the same combination of data space and queue name for more than one correlation service, then change-capture agents will send captured changes to the least-busy correlation service, which might not be the correct correlation service. If the change data is not sent to the correct correlation service, it might be published out of order.

If you configure more than one correlation service in a single data space, the first correlation service that is started sets the size of the data space.

#### Field ten

The parameters in this field are separated by commas.

##### **TCP/IP connection string**

Define a TCP/IP connection string for communication with a distribution service. The string must match the TCP/IP connection string in the service information entry for the distribution service that you want the correlation service to communicate with.

The default values allow the correlation service to communicate with a distribution service that is running on the same LPAR and using port number 5001.

- If the distribution service is configured on a different LPAR, change 0.0.0.0 to specify the IP address or host name of the that LPAR.
- If your network administrator assigns a different port number or name for the distribution service, change the default value of 5001 to that number or name.

The format of the TCP/IP connection string is:

*SKT/ip\_address\_or\_hostname/port\_number*

##### **COLDSTART or WARMSTART**

Specifies whether to cold start or warm start the correlation service.

##### **COLDSTART**

A cold start discards all recovery information and places in active mode all change-capture agents that are known to the correlation service.

If you set the correlation service to perform a cold start, make sure that you change the service information entry to specify WARMSTART after you cold start the correlation service. Otherwise, you might inadvertently cold start the correlation service in the future.

##### **WARMSTART**

A warm start retains the state of change-capture agents that were known to the correlation service at the time that the correlation service was last shut down.

WARMSTART is the default action.

**CSARLSE=*n***

The number of seconds to wait before attempting to release CSA storage when the correlation service shuts down. A value of 0 leaves CSA allocated for reuse when the correlation service is restarted. The default value of CSARLSE is 0, which prevents CSA from being released. If CSARLSE is not 0, the correlation service will release CSA only if no other correlation services are still active in the system.

**INT=*n*** The number of changes to send to the distribution service for a committed unit of recovery (UOR) before checking for incoming change data from change-capture agents or recovery agents.

The more changes in a committed UOR, the longer it takes before the correlation service is done processing the UOR content and can retrieve any more work that a change-capture agent or recovery agent placed on an XM queue.

The INT keyword prevents large transactions from blocking the incoming XM queue while the correlation service is sending change messages to the distribution service. Specifying a non-zero INT value identifies how many changes in a committed UOR are sent to the distribution service before the correlation service interrupts commit processing and checks the XM queue for outstanding XM data grams to be processed. A value of 0 will not interrupt processing of committed messages to look for incoming data messages. 1 is the default value.

Appropriate interrupt values vary. If you have applications that generate a large number of updates, experiment with this parameter to determine an optimum value.

**LOGCONVERRS=Y|N**

Allows you to specify the report logging that the correlation service provides when the correlation service encounters an error while performing either of the following numeric conversions:

- Converting a decimal in zoned-decimal format to packed-decimal format.
- Converting a binary value to packed decimal.

When the correlation service encounters an error during these conversions, it converts invalid data to a decimal value with negative nines (-999's). The actual value returned for the column is based on the precision and scale of the column definition. For example, a column with a data type of DECIMAL(5,2) is converted to -999.99. The correlation service continues processing.

You can specify either of these two values to control the information that the correlation service reports when a conversion results in a value of negative nines:

**Y** Causes the correlation service to generate a set of 0x002f0001 messages to the log. These messages show the original source data the caused the conversion error. This is the default behavior.

**N** Causes the correlation service not to generate any log messages when a conversion occurs.

**NOSTAE**

Disables detection of abends in the correlation service. Do not specify this value unless requested by IBM technical support for diagnostics.

**NAME=*name***

Names the correlation service. If you do not specify this value, the correlation service starts unnamed. You can run only one unnamed correlation service on a z/OS LPAR. IBM recommends that you provide a name for your correlation service.

2. **Optional:** Use the LD TEMP SPACE configuration parameter to set the minimum and maximum sizes for the temporary data sets in which correlation services store messages and pending commits. The default LD TEMP SPACE definition should be adequate under all conditions. The default uses Hiperspace™ for performance and allows the Hiperspace to grow to the largest supported value in increments of 16 MB.

The default value is:

```
HIPERSPACE, INIT=16M, MAX=2048M, EXTEND=16M
```

See “LD TEMP SPACE configuration parameter” on page 230.

## Record selection exit for multiple record layouts

The correlation service supports an option record selection exit that can be developed in assembler language. The purpose of this exit is to accept or reject change the contents of a record to fixup invalid data.

The record selection exit was used in the past purely as a filtering mechanism and was designed to handle records with multiple record layouts. Each layout required a separate table mapping and the exit was invoked each time a database object changed. On each invocation, the exit was passed the name of a table that referenced the database object. The record selection exit would inspect the records contents and then either accept or reject the change depending upon whether the contents of the record matched the mapping (accept) or not (reject).

This kind of functionality was replaced with the ability to define and alter for data capture a view that is defined on the table mapping. The view allows the correlation service to evaluate the WHERE clause defined on the view definition to perform the record selection logic.

For some data sources in the previous release, the record selection exit was also expanded to provide both the before and after images on updates which allowed the record selection exit to perform more sophisticated selection logic based on the values contained in both the before and after images and also afforded the exit the ability to change the contents of the record i/o areas if it so desired. This capability was extended to all data sources in the current release.

Where possible the use of views to perform filtering operations is the preferred approach.

**Note:** Creating views is the recommended method addressing multiple record layouts that are defined with REDEFINES clauses. The record selection exit

is still available, however, and you can use it in cases where data manipulation or transformation is required.

This exit is implemented by assembling and linking the module CACRCSEL into the Classic load library. Sample source for the exit is in the CACRCSEL member in the Classic SCACSAMP library. This assembler module contains a description of the parameters that are passed and logic to accept or reject record data that is passed as matching a mapped table name.

The parameters that are passed to the module CACRCSEL are:

- An uninitialized 256-byte work area. This can be used as a save area so the module can call subroutines. Do not use this area for persistent storage across invocations of the module.
- The address of the record data that is to be matched to a table name. If the data was compressed in the database, it is decompressed prior to calling CACRCSEL. Do not change the record data with CACRCSEL because it contains only the before image for record updates and therefore any necessary changes will not be applied to the after image.
- A fullword value that contains the binary length of the record data.
- An 8-byte, space-padded database type name of the mapped table. This parameter contains one of the following values:
  - '\$IMS '--IMS database mapping
  - '\$VSAM '--VSAM file system mapping
  - '\$ADABAS '-- Software AG Adabas file mapping
- The 8-byte, space-padded table owner ID as specified in the metadata grammar that mapped the table.
- The 18-byte, space-padded table name as specified in the metadata grammar that mapped the table.
- A fullword binary field that contains the change type associated with the passed data. The values in this field are:
  - 1--The change is an update and the passed data is the before image of the update.
  - 2--The change is an insert and the passed data is the data inserted.
  - The change is a delete and the passed data is the record deleted.

The following IMS information is available:

```
IMSSSID DS CL8 IMS SUBSYSTEM ID
IMSJOB DS CL8 CHANGE-CAPTURE AGENT JOB NAME
IMSPSB DS CL8 PSB NAME
IMSTRAN DS CL8 TRANSACTION NAME
IMSUSER DS CL8 USER ID
IMSDBD DS CL8 DBD NAME
IMSSEGM DS CL8 LEAF SEGMENT NAME
IMSLOGN DS CL16 LOG RECORD SEQUENCE NUMBER (in hexadecimal format)
IMSSTCK DS CL16 SYSTEM CLOCK VALUE (in hexadecimal format)
```

Both the log record sequence number and the system clock value are in hexadecimal format.

The following Software AG Adabas information is available:

```

ADADBID DS H DATABASE ID
ADAFNR DS H FILE NUMBER
ADAISN DS F INTERNAL SEQUENCE NUMBER OF RECORD
ADASTCK DS CL8 SYSTEM CLOCK VALUE ASSOCIATED WITH CHANGE
ADAUID DS CL28 ADABAS GLOBAL UNIQUE IDENTIFIER
ADATRAN DS F TRANSACTION SEQUENCE NUMBER

```

If the record selection exit module exists in the Classic load library, it is called each time the correlation service receives a change from a change-capture agent. This call is made once for each table name that matches the database or file record changed. The record selection exit must determine whether the change passed is valid for the table name from the metadata catalog. Accepting or rejecting a table name is indicated by the return code issued by the selection exit. A return code of 0 indicates acceptance of the record data for the table name passed and results in the conversion of the record data to an SQLDA to forward to the publication service.

In cases such as IMS, where the table mapping can include parent segments, the record data parameter contains information for all records in the defined path. When determining offsets to specific fields in the data record, you must account for the concatenated length of all parent segments in the beginning of the data area. To determine offsets to specific data items, you can add debugging logic in the sample record exit to dump passed records to the operator console.

In some instances, you might want to filter certain table names based on the action value passed. For example, an application might be interested in capturing changes to a database only when certain records are inserted. In this case, the selection exit can be used to reject all UPDATE and DELETE messages for a particular table name.

You can modify the following sample JCL and use it to assemble and link the record selection exit:

```

//jobname JOB (ACCTINFO), 'CACRCSEL', CLASS=A,
// MSGCLASS=X, NOTIFY=&SYSUID
//* ASSEMBLE CACRCSEL
//ASSEMBLE EXEC PGM=ASMA90,
// PARM='LIST,NODECK,RENT'
//SYSLIB DD DSN=SYS1.MACLIB, DISP=SHR
//SYSLIN DD DISP=(NEW,PASS), DSN=&&OBJMOD,
// UNIT=SYSDA, SPACE=(TRK,(10,1)),
// DCB=(LRECL=80, BLKSIZE=3200, RECFM=FB)
//SYSUT1 DD DSN=&&SYSUT1, UNIT=VIO, SPACE=(1700,(2000),,ROUND)
//SYSPRINT DD SYSOUT=*
//SYSIN DD DISP=SHR, DSN=CAC.SCACSAMP(CACRCSEL)
//* LINK CACRCSEL
//LINK EXEC PGM=IEWL, PARM='LIST,RENT,REUS', COND=(4,LT)
//SYSLIN DD DISP=(OLD,DELETE), DSN=&&OBJMOD
// DD *
ENTRY CACRCSEL
NAME CACRCSEL (R)
//SYSLMOD DD DISP=SHR, DSN=CAC.SCACLOAD(CACRCSEL)
//SYSUT1 DD UNIT=SYSDA, SPACE=(1024,(120,120),,ROUND),
// DCB=BUFNO=1
//SYSPRINT DD SYSOUT=*

```

---

## Configuring change capture from Adabas databases

The steps for configuring change capture from Adabas databases are a subset of the steps for configuring either Classic event publishing or Classic replication from Adabas databases.

## Change-capture agents for Adabas databases

The change-capture agent for Adabas databases is implemented as an Adabas user exit that is named TRNEX.

Adabas calls the TRNEX user exit as part of the process of modifying a database record. The change-capture agent filters the changes to Adabas files and forwards each change to a correlation service if the following two conditions are met:

- The change occurs to a field that is mapped to a column in a logical table in the metadata catalog.
- The DATA CAPTURE flag for the logical table is set to CHANGES.

These change records that are sent to the correlation service include commit and rollback information, and before and after images of database fields. The correlation service must be active on the LPAR that is running the Adabas nucleus because the change-capture agent communicates with the correlation service by using Cross Memory services.

## Configuring access to Adabas databases

Before you can populate metadata catalogs with the tables that you create with Classic Data Architect, you must configure the data server so that it can access your databases. This access allows you to use Classic Data Architect to validate the tables and views that you map to your Adabas databases. The same steps for configuring this access make it possible for correlation services to access Adabas data directly the first time that they receive change data.

### Before you begin

- Activate dual or multiple protection logging in Adabas.
- Specify the change-capture agent CACEC1A in the transaction data user exit parameter (TRNEX).
- Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB. If you are running correlation services and distribution services in separate data server, this JCL is for the data server where one or more correlation services will run.
- Make sure that the correlation service's STEPLIB is APF-authorized. If the Adabas load libraries cannot be APF-authorized, you must create new libraries and APF-authorize them for use by IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS.

### About this task

A correlation service usually does not access Adabas data directly because correlation services are sent data from change-capture agents. The only time a correlation service access Adabas data directly is when the first change-capture data is detected for a particular file and DBID. When this change occurs, the correlation service issues an Adabas direct call to read the characteristics of all fields in the corresponding file and DBID and to decompress the record that was changed. The result of this direct call is stored in buffers for subsequent use.

### Procedure

To configure access to an Adabas database:

Make the following JCL changes to the data server JCL. You can find a sample in the CACCS member of the sample library.

1. Add the Adabas loadlib to the STEPLIB concatenation. You must also APF-authorize the Adabas loadlib if it is not already authorized.
2. Provide an appropriate ADARUN control card by using the DDCARD DD statement. You can find a sample statement in the member CACADADD in the sample library.

The ADARUN control card must specify:

```
ADARUN SVC=nnn,DB=dbid,MODE=MULTI
```

- *nnn* is the SVC number for the Adabas nucleus to access.
- *dbid* is the default DBID.
- MODE=MULTI specifies that the correlation service will communicate with a multi-user mode Adabas nucleus.

## Running more than one correlation service when capturing from Adabas databases

You can run more than one correlation service on one z/OS LPAR. To make sure that your change-capture agents communicate with the right correlation services, you must name the correlation services.

### Before you begin

- Activate dual or multiple protection logging in Adabas.
- Specify the change-capture agent CACEC1A in the transaction data user exit parameter (TRNEX).
- Make sure that the correlation service's STEPLIB is APF-authorized. If the Adabas load libraries cannot be APF-authorized, you must create new libraries and APF-authorize them for use by IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS.

### About this task

You would need to have more than one correlation service if you had both a test and production database on the same system. If you were to use only one correlation service, and either encountered an error or you had to stop the correlation service for the test system, then not only would the test system go into recovery mode, but the production system would, too.

### Procedure

To run more than one correlation service on one z/OS LPAR, follow these steps for each correlation service:

1. Create a service information entry for the correlation service in the configuration file of the data server where the correlation service will run. See "Defining correlation services" on page 85.
2. Add a NAME parameter and a value for that parameter after field ten in the service information entry for the correlation service. The value is the name that you want to assign to your correlation service. The name must be eight characters or fewer. Both the active and recovery change-capture agents need to reference your correlation service by this name.
3. Update the CACE1OPT (options table) source module to include the name of the correlation service in the constant labeled SRVRNAME.
4. Assemble the updated CACE1OPT source module. The CACE1OPT module is used by several load modules. After making changes specific to your environment, submit the following JCL to re-link the affected load modules:

```

//valid job card here
//*****
/* JOB CONTROL STATEMENTS TO ASSEMBLE A MODIFIED OPTIONS TABLE *
/* (CACE1OPT) *
//*****
//ASSEMBLE EXEC PGM=ASMA90,PARM='LIST,NODECK,RENT'
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//SYSIN DD DISP=SHR,DSN=your.source.input.library(CACE1OPT)
//SYSLIN DD DISP=SHR,DSN=your.assembly.output.library(CACE1OPT)
//SYSUT1 DD DSN=&SYSUT1,UNIT=VIO,SPACE=(1700,(2000),,ROUND)
//SYSPRINT DD SYSOUT=*
//*****
/* JOB CONTROL STATEMENTS TO LINK A MODIFIED OPTIONS TABLE *
/* (CACE1OPT) WITH ADABAS CHANGE_CAPTURE AGENTS. *
//*****
//LINK EXEC PGM=IEWL,
// PARM='MAP,LIST,AMODE=31',COND=(4,LT)
//LOAD DD DISP=SHR,DSN=your.testing.load.library
//OBJ DD DISP=SHR,DSN=your.assembly.output.library
//SYSLMOD DD DISP=SHR,DSN= your.testing.load.library
//SYSUT1 DD UNIT=3390,SPACE=(1024,(120,120),,ROUND),DCB=BUFNO=1
//SYSPRINT DD SYSOUT=*
//SYSLIN DD *
INCLUDE OBJ(CACE1OPT)
INCLUDE LOAD(CACECA1A)
ENTRY MSLAAPRE
NAME CACECA1A(R)
INCLUDE OBJ(CACE1OPT)
INCLUDE LOAD(CACEC1AR)
SETCODE AC(1)
ENTRY MAIN
NAME CACEC1AR(R)
/*
//

```

5. Add the parameter `SERVER=svrname` to the JCL execution parameter of the recovery agent for Adabas. `svrname` is the name that you assigned to the correlation service.
6. If you did not yet start the correlation service or the correlation service is not running, stop and restart the Adabas nucleus.
7. If the correlation service is running, follow these steps:
  - a. Stop the Adabas nucleus.
  - b. Stop and restart the correlation service.
  - c. After the correlation service finishes initializing, restart the Adabas nucleus.

## Installing change-capture agents for Adabas databases

To begin capturing changes that are made to an Adabas database, you need to install the change-capture agent.

### Before you begin

- Make sure that the corresponding correlation service is running, the distribution service that the correlation service communicates with is running, and your publications are defined.

### Procedure

To start the process of capturing changes:

1. Install the change-capture agent, which is the exit `CACECA1A` in the IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS installation library.

Use either of these two methods:

- Copy the CACECA1A exit into another load library that is in the Adabas nucleus STEPLIB.
  - Add a DD statement in the STEPLIB to define the IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS installation library.
2. Allocate the appropriate protection log data sets and include them in the Adabas nucleus JCL.
  3. Include the following ADARUN parameter for the nucleus:  
ADARUN TRNEX=CACECA1A
  4. Create a recovery data set. If a change-capture agent is started without a correlation service running, the agent records a restart point in a data set. To enable the change-capture agent to record the restart point, add a recovery data set with the DD name CACRCV to the ADABAS nucleus JCL. You must define the data set as a sequential file with fixed-blocked format, record length of 80 bytes, and block size of 80 bytes. The file is a single-record file, so you need to allocate only one track.
  5. Restart the Adabas nucleus so that it recognizes the new ADARUN parameters.
  6. Verify the installation by starting the Adabas nucleus and looking for this operator message in the Adabas nucleus JCL messages:  
CACH001I CHANGE CAPTURE AGENT 'nnnn' INSTALLED FOR SERVER '(no name)'

---

## Configuring change capture from CA-IDMS databases

The steps for configuring change capture from CA-IDMS databases are a subset of the steps for configuring either Classic event publishing or Classic replication from CA-IDMS databases.

### Change-capture agents for CA-IDMS databases

The change-capture agent for CA-IDMS databases is implemented as a CA-IDMS database exit that is named IDMSJNL2.

There is one IDMSJNL2 exit point for each central version. A central version calls the IDMSJNL2 exit before and after journal records are written to the CA-IDMS journal files and when journal files are closed. These records include information about the start and end of transactions and about before and after images of database record and set updates.

The IDMSJNL2 exit extracts transaction and record image information from journal records after the records are successfully written to the database journal file. The exit then sends this extracted information to the correlation service by using Cross Memory services.

### Configuring connectivity to CA-IDMS database systems

You must configure CA-IDMS and modify the JCL for the data server where the correlation service will run.

#### Before you begin

Copy the data server JCL, which is in the CACCS member of the SCACSAMP data set, to your PROCLIB. If you are running correlation services and distribution services in separate data server, this JCL is for the data server where one or more correlation services will run.

## About this task

There are two reasons for configuring connectivity to CA-IDMS:

- When you create a CA-IDMS table in Classic Data Architect, you can choose to have Classic Data Architect discover the records that you want to capture changes from. Discovery requires a connection to your CA-IDMS system.
- Correlation services must access CA-IDMS data to retrieve path (record owner) records associated with changed records if the path contains more than one record.

A correlation service communicates with CA-IDMS by using the CA-IDMS external run-unit interface. The correlation service loads the CA-IDMS access modules that you specify when you are creating CA-IDMS tables in Classic Data Architect. If you do not specify the access module for a CA-IDMS table, the correlation service loads the CA-IDMS interface module IDMS. This module can access CA-IDMS in either central version or local modes.

All sample JCL and examples that are provided are configured to access CA-IDMS in central version mode. If you have a specific need for local mode access, see the CA-IDMS documentation for the required JCL changes.

## Procedure

To configure a data server and correlation services to communicate with one or more CA-IDMS central versions:

1. Verify that the MAXERUS setting for the central versions is sufficient to support access by the data server to CA-IDMS.

In central version mode, the CA-IDMS data access component in the data server connects to the CA-IDMS system as an external run-unit. An external run-unit is started during the processing of the CREATE TABLE, CREATE INDEX, and ALTER TABLE statements, connecting to a CA-IDMS dictionary to extract from and validate information in the appropriate subschema and schema.

In central version mode, the CA-IDMS data access component in a correlation service connects to the CA-IDMS central version as one external run-unit. If you run multiple correlation services that need to access the same central version, each correlation service counts as one external run-unit.

The MAXERUS setting for a central version must be high enough to accommodate the run-units for the data server and the run-units for the correlation services that need to access that central version.

2. Set up APF authorization of the CA-IDMS LOADLIB.

The correlation service's STEPLIB must be APF-authorized. The CA-IDMS.LOADLIB is not usually APF-authorized, and some utility programs in that library will fail if they are run from an APF-authorized STEPLIB concatenation.

To run a correlation service, you must create a separate authorized copy of the CA-IDMS.LOADLIB.

If you are capturing changes to CA-IDMS records that are using Presspack compression, you also must authorize the library that contains DCTABLE modules and include it in the data server's STEPLIB concatenation.

3. Pass the correct user context to CA-IDMS run units.

By default, all CA-IDMS batch run-units connect to CA-IDMS with a blank user context. This is interpreted in CA-IDMS as PUBLIC access to the data.

To establish the correct user name for each run-unit created under the data server:

- a. Re-link the CA-IDMS module IDMSSTRT, distributed with CA-IDMS, with a USRIDXIT exit module supplied with IBM WebSphere Classic Data Event Publisher for z/OS and IBM WebSphere Classic Replication Server for z/OS. This IDMSSTRT module is for use only by these two products and should be placed in a library other than the CA-IDMS LOADLIB.
- b. Locate the new IDMSSTRT in the separate authorization LOADLIB that was created for CA-IDMS.

To create a new IDMSSTRT module:

- a. Use the CACIDUXT sample JCL in the SAMPLIB samples library to assemble the CACIDUXS exit source in the SAMPLIB
- b. Link edit it into a new IDMSSTRT module
- c. Link the new module into a library that is used only by IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS.
- d. Include this library in the STEPLIB concatenation of the start up JCL for the data server prior to the CA-IDMS LOADLIB.

Creating an IDMSSTRT module ensures only that all run-units established with a CA-IDMS central version have the correct user name associated with them. It does not ensure the security of the CA-IDMS data itself. For data to be secure, the CA-IDMS central version must have security enforcement active. See your CA-IDMS System Administrator to validate that CA-IDMS data security is enforced.

4. **Optional:** Set up access to multiple CA-IDMS central versions.
5. Update the data server JCL.
  - a. Add the authorized version of the CA-IDMS LOADLIB to the STEPLIB concatenation.
  - b. Include a SYSCTL DD statement for all central versions that the data server will connect to.
  - c. Ensure that the STEPLIB includes any access modules that are needed to access the SYSCTL files.
  - d. Include a SYSIDMS DD statement that includes the following parameters to ensure that the data server is notified correctly when the CA-IDMS CV is not available:

```
CVRETRY=OFF
REREAD_SYSCTL=ON
DMCL=DMCL name that contains the SYSTEM segment
```
  - e. Include the data set for the DDLDCMSG area in the JCL for the batch job.

## Configuring data servers to access multiple CA-IDMS central versions

You can configure access for data servers to multiple CA-IDMS central versions.

### About this task

Each CA-IDMS central version has a unique SYSCTL file associated with it. When added to the data server JCL, the default SYSCTL DD name defines the default CA-IDMS central version to communicate with. You can allocate multiple SYSCTL data sets with unique DD names to a single data server and for each logical table select the appropriate data set using a custom built CA-IDMS ACCESS LOADMOD that references the appropriate SYSCTL DD name.

## Procedure

To build an access module that references an alternate SYSCTL DD name called SYSCTL1:

1. Code an assembler IDMSOPTI module containing the following assembler macro statement:  

```
IDMSOPTI CENTRAL=YES,SYSCTL=SYSCTL1  
END
```
2. Assemble the IDMSOPTI module supplied in the SCACSAMP library member CACIDACM.
3. Re-link the supplied batch access module named IDMS, and include the IDMSOPTI module that you assembled in the previous step. Here are sample link-edit control statements to build the new access module:
  - INCLUDE IDMSLOAD (IDMS)
  - ENTRY IDMS
  - NAME *new-module-name*(R)

Be sure to create a new name for the CA-IDMS module because the default module must be left as-is for other CA-IDMS batch applications. Also, be sure to link the new module into a library that is accessible to the startup JCL for the data server.

You can use IDMSOPTI modules to manage default databases and other CA-IDMS-specific parameters. For more information, see the *CA/CA-IDMS System Operations* manual.

## Running more than one correlation service when capturing from multiple CA-IDMS central versions on the same z/OS LPAR

If you have multiple CA-IDMS central versions running on the same LPAR, the change-capture agent for each central version must send change data to a separate correlation service rather than to the same correlation service.

### About this task

Each correlation service must have a unique name. You specify the names using the NAME parameter in the service information entries for the correlation services.

## Procedure

To use named correlation services:

1. In the service information entry of each correlation service, add a NAME parameter (such as NAME=CA-IDMS\_ central\_ version).
2. For each change-capture agent that you are running, perform these steps:
  - a. Update member CACE1OPT by including the name of the correlation service as the value of the SERVER parameter. Member CACE1OPT is located in the SCACSAMP library.
  - b. Reassemble the CACE1OPT module and re-link it with the IDMSDBIO module. You can use member CACIDLDB in the SCACSAMP library to perform this step.
3. When you want to run the recovery agent, change the JCL to run the recovery agent by adding a SERVER= parameter to the execution parameter (such as SERVER=CA-IDMS\_ central\_ version).

## Local mode versus central version mode for CA-IDMS databases

It is not possible to merge journal files from two different sources. For example, if you run a batch program in local mode with journaling, and you run another program in the CA-IDMS central version, you cannot merge the journal files.

In such situations, if the local mode files are for databases other than those accessed from the central version, you can configure a change-capture agent and correlation service for those databases. This change-capture agent would be separate from the capture agent associated with the central version.

Run the local mode clients with CA-IDMS journaling active. Journaling in local mode requires that you run local mode clients using a CA-IDMS DMCL that defines disk or tape journals. For more information about local mode journaling, see the *CA-IDMS Database Administration* guide.

When local mode clients are updating databases that are normally under the control of a central version, there are multiple sets of journals for each database. Local mode clients can update these databases only if the central version is down or if the databases and areas have been varied offline from the central version.

If change capture is required on these databases for updates that occur during both central version-mode and local-mode processing, the following procedures are required.

- If a central version is shutdown to enable local mode processing, before you start the batch job make sure that the change-capture agent that is associated with the central version is not in recovery mode.
- If the areas are being varied offline from the central version to enable the local mode processing, ensure that a journal swap is done after the areas are successfully offline. In addition, before you start the batch job make sure that the change-capture agent that is associated with the central version is not in recovery mode.

## Ensuring that the minimum number of journals is available for recovery

Although recovery agents can recover change data from archived journals, recovering change data is faster when the recovery agent can read unarchived journals. CA-IDMS typically archives journals automatically when they become full. You might want to allow a certain number of active journals to remain unarchived so that they can be used by the recovery agent.

### Procedure

To keep enough journals online for the recovery agent, follow either of these steps:

- Run the recovery agent as part of your archiving procedure.

Use the COUNT parameter for recovery agents to count the number of journals and to skip automatic archiving if the minimum number of journals is not available. Include the following EXEC statement in the JCL for your CA-IDMS journal offloading process:

```
//CACEC1DR EXEC PGM=CACEC1DR,PARM='COUNT'
```

An example of this EXEC statement is in member CACIDJNL in the SCACSAMP data set. You can use the returned value to prevent journal archiving unless there are a specified number of full (unarchived) journals that are available for

recovery. Because this change reduces the number of archived journals that are available to the central version, you might want to increase the number of online journal files that the central version uses. Doing so can prevent CA-IDMS from halting because an archived journal is unavailable.

- Allow a certain number of active journals to remain unarchived so that they can be used by the recovery agent. The number of unarchived journals that you need to retain is based on the size of the journal files, the frequency at which journals are filled up (that is, the volume of changes in your CA-IDMS system), and how quickly you can respond to a situation that requires recovering data.

## Configuring automatic recovery of change data from CA-IDMS databases

You can set up automatic monitoring to detect whether a change-capture agent goes into recovery mode. You can also configure the recovery agent to start recovering change data automatically.

### Before you begin

You must be running CA-IDMS in central-version mode.

### Restrictions

If the recovery agent determines that any required journals are archived, it returns an error message and stops. You must manually recover data from archived journals.

If you are running a test environment and want to cold start a correlation service to exit recovery mode, do not modify the startup JCL for your central version to start recovery automatically. If a change-capture agent goes into recovery mode, you can cold start the correlation service.

### About this task

If you do not configure the recovery agent to start automatically, you must follow the manual procedures for recovering change data.

### Procedure

To configure automatic recovery of change data from CA-IDMS databases:

1. To set up automatic monitoring to detect whether a change-capture agent is in recovery mode, follow these steps:
  - a. **Optional:** Monitor periodically as part of your process for offloading journals. Integrate the following jobstep into existing JCL for the CA-IDMS journal offload process, if that process is triggered when a journal is full. This jobstep can be used to prevent the offloading of journals during the recovery process:

```
//CHECK EXEC PGM=CACEC1DR,PARM='MONITOR,RESTARTWAIT=0S'
```

The value 0S for the RESTARTWAIT keyword causes the recovery agent to terminate when it catches up with the current position in the active journal file.
  - b. Run JCL similar to the following example each time that the CA-IDMS Central Version is started. The ellipsis after CV in the second statement indicates that you can use optional keywords:

```
//MONITOR EXEC PGM=CACEC1DR,PARM='MONITOR,RESTARTWAIT=10S,TIMEOUT=5M'
//...
//RECOVER EXEC PGM=CACEC1DR,PARM='CV,...',COND=(0,NE,MONITOR) //...
```

2. If you want the recovery agent to recover change data automatically if a change-capture agent is in recovery mode, modify the JCL that offloads the central version journals to include JCL that is similar to the following example. The ellipsis after CV in the second statement indicates that you can use optional keywords:

```
//CHECK EXEC PGM=CACEC1DR,PARM='MONITOR,RESTARTWAIT=0S'
//...
//RECOVER EXEC PGM=CACEC1DR,PARM='CV,...',COND=(0,NE,CHECK)
//...
```

If you use the optional keyword `ACTIVATE`, IBM recommends to set the value to `N` in a production environment. Using this value requires you to shut down and restart the central version to resume change capture.

## Starting the process of capturing changes from CA-IDMS databases

To begin capturing changes that are made to a CA-IDMS database, you need to do the following steps:

- Install the change-capture agent.
- Create a recovery data set.
- Modify the automatic procedures for archiving journal files.

### Procedure

To start the process of capturing changes:

1. Create a recovery data set and add it to the startup JCL for the CA-IDMS central version. This data set is used to record information about where the recovery process should begin within the CA-IDMS journal files if IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS enters recovery mode.
  - a. Create the recovery data set by allocating a 1-track, 80-byte sequential data set with the following attributes:

```
LRECL=80,DSORG=PS,RECFM=FB
```
  - b. Add the recovery data set to the CA-IDMS central version JCL by adding a `CACRCV DD` statement that references the newly allocated data set as follows:

```
//CACRCV DD DISP=SHR,DSN=name_of_data_set
```
2. Relink the CA-IDMS database I/O module (`IDMSDBIO`) to include the `IDMSJNL2` exit. The `CACIDLDB` member of the `SCACSAMP` library contains sample JCL to relink `IDMSDBIO`. The JCL copies the existing `IDMSDBIO` module to the name `@BKPD BIO`. A new `IDMSDBIO` module is then created by linking the `IDMSJNL2` exit with the `@BKPD BIO` module.
3. If you have an existing `IDMSJNL2` exit, determine whether you need to stack the exit with the IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS version of the exit. Both products ship with a module called `CACECA1D`. This module contains the `IDMSJNL2` exit code and a CSECT name `IDMSJNL2` which is referenced in the CA-IDMS `IDMSDBIO` module as a weak external. When the `CACECA1D` module is linked with `IDMSDBIO`, the `IDMSJNL2` external is resolved and the change-capture agent is active.

To stack the exits, follow these general instructions:

- a. Rename your exit CSECT from IDMSJNL2 to IDM2JNL2. If IDM2JNL2 is resolved by the IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS version of the exit, this version of the exit will automatically call your exit after receiving control from IDMS.
- b. Ensure that the link-edit control card that is used to relink IDMSDBIO includes the following card prior to the INCLUDE SYSLIB(@BKPDBIO) card:

```
REPLACE IDMSJNL2
```

4. Add the sample SCACSAMP member CACIDTRM to the end of the CA-IDMS central version JCL. This step provides notification to the correlation service when the central version is shutdown. This notification tells the correlation service the state of the change-capture agent for the central version.

In the agent parameter on the CACIDTRM job card of the CACIDTRM, specify the name of the change-capture agent that is associated with this central version. The name of the change-capture agent is IDMS\_ *nnn* where *nnn* is the number for this central version as specified in the SYSTEM statement for the central version, as in the following example:

```
// CACIDTRM EXEC PGM=CACE1TRM,  
// PARM='AGENT=IDMS_ nnn'
```

5. Verify the installation of the exit by starting the central version and looking for the following operator message in the central version JES messages:

```
CACH001I CHANGE-CAPTURE AGENT 'IDMS_ nnn' \  
INSTALLED FOR SERVICE 'nnnnnnnn'
```

This message appears only after the first journaled change takes place within the Central Version.

6. Verify that the correlation service receives a termination message when you shutdown the central version. The change-capture agent for the central version must be active and able to communicate with the correlation service. Shutdown the central version. You should see the following message:

```
CACG114I SHUTDOWN RECEIVED FROM ACTIVE AGENT 'IDMS_ nnn'
```

If you see this message, you can restart the central version.

7. **Optional:** Relink the Presspack support module.

If the tables that you are capturing changes from use the Presspack support module to compress data, relink the Presspack support module CACPPK to include the IDMS interface module so that the correlation service can decompress the data that is stored in the central version journals. Sample JCL for relinking is in the SCACSAMP member CACIDLPP.

If you are using other compression or decompression modules, or if you are using DC tables used by the Presspack support module, be sure to include these authorized libraries in the STEPLIB of the correlation service.

If the Presspack support module fails to decompress a record, the following message is written to the log:

```
Presspack RC=nn
```

*nn* represents the return code. The following list shows the return codes from IDMS R14.0:

- |    |                               |
|----|-------------------------------|
| 00 | Decompression successful.     |
| 04 | GETSTG failure.               |
| 08 | Call from system mode module. |
| 12 | DCT load failure.             |

- 16 DCT not valid.
- 20 Record or table not compressed by CA-IDMS Presspack.
- 24 Load of IDMPRES failed.
- >100 Error during decompression; PRESSTO return code = return code minus 100.

## Configuring change capture from IMS databases

The steps for configuring change capture from IMS databases are a subset of the steps for configuring either Classic event publishing or Classic replication from IMS databases.

### IMS environments and supported database types

You can capture changes only to data that is in a database of a supported type and supported IMS version.

The following table lists that supported database types.

*Table 11. IMS environments and supported database types*

IMS environment	Supported database types
DB Batch	Full-function
TM Batch	None
DB/DC	Full function DEDB
DBCTL	Full-function DEDB
DCCTL	None

The supported IMS versions are 7, 8, and 9.

### Change-capture agents for IMS databases

A change-capture agent for IMS databases is the IMS Logger exit (DFSFLGX0) that is supplied by IBM WebSphere Classic Event Publisher for z/OS and IBM WebSphere Classic Replication Server for z/OS.

When an IMS control region initializes, IMS issues a load request for DFSFLGX0. The types of IMS control regions that can run this change-capture agent are:

- DB/DC subsystems
- DBCTL subsystems
- Batch jobs

The name of the change-capture agent is the name of the batch job or started task. If batch jobs have multiple job steps that update IMS data, each job step is considered a separate instance of the change-capture agent's execution.

When a change-capture agent is invoked by IMS in a DB/DC, DBCTL, or batch IMS control region, IMS passes the IMS Logger exit a pointer to a buffer that contains one or more IMS log records that were successfully written to an IMS log file.

The log records written by the control region are buffered. When a buffer is filled and successfully written to the log file, the change-capture agent is called. The

change-capture agent sends a single Cross Memory data gram to the correlation service. In the data gram are any log records that IBM WebSphere Classic Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS use to track the state of the current units-of-work that are in-flight and any type 99 data capture log records that were in the buffer that was passed by IMS.

XM data grams represent a unidirectional asynchronous method of communication with the correlation service. The change-capture agent puts the XM data gram on a memory queue, and the correlation service gets the message when it is not busy doing other work. Bi-directional communications occur between the change-capture agent and the correlation service using common memory (CSA).

A change-capture agent is in one of two modes:

**Active mode**

A correlation service is active, not reporting any errors, and the change-capture agent is successfully communicating with the correlation service.

**Recovery mode**

Communications were never successfully established with a correlation service, the correlation service reported some form of error, or the change-capture agent detected an error condition.

## Augmenting DBDs

To have IMS generate IMS data capture records, you must augment the DBD for which change data needs to be captured. You use the EXIT parameter to specify IMS Data Capture information.

**About this task**

All updates to an IMS database are logged as “type 50” log records, which contain sufficient information to enable database restart and recovery operations. Change capture requires the capture of additional information, such as the complete “before image” of a record before and update or a delete.

Log records with this additional information is written as “type 99” on subsets of DBDs and SEGM for which change capture is required. Such “type 99” log records are generated by IMS whenever the EXIT parameter is specified on the on the DBD, SEGM definitions, or both.

You can specify the EXIT parameter in two locations:

**The DBD definition**

Specifying the EXIT parameter here generates “type 99” log records for every change to the database.

**One or more SEGM definitions**

Specifying the EXIT parameter here limits the generation of “type 99” log records to changes made to those segments only. If only a subset of the segments are required for change capture, you can avoid unnecessary overhead from generating “type 99” log records for the other segments.

For a list of the possible values for the EXIT parameter, see “EXIT parameter in DBD and SEGM statements” on page 105.

These DBD modifications affect only the DBD definition that is stored in the DBD/ACB library and do not affect the physical database.

### Procedure

To augment a DBD that defines the segments that you plan to capture changes from:

1. Supply an EXIT keyword on the DBD statement to define default values for all segments in the DBD.
2. **Optional:** Supply an EXIT keyword on SEGM statements to override the default values for the DBD.
3. Run DBDGEN for the updated DBD.
4. Run the ACBGEN utility to update all PSBs that reference the DBD.
5. Put the updated DBD and PSB members into your production ACB libraries.

After you complete this task, you can use Classic Data Architect to create tables that map to your IMS database.

### EXIT parameter in DBD and SEGM statements

To have IMS generate IMS data capture records, the DBD for which information needs to be captured must be augmented to specify what information you want captured.

### Purpose

Use the EXIT parameter to specify IMS data capture information. The EXIT parameter is supported for the DBD control statement and the SEGM control statement. Supplying a keyword with the EXIT parameter on the DBD statement defines default values for all segments in the DBD. Specifying a keyword with the EXIT parameter on the SEGM statement allows you to override the default values.

### Format

The format of the EXIT parameter is:

```
EXIT=(Exit-Name,KEY|NOKEY,DATA|NODATA,PATH|NOPATH, (CASCADE|NOCASCADE, KEY|NOKEY,\
DATA|NODATA,PATH|NOPATH), LOG|NOLOG)
```

### Keywords

Table 12. Keywords for the EXIT parameter

Keyword	Purpose
Exit-Name	This parameter is used to specify the name of a DPropNR synchronous data capture exit, or the fact that there is no exit (by specifying *), or NONE to deactivate an exit routine on a SEGM statement.  Classic event publishing and Classic replication do not use data capture exits. However, it can co-exist with DPropNR or your own exits. If you do not have any data capture exits, specify * for the Exit-Name parameter.
KEY NOKEY	Identifies whether you want the IMS Data Capture log records to contain physical path concatenated key information for a segment that is deleted, inserted, or updated. The default is KEY.

Table 12. Keywords for the EXIT parameter (continued)

Keyword	Purpose
DATA NODATA	Identifies whether you want physical segment data included in the IMS Data Capture log records for a segment that is deleted, inserted, or updated. The default is DATA.
PATH NOPATH	Identifies whether you want physical segment data included in the IMS Data Capture log records for the parents of a segment that is deleted, inserted, or updated. The default is NOPATH.
CASCADE NOCASCADE	Identifies whether you want IMS Data Capture log records to contain cascade delete information for a segment that has been deleted that has child segments. When CASCADE is specified, the next three sets of keywords can be specified to identify what kind of information you want included in the IMS Data Capture log record for the deleted child segments.
KEY NOKEY	Identifies whether you want the concatenated key information for a child segment included in the IMS Data Capture log record. The default is KEY.
DATA NODATA	Identifies whether you want the physical segment data included in the IMS Data Capture log record for a deleted child segment. The default is DATA.
PATH NOPATH	Identifies whether you want physical segment data included in the IMS Data Capture log records for the parents of a child segment that is deleted. The default is NOPATH.
LOG NOLOG	Identifies whether you want IMS Data Capture log records to be generated to the IMS log files. If an * is specified for Exit-Name, the default is LOG. You can specify NOLOG on individual SEGM statements, if these segments do not contain data that is to be captured. Alternately, you can specify NOKEY, NODATA for segments that you do not want data captured for.

### Augmentation of DBDs for root-only databases or for child segments

Follow these guidelines when augmenting DBDs for root-only databases and child segments.

It is important that you chose the right level of augmentation. If you add more EXIT parameters than required or if you use unnecessary options within the EXIT parameters, you will generate unnecessary load on your system. If you add fewer EXIT parameters or options than required, change-capture agents might not capture all of the changes that you want.

### Augmentation of DBDs for root-only databases

For root-only databases, specify the following EXIT parameter on the DBD statement:

```
EXIT=(*,NOKEY,DATA,NOPATH,(NOCASCADE),LOG)
```

Specifying these values in the EXIT parameter causes the capture of all changes to the root segment and informs IMS that path data and cascade delete information is

not required. These values do not request concatenated key information because that information will be available in the before and after images for the root segment. Not requesting concatenated key information reduces the size of the IMS data capture log records that are created.

### **Augmentation of DBDs for child segments**

When you are augmenting DBD's with child segments, consider whether your database is designed in Third Normal Form, and how many child segments physically exist. Third Normal Form means that each table that is defined in Classic Data Architect for change capture maps to fields of only one child segment in the IMS database plus the key fields of all parent segments in the path. Augment and map only child segments from which you want to capture changes. For each segment that you want to ignore, specify the following EXIT parameter in the SEGM statement:

```
EXIT=(*,NOKEY,NODATA,NOPATH,(NOCASCADE),NOLOG)
```

When your IMS database is designed in Third Normal Form, presumably the foreign key information required is contained in the IMS concatenated key. Code the following EXIT statement for the child segment being monitored:

```
EXIT=(*,KEY,DATA,NOPATH,(?),LOG)
```

The question mark represents the cascade delete parameter and options that are in effect.

Your IMS database might be in Third Normal Form, but the application that is receiving the messages from the distribution service does not recognize the segment data and concatenated key data as sufficient to create a foreign key for its own use. In these situations you must capture path data. Minimally you must code the following

```
EXIT=(*,KEY,DATA,PATH,(?),LOG)
```

## **Running more than one correlation service when capturing from IMS**

If you have multiple IMS subsystems defined on the same z/OS LPAR, you can configure the change-capture agent for each subsystem to send change data to a separate correlation service rather than to one correlation service.

### **About this task**

By default, if you set up change capture from multiple IMS subsystems, all of the change-capture agents send change data to a single correlation service.

### **Procedure**

To run more than one correlation service:

1. Create a service information entry for every correlation service that you want to run. Create each service information entry in the configuration file for the data server where the correlation service will run. In field 10 of each service information entry, add a NAME parameter to the service information entry of the correlation service (such as NAME=IMS\_subsystem). Only one correlation service on a z/OS LPAR can remain without a name.
2. For each change-capture agent (except the change-capture agent that will communicate with the unnamed correlation service), follow these steps:

- a. Update the CACE1OPT source file by adding the server name. The source code for the CACE1OPT module is located in the SCACSAMP library.
- b. Reassemble the CACE1OPT module.
- c. Create a customized version of the IMS Logger Exit (DFSFLGX0) that identifies the name of the correlation service that the change-capture agent communicates with.
- d. Re-link the IMS Logger Exit (DFSFLGX0). Member CACIMSLX in the SCACSAMP data set contains sample JCL for re-linking the exit.

## Example

For example, you have two different IMS systems, IMST for test and IMSP for production. You create two instances of the correlation service and name them IMST and IMSP. You create two customized versions of the IMS Logger Exit, DFSFLGX0. One version uses the name IMST; you place the exit in the IMST RESLIB load library. You place the second version of the exit, which communicates with the correlation service that is named ISMP, in the IMSP RESLIB load library.

The name of the IMS Logger Exit load module, which is the change-capture agent, is determined by IMS and is hard coded to be DFSFLGX0. Therefore, you must do either of the following tasks:

- You must have IMS RESLIBs with different names.
- You must place each version of DFSFLGX0 in a different APF-authorized load library (for DB/DC or DBCTL subsystems) and modify your IMS batch and started task JCL to reference the correct version of DFSFLGX0.

## Creating recovery data sets before activating change-capture agents for IMS data sources

A recovery data set is a single-record, 80-byte fixed-length sequential data set that a change-capture agent updates when the agent is active and the correlation service is not.

### About this task

When activated, the change-capture agent records in the recovery data set information about the first log record that was passed in the IMS log buffer on the first IMS Logger exit write call.

The recovery agent requires the information in the recovery data set if you start a change-capture agent without a correlation service being active.

### Procedure

To create recovery data sets before activating change-capture agents:

1. Allocate an 80-byte fixed-length physical sequential data set for each change-capture agent. Define the block size as 80-bytes and allocate only one block. No secondary extents are required because the data set contains only a single record.

Use the following naming standard for recovery data sets:

*&HIGH-LEVEL-QUALIFIER.SECOND-LEVEL-QUALIFER.&JOBNAME*

*&HIGH-LEVEL-QUALIFIER*

A high-level data set name qualifier that the IMS control region job or started task name can access and update.

### *SECOND-LEVEL-QUALIFER*

A constant of up to 8 characters that groups data sets together.

### *JOBNAME*

The job name or started task name of the IMS control region that you are capturing changes from.

2. Update your IMS JCL to include a reference to a recovery data set for each change-capture agent that is running at your site.

## **Installing change-capture agents for IMS databases**

To begin capturing changes that are made to an IMS database, you need to install the change-capture agent.

### **Before you begin**

Make sure that the correlation service that will communicate with the change-capture agent is running.

### **Procedure**

To install change-capture agents for IMS databases:

1. If you implemented your own IMS Logger exit or are using a different exit, you can invoke that exit with the version of the IMS Logger exit supplied by IBM WebSphere Classic Data Event Publisher for z/OS and IBM WebSphere Classic Replication Server for z/OS.

Use the SCACSAMP member CACIMSLX to relink the IMS Logger exit. Modify the JCL to include a reference to the load library where the existing DFSFLGX0 exit resides on the SYSLIB DD statement.

The SCACSAMP member CACIMLEX is a sample relink job that will create a backup of the exit that you are using. This job then relinks the IMS Logger exit with your exit. Your exit must be named DFSFLGX0 for the call to succeed. The exit is invoked after the active change-capture agent completes its processing.

2. Install the version of the IMS Logger exit that is provided by IBM WebSphere Classic Data Event Publisher for z/OS and IBM WebSphere Classic Replication Server for z/OS. Use either of the following methods:

#### **For large-scale deployments**

Copy module DFSFLGX0 from the IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS distribution libraries into the SDFSRESL.

You have a large-scale deployment when either of the following conditions is true:

- You are planning to augment the majority of your IMS databases for change capture.
- You are augmenting an IMS database for change capture that is updated by the majority of your IMS applications.

#### **For small-scale deployments**

If you are planning a smaller-scale implementation that only monitors a small number of IMS databases that are updated by a small number of IMS applications, concatenate the IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS load library into your IMS batch jobs and started task procedures for the online DB/DC or DBCTL regions.

In a small-scale implementation, the number of IMS batch jobs and started task procedures that need to be updated are reduced. However, if you forget to update one of your IMS applications that updates a monitored database, these changes are lost, and the correlation service has no knowledge that the loss occurred.

If you install the change-capture agent in SDFSRESL and are performing only a small-scale implementation, the correlation service still tracks all IMS control regions that are referencing SDFSRESL where the change-capture agent is installed, even though many of these IMS applications do not update databases that are being monitored for changes. Likewise, if these change-capture agents go into recovery mode, you have to recover these failed agents, even though no IMS changes are being captured, which means more work for you.

3. After making the necessary changes to the IMS region JCL, verify the installation by starting the DB/DC or DBCTL region where the change-capture agent is installed and looking for the following operator message in the IMS region JES messages:

```
CACH001I CHANGE CAPTURE AGENT 'IMS_XXXX' INSTALLED FOR SERVICE '(noname)'
```

4. Verify that the termination message is working correctly by stopping the IMS region.

The DB/DC or DBCTL region must be running in active mode and communicating successfully with the correlation service at the time that you stop the IMS region. The change-capture agent should issue the following message even if it is in recovery mode, though it will not issue the message if it deactivated itself because it detected a serious error or abend:

```
CACH098I IMS TERMINATION NOTIFICATION RECEIVED
```

If the change-capture agent is not in recovery mode, it should next issue this message:

```
CACH099I SENDING IMS TERMINATION NOTIFICATION
```

This message says that the change-capture agent is notifying the correlation service that the IMS region is shutting down.

The correlation service should issue the message:

```
CACG114I SHUTDOWN RECEIVED FROM ACTIVE AGENT 'IMS_XXXX'
```

The correlation service will issue this message only if the change-capture agent is in active mode. Otherwise, all active messages (including the shutdown message) are disabled, because recovery is necessary.

**For batch jobs:** If the change-capture agent is running as part of a batch job, these messages appear when the batch job terminates.

## Configuring the tracking of log files

Use the log tracking utility to track the IMS log files that are associated with a change-capture agent. If the change-capture agent goes into recovery mode, the recovery agent will be able to identify the IMS log files that contain records that need to be recovered.

### About this task

Implementing IMS log file tracking differs slightly between DB/DC, DBCTL, and IMS batch environments, but in all of these environments IMS log files are defined to the IMS subsystem where change capture is taking place.

## Procedure

To use the log tracking utility:

1. Create a data set for tracking IMS log files.

The name of the file that the IMS recovery agent attempts to open is:

*name\_of\_recovery\_data\_set*.LOGS

Therefore, when you define an IMS log file tracking data set, you must follow this naming convention.

2. If you are capturing changes from IMS DB/DC or DBCTL subsystems, add the log tracking utility as a job step to the log archive JCL.

Member CACIMSLT in SCACSAMP contains sample JCL for this job step.

3. If you are using IMS batch jobs to make changes to your data, add job steps for the log tracking utility after each job step that changes data.

Member CACIMSLT in SCACSAMP contains sample JCL for this job step.

4. Modify the JCL for the log tracking utility as follows:

- a. In the PARM value for the EXEC statement in the JCL to run the log tracking utility, use any of the keywords listed in the following table.

*Table 13. Keywords to control the behavior of the log tracking utility*

Keyword	Value	Description
DIALOGS	Y   N	<p>Specifies whether the log tracking utility collects information about a secondary IMS log file:</p> <p><b>Y</b>      The IMS control region is using dual logging.</p> <p><b>N</b>      The IMS control region is using single logging.</p> <p>The default value is N. PARM='DUALLOGS=N'</p>
ECHO	Y   N	<p>Specifies whether the log tracking utility issues informational WTO messages:</p> <p><b>Y</b>      Issue informational WTO messages.</p> <p><b>N</b>      Do not issue informational WTO messages.</p> <p>The default value is Y. PARM='DUALLOGS=Y,ECHO=N'</p>

Table 13. Keywords to control the behavior of the log tracking utility (continued)

Keyword	Value	Description
MAXLOGS	Number	<p>Identifies the maximum number of entries that are to be maintained in the IMS log file tracking data set. Specifying a value of 0, or not supplying a MAXLOGS value causes the log tracking utility to maintain an unlimited number of IMS log file entries.</p> <p>In general, IMS log files are defined as generation data sets and only a fixed number of generations are retained. If your IMS log files are so defined, supply a value for MAXLOGS that matches the number of generations that are retained. Otherwise, the log tracking utility will track log files that no longer exist.</p> <p>If you specified dual logging for a change-capture agent, the log tracking utility doubles the MAXLOGS value that you enter because the log tracking utility assumes that an equal number of primary and secondary IMS log files are being retained.</p> <p>PARM= ' DUALLOGS=N,ECHO=N,MAXFILES=5 '</p>

b. Use fixed DD names to identify these items:

- The primary log files (referenced by the CACLOG1 DD statement) to be registered
- The secondary log files (referenced by the CACLOG2 DD statement) to be registered
- The name of the IMS log file tracking data set (referenced by the CACTRACK DD statement) to be updated

### Optional log file tracking for IMS databases

Given the information contained in an IMS log file tracking data set, the recovery agent can identify the IMS log files that contain records that need to be recovered before you can return a change-capture agent to active mode.

Log file tracking tracks the IMS log files that are associated with a change-capture agent. When log file tracking is implemented, all IMS log files that are associated with a change-capture agent are tracked whether the change-capture agent is in active or recovery mode. Log file tracking requires a data set that contains information about the IMS log files with relevant records. This data set is an 80-byte sequential data set that can contain multiple records. You can control how many log entries are maintained for each agent being tracked. You need to create this file manually.

The log tracking utility manages the content of an IMS log file tracking data set. The data set for tracking IMS log files contains a record for each primary and secondary IMS log file that is created by IMS for a given change-capture agent. The content of each record contains the information that is listed in the following table.

Table 14. The contents of a data set that is used for tracking log files

Name	Starting offset	Length	Description
Log file type	1	1	Type of IMS log file. The possible values are as follow: <b>1</b> Primary log file <b>2</b> Secondary log file
First log record suffix	2	16	The suffix of the first IMS log record in the log file. The IMS log record suffix consists of an 8-byte system clock value and a 8-byte double-word IMS log sequence number.
Log file name	18	44	Fully-qualified data set name of the IMS log file.

Based on a restart point, the IMS recovery agent can determine whether the IMS log file was created before or after the agent went into recovery mode. Any file that is created after the restart point must be recovered. Also, the last log file with a system clock value that starts before the restart point is necessary. It is assumed that this IMS log file contains the IMS log record that the restart point is referring to.

When reporting about IMS log files that are required in the recovery process, the IMS recovery agent identifies primary IMS system log files, or archived IMS system log files in preference to secondary IMS system or archived log files. This behavior is modified when there is no primary IMS system or archived log file information and only secondary IMS log file information is available.

If the data set for tracking IMS log files gets corrupted, you will need to update its content either by running the IMS log tracking utility manually or by physically modifying the content of the data set. For best results, make sure that IMS log files are listed in the sequence in which they were created. Also, make sure that secondary IMS log files are listed immediately after the corresponding primary log file entry. If you do not follow these guidelines, the IMS recovery agent will terminate and report errors when duplicate IMS log files are supplied.

## Restart points for IMS databases

Every change-capture agent that is in recovery mode has a restart point. A restart point is a location within an IMS log file where the recovery process needs to begin.

You must input the following items into the recovery process:

- The IMS log file that contains the restart point.
- All log files created after the restart point that contain changes to the databases that you are capturing changes from, up until the point that IMS is shutdown, or a quiesce point is reached.
- If the agent associated with an IMS control region is in recovery mode and that IMS job or subsystem is run again before the recovery process is completed, any IMS log files created in subsequent executions.

For each change-capture agent that is in recovery mode, there is a specific record within an IMS log file where recovery needs to start. This record is the first IMS log record that was written after the active change-capture agent was called for the first IMS Logger exit write operation.

- For an IMS batch application, the restart point identifies the first IMS log record in the log file.
- For a DC/DC or DBCTL subsystem, the restart point generally identifies the first IMS log record in the currently active online log data set.

When a UOR (unit of recovery) is committed by the correlation service, the agent's restart point is the IMS log record that is associated with the first change received from the oldest living UOR when the committed UOR started. Each UOR that is committed pushes the restart point farther into the future. The future in this context is forward in the stream of database changes represented in the IMS log files.

The following information is associated with a restart point:

- DB2 timestamp when the change-capture agent went into recovery mode.
- Internal timestamp when the change-capture agent went into recovery mode.
- The IMS log record suffix of the first IMS log record that was recorded by the change-capture agent or the first type 99 log record that was received for the UOR with changes that need to be recovered.
- The IMS log record suffix of the first log record that was recorded by the change-capture agent or the first type 99 log record for the oldest UOR in existence when the UOR with changes that need to be recovered was created.

Although multiple UORs might be in-flight when a change-capture agent goes into recovery mode, the restart point is recorded for the UOR that is the oldest. To illustrate how restart points progress forward in time, the following figure shows a DB/DC or DBCTL subsystem that has four UORs that are being tracked.

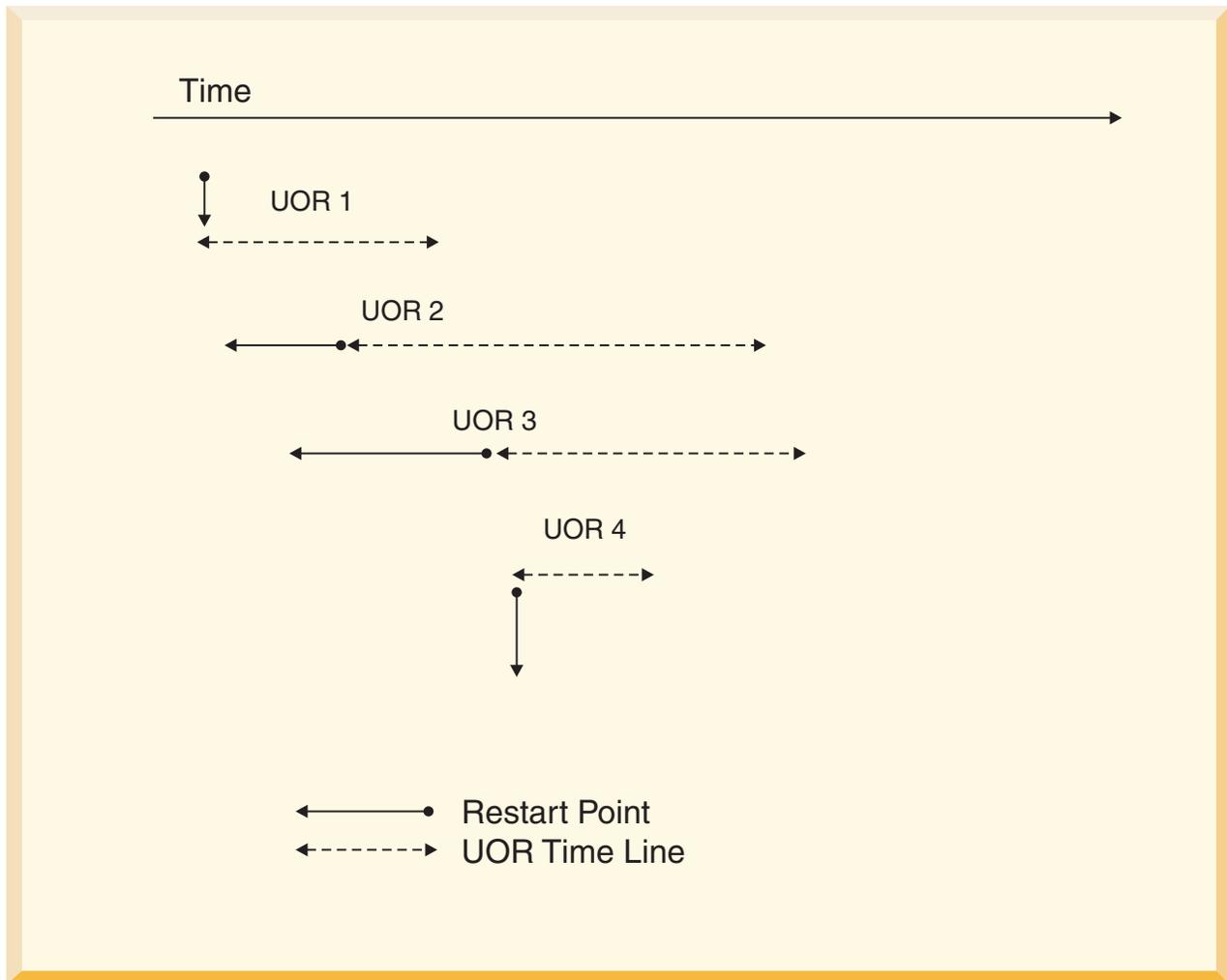


Figure 4. Four UORs that are being tracked in a DB/DC or DBCTL subsystem

The following points explain the figure:

1. The restart point for UOR 1 is the time at which that UOR started.
2. Because UOR 1 and UOR 2 overlap in time, the restart point for UOR 2 is the time at which that UOR 1 started.
3. UOR 2 and UOR 3 also overlap in time; therefore, the restart point for UOR 3 is the time at which UOR 2 started.
4. UOR 4 does not overlap with any other UOR, so its restart point is the time at which it started.

The DB2 timestamp for a restart point is based on the system clock value for the oldest living UOR. The system clock value is either the IMS log record or the first type 99 log record, if a UOR was in-flight when the system went into recovery, or the system clock value of the first log record recorded by the change-capture agent. The time has millisecond resolution. The internal timestamp consists of a DB2 timestamp with a resolution of one-ten-thousandth of a second, based on the oldest living UOR system clock value.

The combination of the system clock value and the log sequence number, both of which are in the IMS log record suffix, identify a unique IMS log record that is

used to establish the restart point. The 8-byte system clock value identifies when the IMS log record was created. The 8-byte double-word IMS log record sequence number is generated by IMS.

---

## Configuring change capture from CICS VSAM files

The steps for configuring change capture from VSAM files that are under the control of CICS Transaction Server are a subset of the steps for configuring either Classic event publishing or Classic replication.

### Three types of change-capture agent for CICS VSAM files

Three different types of change-capture agent are available for capturing changes from CICS VSAM files.

#### File control agents

The file control agent consists of a CICS global user exit (GLUE) and a CICS task-related user exit (TRUE), which together capture deletes, inserts, and updates to VSAM data sets. After you define the exits in CICS, you can run CICS transactions to enable or disable the file control agent.

When you need to recover change data, you can use the auto-journal agent or your own method of recovery. If you want to use the auto-journal agent, you must set up auto-journal logging in your VSAM file definitions before you begin capturing changes. The auto-journal agent is configured with a service information entry in the configuration file for the data server and runs in the same address space as the correlation service. The auto-journal agent can process multiple auto-journal log streams in parallel. After consuming all data from all log streams, the auto-journal agent marks the file control agent as active.

This method of recovering change data does not require VSAM files to be recoverable.

#### Auto-journal agents

The auto-journal agent captures changes by processing one or more auto-journal log streams in parallel. You must configure auto-journaling in the VSAM files that you want to capture changes to. Starting at a previously recorded restart point or at a point that you specify, the auto-journal agent continuously reads the auto-journal logs and forwards changes to the correlation service.

By logically demarcating transactions, the auto-journal agent generates the appropriate COMMIT messages for the correlation service. The auto-journal agent assumes that transactions complete within a certain time interval. The default for this interval is 10 seconds, although you can change the interval to meet your own requirements.

The auto-journal agent can process multiple auto-journal log streams in parallel. After consuming all data from all log streams, the agent continues to check for additional data.

You configure the auto-journal agent with a service information entry in the configuration file for the data server in which the agent runs.

## Log-reading agents

The log-reading agent reads commit and rollback synchpoint notifications from the system log. The agent requires the CICS system log for obtaining the demarcations for units of recovery. You must retain system logs for at least one day. Otherwise, the system logger could physically delete log data before the log-reading agent reads that log data.

This agent also requires at least one forward recovery log.

The log-reading agent is configured with a service information entry and runs in the same address space as the correlation service does. The log-reading agent also recovers data. The recovery process is automatic and the agent returns to active mode after recovering data.

This agent requires the RECOVERY option in VSAM file definitions so that after images are written to the selected forward recovery log.

## Configuring change capture from CICS VSAM files by using file control exits

The steps for configuring change capture from VSAM files by using file control exits are a subset of the steps for configuring either Classic event publishing or Classic replication.

### Defining file control agents to CICS

You define file control agents by creating new CICS resource definitions for the global user exit and the task-related user exit and then making those definitions available to CICS.

#### Procedure

To define a file control agent:

1. Install the task-related user exit program.
  - a. In CEDA, use this command to define the task-related user exit.

```
CEDA DEFINE PROGRAM(program_name) GROUP(group_name)
```

*program\_name*

Use the name "CACCTRU".

*group\_name*

The name depends on your CICS installation. In general, the name should identify a group that is automatically installed when you start CICS.

Accept the default values for all parameters except the following parameters:

#### DESCRIPTION

Value: File Control TRUE exit

#### LANGUAGE

Value: Assembler

#### DATALOCATION

Value: Any

#### EXECKEY

Value: CICS

**CONCURRENCY**

Value: Threadsafe

- b. Install the new definition with the CEDA INSTALL command so that the exit is available.
2. Install the global user exit program.
    - a. In CEDA, use this command to define the global user exit program.

```
CEDA DEFINE PROGRAM(program_name) GROUP(group_name)
```

*program\_name*

Use the name "CACCGLU".

*group\_name*

The name depends on your CICS installation. In general, the name should identify a group that is automatically installed when you start CICS.

Accept the default values for all parameters except the following parameters:

**DESCRIPTION**

Value: File Control GLUE exit

**LANGUAGE**

Value: Assembler

**DATALOCATION**

Value: Any

**EXECKEY**

Value: CICS

**CONCURRENCY**

Value: Threadsafe

- b. Install the new definition with the CEDA INSTALL command so that the exit is available.

## Defining in CICS the programs that enable and disable file control agents

After installing a file control agent by defining and installing a task-related exit and global user exit in CICS, define in CICS the programs that will enable and disable the file control exit.

### About this task

To simplify the task of enabling this agent, define a CICS transaction to enable the file control agent. Any time that you need to enable the file control agent, you only need to run the CICS transaction.

### Procedure

To define the programs that enable and disable a file control agent:

1. In CEDA, use this command to define the program that will enable the file control agent.

```
CEDA DEFINE PROGRAM(program_name) GROUP(group_name)
```

*program\_name*

Use the name "CACCEA".

*group\_name*

The name depends on your CICS installation. In general, the name should identify a group that is automatically installed when you start CICS.

Accept the default values for all parameters except the following parameters:

**DESCRIPTION**

Value: Enable File Control Agent

**LANGUAGE**

Value: Assembler

**DATALOCATION**

Value: Below

**EXECKEY**

Value: User

**CONCURRENCY**

Value: Quasirent

2. In CEDA, use this command to define the program that will disable the file control agent.

```
CEDA DEFINE PROGRAM(program_name) GROUP(group_name)
```

*program\_name*

Use the name "CACCDIS".

*group\_name*

The name depends on your CICS installation. In general, the name should identify a group that is automatically installed when you start CICS.

Accept the default values for all other parameters except the following parameters:

**DESCRIPTION**

Value: Disable File Control Agent

**LANGUAGE**

Value: Assembler

**DATALOCATION**

Value: Below

**EXECKEY**

Value: User

**CONCURRENCY**

Value: Quasirent

## Defining CICS transactions that run the programs that enable or disable file control agents

You must define a CICS transaction to use for enabling and disabling file control agents.

### Procedure

To define CICS transactions for enabling and disabling a file control agent:

1. In CEDA, use this command to define the CICS transaction that will run CACCENA, the program that enables the file control agent:

```
CEDA DEFINE TRANSACTION(transaction_name) GROUP(group_name)
```

*transaction\_name*

The name of the transaction.

*group\_name*

The name depends on your CICS installation. In general, the name should identify a group that is automatically installed when you start CICS.

Accept the default values for all parameters except the following parameters:

**DESCRIPTION**

Value: Enable File Control Agent

**PROGRAM**

Value: CACCENA

2. In CEDA, use this command to define the CICS transaction that will run CACCDIS, the program that disables the file control agent:

```
CEDA DEFINE TRANSACTION(transaction_name) GROUP(group_name)
```

*transaction\_name*

The name of the transaction.

*group\_name*

The name depends on your CICS installation. In general, the name should identify a group that is automatically installed when you start CICS.

Accept the default values for all other parameters except the following parameters:

**DESCRIPTION**

Value: Disable File Control Agent

**PROGRAM**

Value: CACCDIS

## Defining CICS transactions to disable file control agents

If you want to stop capturing changes, you must disable the file control agent. You must first define a CICS transaction so that you can disable the file control agent when you need to.

### About this task

To stop capturing changes, you disable the file control agent before stopping the correlation service. If you do not stop the file control agent before stopping the correlation service, the file control agent will go into recovery mode.

After you define the CICS transaction to disable the file control agent, you can run the transaction any time that you want to stop capturing changes.

### Procedure

To define a CICS transaction to disable a file control agent:

1. In CEDA, use this command to define the program that will disable the file control agent.

```
CEDA DEFINE PROGRAM(program_name) GROUP(group_name)
```

*program\_name*

Use the name "CACCDIS".

*group\_name*

The name depends on your CICS installation.

Accept the default values for all other parameters except the following parameters:

**DESCRIPTION**

Value: Disable File Control Agent

**LANGUAGE**

Value: Assembler

**DATALOCATION**

Value: User

**EXECKEY**

Value: User

**CONCURRENCY**

Value: Quasirent

2. In CEDA, use this command to define the CICS transaction that will run CACCDIS:

```
CEDA DEFINE TRANSACTION(transaction_name) GROUP(group_name)
```

*transaction\_name*

This name must match the TRANSID in the definition of CACCDIS.

*group\_name*

This name must match the GROUP name in the definition of CACCDIS.

Accept the default values for all other parameters except the following parameters:

**DESCRIPTION**

Value: Disable File Control Agent

**PROGRAM**

Value: CACCDIS

## Activating auto-journal logging for VSAM files

Before you start capturing changes to VSAM files, you must activate auto-journal logging for those files if you plan to use the auto-journal agent to recover change data. If you plan to recover change data by using your own method, you can skip these steps.

### Before you begin

The recovery agent uses auto-journal logs to recover change data. If you do not activate auto-journal logging before starting change capture, you will not be able to recover change data with the recovery agent if an error puts the system into recovery mode.

### About this task

The values that you specify for the parameters for auto-journal logging determine the data changes that are logged. Be sure that the values that you set match the type of data changes that you want to capture.

For information about the parameters for auto-journal logging, see Enabling CICS TS autojournaling.

## Procedure

To activate auto-journal logging for a specific file, update the following parameters in the CICS file definition using the CICS transaction CEDA:

### JOURNAL

Specifies the auto-journal that will receive change data. The value must be between 01 and 99.

### JNLRead

Recommended value: Updateonly.

Determines whether read operations are recorded in the auto-journal. You must specify Updateonly or All.

Possible values:

**None** No read operations are recorded.

#### Updateonly

Reads for updates are included in the auto-journal.

#### Readonly

Only ordinary reads are included in the auto-journal.

**All** All reads are included in the auto-journal.

### JNLSYNCRRead

Recommended value: Yes

Possible values:

**Yes** Auto-journal records that are generated by read requests are written synchronously.

**No** Auto-journal records that are generated by read requests are written asynchronously.

### JNLUpdate

Recommended value: Yes

Possible values:

**Yes** Rewrite and delete operations are recorded in the log.

**No** Rewrite and delete operations are not recorded in the log.

### JNLAdd

Recommended value: After

Possible values:

**After** Journal the file control write operation after the VSAM I/O operation.

**ALL** Record the write file control operation both before and after the completion of the VSAM I/O operation.

#### BEFORE

Journal the file control write operation before the VSAM I/O operation.

#### NONE

Do not journal add operations.

### JNLSYNCRWrite

Recommended value: Yes

Possible values:

- Yes** Auto-journal records generated via write requests are written synchronously.
- No** Auto-journal records generated via write requests are written asynchronously.

## Configuring named correlation services for the file control agent for CICS VSAM files

If you are capturing from CICS VSAM files by using a file control agent, you must name the correlation service. The file control agent and the correlation service run in different address spaces, so the file control agent communicates with the correlation service by using cross memory services.

### Before you begin

There must be a service information entry for the correlation service in the configuration file for the data server where the correlation service will run.

### About this task

You can use the following procedure to name a correlation service for the first time or to rename a correlation service.

### Procedure

To use a named correlation service or to rename a correlation service:

1. Modify the service information entry for your correlation service by adding a `NAME=svrname` parameter, where `svrname` is the name that you want to assign to your correlation service. The name must be eight characters or fewer. Both the active and recovery change-capture agents need to reference your correlation service by this name.
2. Update the CACE1OPT source module to include the name of the correlation service in the constant labeled `SRVRNAME`.
3. Assemble the updated CACE1OPT source module.

The CACE1OPT module is used by the load modules `CACCGLU` and `CACCTRU`. Those load modules must be in your CICS load library.

```
//valid job card here
//*****
//* JOB CONTROL STATEMENTS TO ASSEMBLE A MODIFIED OPTIONS TABLE *
//* (CACE1OPT) *
//*****
//ASSEMBLE EXEC PGM=ASMA90,PARM='LIST,NODECK,RENT'
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//SYSIN DD DISP=SHR,DSN=your.source.input.library(CACE1OPT)
//SYSLIN DD DISP=SHR,DSN=your.assembly.output.library(CACE1OPT)
//SYSUT1 DD DSN=&SYSUT1,UNIT=VIO,SPACE=(1700,(2000),,ROUND)
//SYSPRINT DD SYSOUT=*
```

- Link the load modules directly into your CICS load library. You can use the sample JCL in member `CACVSMXLX` in the `SCACSAMP` data set:

```
//*CACVSMXLX PROVIDE VALID JOB CARD
//*****
//* *
//* JOB CONTROL STATEMENTS TO LINK A MODIFIED OPTIONS TABLE *
//* (CACE1OPT) WITH THE VSAM FILE CONTROL CHANGE CAPTURE AGENT *
//* *
//*****
```

```

//LNKOPT PROC CAC='CAC' EXADAS HIGH LEVEL QUALIFIER
//*
//LINK EXEC PGM=IEWL,
// PARM='MAP,LIST,AMODE=31,RENT,REUS,REFR'
//LOAD DD DISP=SHR,DSN=&CAC..V3R0M00.SCACLOAD
//OBJ DD DISP=SHR,DSN=&&OBJ
//SYSLMOD DD DISP=SHR,DSN=&CAC..V3R0M00.SCACLOAD
//SYSUT1 DD UNIT=3390,SPACE=(1024,(120,120),,ROUND),DCB=BUFNO=1
//SYSPRINT DD SYSOUT=*
// PEND
//LNKOPT EXEC LNKOPT
//SYSLIN DD *
INCLUDE OBJ(CACE1OPT)
INCLUDE LOAD(CACCGLU)
MODE RMODE(ANY)
ENTRY MSLAVGLU
NAME CACCGLU(R)
INCLUDE OBJ(CACE1OPT)
INCLUDE LOAD(CACCTRU)
MODE RMODE(ANY)
ENTRY MSLAVTRU
NAME CACCTRU(R)
/*

```

- Copy the load modules from the loadlib specified in the JCL. After making changes specific to your environment, submit the following JCL to re-link the affected load modules:
4. Follow these steps if these conditions are true:
    - You already configured a named correlation service.
    - You enabled the file control agent that works with the named correlation service.
    - You started the named correlation service.
    - You followed steps 1 through 3 to rename the correlation service.
      - a. Disable the file control agent.
      - b. Stop and restart the correlation service.
      - c. Enable the file control agent.

## Enabling file control agents

Enable the file control agent so that it can be invoked for change capture.

### Before you begin

If you are doing Classic event publishing, make sure that the following conditions are true:

- If you are using one data server, that data server is configured and running, your correlation service is running, and your distribution service is running.
- If you are using two data servers or more, make sure that the data server where your correlation service is configured is configured and running, and that your correlation service is running.
- Make sure that connectivity is configured between the data server where the correlation service is running and your CICS VSAM files.

After you enable a file control agent, change capture begins.

### Procedure

To enable a file control agent:

Submit to CICS the TRANSID of the transaction that you defined to enable the file control agent.

## Configuring and running auto-journal agents

The steps for configuring change capture from VSAM files by using auto-journal agents are a subset of the steps for configuring either Classic event publishing or Classic replication.

You create a service information entry for the auto-journal agent in the configuration file where the agent will run. Member CACCSCF in the SCACSAMP data set contains a sample configuration file.

The steps for starting an auto-journal agent depend on whether the data server where the agent is configured is running.

### Creating service information entries for auto-journal agents

Auto-journal agents run as services within data servers. To configure an auto-journal agent, you create a service information entry in the configuration file for the data server in which the auto-journal agent will run.

#### Restrictions

The service information entry for the auto-journal agent must come after the service information entry for the correlation service that the agent will communicate with.

#### Procedure

To create a service information entry for an auto-journal agent:

In the configuration file (member CACCSCF in the SCACCONF data set), create a service information entry. Use the following values for the fields:

##### Field one

Use the name CACECA1J.

##### Field two

Specify the name of the agent. If you want to use the auto-journal agent as a recovery agent to the file control agent, the value in this field must be the jobname of the CICS region where the file control agent executes.

##### Field three

2

##### Field four

- Specify 0 if the auto-journal agent performs data recovery for a file control agent. This value prevents the auto-journal agent from starting automatically when you start the data server. You can start the agent with the START command when you need to recover data.
- Specify 1 if the auto-journal agent performs change capture. The auto-journal agent starts automatically when you start the data server.

##### Field five

1

##### Field six

1

**Field seven**

Leave this value set to 4 unless you are asked to change it by IBM® Software Support.

**Field eight**

Specify the maximum amount of time that a service will wait for an expected response before terminating a connection.

Use MS for milliseconds, S for seconds, and M for minutes.

**Field nine**

Specify the amount of time the agent will delay between encountering an end of stream condition for all log streams and attempting to read the log streams again.

Use MS for milliseconds, S for seconds, and M for minutes.

**Field ten****SERVER=*name***

Specify the name of the correlation service that processes the change data that is sent by the auto-journal agent.

**THROTTLE=*nnnn***

Optional: Specify the maximum number of messages that the recovery agent can put in the change data queue of the correlation service. Specifying a value of 0 disables throttling of messages. The maximum value is 9999.

Using the THROTTLE parameter can prevent the recovery agent from completely filling the correlation service's change data queue with messages. The auto-journal agent can fill up this queue if the point at which recovery begins is significantly behind the current activity point. Throttling can ensure that the message queue has available space for messages from any active change-capture agents that are communicating with the correlation service.

**Log streams**

List the log streams to read, separated by spaces. Specify one or more user log streams. The log streams can be listed in any order.

**COLDSTART**

Optional: Use this parameter to override the restart information that is stored by the correlation service. In addition to COLDSTART, you must specify either of these two keywords:

**OLDEST**

Specifies to start processing at the beginning of the log stream.

**TIME *mmdyyy hhmms***

Specifies to start recovery processing at the indicated time. The time should occur during a period of CICS inactivity. If the time is set to an active period, all activity for a specific unit of work might not be captured, which could result in errors or missed updates.

**INTERVAL=*nnn***

Optional: Use this parameter to override the transaction demarcation interval that is used by default during auto-journal processing. This value represents the assumed duration in seconds

during which all changes for a unit of work are expected to be complete. The default value is 10. The minimum value is 1.

## RECOVERY

Use this parameter if you want to use the auto-journal agent to recover change data when a file control agent enters recovery mode. The auto-journal agent processes the identified auto-journal log streams to the end of the streams and then returns the file control agent to active mode.

If you do not use this parameter, the auto-journal agent will continue to process auto-journal log streams, and wait for new data to be written to the log streams as necessary until the auto-journal agent is stopped or the server region is terminated.

## Example

Here is a sample information entry for an auto-journal agent:

```
SERVICE INFO ENTRY = CACECA1J CICSD 2 1 1 1 1 5M 5S \  
SERVER=CJWCS1 \  
INTERVAL=8 \  
CICSD.CICSD.DFHJ01 \  
CICSD.CICSD.DFHJ02 \  
CICSD.CICSD.DFHJ03 \  
CICSD.CICSD.DFHJ04 \  
CICSD.CICSD.DFHJ05
```

## Starting change capture from CICS VSAM files by using auto-journal agents

The steps for starting change capture depend on whether the correlation service and distribution service are running on the same LPAR or separate LPARs.

### Before you begin

- If the correlation service and distribution service are configured in one data server, ensure that the data server is not running.
- If the correlation service and distribution service are configured in separate data servers, ensure that the data server where the correlation service is configured is not running.
- Ensure that field four of the service information entries for the correlation service, auto-journal agent, and distribution service is set to 1.
- Ensure that the service information entry for the auto-journal agent comes after the service information entry for the correlation service.

### Procedure

To start change capture with auto-journal agents:

- If the correlation service and distribution service are configured in one data server, start the data server. See “Starting data servers” on page 139.
- If the correlation service and distribution service are configured in separate data servers:
  1. Start the data server where the distribution service is configured. See “Starting data servers” on page 139.
  2. Start the data server where the correlation service is configured. See “Starting data servers” on page 139.

## Starting and stopping auto-journal agents when the data server where they are configured is running

You can start and stop an auto-journal agent without stopping and restarting the data server in which the agent is configured.

### Procedure

To start or stop an auto-journal agent:

- To start an auto-journal agent, issue the following command in an MTO interface:

```
F CACCS,START,SERVICE=name
```

where *name* is the unique name for the agent. This name is in field two of the service information entry for the agent.

- To stop an auto-journal agent, issue the following command in an MTO interface:

```
F CACCS,STOP,SERVICE=name
```

where *name* is the unique name for the agent. This name is in field two of the service information entry for the agent.

## Configuring change capture from CICS VSAM files by using log-reading change-capture agents

The steps for configuring change capture from VSAM files by using log-reading agents are a subset of the steps for configuring either Classic event publishing or Classic replication.

### Retention period and AUTODELETE of log data

You must retain CICS log data long enough that you can recover change data if your event publishing configuration goes into recovery mode. If the retention period and AUTODELETE specifications are met, CICS purges completed units of work on the system log file when CICS terminates.

Set retention period and AUTODELETE specifications of the system, user, and log of log streams so that the data remains in the log stream for the longest period of recovery that you want.

### Defining log-reading agents for CICS VSAM files

You create log-reading agents for CICS VSAM data sources as service information entries in member CACCSCF in the SCACCONF library.

### Restrictions

The service information entry for the log-reading agent must come after the service information entry for the correlation service that the agent will communicate with.

### About this task

You can create more than one log-reading agent. For each agent, create a separate service information entry. In the service information entry, provide a unique name in field two.

### Procedure

To define a log-reading agent for CICS VSAM files:

1. In member CACCSCF in the SCACCONF library, uncomment the service information entry for the agent (VSAMECA).
2. Provide values for the following fields:

**Field two**

Specify the name of agent. The default name is VSAMECA, but the name can be any unique value. If you want to run more than one log-reading change-capture agent, each agent must have a unique name.

**Field four**

Set the minimum number of tasks to 1 if you want the agent to start when you start the data server. If you want to manually start the agent with an operator command, set it to 0.

**Field eight**

This field is used when requesting information from the correlation service for either a restart Log Sequence Number (LSN) or a throttling confirmation. This value should be set large enough to allow the server time to receive throttled messages. A value of 5 minutes (5M) should be adequate in most cases.

**Field nine**

This field sets a polling frequency for recovery restart and rules confirmation messages. A value of 5 seconds (5S) is recommended. For the change-capture agent, this is the amount of time to wait when all of the log streams have reached end of file. Decreasing this value will reduce the latency in which changes are reflected to the correlation service at the expense of increased overhead.

3. After field 9, specify the following optional parameters in any order:

**APPLID=ID**

If multiple CICS tasks are writing to the same set of logs, specify the application ID (APPLID) of the specific CICS implementation from which you want to capture changes. If you do not specify an application ID, changes from all CICS implementations in the log streams are processed by the change-capture agent.

**COLDSTART**

This keyword causes the change-capture agent to ignore the restart point and to start processing at the position specified on the SERVICE INFO ENTRY for the change-capture agent. In normal operations, this keyword should not be required. If the change-capture agent is stopped and restarted, it will resume processing at a position to gather all the information from when it was last stopped.

**THROTTLE=n**

Specifies a throttling level for change messages when the agent is in recovery mode and must catch up with the current activity point in the log stream. Throttling prevents over-running the raw change data queue for the correlation service when the recovery point in the log stream is significantly behind the current activity point in the log. The value specified for this parameter indicates the maximum number of messages that will be in the raw data queue at any point in time. The default value is THROTTLE=1024.

**Parameters for starting change capture at specific times**

**STARTUP**

Start processing at the time CICS was last started.

**OLDEST**

Start processing at the beginning of the log stream file.

**YOUNGEST**

Start processing with the next record written.

**TIME** *mmddyyyy hhmmss*

Start processing at the indicated time. *mmddyyyy* is the month, day and year to start. *hhmmss* is the hour, minute, and second to start.

**Log streams**

List the log streams, separated by spaces, that the change-capture agent must read. Specify the system log stream and one user log stream. If you specify more than one user log stream, you must also specify the log of logs log stream. The log streams can be listed in any order.

**SERVER=name**

Specifies the name of the named correlation service that you want this change-capture agent to communicate with.

The following sample shows a service information entry for the CICS VSAM change-capture agent:

```
SERVICE INFO ENTRY = CACECA1V VSAMECA 2 1 1 1 4 5M 5S \
APPLID=CICSAPPL STARTUP CICSUID.CICSAPPL.DFHLOG \
CICSUID.CICSAPPL.DFHJ01 CICSUID.CICSAPPL.DFHJ02
CICSUID.CICSVR.DFHGLGLOG
```

**Starting change capture from CICS VSAM files with log-reading agents**

The steps for starting change capture depend on whether the correlation service and distribution service are running on the same LPAR or separate LPARs.

**Before you begin**

- If the correlation service and distribution service are configured in one data server, make sure that the data server is not running.
- If the correlation service and distribution service are configured in separate data servers, make sure that the data server where the correlation service is configured is not running.
- Make sure that field four of the service information entries for the correlation service, log-reading agent, and distribution service is set to 1.
- Make sure that the service information entry for the log-reading change-capture agent comes after the service information entry for the correlation service.

**Procedure**

To start change capture with log-reading change-capture agents:

- If the correlation service and distribution service are configured in one data server, start the data server. See “Starting data servers” on page 139.
- If the correlation service and distribution service are configured in separate data servers:
  1. Start the data server where the correlation service is configured. See “Starting data servers” on page 139.

2. Start the data server where the distribution service is configured. See “Starting data servers” on page 139.

## Starting and stopping log-reading agents when the data server where they are configured is running

You can start and stop a log-reading agent without stopping and restarting the data server in which the agent is configured. For example, you might want to change the startup parameters for the agent.

### Procedure

To start or stop a log-reading agent:

- To start a log-reading agent, issue the following command in an MTO interface:

```
F CACCS,START,SERVICE=task_name
```

where *task\_name* is the unique task name for the agent. This name is in field two of the service information entry for the agent.

When the agent starts, you will see several messages in the system log. The two primary messages are listed here. The first message indicates that the agent is active:

```
CACH105I CICS VSAM CAPTURE: Vv.r.m mmdyyyy READY
```

*Vv* Indicates the version.

*r* Indicates the release level.

*m* Indicates the modification level.

*mmdyyyy*

Shows the month, day, and year that the agent became active.

The second message indicates the time that processing began:

```
CACH106I START PROCESSING AT mm/dd/yyyy hh:mm:ss
```

- To stop a log-reading agent, issue the following command in an MTO interface:

```
F CACCS,STOP,SERVICE=task_name
```

where *task\_name* is the unique task name for the agent. This name is in field two of the service information entry for the agent.

When the agent terminates, this message appears in the system log:

```
CACH111I CICS VSAM CAPTURE: TERMINATING  
CAC00205I STOP PROCESSING COMPLETED SUCCESSFULLY
```

## Configuring access to CICS VSAM files for change capture

You must configure access to your CICS VSAM file from the data server.

### About this task

The data server makes no attempt to open the VSAM files where your source data is located.

### Procedure

To configure access to CICS VSAM files for change capture:

1. Configure VTAM resource definitions.
2. Configure CICS resource definitions.



## Configuring CICS resource definitions for change capture from CICS VSAM files

As part of configuring change capture from CICS VSAM files, you must configure CICS resource definitions.

### Procedure

To configure CICS resource definitions for change capture from CICS VSAM files:

1. Add ISC=YES to the CICS system initialization table (DFHSIT) definition or initialization overrides. Recycle CICS.
2. Copy the load module CACCIVS from the load library to the CICS user load library.
3. Install the IBM Language Environment® (LE) in CICS.
4. Add member CACCDEF in the SCACSAMP data set to the CICS transaction, program, connection, session, and file definitions that are required for change capture. Sample CICS transaction, program, connection, session, and file definitions are supplied in CACCDEF.
5. If you are going to capture changes with a change-capture agent that reads forward recovery logs and system logs, do the following steps:
  - a. Set RECOVERY to ALL in the file definition for each of your source VSAM files. This setting allows you to capture before and after images.
  - b. Set FWDRECOVLOG in the file definition for each of your source VSAM files. This parameter determines the journal to which the after images are written in support of forward recovery.
6. Define a log stream into the MVS logger subsystem. You can do so with the following JCL, after replacing CICSUID, CICSAPPL, and DFHJ01 with values specific to your configuration:

```
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DATA TYPE(LOGR) REPORT(YES)
DEFINE LOGSTREAM NAME(CICSUID.CICSAPPL.DFHJ01)
HLQ(xxxxxxx) MODEL(NO) STG_DATACLAS(xxxxxxx)
LOWOFFLOAD(0) HIGHOFFLOAD(80)
RETPD(n) AUTODELETE(YES)
DASDONLY(YES) DIAG(NO)
MAXBUFSIZE(65532)
/*
```

7. Run the CACCDEF job.
  - a. Update the job card with specifications for your site.
  - b. Update the STEPLIB with the correct CICS library.
  - c. Update the DFHCSD DD statement with the correct CSD file.
  - d. Add the following use journalmodel definition to the end of CACCDEF. You can modify &USERID, &APPLID, and &JNAME or leave them as they are.

```
DEFINE JOURNALMODEL (DFHJ01)
GROUP(CACVSAM)
DESCRIPTION (USER LOG STREAM)
JOURNALNAME(DFHJ01)
TYPE(MVS)
STREAMNAME (&USERID..&APPLID..&JNAME)
```
8. Install the new CICS resource definitions with the following CICS transaction:

```
CEDA INSTALL GROUP(CACVSAM)
```

9. Add the CACVSAM group to your start-up group with the following CICS transaction, where `xxxxxxx` is the name of the start-up group from your SIT table:

```
CEDA ADD GR(CACVSAM) LIST(xxxxxxx)
```

---

## Configuring change capture from VSAM files

The steps for configuring change capture from VSAM files that are not under the control of CICS Transaction Server are a subset of the steps for configuring either Classic event publishing or Classic replication.

### Before you begin

For Classic event publishing, perform all of the configuration steps that precede the implementation of NVA instances. See “Configuring Classic event publishing with one data server from VSAM files” on page 26 or “Configuring Classic event publishing with two data servers from VSAM files” on page 27 for guidance.

### Restrictions

Implementing an NVA instance involves customizing the instance and then loading and enabling the instance. An NVA instance can capture changes only to VSAM files that are on the same z/OS LPAR as itself.

### About this task

To implement an NVA instance, you do not need to change your native applications or make changes to the jobs or started tasks that launch those applications. If your site implements storage control exits, you might need to increase the region size of a job step if a native application is using most of the current region and an NVA instance does not have enough above-the-line storage available.

### Procedure

To implement one or more NVA instances:

1. Customize each NVA instance that you want to implement. See “Customizing NVA instances” on page 135.
2. Load and enable each NVA instance. See “Administering NVA instances with the NVA SVC Manager” on page 209.

You must administer each NVA instance through the NVA SVC Manager.

## NVA instances for capturing changes to VSAM files

NVA instances capture changes that are made by non-CICS application programs to VSAM files.

In this documentation, non-CICS application programs are collectively referred to as *native applications* because they update VSAM files when those files are not under control of a database or transaction manager. Native applications are z/OS jobs or started tasks that are usually part of a series or stream of batch jobs. A series of batch jobs might run daily, weekly, monthly, or at other intervals during a *batch window*, a period of time when a number of related VSAM files are taken

offline so that other applications cannot access and update the files. The batch jobs serially update the VSAM files as quickly as possible, and the files are then again made available to other applications.

An *NVA instance* is a combination of a VSAM JRNAD exit and SVC intercept routines that intercept SVC open and close requests. The routines and the exit are distributed together as load module CACNVA00 in the SCACLOAD data set. After the load module is loaded and installed in DLPA and then enabled with the NVA SVC Manager, the NVA instance intercepts SVC open and close requests for VSAM files that the instance is monitoring for changes. If the requests are for VSAM files that are not of interest, the NVA instance passes the requests on to the next SVC in the SVC chain.

However, if the requests are for VSAM files that the NVA instance is monitoring for changes, an NVA VSAM JRNAD exit routine is introduced into the processing flow so that NVA can be made aware of the changes to the VSAM file. These VSAM file changes will be sent to the correlation service for processing.

### **Support for multiple NVA instances**

You can install multiple NVA instances on a single z/OS LPAR. Multiple NVA instances are useful for testing. You might also find other uses for them. Each NVA instance must use a separate correlation service, metadata catalog, file table, and must be contained in a uniquely named load module.

## **Customizing NVA instances**

You must customize NVA instances before you install them on your system.

### **Restrictions**

Only one NVA instance can capture the changes that are made to any one VSAM file. Two NVA instances cannot capture changes for the same VSAM file.

### **About this task**

Customizing an NVA instance involves specifying which correlation service you want the NVA instance to communicate with, as well as setting optional filters.

You specify the correlation service and the filtering values by customizing an options module and then linking that options module to the NVA instance.

### **Procedure**

To customize an NVA instance:

1. For each NVA instance that you want to configure, copy the options module (member CACE1NVA in the SCACSAMP data set) and make the following edits in the copy:
  - a. Change the value of the SrvNam parameter to the name of the correlation service that you want the NVA instance to communicate with. The name must match the value of the NAME parameter in the service information entry for the correlation service.

If you do not specify a name, the default value NONAME is used. You can run only one unnamed correlation service on a single z/OS LPAR.

- b. **Optional:** Filter the changes that the NVA instance will capture. Use the `FilterUser` parameter to specify the user ID that will make the changes that the NVA instance will capture. Use the `FilterJobNm` parameter to specify the name of the job that will make the changes that the NVA instance will capture.

Using filters can limit the number of VSAM files that the NVA instance monitors for changes. Filtering can also reduce the number of messages that the NVA instance sends to the correlation service.

The value that you specify for a job or a user ID can be in one of the following formats:

**A name of one to eight characters.**

The name must exactly match the user ID or name of the job that is available when the NVA instance processes an open request.

You can use an asterisk (\*) as a wildcard character at the end of the name or as the only character in the name. Using the asterisk alone is equivalent to turning filtering off.

- @ You can specify this symbol for either the user ID or for the job name. This symbol specifies that you want to use the default filtering value from the options module that is linked to the NVA load module.

**Null value**

A null value turns off filtering. For example, you could specify `U=` to turn off filtering by username or `J=` to turn off filtering by job name.

- c. **Optional:** Use the `PgmExcl` parameter to exclude changes that are made by one or more applications. Separate each name with a comma. Do not remove the default values, which exclude changes that are made by CICS applications or IDCAMS file maintenance.
2. Use the high-level assembler (ASMA90) to assemble each options module.
    - a. Create the assembly JCL to concatenate the SCACMAC macro library as the last data set that is referenced on the SYSLIB DD statement.
    - b. **Optional:** Change the name of the output member to match the name of the options module.
    - c. Submit the job.
  3. Use the sample JCL in member CACNVALX in the SCACSAMP data set to link each options module to its corresponding NVA instance.
    - a. Change the high-level qualifier from CAC to the high-level qualifier where the Classic libraries are installed.
    - b. Verify that the CHANGE statement is as follows:

```
CHANGE name-of-options-module(CACE1OPT)
```

*name-of-options-module* must match the CSECT name that is specified in the assembled options module.
    - c. Change the OBJ DD to match the name of the data set where the assembled options module is located.
    - d. Change the LOAD DD to match the name of the NVA load module that you want to link to the options module.
    - e. If you plan to run more than one NVA instance, change the value of the NAME parameter. Each NVA instance must have a unique name for its load module. If you plan to use the default NVA instance, leave the default name.
    - f. Submit the job.

After the CACNVALX job successfully completes, you can install the NVA instance.



---

## Chapter 3. Administering event publishing

Administering Classic event publishing involves administering data servers, correlation services, distribution services, and change-capture agents.

---

### Administering data servers for Classic event publishing or Classic replication

Included with IBM WebSphere Classic Data Event Publisher for z/OS and IBM WebSphere Classic Replication Server for z/OS is a z/OS MTO (Master Terminal Operator) interface that you can use to monitor and control data server operations.

Data servers are designed to run continuously. With the MTO interface, you can issue commands to display the different services that are active within a data server and the amount of memory that is available. You can also issue commands to start and stop data servers.

You can run commands in the following format:

F *name\_of\_job,command*

- F is the abbreviation for the z/OS MODIFY command.
- *name\_of\_job* is the name of the started task to communicate with.
- *command* is the command to pass to the started task.

For example, the following commands is issued for the data server t9396840:

F CACDS.t9396840,display,users

### Starting data servers

When you start a data server, you start all of the services with uncommented service information entries in the data server's configuration file.

#### Procedure

To start a data server, perform either of the following steps:

- Issue a console command to start the data server JCL procedure:

S *procname*

where *procname* is the 1-8 character PROCLIB member name to be started. When you issue commands from the SDSF product, prefix all operator commands with the forward slash ( / ) character.

- Submit a batch job.

### Displaying information about data servers

The DISPLAY command outputs a formatted list of the selected information about a data server.

#### Procedure

To display information about a data server, issue one of the following commands with the MTO interface:

- To display current usage information on services, users, configurations, and the memory pool, issue this command:

`F name,DISPLAY,ALL`

*name* The name of the task or batch job started by the data server.

- To display a list of the configurations that are currently active in the data server, issue this command:

`F name,DISPLAY,CONFIGS`

*name* The name of the task or batch job started by the data server.

- To display the current use of the memory pool in the data server, issue this command:

`F name,DISPLAY,MEMORY`

*name* The name of the task or batch job started by the data server.

The following information is displayed about overall data server memory usage:

#### **TOTAL MEMORY**

The total size in kilobytes of the message pool that was allocated.

**USED** The amount of memory that is currently being used out of the message pool. This value is expressed in kilobytes followed by the percentage of the current message pool that is being used.

#### **MAX USED**

The maximum amount of the message pool that was ever used. This value is expressed in kilobytes followed by the percentage of the message pool that was ever used.

- To display a list of all running services in the data server, issue this command:

`F name,DISPLAY,SERVICES`

*name* The name of the task or batch job started by the data server.

When information is requested about the services that are active within a data server, a WTO display message is generated for each service that is active. For each service, the following information is displayed:

#### **SERVICE**

First eight characters of field 2 (service name) from the service information entry definition.

**TYPE** Load module name of the service - field 3 on the service information entry.

#### **TASKID**

TCB address (in decimal notation) of the service instanced that is displayed.

#### **TASKNAME**

Same as TYPE.

#### **STATUS**

One of the values that is displayed in Table 15.

**USER** The user ID that is currently being serviced. Generally, this value will be blank.

The following table lists the most common statuses:

*Table 15. States and descriptions*

Status	Description
QUIESCE	Unused.

Table 15. States and descriptions (continued)

Status	Description
READY	Idle and waiting for requests.
RECEIVING	Receiving a request.
RESPONDING	Sending a response.
STOP	Processing a STOP,ALL request.

## Displaying the values of single configuration parameters

You can use the GET command to display the value of a single configuration parameter.

### Before you begin

Issue the DISPLAY,CONFIG command to list the configuration parameters for the data server.

### Procedure

To display the value of a single configuration parameter:

```
F name,GET,NAME=configuration-name,ORD=ordinal-number
```

*name* The name of the data server started task or batch job.

#### NAME

The name of the configuration name of the configuration member from which to display the value.

**ORD** The number given for the configuration parameter by the DISPLAY,CONFIG command.

## Modifying configurations while data servers are running

You can modify the settings of the configuration parameters that a data server is using while that server is running. You can also use this command to create new service information entries for an active data server or enterprise server.

### Before you begin

Do not reset SERVICE INFO ENTRY values that have active instances. Issue the STOP,SERVICE command to stop all active instances first.

### Restrictions

Some of the new settings take effect immediately, and others take effect when the next query is processed for a particular user (in Classic federation) or the next time a service is started.

Any changes made to an active configuration remain only for the duration of the active configuration unless the FLUSH command is issued to commit the change permanently. You should use the FLUSH command only after your production system is stable because all comments (including commented-out configuration parameter definitions) are lost when a configuration member is dynamically saved to disk.

Configuration values that contain embedded spaces and special characters must be enclosed in either quotation marks ( " ") or apostrophes ( ' ').

Timing values must be suffixed with either M (minutes), S (seconds), or MS (milliseconds), for example, 5M, 5S, or 500MS.

### About this task

The command to modify a configuration takes this general form:

```
F name,SET,NAME=name,ORD=number,VALUE=value
```

*name* The name of the task or batch job started by the data server.

#### NAME

The name of the configuration.

**ORD** The number that is given for the configuration parameter by the DISPLAY,CONFIG command.

#### VALUE

Can have either of the following values:

- The value that you want to set for the configuration parameter at the specified line number.
- The service info entry that you want to create.

### Procedure

To modify configurations for running data servers or enterprise servers (in Classic federation), issue one of the following commands from the MTO interface:

- To modify a configuration parameter, issue this command:

```
F name,SET,NAME=name,ORD=number,VALUE=value
```

- To reset a configuration parameter to the default value, issue this command:

```
F name,SET,NAME=name,ORD=number,VALUE=NULL
```

- To create a new service info entry, issue this command with an ordinal value that is greater than the highest existing service info entry in the configuration:

```
F name,SET,NAME=name,ORD=number,VALUE='service_info_entry'
```

IBM recommends creating new service information entries by stopping the data server and editing the configuration file.

## Saving changes to configuration files

If you modified a configuration file dynamically, you can save your changes with the FLUSH command.

### Before you begin

Back up your initial configuration file. All comments in the initial configuration will be lost when you issue this command.

### Procedure

To save changes that you made dynamically to configuration files, issue the following command using the MTO interface:

```
F name,FLUSH,NAME=name
```

*name* The name of the task or batch job started by the data server.

## NAME

The name of the configuration member from which to display the value.

## Stopping data servers

Stopping a data server stops all of the services that are running within it.

### Procedure

To stop a data server, issue the following command in an MTO interface:

```
F name,STOP,ALL
```

*name* The name of the data server started task or batch job.

## Displaying log messages that are written to SYSTEM DD

You can dynamically modify the logger service (CACLOG) to display on the log messages from the services that are running in a data server. The logger service continues to write messages to the location that you specified in field 10 of the service information entry for that service.

### About this task

The output is formatted to text, but the output does not include descriptive text for error messages.

### Procedure

To display log messages, issue the following command in an MTO interface:

```
F name,MODIFY,LOG,OUTPUT=DISPLAY
```

*name* The name of the task or batch job started by the data server.

**Note:** If you change the name of the logger service from the default name LOG when you define the logger service, change LOG to the new name.

To stop displaying log messages, issue the following command in an MTO interface:

```
F name,MODIFY,LOG,OUTPUT=DEFAULT
```

*name* The name of the task or batch job started by the data server.

**Note:** If you change the name of the logger service from the default name LOG when you define the logger service, change LOG to the new name.

## Viewing log messages with the log print utility (CACPRTLG)

With the log print utility (CACPRTLG), you can format and display messages that are written to a log. You can also summarize the log messages or filter them.

### Before you begin

Set the following configuration parameters in the configuration file for the data server:

- MESSAGE POOL SIZE
- NL
- NL CAT

## Procedure

To view log messages:

1. Configure CACPRTLГ. See “Parameters for configuring the log print utility (CACPRTLГ).”
2. Create filters for the output. See “Filters for modifying output from the log print utility (CACPRTLГ)” on page 145.
3. Run CACPRTLГ. There are two ways to run this utility:
  - Run CACPRTLГ as a step in the same job used to run the data server. If you are using a temporary data set that is defined in the CACLOG DD statement, you must run CACPRTLГ as a subsequent step in the data server job.
  - Run CACPRTLГ as a separate job from the data server job or started task.

## Parameters for configuring the log print utility (CACPRTLГ)

You supply values to the PARM parameter of the CACPRTLГ EXEC statement to determine which information CACPRTLГ displays and where CACPRTLГ extracts the information from.

The following list shows the possible values for the PARM parameter.

### **SUMMARY=N**

Displays all of the messages that are in the log if you configured the logger service to write to the CACLOG DD statement.

### **SUMMARY=Y**

Displays a report about the contents of the log if you configured the logger service to write to the CACLOG DD statement.

### **SUMMARY=N STREAM=log\_stream**

Displays all of the messages that are in the log if you configured the logger service to write to a log stream. *log\_stream* must be a valid log stream that contains data that was written by the logger service. If you use the STREAM keyword, remove the CACLOG DD statement from the JCL for the log print utility.

### **SUMMARY=Y STREAM=log\_stream**

Displays a report about the content of the log, if you configured the logger service to write to an log stream. *log\_stream* must be a valid log stream that contains data that was written by the logger service. If you use the STREAM keyword, remove the CACLOG DD statement from the JCL for the log print utility.

### **SUMMARY=N STREAM=log\_stream PURGE**

Displays the contents of the log, if you configured the logger service to write to a log stream. Marks for deletion all of the log messages that are in the log stream and that are older than the value of the STARTTIME filter criterion for the log print utility. *log\_stream* must be a valid log stream that contains data that was written by the logger service. If you use the STREAM keyword, remove the CACLOG DD statement from the JCL for the log print utility.

### **SUMMARY=N PURGE STREAM=log\_stream**

This example is identical to the previous example, except for the order of the PURGE and STREAM keywords.

### **SUMMARY=Y STREAM=log\_stream PURGEALL**

Displays a report of the content of the log if you configured the logger service to write to a log stream. *log\_stream* must be a valid log stream that contains

data that was written by the logger service. If you use the STREAM keyword, remove the CACLOG DD statement from the JCL for the log print utility.

This example also marks for deletion all of the log messages that are in the log stream.

**SUMMARY=N STREAM=log\_stream PURGEALL**

Displays all of the messages that are in the log, if you configured the logger service to write to a log stream. Also, this option marks for deletion all of the log messages that are in the log stream.

*log\_stream* must be a valid log stream that contains data that was written by the logger service. If you use the STREAM keyword, remove the CACLOG DD statement from the JCL for the log print utility.

**Filters for modifying output from the log print utility (CACPRTLG)**

You can use SYSIN control cards to filter and display only a subset of the log messages. With these control cards, you can display messages for a specific time-frame, a specific task, a range of return codes, or any combination of the elements that are listed in the log summary report.

The format of the SYSIN filtering is exactly the same as the format of the summary report. So, you can run a summary report, find the criteria that would be relevant for you to filter on, then submit a SYSIN control card with those criteria. You can find sample JCL to run a summary report in member CACPRTLS in the SCACSAMP data set.

The following list presents the available filtering criteria. Although the criteria are presented in uppercase, you can specify them in mixed case because the log print utility will fold the characters into uppercase. All filter criteria must be followed by an equal sign and a value.

**STARTTIME='YYYY/MM/DD HH:MM:SS:thmi'**

Specifies the beginning of the duration of time that you want log information from. When you request the log information for a particular data server, you might find it helpful to review the JES output for the data server job to obtain the start time.

- *t* is tenths of a second
- *h* is hundredths of a second
- *m* is milliseconds
- *i* is ten-thousandths of a second

**STOPTIME='YYYY/MM/DD HH:MM:SS:thmi'**

Specifies the end of the duration of time that you want log information from.

- *t* is tenths of a second
- *h* is hundredths of a second
- *m* is milliseconds
- *i* is ten-thousandths of a second

**MINRC**

Specifies a numeric value that represents the lowest return code that you want to be reported.

**FILTERS**

Specifies tracing filters to use in the report. Use only in conjunction with IBM support.

**EXFILTERS**

Specifies tracing filters not to use in the report. Use only in conjunction with IBM support.

**MAXRC**

Specifies a numeric value that represents the highest return code that you want to be reported.

**TASKS**

Specifies a task number (service) to filter the log information by. Although this criterion is helpful if you are diagnosing a problem with a specific task, generally you should not use this criterion. If this criteria is used with multiple values, each separate line must start with the TASKS keyword, an equal sign, and the comma-delimited list of task numbers enclosed within parenthesis.

**NODES**

Specifies a specific node (address space or data server) for which the log print utility should return information. This value is a comma-delimited list enclosed with parentheses. Each line of node filters must be preceded by the NODES keyword and an equal sign.

**SPCRC**

Specifies a list of specific return code values for which the log print utility should return log records. Use only in conjunction with IBM support.

---

## Administering correlation services

You can generate reports on the activity of correlation services. You can also activate and deactivate publications (in Classic event publishing) and Q subscriptions (in Classic replication) by issuing commands to correlation services.

### Starting correlation services

You can start a correlation service without stopping and restarting the data server in which the correlation service is configured.

**About this task**

IBM recommends that you start correlation services by starting the data servers in which those services run. However, there might arise situations in which it is necessary to use this command. In such situations, IBM recommends consulting with IBM support.

**Procedure**

To start a correlation service:

Issue the following command in an MTO interface on the z/OS LPAR where the correlation service is configured:

```
F name,START,SERVICE=CACECA2
```

*name* The name of the started task or batch job to run on the data server.

### Generating reports on the activity of correlation services

After you start a correlation service, you can use an MTO command to display a report on its activities.

## About this task

This command displays a report on the activity of the correlation service and all change-capture agents that send change-capture data to the correlation service.

## Procedure

To generate a report on a correlation service:

Issue the following command in an MTO interface on the z/OS LPAR where the correlation service is running:

```
F name,CMD,service_name, 'REPORT'
```

*name* The name of the started task or batch job to run the data server.

*service\_name*

The name of the correlation service. This name is in field 10 of the service information entry for the correlation service.

If the correlation service is unnamed, use the value from field 2 of the service information entry for the correlation service.

The report lists the change-capture agents that are communicating with the correlation service. The change-capture agents can appear in the following orders:

- If the change-capture agents started while the correlation service was active, the change-capture agents are listed in the order that they were started.
- If the correlation service is warm-started and the change-capture agents are in recovery mode, the change-capture agents are listed in alphabetical order.
- If some change-capture agents are started and other change-capture agents are in recovery mode, the started change-capture agents are listed after the change-capture agents that are in recovery mode.

For each change-capture agent, the report displays the following information:

**Agent** The name of the change-capture agent.

### Processed

The number of committed transactions (UORs) that the correlation service received from the change-capture agent and processed.

### Sent to Distribution Service

The number of UORs that the correlation service sent to a distribution service.

### Pending

The number of UORs that need to be processed.

It is possible for the Confirmed count not to equal the Processed count even if the Pending count is zero. For example, the correlation service will not send a committed UOR to the distribution service if the changes in the UOR were to data that is not mapped to a logical table. Also, a record selection exit might block a committed UOR from being sent to the distribution service.

**State** One of the following values:

**Active** The change-capture agent is active.

### Shutdown

The change-capture agent notified the correlation service that it is no longer active.

**Recovery**

The change-capture agent is in recovery mode.

**Error** Either the change-capture agent reported an error or the correlation service reported an error concerning the change-capture agent.

After the list of change-capture agents, message CACG151I displays the total number of change-capture agents that are communicating with the correlation service.

At the end of the report, message CACG152I displays the following information:

**PENDINGQ**

Identifies the number of committed UORs that need to be sent to the distribution service.

**MSGQ**

Identifies the number of messages that are in the message store. The number includes data changes for any committed UORs that need to be sent to the publication service and all messages for any UORs that are in-flight.

**UORs** The number of UORs that were sent to the distribution service but that are not yet confirmed as received by the distribution service.

**MSGs**

The number of messages that were sent to the distribution service.

**Example**

```

CAC00200I CMD,XM1/IMSX/IMSX/200,REPORT
CACG150I CORRELATION SERVICE ACTIVITY REPORT
***** Transactions *****
Agent      Processed  Sent to Distribution Service Pending State
-----
VSAVSAMECA 0000002    0000002        0000000 Active

CACG151I END OF REPORT, AGENT TOTAL=1
CACG152I PENDINGQ(0) MSGQ(0) UNCONFIRMED(0)

```

## CSA reporting and maintenance utility

You can use the CSA reporting and maintenance utility to view and release the contents of the CSA storage that is used for communications between change-capture agents and the correlation service.

SCACSAMP member CACE2RPT is sample JCL that you can use to invoke the CSA reporting and maintenance utility. This utility is controlled with keywords in the PARM parameter of the EXEC statement. The following keywords are supported:

**REPORT | NOREPORT**

Identifies whether the CSA reporting and maintenance utility generates an Event Publisher Correlation Service Report that describes the CSA usage and state. The report is for either an unnamed correlation service or the correlation service that is identified by the SERVER command line parameter.

**RELEASE | NORELEASE**

Identifies whether to release (free) or retain the CSA that is being used by a named or unnamed correlation service. Under normal circumstances, you do not want to release the CSA storage. If you encounter error conditions

where a correlation service might not start and reports problems related to CSA, you can run the CSA reporting and maintenance utility and specify the RELEASE option to free the currently allocated CSA storage. When you release CSA storage, ensure that the correlation service that owns the storage is not running.

**SERVER=*name***

Identifies the name of the correlation service to be reported on, or the name of the correlation service that owns CSA storage that is being released. If this parameter is not specified, the CSA reporting and maintenance utility uses the CSA storage that is associated with the unnamed correlation service.

The name of the correlation service comes from the value of the NAME parameter in the service information entry for the correlation service.

**Example job to release CSA**

```
//CACE2RPT JOB (POK,999),MSGLEVEL=(1,1),MSGCLASS=H,
// CLASS=A,NOTIFY=&SYSUID
//CACE2RPT PROC CAC='CAC' INSTALLED HIGH LEVEL QUALIFIER
//*
//CACE2RPT EXEC PGM=CACECA2U,
// PARM='REPORT,RELEASE' TO GET REPORT AND RELEASE CSA
//* PARM='REPORT' TO GET A REPORT ONLY
//STEPLIB DD DISP=SHR,DSN=&CAC..V8R2M0.SCACLOAD
//CTTRANS DD DISP=SHR,DSN=&CAC..V8R2M0.SCACSASC
//SYSPRINT DD SYSOUT=*
//SYSTEM DD SYSOUT=*
//
PEND
//RUNE2RPT EXEC CACE2RPT
```

**Example of output for the REPORT parameter**

The following output was given for an active, unnamed correlation service:

```
+----- Event Publisher Correlation Service Report -----
|
| Correlation Service CSA E2CSGBLA, Lth=2588, Lock=x0000
| Service maximum count=1
| Allocated Service blocks=1
| Agents in Recovery Mode=3
|
| Server AREA XADECA2_WCA008CS, Lth=1024, Free=488 (Terminated)
| Server Name 'WCA008CS'
| Catalog Name 'UPTONG.V8R2M00.CATALOG,TCP/9.30.136.90/7026'
| Savant Filters( IDMS IMS ), ECB(x80000000)
|
| Agent 'IMS_AJW00850 ' CLASS(4) IN RECOVERY MODE
| AS OF 2005/10/26 14:49:52.17489f
| Agent 'IMS_WCA00801 ' CLASS(4) IN RECOVERY MODE
| AS OF 2005/10/14 15:47:11.00000f
| Agent 'IMS_WCA00802 ' CLASS(4) IN RECOVERY MODE
| AS OF 2005/10/14 16:12:26.00000f
|
+-----
```

**Correlation Service CSA**

This section of the report begins with the named token that is assigned to the CSA that WebSphere Classic Event Publisher for z/OS uses to track all of the correlation services that are on the image.

**Lth**     The length of the allocated CSA.

**Lock** The state of a lock word that is used to serialize access to the CSA.

**Service maximum count**

The number of CSA service control blocks that are active. This number represents the number of correlation services that are being tracked in CSA for the named correlation service. This number should always be 1.

**Allocated Service blocks**

The number of CSA service control blocks that are allocated. This number should always be 1.

**Agents in Recovery Mode**

The number of change-capture agents that are being tracked and that are in recovery mode.

**Server Area**

This section of the report begins with the named token of the CSA that is allocated to track the correlation service that is being reported on.

**Lth** The length of the allocated CSA.

**Free** The amount of the allocated length that is not being used.

**Keyword that indicates the status of the correlation service**

**Active** The correlation service is running.

**Initializing**

The correlation service is starting up.

**Terminated**

The correlation service is not running.

**Server Name**

The name of the correlation service, if a value is specified for the NAME parameter in the service information entry for the correlation service.

**Catalog Name**

The name of the data set of the metadata catalog that the correlation service is using. The data set name is extracted from the CACCAT DD statement.

**A protocol string**

The protocol string that is used by the correlation service to communicate with a distribution service.

**Savant Filters**

The types of data sources that the change data that is sent to the correlation service is coming from.

**ECB** The content of a change control block that is used by the change-capture agents to notify the correlation service that the state of a change-capture agent changed.

**List of change-capture agents that communicate with the correlation service and that are in recovery mode**

For each change-capture agent, the following information is provided:

**Agent** The name of the change-capture agent that is in recovery mode.

**CLASS**

The data source that the change-capture agent is sending change data from. The possible values are:

0	Adabas
3	CA-IDMS
4	IMS
6	VSAM

**AS OF**

The date and time in DB2 format that the change-capture agent went into recovery mode.

## Stopping correlation services

Correlation services are designed to run continuously, but sometimes you need to stop them.

### Before you begin

Stop all databases for which changes are sent to the correlation service that you are shutting down.

#### For Adabas, CA-IDMS, and IMS databases

You must shut down the databases from which you are capturing changes before you shut down the correlation service. The process that shuts down the databases must notify the correlation service that the databases are being shut down. If you do not shut down or halt the databases before shutting down the correlation service, the change-capture agents will go into recovery mode. If they do, see the recovery procedures for your data source.

#### For CICS VSAM files when you are capturing from forward recovery logs and system logs

The change-capture agents can tell when databases are being shut down. The change-capture agents notify the correlation service automatically.

#### For CICS VSAM files when you are capturing with file control agents

Disable the file control agent that communicates with the correlation service that you want to stop.

### About this task

When there is a system failure in IBM WebSphere Classic Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS, you must stop the correlation service. System failures might result from problems such as these:

- The correlation service loses contact with the distribution service.
- There is an out-of-memory condition.
- The correlation service report fails to write to an XM queue.

**Attention:** IBM recommends that you stop correlation services by stopping the data servers in which those services run. However, there might arise situations in which it is necessary to use this command. In such situations, IBM recommends consulting with IBM support.

### Procedure

To shut down a correlation service, issue one of these commands in an MTO interface on the z/OS LPAR where the correlation service is running:

- If you are running more than one correlation service, you can stop any of them by issuing this command:

```
P name_of_correlation_service
```

*name\_of\_correlation\_service*

The value of the NAME parameter in the service information entry for the correlation service.

- If you are running only one correlation service, you can stop it by issuing this command:

```
F name,STOP,SERVICE=CACECA2
```

*name* The name of the data server started task or batch job.

- You can stop any correlation service, regardless of whether it has a name, by issuing this command:

```
F name,STOP,TASKID=task ID
```

*name* The name of the data server started task or batch job.

## Considerations for recycling correlation services

Stopping and restarting a correlation service does not guarantee that the change-capture agents that are linked to the correlation service will be returned to active mode, even if you specify a cold start in the service information entry for the correlation service.

If you are not going to stop all change-capture agents that are linked to the correlation service before stopping and starting the correlation service, use the following checklist:

- For IMS databases, check the database server system log. Verify that the CACH001 message was issued to indicate that the change-capture agent was successfully installed into the database system. In most cases, message CACH001 is issued almost immediately after the IMS control region opens an IMS log file.
- Check the correlation service system log for any messages that are related to the change-capture agents. All messages that are related to the status of a change-capture agent are prefixed with CACG and include the agent name. The first message for a change-capture agent will usually be either CACG109I, which indicates that the agent is sending messages to the correlation service, or CACG111W, which indicates that the agent is in recovery mode. If the correlation service is cold started, the message CACG113I is issued.

## Reinitializing correlation services

If you add or delete publications or Q subscriptions, or if you modify the content of a metadata catalog, you can reinitialize your correlation service so that it recognizes the changes without having to stop and restart the correlation service.

### Before you begin

### Restrictions

### Procedure

To reinitialize a correlation service:

Issue the following command in an MTO interface on the z/OS LPAR where the correlation service is running:

```
F name,CMD,service_name,'REINIT'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the correlation service. This name is in field 10 of the service information entry for the correlation service.

If the correlation service is unnamed, use the value from field 2 of the service information entry for the correlation service.

## Activating publications and Q subscriptions

Before you can start publishing data (in Classic event publishing) or replicating data (in Classic replication) from source tables or views, the publications and Q subscriptions that use those source tables or views must be active.

### Procedure

To activate a publication (for Classic event publishing) or a Q subscription (for Classic replication), follow either of these steps:

Issue the following command in an MTO interface on the z/OS LPAR where the correlation service is running:

```
F name,CMD,service_name,'activate,subname=name_of_publication_or_Q_subscription'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the correlation service. This name is in field 10 of the service information entry for the correlation service.

If the correlation service is unnamed, use the value from field 2 of the service information entry for the correlation service.

*name\_of\_publication\_or\_Q\_subscription*

The name of the publication or Q subscription that you want to activate.

## Deactivating publications and Q subscriptions

You deactivate a publication in Classic event publishing to stop publishing changes for the publication. You deactivate a Q subscription in Classic replication to stop replicating changes for the Q subscription.

### Before you begin

The publication or Q subscription that you want to deactivate must currently be active.

### About this task

The deactivation occurs at the next transaction boundary.

For Classic replication, this command is the equivalent of the STOP QSUB command in ASNCLP.

### Procedure

To deactivate a publication (in Classic event publishing) or a Q subscription (in Classic replication), follow either of these steps:

Issue the following command in an MTO interface on the z/OS LPAR where the correlation service is running:

```
F name,CMD,service_name,'deactivate,subname=name_of_publication_or_Q_subscription'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the correlation service. This name is in field 10 of the service information entry for the correlation service.

If the correlation service is unnamed, use the value from field 2 of the service information entry for the correlation service.

*name\_of\_publication\_or\_Q\_subscription*

The name of the publication or Q subscription that you want to deactivate.

## Deactivating all of the publications or Q subscriptions that use a send queue

You can deactivate all of the publications (in Classic event publishing) or Q subscriptions (in Classic replication) that use the same send queue by deactivating that send queue.

### Before you begin

You must know which correlation service is processing the change data for the sources of the publications or Q subscriptions that are using the send queue.

### Procedure

To deactivate a send queue:

Issue the following command in an MTO interface on the z/OS LPAR where the correlation service is running:

```
F name,CMD,service_name,'deactivate,sendqueue=name_of_send_queue'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the correlation service. This name is in field 10 of the service information entry for the correlation service.

If the correlation service is unnamed, use the value from field 2 of the service information entry for the correlation service.

*name\_of\_send\_queue*

The name of the send queue that you want to deactivate.

## Reinitializing publications and Q subscriptions

Use the REINIT command if you changed a publication (in Classic event publishing) or a Q subscription (in Classic replication) and want the changes to take effect.

### Before you begin

You must know which correlation service is processing the change data for the source of the publication or Q subscription.

### Restrictions

## Procedure

To reinitialize a publication (in Classic event publishing) or a Q subscription (in Classic replication):

Issue the following command in an MTO interface on the z/OS LPAR where the correlation service is running:

```
F name,CMD,service_name,'reinit,subname=name_of_publication_or_Q_subscription'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the correlation service. This name is in field 10 of the service information entry for the correlation service.

If the correlation service is unnamed, use the value from field 2 of the service information entry for the correlation service.

*name\_of\_publication\_or\_Q\_subscription*

The name of the publication or Q subscription that you want to reinitialize.

---

## Monitoring change capture

You can issue commands to a distribution service to display reports about the status of your Classic event publishing or Classic replication environment.

To display the commands that are available for distribution services, in an MTO interface type CMD,DS,HELP.

## Starting distribution services

You can start a distribution service without stopping and restarting the data server in which the correlation service is configured.

### About this task

If you stop and then restart a distribution service, you must also stop and restart the correlation services that communicate with the distribution service.

**Attention:** Stopping and restarting a distribution service can cause your environment to go into recovery mode. IBM recommends that you start distribution services by starting the data servers in which those services run. However, there might arise situations in which it is necessary to use this command. In such situations, IBM recommends consulting with IBM support.

## Procedure

To start a distribution service:

Issue the following command in an MTO interface on the z/OS LPAR where the distribution service is configured:

```
F name,START,SERVICE=service_name
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the distribution service. This name is in field two of the service information entry of the distribution service that you want to start.

## Displaying information about the configuration of publications and Q subscriptions

After you start a distribution service, you can use MTO commands to display information about how publications (in Classic event publishing) and Q subscriptions (in Classic replication) are configured. The same commands provide metrics for tracking the quantity of data that is captured for the publications and Q subscriptions.

### Procedure

To display information about the configuration of publications and Q subscriptions, use either of the following commands in an MTO interface on the z/OS LPAR where the distribution service is running:

- To display information about the configuration of all publications and Q subscriptions, issue this command:

```
F name,CMD,service_name,'DISPLAY,SUBS'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the distribution service. This name is in field 2 of the service information entry for the distribution service.

- To display information about the configuration of a single publication or subscription, issue this command:

```
F name,CMD,service_name,"DISPLAY,SUB,NAME=name_of_publication_or_Q_subscription"
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the distribution service. This name is in field 2 of the service information entry for the distribution service.

*name\_of\_publication\_or\_Q\_subscription*

The name of the publication or Q subscription.

For each publication or Q subscription, the following information is displayed:

#### **SUBNAME**

The name of the publication or Q subscription.

#### **SUB ID**

The ID of the publication or Q subscription. This ID is generated by the distribution service and is used in administrative messages to identify the publication or Q subscription.

#### **SENDQ**

The name of the send queue that is used by the publication or subscription.

#### **SUBTYPE**

The format of messages that are written for the publication or Q subscription.

**CMF** Compact message format, which is used in Classic replication.

**DLM** Delimited format, which is used in Classic event publishing.

**XML** XML, which is used in Classic event publishing.

#### **TOPIC**

## STATE

## INSERTS

The number of inserts that occurred in the source table for the publication or Q subscription.

## UPDATES

The number of updates that occurred in the source table for the publication or Q subscription.

## DELETES

The number of deletes that occurred in the source table for the publication or Q subscription.

## Displaying information about send queues

You can use MTO commands to display information about the send queues that your publications and Q subscriptions are using.

### Before you begin

- The distribution service that is processing change data for your publications and Q subscriptions must be running.
- The send queues that you want to use for event publishing or Classic replication must already exist.

### Procedure

To display information about send queues, issue either of the following commands in an MTO interface on the z/OS LPAR where the distribution service is running:

- To display information about all of the send queues that are being used by the publications and Q subscriptions that a distribution service is processing change data for, issue this command:

```
F name,CMD,service_name, 'DISPLAY,SENDQUEUES'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the distribution service. This name is in field 2 of the service information entry for the distribution service.

- To display information about only one of the send queues that are being used by the publications and subscriptions that a distribution service is processing change data for, issue this command:

```
F name,CMD,service_name, 'DISPLAY,SENDQUEUE,NAME=name_of_send_queue'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the distribution service. This name is in field 2 of the service information entry for the distribution service.

*name\_of\_send\_queue*

The name of the send queue that you want to display information for.

For each send queue, the following information is displayed:

### SENDQUEUE NAME

The name of the publishing queue map (in Classic event publishing) or replication queue map (in Classic replication).

**SENDQ**

The name of the send queue.

**MAX SIZE**

The size (in kilobytes) of the largest message that was placed on the send queue.

**STATE**

The state of the send queue.

**A** Indicates that the send queue is active. In this state, the send queue is available for event publishing or Classic replication, even if the publications or Q subscriptions that are associated with the send queue are not active.

**I** Indicates that the send queue is inactive.

**STATE TIME**

The time at which the state of the send queue was last reported.

**# of TRAN MESSAGES**

The number of transaction-level messages that were sent on the send queue by all publications or Q subscriptions that are using the send queue.

**# OF CONTROL MESSAGES**

In Classic replication, the number of messages that were sent to the Q Apply program.

## Generating reports on the activity of distribution services and publication services

After you start a distribution service, you can use an MTO command to display a report on its activities and the activities of the publication services that the distribution service started.

**Procedure**

To generate a report on a distribution service:

Issue the following command using an MTO interface on the z/OS LPAR where the distribution service is running:

```
F name,CMD,service_name, 'REPORT'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the distribution service. This name is in field 2 of the service information entry for the distribution service.

**Control table information****Number of subscriptions received**

The number of publications and Q subscriptions that the distribution service is tracking.

**Number of tables**

The number of mapped tables and views that are sent from the correlation service to the distribution service. This number can indicate errors in the metadata exchange between the correlation service and the distribution service.

**Number of send queues**

The number of send queues that are being used by the publications and Q subscriptions that the distribution service is tracking.

**Information for each change-capture agent**

**Agent** The name of the change-capture agent.

**Inserts**

The number of inserts that were sent by the change-capture agent.

**Updates**

The number of updates that were sent by the change-capture agent.

**Deletes**

The number of deletes that were sent by the change-capture agent.

**Staged**

The number of changes that the distribution service received from the change-capture agent and is currently staging while awaiting a commit from the change-capture agent.

**Sent**

The number of changes that the distribution service received from the change-capture agent and sent to the publication service.

**State**

The state of the change-capture agent. The two possible states are "Active" and "Recovery."

**CPU Time (MS)**

The cumulative amount of CPU time that the distribution service consumed to process changes from the change-capture agent.

**Information from publication services****SUB NAME**

The name of the publication or Q subscription.

**TYPE**

Indicates whether the object is a publication or a Q subscription.

**XML**

Indicates that the object is a publication.

**CMF**

Is an acronym for *Compact Message Format* and indicates that the object is a Q subscription.

**STAT**

The state of the publication or Q subscription.

**TABLE NAME**

The name of the source table for the publication or Q subscription.

**CHANGES**

The number of changes (including inserts, updates, and deletes) that were published for the publication or replicated for the Q subscription.

**INSERTS**

The number of inserts that were published for the publication or replicated for the Q subscription.

**UPDATES**

The number of updates that were published for the publication or replicated for the Q subscription.

**DELETES**

The number of deletes that were published for the publication or replicated for the Q subscription.

## Displaying the WebSphere MQ configuration for change capture

You can use an MTO command to display the names of the WebSphere MQ queue manager and restart queue that are used for communications by the change capture components. The command also shows the heartbeat interval and (for Classic replication) the name of the administration queue.

### Before you begin

- The distribution service that is processing change data for your publications and subscriptions must be running.
- The WebSphere MQ objects that you want to use for event publishing or Classic replication must already exist.

### Procedure

To display information about the WebSphere MQ configuration for change capture:

Issue the following command in an MTO interface on the z/OS LPAR where the distribution service is running:

```
F name,CMD,service_name,'DISPLAY,CAPPARMS'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the distribution service. This name is in field 2 of the service information entry for the distribution service.

The following information is displayed:

#### QUEUE MANAGER

The name of the WebSphere MQ queue manager that is managing the administrative queue and the restart queue (in-doubt resolution queue).

#### ADMINQ

The name of the administrative queue that is used by the Q Apply program to send messages to the change-capture components. This queue is used only in Classic replication.

#### RESTARTQ

The name of the queue that stores the restart information that is required when a correlation service warm starts or when a change-capture agent or correlation service goes into recovery mode.

#### HEARTBEAT INTERVAL

During periods when there are no changes to source data, the interval at which the Q Apply program receives a message to indicate that the change capture components are still active. Setting the heartbeat interval to 0 suppresses these messages.

## Displaying metrics about the correlation services that are connected to a distribution service

You can use an MTO command to display metrics about the progress of the correlation services that are connected to a distribution service.

### Before you begin

The distribution service that is processing the change data for your publications (in Classic event publishing) or Q subscriptions (in Classic replication) must be running.

### Procedure

To display these metrics:

Issue the following command in an MTO interface on the z/OS LPAR where the distribution service is running:

```
F name,CMD,service_name, 'DISPLAY,CS'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the distribution service. This name is in field 2 of the service information entry for the distribution service.

The following information is displayed:

#### CSNAME

The name of the correlation service. This value comes from the NAME parameter in the service information entry for the correlation service. If the correlation service is unnamed, the value (NONAME) appears in this column.

#### HOSTNAME

The host name of the z/OS image where the correlation service is running.

#### USERID

This field is reserved for future use.

#### INSERTS

The number of inserts that were in the change data that the correlation service sent to the distribution service.

#### UPDATES

The number of updates that were in the change data that the correlation service sent to the distribution service.

#### DELETES

The number of deletes that were in the change data that the correlation service sent to the distribution service.

#### RESTART TOKEN

The restart token from the last commit that was received from the correlation service.

## Displaying metrics about publications in Classic event publishing

You can use an MTO command to display information about publications that are formatted in XML or the delimited format.

### Before you begin

The distribution service that is processing change data for your publications must be running.

### Procedure

To display metrics about publications:

Issue the following command in an MTO interface on the z/OS LPAR where the distribution service is running:

```
F name,CMD,service_name,'DISPLAY,REPORT,XML'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the distribution service. This name is in field 2 of the service information entry for the distribution service.

The following information is displayed:

**SUB NAME**

The name of the publication.

**TYPE**

**XML** Indicates that the messages in the publication use the XML format.

**DLM** Indicates that the messages in the publication use the delimited format.

**STAT** The state of the publication.

**TABLE NAME**

The name of the source table for the publication.

**CHANGES**

The number of changes (including inserts, updates, and deletes) that were published for the publication.

**INSERTS**

The number of inserts that were published for the publication.

**UPDATES**

The number of updates that were published for the publication.

**DELETES**

The number of deletes that were published for the publication.

**COMMITTS**

The number of commits that were published for the publication.

## Displaying information about publication services

You can use an MTO command to display the number of inserts, updates, and deletes that your publication services have processed.

### Before you begin

The distribution service that is processing the change data for your publications or Q subscriptions must be running.

### About this task

In most cases, a distribution service starts only one publication service. This report can be useful for troubleshooting your configuration. You can see whether the publication service processed the expected number of changes.

### Procedure

To display information about publication services:

Issue the following command in an MTO interface on the z/OS LPAR where the distribution service is running: This is the distribution service that started the publication services.

```
F name,CMD,service_name,'REPORT,THREADS'
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the distribution service. This name is in field 2 of the service information entry for the distribution service.

The following information is displayed:

#### **TASKID**

The ID of the thread of the running publication service.

**TYPE** The format of message that the publication service wrote to a send queue.

**CMF** Compact message format, which is used in Classic replication.

**DLM** Delimited format, which is used in Classic event publishing.

**XML** XML, which is used in Classic event publishing.

**CPU** The cumulative amount of CPU time that the publication service consumed.

#### **CHANGES**

The cumulative number of changes that the publication service has converted to messages of the given format.

#### **INSERTS**

The number of inserts that the publication service has converted to messages of the given format.

#### **UPDATES**

The number of updates that the publication service has converted to messages of the given format.

#### **DELETES**

The number of deletes that the publication service has converted to messages of the given format.

## **Stopping distribution services**

Distribution services are designed to run continuously, but sometimes you need to stop them.

### **About this task**

IBM recommends that you stop distribution services by stopping the data servers in which those services run. However, there might arise situations in which it is necessary to use this command. In such situations, IBM recommends consulting with IBM support.

### **Procedure**

To shut down a distribution service, issue one of these commands in the MTO interface on the z/OS LPAR where the distribution service is running:

- You can stop the distribution service by using its name:

```
F name,STOP,SERVICE=service_name
```

*name* The name of the data server started task or batch job.

*service\_name*

The name of the distribution service. This name is in field 2 of the service information entry for the distribution service.

- You can stop the distribution service by using its task ID:

```
F name,STOP,TASKID=task_ID
```

*name* The name of the started task or batch job that runs on the data server.

---

## Recovering change data from Adabas databases

You can recover change data from Adabas databases either manually or by setting up automatic recovery.

### Recovering change data automatically from Adabas databases

You can set up an automatic procedure that recovers change data from PLCOPY files when one or more errors occur that cause the change-capture agent to stop capturing data.

#### About this task

Setting up automatic recovery is optional. You can configure WebSphere Classic change publishing without making use of this feature. You can set up this feature at any time.

#### Procedure

To set up automatic recovery of change data:

1. Add JCL to your automated protection log ADARES PLCOPY procedures to run the recovery agent with the MONITOR parameter.

In an EXEC statement, execute CACEC1AR, using MONITOR for the value of the PARM parameter and using a value for the RESTARTWAIT keyword.

The recovery agent checks the state of recovery and checks for a running correlation service. If the change-capture agent is in recovery mode and a correlation service is active and ready to receive messages, the recovery agent ends with a return code of 0; otherwise, it ends with a return code of 4.

2. Add JCL to your PLCOPY procedure to start a recovery agent, if the change-capture agent is in recovery mode, at the conclusion of the protection log copy processing.

In an EXEC statement, execute CACEC1AR, using PLCOPY for the value of the PARM parameter. You can use additional keywords for the PARM parameter to control the behavior of the recovery agent. See “Options for controlling recovery agents for Adabas databases” on page 165.

After the automated procedure starts, the recovery agent reads through the available PLCOPY files and looks for the recovery starting point.

For example, the two JCL statements that you add might look like these statements:

```
//CHECK EXEC PGM=CACEC1AR,PARM='MONITOR,RESTARTWAIT=0S'  
//...  
//RECOVER EXEC PGM=CACEC1AR,PARM='PLCOPY,...',COND=(0,NE,CHECK)  
//...
```

## Recovering change data manually from Adabas databases

When one or more errors cause a change-capture agent to stop capturing data, you can recover data from PLCOPY files by manually running the recovery agent.

### About this task

The recovery agent searches for the restart point in any ADARES PLCOPY files that you specify. When you specify more than one PLCOPY file, the recovery agent searches the files in the specified sequence and verifies that the files are in chronological order. After the specified starting point is found, change data for all candidate database modifications is sent to the correlation service until all the PLCOPY file data is processed.

When the change-capture agent is in recovery mode, the most effective means of recovery is to run the recovery agent each time that the dual or multiple protection log is switched and the ADARES PLCOPY operation is done. This method allows the recovery agent to process the recovery data as quickly as it becomes available. You can run the agent iteratively until the database can be stopped and all changed data can be processed. You can then return the change-capture agent to active mode.

### Procedure

To run the recovery agent manually:

1. Switch the dual or multiple protection logs by using the Adabas operator command FEOFPL.
2. Run the ADARES PLCOPY utility.
3. Execute a batch job to start CACEC1AR, using PLCOPY for the value of the PARM parameter. You can find a sample in the member CACADR1 in the sample library. You can use additional keywords for the PARM parameter to control the behavior of the recovery agent. See "Options for controlling recovery agents for Adabas databases."
4. Repeat steps 1 through 3 until you can either quiesce the database or quiesce activity on the files from which you are capturing changes.
5. After all updates to the files from which you are capturing changes are committed, repeat steps 1 through 3 one last time so that the recovery agent can catch up in the PLCOPY files.

After the recovery agent reaches the oldest, non-committed transaction in the PLCOPY files, you can activate the change-capture agent.

## Options for controlling recovery agents for Adabas databases

The values of the PARM parameter of the JCL EXEC statement control the behavior of the recovery agent.

### Format

The format of the parameter for running PLCOPY is as follows:

```
PARM='PLCOPY,Addtn|keyword=vvvv,...'
```

The format of the parameter for generating a report is as follows:

```
PARM='REPORT,Addtn|keyword=vvvv,...'
```

The format of the parameter for checking whether a change-capture agent is in recovery mode is as follows:

PARM= 'MONITOR,Addtnlkeyword=vvvv,...'

## Values

### PLCOPYY

Initiates recovery from one or more Adabas sequential protection log files that are written by the Adabas ADARES utility.

### REPORT

Requests a report of the current recovery sequence and timestamp for the identified agent. The AGENT keyword is required with the REPORT option.

The recovery sequence identifies the transaction at which the recovery agent begins processing and consists of the Adabas Global Unique Identifier. The timestamp allows the recovery agent to find the block that should contain the restart point that is identified by the recovery sequence. The recovery agent then looks in that block for the start of that transaction. If the recovery agent finds the start of the transaction, the recovery agent starts sending data to the correlation service. If the recovery agent does not find the start of the transaction, the recovery agent returns an error.

### MONITOR

Allows the recovery agent to check for the recovery state. This function enables automatic starting of recovery when a change-capture agent fails.

### Addtnlkeyword=vvvv

One or more additional keyword parameters, which can or must be included in the parameter string to control recovery processing.

The following table lists all of the additional keyword parameters and indicates in which functions they are used.

Table 16. Use of the additional keyword parameters

Additional keyword parameters	Default value	PLCOPYY function	REPORT function	MONITOR function
ACTIVATE	N	Optional	Optional	N/A
AGENT	None	Required	Required	Required
RESTARTWAIT	0	N/A	N/A	Optional
SERVER	unnamed	Optional	Optional	Optional
THROTTLE	512	Optional	N/A	N/A
TIMEOUT	5M	Optional	Optional	Optional

### ACTIVATE={Y | N}

Specifies whether to enable the change-capture agent after the recovery agent finishes recovering change data. Specifying Y informs the correlation service that all recovery messages were sent and that the change-capture agent can begin forwarding messages again after the final recovery message is processed.

This parameter is optional on both the PLCOPY and REPORT functions. The default value is N, which leaves the change-capture agent in recovery mode at the completion of processing. This parameter is not used by the MONITOR function.

**AGENT=agentname**

Identifies the name of the change-capture agent. This name must be ADABAS\_#####, where ##### is the database ID number. This parameter is required on the REPORT and MONITOR functions. For the PLCOPY function, the DBID is in the protection log record and is used to automatically determine the agent name.

**RESTARTWAIT=n{H | M | S}**

Defines the wait interval in hours, minutes, or seconds for the MONITOR function. The default unit is S. Specifying 0 indicates that the process is to execute one time only. Specifying any other value indicates a delay time and causes the MONITOR function to execute continuously.

This parameter is optional on the MONITOR function. The default value is 0. If this parameter is not specified or set to 0, only one check is performed. This parameter is not used by the PLCOPY or REPORT function.

**SERVER=servername**

Identifies a correlation service by a specific name. The correlation service with the specified name processes the request from the recovery agent.

This parameter is optional on the PLCOPY, REPORT, and MONITOR functions. If this parameter is not specified, the default unnamed correlation service processes the request of the recovery agent.

**THROTTLE=#####**

Prevents the recovery agent from overrunning the correlation service with change messages. The throttle value defines the maximum number of messages to be queued to the correlation service at any time. The throttle ensures available space in the message queue for any change-capture agents that communicate with the correlation service. Multiple change-capture agents for multiple Adabas databases can communicate with a single correlation service. Specifying a value of 0 disables throttling of messages.

This parameter is optional on the PLCOPY function. If this parameter is not specified, the default value of 512 is assumed. This parameter is not used by the REPORT or MONITOR function.

**TIMEOUT=nm{H | M | S}**

Defines a timeout value for controlling communications between the recovery agent and the correlation service. The default unit is S. If the correlation service is unable to respond within the specified timeout value, the recovery agent terminates with an error.

In the MONITOR function, the timeout value is used to control the length of time that monitoring will continue without any contact with the active change-capture agent. The default value is 5M.

## Examples

The following examples show parameter strings that can be used for various operations:

- To generate a report of the current recovery sequence and timestamp:  
PARM='REPORT,AGENT=ADABAS\_00001'
- To check if a change-capture agent is in recovery state:

```
PARM='MONITOR,AGENT=ADABAS_00001'
```

- To execute a recovery and return to active mode upon successful completion:

```
PARM='PLCOPY,ACTIVATE=Y'
```

## Moving from recovery mode to active mode for Adabas databases

After you recover change data and are fully caught up in the PLCOPY files, you can return the change-capture agent to active mode.

### Before you begin

Do not move the change-capture agent from recovery mode to active mode until all changes that are recorded in Adabas are successfully processed by the recovery agent. The recovery agent can be run periodically, whenever ADARES PLCOPY files are available for processing. Continue to run the recovery agent until changes in Adabas can be halted and all changed data is processed. You must complete the recovery process to avoid losing changes.

### Procedure

There are three methods for moving from recovery mode to active mode:

- When you run the recovery agent for the last time, specify `ACTIVATE=Y` in the input parameters.
  - If you are running the recovery agent manually, specify `ACTIVATE=Y` the last time that you run the recovery agent with `PLCOPY` for the `PARM` parameter.
  - If you are running the recovery agent automatically, specify `ACTIVATE=Y` to execute the recovery agent with `REPORT` for the `PARM` parameter.
- In the procedure that you use to allow database changes to resume, add a new job step. The new job step executes the recovery agent with the `REPORT` parameter and specifying `ACTIVATE=Y`.
- Execute the recovery agent as a batch job, using the `REPORT` parameter and `ACTIVATE=Y`.

After the recovery agent completes successfully, the change-capture agent is returned to active mode.

---

## Recovering data from CA-IDMS databases

When a change-capture agent enters recovery mode, you must run the recovery agent to recover change data from the time that the change-capture agent stopped running to the time that the change-capture agent is started again.

### Recovery mode for CA-IDMS databases

If a change-capture agent or the correlation service encounters an error while processing change data, the change-capture agent goes into recovery mode. When the change-capture agent is in recovery mode, you need to recover the change data that your applications continue to generate. You can recover this change data with the recovery agent.

The recovery agent can read the change data from the moment that the change-capture agent stopped until the most recent records in your journal files.

When a change-capture agent becomes inactive, a starting point is written in the correlation service's recovery data set. The recovery agent begins reading journal records from the record that was created at the restart point and continues reading forward in time through journal files. When the recovery agent reads the journal records, it sends messages that contain this data to the correlation service.

The recovery agent can recover change data from the following sources:

- When you run a recovery agent in CV mode, you can recover data from currently active CA-IDMS Central Version disk journal files.
- When you run a recovery agent in LOCAL mode, you can recover data from CA-IDMS local client journal files that are associated with a local mode batch job.
- When you run a recovery agent in ARCHIVE mode, you can recover data from CA-IDMS Archive journal files on tape or disk.

The most effective means of recovery is to run the CA-IDMS recovery agent using the CA-IDMS Central Version disk journal files. The recovery agent can run continuously while the Central Version is active and can forward changes made to the CA-IDMS database with minimal latency.

## Recovery agents for CA-IDMS

The load module CACEC1DR is the CA-IDMS recovery agent. This module is used to perform recovery operations when the change-capture agent goes into recovery mode due to an error in processing encountered by either the change-capture agent or the related correlation service.

The recovery agent is a program that you can start manually or initiate by integrating into an automatic procedure, such as one that archives journals.

The recovery agent monitors the online central version journals J1JRNL through JnJRNL, where *n* is any number. After you start it, the CA-IDMS recovery agent receives an initial restart position in the journal from the correlation service. The agent then searches the allocated journals for the restart position and sends information from that point forward to the correlation server. All data from the restart point to the current time must be available in the uncondensed portion of the journal. The recovery agent does not process condensed journal segments.

You can recover changes from central version disk journals if one or more of the disk journals were archived to tape. Recovering these changes is possible only if all of the journal records since the last restart point are available in journal segments that have not been offloaded. The recovery agent attempts to catch up with the active central version and then continues to monitor the active journal for new database changes.

Because the recovery agent receives the initial restart position from the correlation service, if the correlation service is not running, the recovery agent terminates.

In addition to recovering lost changes, recovery agents can perform the following tasks:

- Monitor the state of the change-capture agent and the correlation service. Based on the state of the change-capture agent and correlation service, the recovery agent can be used together with the CA-IDMS journal offload process to determine when journals can be archived. This collective process then provides a

mechanism to maintain a fixed number of unarchived central version journals so that the recovery agent can recover changes quickly.

- Monitor a CA-IDMS central version to determine whether a change-capture agent is in recovery mode.
- Force a change-capture agent back into active mode.

## Recovery point files for CA-IDMS databases

The change-capture agent stores an initial recovery point for the correlation service in an 80-byte file when the CA-IDMS central version is started without an active correlation service.

This recovery point is retained until a correlation service is started and receives the initial recovery information from the change-capture agent. The correlation service constantly updates the restart point each time that the distribution service confirms receipt of committed transactions. Saving an initial recovery point ensures that change data will not be lost even if the CA-IDMS central version is run multiple times without starting a correlation service.

The attributes of the recovery point file are:

```
LRECL=80,DSORG=PS,RECFM=FB
```

The blocksize of the file does not matter, and you can allocate the file to use one track because the file will never contain more than a single record.

If you stop and restart CA-IDMS after the CACH002A message (which is CACH002A EVENT PUBLISHER SERVICE '(service name)' NOT FOUND BY CHANGE-CAPTURE AGENT 'IDMS\_nnn ', REPLY 'R' OR 'A' RECOVER/ACTIVE), you will see the following message:

```
CACH003A RECOVERY RECORD EXISTS FOR AGENT '.....', REPLY  
'R' OR 'A' RECOVERY/ACTIVE
```

This message indicates that a recovery point from a prior CA-IDMS start exists and that you need to recover change data. This message will appear each time you start CA-IDMS until a correlation service is available to receive the recovery point from CA-IDMS.

This message will be displayed even if the operator starts the correlation service prior to restarting CA-IDMS, unless the operator responds to the original CACH002A message by selecting A.

## Optional keywords for recovery agents that run in CV mode

You can use these optional keywords when you run a recovery agent in with the CV (central-version mode) parameter to recover data from one or more journals.

**ACTIVATE={Y | N}**

Specifies whether to enable the change-capture agent after the recovery agent successfully completes.

Using this keyword with the CV parameter is recommended in only two situations:

- Set ACTIVATE to N if you want the recovery agent to run continuously (with RESTARTWAIT set to a non-zero value) until the central version is shut down and the recovery agent reaches the end of the current journal.

At this point, the recovery agent stops and does not activate the change-capture agent. You need to start the change-capture agent by restarting the central version.

ACTIVATE=N is the recommended setting for production environments. This setting ensures that a quiesce point is established in the CA-IDMS central version.

- Set ACTIVATE to Y if you want the recovery agent to run continuously (with RESTARTWAIT set to a non-zero value) until the recovery agent reaches the end of the most current journal. At this point, the recovery agent forces change capture to begin, even though the central version is still running.

IBM recommends using ACTIVATE=Y for test environments only, not for production environments.

If you do not use this keyword and the value of RESTARTWAIT is a non-zero value, the recovery agent stops after reaching the end of the current journal. The change-capture agent remains in recovery mode unless the recovery agent detects that the CA-IDMS central version was shut down.

#### **RESTARTWAIT=*nn*{M|S}**

Defines the wait interval in minutes or seconds. Each time the recovery agent catches up to the last record written by an active CA-IDMS central version, the recovery agent suspends processing for the specified interval before querying the active journal for new change records.

Use M for minutes or S for seconds. The default is S. If you do not specify this keyword or you give it the value of 0, the recovery agent stops after catching up with the current position in the active journal file.

```
PARM='CV,RESTARTWAIT=5S'
```

#### **THROTTLE=*nnnn***

Prevents the recovery agent from overrunning the correlation service with change messages. The throttle value defines the maximum number of messages to be queued to the correlation service at any time. This value ensures available space in the message queue for any change-capture agents that communicate with the correlation service.

If you do not specify a value, the default value is 512. A value of 0 disables throttling of messages.

```
PARM='CV,THROTTLE=1024'
```

#### **TIMEOUT=*nn*{M|S}**

Defines a timeout value when throttling is used. Throttling relies on the correlation service responding to requests that messages be received. If the correlation service is unable to respond within the specified TIMEOUT value, the recovery server stops with an error.

If you do not specify a value, the default value is 5 minutes.

```
PARM='CV,THROTTLE=1024,TIMEOUT=1M'
```

## **Recovering change data manually from CA-IDMS databases**

You can run the recovery agent to detect whether a change-capture agent is in recovery mode. If the recovery agent detects that you need to recover change data, you can choose the recovery procedure that fits your environment.

### **Before you begin**

Make sure that the job to offload journals and the job to start the central version are not modified to run the recovery agent automatically.

### Procedure

To recover change data manually from CA-IDMS databases:

1. Periodically run the recovery agent to generate a report on the status of the change-capture agents. See “Generating reports on recovery sequences for active change-capture agents for CA-IDMS databases.”
2. If a change-capture agent goes into recovery mode, follow these steps:
  - a. If you running test environment and want to cold start the correlation service to exit recovery mode, cold start the correlation service.
  - b. If the journals that are required were archived, recover change data from the archived journals. See “Recovering data from archived central version journal files” on page 175.
  - c. If you are running CA-IDMS in central-version mode and none of the journals are archived, run the recovery agent to recover change data from online journals. See “Recovering data from disk journal files that were written by CA-IDMS central versions” on page 173.
  - d. If you are running CA-IDMS in local mode, run the recovery agent to recover change data from single tapes or journals. See “Recovering from single tapes or disk journal files that were written by local mode CA-IDMS applications” on page 174.
3. If you are running in central-version mode, shut down and restart the central version to resume capturing change data. You do not need to follow this step if you are running in local mode or if you decide to cold start the correlation service.

### Generating reports on recovery sequences for active change-capture agents for CA-IDMS databases

Use the REPORT parameter for recovery agents to generate reports that show the current sequence of journals from which change data needs to be recovered and the restart point for a particular change-capture agent.

### Procedure

To generate these reports:

Include the following EXEC statement in your JCL to run the recovery agent:

```
//CACEC1DR EXEC PGM=CACEC1DR,PARM='REPORT,AGENT=name'
```

CACEC1DR is the name of the load module for the recovery agent. The keyword AGENT identifies the change-capture agent for which you want to generate the report.

The format for the names of change-capture agents is as follows:

- For change-capture agents that run in central versions, the format is *IDMS\_central\_version\_name*.
- For local change-capture agents, the format is *IDMS\_name*, where *name* is the name of the z/OS job or the started task that performed the database update.

Example JCL for the recovery agent:

```

//CACIDJNL PROC CAC='CAC', INSTALLED HIGH LEVEL QUALIFIER
// IDMS='CAI.IDMS.R160'
//*

//LOCALJN EXEC PGM=CACEC1DR,PARM='REPORT,AGENT=IDMS_160'
//STEPLIB DD DISP=SHR,DSN=&CAC..V8R2M00.SCACLOAD
//CTRANS DD DISP=SHR,DSN=&CAC..V8R2M00.SCACSASC
//J1JRNL DD DISP=SHR,DSN=&IDMS..DEV.J1JRNL
//J2JRNL DD DISP=SHR,DSN=&IDMS..DEV.J2JRNL
//J3JRNL DD DISP=SHR,DSN=&IDMS..DEV.J3JRNL
//J4JRNL DD DISP=SHR,DSN=&IDMS..DEV.J4JRNL
//SYSPRINT DD SYSOUT=*
//SYSTEM DD SYSOUT=*

```

## Recovering data from disk journal files that were written by CA-IDMS central versions

Use the CV (central-version mode) parameter for recovery agents to recover data from one or more disk journal files written by a CA-IDMS central version. The central version can be either running or stopped.

### Before you begin

You must be running CA-IDMS in central-version mode.

### About this task

CA-IDMS journals are read from the *J*n*JRNL* DD statements that are allocated in the recovery agent's execution JCL. When you use the central-version-mode parameter, you can include multiple journal files so that all journal files that are allocated in the central version startup JCL are processed by the recovery agent.

When you allocate multiple files for central-version mode, make sure that the order of data set allocations to *J*n*JRNL* matches the processing order that is defined in the CREATE DISK JOURNAL statements in the DMCL.

### Procedure

To recover data from one or more disk journal files written by a CA-IDMS central version:

1. Include the following EXEC statement in your JCL to execute the recovery agent. The ellipsis after CV in the second statement indicates that you can use optional keywords:

```
//CACEC1DR EXEC PGM=CACEC1DR,PARM='CV,...'
```

CACEC1DR is the name of the load module for the recovery agent.

2. Allocate a DD statement for every journal file that you want to recover data from.
3. Run the recovery agent continuously until you can shut down the central version.
4. Shutdown the central version.
5. Make sure that the recovery agent is finishes processing all journals and then terminates.
6. Restart the central version.

Example of what recovery execution JCL:

```

//JOBNAME JOB (ACCT,INFO), 'CA-IDMS RECOVERY', CLASS=A, MSGCLASS=X,
// NOTIFY= &SYSUID
//CACEC1DR EXEC PGM=CACEC1DR,
// PARM='CV,RESTARTWAIT=5S,TIMEOUT=5M'
//STEPLIB DD DISP=SHR,DSN=CAC.LOADLIB
//CTRANSDD DISP=SHR,DSN=SYS2.SASC.C650.LINKLIB
//** CV JOURNAL DATASETS AS DEFINED IN THE DMCL SOURCE
//** FOR CV 120.
//J1JRNL DD DISP=SHR,DSN=CAI.CA-IDMS.J1JRNL
//J2JRNL DD DISP=SHR,DSN=CAI.CA-IDMS.J2JRNL
//J3JRNL DD DISP=SHR,DSN=CAI.CA-IDMS.J3JRNL
//J4JRNL DD DISP=SHR,DSN=CAI.CA-IDMS.J4JRNL
//SYSPRINT DD SYSOUT=*
//SYSTEMDD SYSOUT=*
//

```

## Recovering from single tapes or disk journal files that were written by local mode CA-IDMS applications

Use the LOCAL parameter for recovery agents to recover data from a single tape or disk journal that was written by a CA-IDMS application that is running in local mode.

### About this task

In CA-IDMS, you can take a database offline from a central version, back up the database, and then run a batch job to update the database, using a local journal to capture the changes and possibly making interim commits. If the batch job fails, the CA-IDMS batch recovery job can rollback the changes to the last commit.

Recovery agents can read the same local journal files that are read by CA-IDMS batch recovery jobs. After a CA-IDMS batch recovery job completes, you can run a recovery agent if you want to capture the committed change data that the local journal contains.

You must run a recovery agent to capture this data before you bring the affected database online in the central version.

### Procedure

To recover data from single tapes or disk journal files that were written by local mode CA-IDMS applications:

Include the following EXEC statement in your JCL to run the recovery agent:

```
//CACEC1DR EXEC PGM=CACEC1DR,PARM='LOCAL,optional_keyword=value'
```

CACEC1DR is the name of the load module for the recovery agent.

The optional keywords are:

#### **ACTIVATE={Y | N}**

Specifies whether to activate the change-capture agent after the recovery agent successfully processes the entire local journal and the correlation service confirms receipt of the data. Specify Y to inform the correlation service that all recovery messages were sent. The change-capture agent will begin forwarding messages again after the final recovery message is processed by the correlation service.

#### **AGENT=agent-name**

Identifies the name of the change-capture agent when you set

PARM=LOCAL or the central version number is 0. For local change-capture agents, this name must be IDMS\_##### where ##### is the original z/OS job or started task name that performed the database update. For central version systems, the name is IDMS\_##### where ##### is the central version started task name.

```
PARM='LOCAL,AGENT=IDMS_IDMJUPDT'
```

**SERVER=***name*

Specifies the name of the correlation service that you want the recovery agent to communicate with, if you are using named correlation services.

Example JCL for the recovery agent:

```
//CACIDJNL PROC CAC='CAC', INSTALLED HIGH LEVEL QUALIFIER
// IDMS='CAI.IDMS.R160'
//*
//LOCALJN EXEC PGM=CACEC1DR,PARM='LOCAL,AGENT=IDMS_IDMJUPDT'
//STEPLIB DD DISP=SHR,DSN=&CAC..V8R2M00.SCACLOAD
//CTRANS DD DISP=SHR,DSN=&CAC..V8R2M00.SCACSASC
//J1JRNL DD DISP=SHR,DSN=<LOCAL JOURNAL>
//SYSPRINT DD SYSOUT=*
//SYSTEM DD SYSOUT=*
// PEND
```

## Recovering data from archived central version journal files

Use the ARCHIVE parameter for recovery agents to recover data from archived journals.

### Procedure

To recover data from archived central version files:

1. Include the following EXEC statement in your JCL to run the recovery agent.  

```
//CACEC1DR EXEC PGM=CACEC1DR,PARM='ARCHIVE,RESTARTWAIT=0S,AGENT=name'
```

CACEC1DR is the name of the load module for the recovery agent. The keyword AGENT identifies the change-capture agent for which you want to generate the report. This keyword is required for processing archives.
2. In the JCL to run the recovery agent, supply the names of one or more archived journals that contain data to recover. You can supply the names of all of the archived journals at one time. See the example after the last step.
3. Run the recovery agent.
4. If you did not supply all of the names of the archived journal files, run the recovery agent again with however many archived journals you want to include. Repeat this step until you reach a time in which you can shut down the central version.
5. Shut down the central version.
6. Run the recovery agent one final time, supplying the names of the remaining journal files that it needs to process, including the journal file that was offloaded when you shut down the central version. After reaching the end of the last journal, the recovery agent restarts the correlation service and change-capture agent.
7. Restart the central version.

### Example

In this example, 4 archived journals were input to the recovery agent. Each archive contains a full day's journal activity. The date at the end of the archived DSN shows the date the data was archived. The SYSJRNL*n* file shows which CV disk

journal was offloaded to create this file, where *n* matches the CA-IDMS CV disk journal number. The CV DISK journal DD's are commented out, emphasizing that the recovery agent cannot recover data from archived journals and disk journals at the same time.

```
//RCVR EXEC PGM=CACECIDR,
//          PARM='ARCHIVE,RESTARTWAIT=0S,AGENT=IDMS_160'
//STEPLIB DD DISP=SHR,DSN=&CAC..V8R2M00.SCACLOAD
//CTRANS  DD DISP=SHR,DSN=&CAC..V8R2M00.SCACSASC
//* ARCHIVE FILES
//J1JRNL  DD DISP=SHR,DSN=ARCHIVE.IDMS.R160.SYSJRNL4.02062006
//J2JRNL  DD DISP=SHR,DSN=ARCHIVE.IDMS.R160.SYSJRNL1.02072006
//J3JRNL  DD DISP=SHR,DSN=ARCHIVE.IDMS.R160.SYSJRNL2.02082006
//J4JRNL  DD DISP=SHR,DSN=ARCHIVE.IDMS.R160.SYSJRNL3.02092006
//* DISK JOURNALS
//*J1JRNL DD DISP=SHR,DSN=&IDMS..DEV.J1JRNL
//*J2JRNL DD DISP=SHR,DSN=&IDMS..DEV.J2JRNL
//*J3JRNL DD DISP=SHR,DSN=&IDMS..DEV.J3JRNL
```

Figure 7. Naming four archived journals in the JCL for the recovery agent

This example shows that you can provide as many archived journal files as needed, as long as you specify the JnJRNL DD value in the order that archived files were created. You can have many more than the JnJRNL DD's that are assigned to the CV disk journals. The JnJRNL DD is only an ascending sequence of archived journal files.

You can run the recovery agent multiple times, specifying a different range of archived journals each time. Each prior successful run will update the recovery start point being maintained by the CS. However, always include the last file input to the previous successful run of the recovery agent as the first archive journal of the next run.

The following example illustrates what the next run of the recovery agent might be after the prior example was successful.

```
//RCVR EXEC PGM=CACECIDR,
//          PARM='ARCHIVE,RESTARTWAIT=0S,AGENT=IDMS_160'
//STEPLIB DD DISP=SHR,DSN=&CAC..V8R2M00.SCACLOAD
//CTRANS  DD DISP=SHR,DSN=&CAC..V8R2M00.SCACSASC
//* ARCHIVE FILES
//J1JRNL  DD DISP=SHR,DSN= ARCHIVE.IDMS.R160.SYSJRNL3.02092006
//J2JRNL  DD DISP=SHR,DSN=ARCHIVE.IDMS.R160.SYSJRNL4.02102006
//J3JRNL  DD DISP=SHR,DSN=ARCHIVE.IDMS.R160.SYSJRNL1.02112006
//J4JRNL  DD DISP=SHR,DSN=ARCHIVE.IDMS.R160.SYSJRNL2.02122006
//* DISK JOURNALS
//*J1JRNL DD DISP=SHR,DSN=&IDMS..DEV.J1JRNL
//*J2JRNL DD DISP=SHR,DSN=&IDMS..DEV.J2JRNL
//*J3JRNL DD DISP=SHR,DSN=&IDMS..DEV.J3JRNL
```

Figure 8. Naming the last input archived journal and the next three archived journals

### Reverting to active mode during testing of your change capture environment for CA-IDMS databases

When you implement IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS on test servers to determine how best to deploy the application, the system will probably go into recovery mode regularly. To make development easier, there is a way for you to put the test system back into active mode without requiring you to complete the standard recovery process.

## Procedure

To return to active mode during testing:

1. **Optional:** If possible, stop CA-IDMS. Because it is frequently not possible to stop CA-IDMS, these steps will still work if CA-IDMS is running. If you are unable to stop CA-IDMS, run these steps when CA-IDMS is least busy.
2. If a CACH002A message appears on the operator screen, respond with A.
3. If you did not respond with A in step 2, configure and run the recovery agent using the following parameters:

```
REPORT,AGENT=IDMS_nnn,ACTIVATE=Y
```

If the correlation service is a named server, use the following parameters:

```
REPORT,SERVER=server_name,AGENT=IDMS_nnn,ACTIVATE=Y
```

These parameters allow you to skip the recovery of data and to switch to active mode with the correlation service and change-capture agent running.

If the correlation service goes into active mode and then immediately reverts to recovery mode, the recovery agent was probably started when there were in-flight CA-IDMS transactions. If the correlation service determines that it received an incomplete transaction, the correlation service immediately enters recovery mode.

4. If the correlation service enters recovery mode, repeat step 3 until the correlation service remains in active mode.

---

## Recovering change data from IMS databases

If your IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS system enters recovery mode, follow these steps to capture changes until you can reactivate your change-capture agent.

### Procedure

To recover change data:

1. **Optional:** If no correlation service was running when you started the change-capture agent, run the following batch jobs:
  - a. Run the "Check agent" batch job determine whether the change-capture agent is in recovery mode. If so, identify the exact starting point in the IMS log files where the recovery process needs to be initiated.
  - b. If the correlation service does not report that the change-capture agent is in recovery mode, run the "Set agent in recovery mode" batch job.
2. Use DBRC to identify the log files that you need for recovery.
3. Run the "Check agent" batch job.
4. Run the "Log file recovery" batch job.
5. Generate an activity report on the correlation service.

When the correlation service is finished processing the changes in the current log file, perform steps a through c again if more log files need processing.

If you do not stop IMS during the recovery process, more log files will need processing because changes will continue to be made. Continue to repeat steps a through c until there is a period of low activity that will allow the recovery agent to catch up completely in the logs.

6. Run the "Place agent in active mode" batch job after the correlation service is finished processing the changes in the last log file.

After all IMS agents in recovery are placed into active mode, recovery is complete and you can resume normal IMS and change capture operations.

## Recovery process for IMS databases

The recovery process involves recovering from IMS log files all of the change data records up to the most recent uncommitted transaction. You use the recovery process to push the restart point farther into the future for one or more change-capture agents that are in recovery mode.

How complicated the recovery process is and how long it takes to complete depend on three factors:

- The complexity of the recovery situation
- The length of time that the change-capture agents are in recovery mode
- Whether the IMS control region (for a change-capture agent that is in recovery mode) is still active or whether a quiesce point was reached

Recovering change data does not require the use of archived online log files. If the original online log (primary or secondary) was not overwritten due to log roll-over, this log can be used in the recovery process. If change data is recovered quickly enough and the online logs are still available, the recovery process is generally more efficient because typically the archived logs are written to tape, which is slower to process than the DASD online logs.

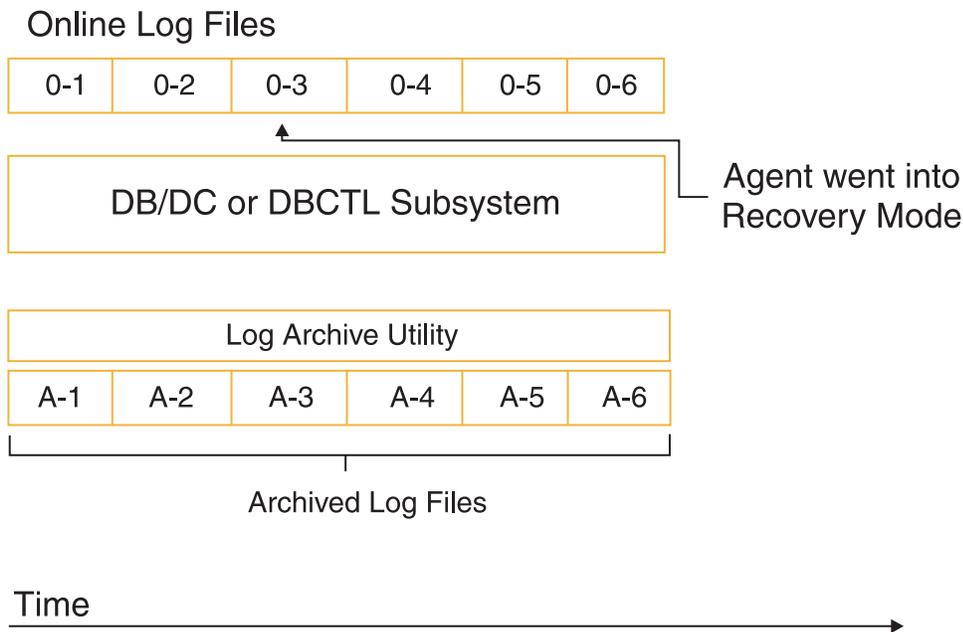
The recovery agent is capable of processing an active online log when recovering change data. The recovery agent detects when it reaches the current logical end-of-file (which is the record for which the timestamp value is less than the timestamp value for the previous record) and stops the recovery process when this condition is detected.

The diagrams in the following sections show two recovery situations that you might encounter.

- Single DB/DC or DBCTL subsystem in recovery mode
- Single DB/DC or DBCTL subsystem with two IMS batch jobs that are in recovery mode

### Single DB/DC or DBCTL subsystem in recovery mode

In this example, a single IMS DB/DC or DBCTL subsystem is being monitored. The diagram shows that while the subsystem was running, it used six online log files. While the subsystem was using the third online log file, the IMS change-capture agent went into recovery mode, perhaps due to an XM queue overrun.



#### Example 1:

1. When you configure Classic Event Publisher for IMS, you set up the IMS Log File Tracking Utility.
2. The change-capture agent goes into recovery mode, although you are unaware of this change. You might not know that the agent is in recovery mode if, for example, you did not see the WTO messages that the agent issued.
3. You stop the IMS subsystem.
4. You run the recovery agent in Check Status mode to get the status of the change-capture agent. The recovery agent reports that the change-capture agent is in recovery mode. The IMS log files A-3, A-4, A-5, and A-6 were archived and are required for the recovery process. Because you stopped the IMS subsystem, the recovery process can be completed in one operation by supplying the names of archived IMS log files A-3 through A-6.
5. You run the recovery agent in Log File Recovery mode, supplying in the control file the names of the archived IMS log files A-3 through A-6. When the recovery agent finishes processing these IMS log files, the recovery process is complete.
6. You run the recovery agent in "Activate change-capture agent" mode.
7. You restart the DB/DC or DBCTL subsystem.

#### Example 2:

1. When you configure change capture, you modify the Log Archive Utility job to include the IMS Log File Tracking Utility.
2. The change-capture agent goes into recovery mode, and you see the MTO message from the correlation service that reports the status of the change-capture agent.
3. You run the recovery agent in Check Status mode to identify the restart point and the log file in which the restart point is located. The recovery agent reports a restart point that is located somewhere in the third IMS online log file.

If the Log Archive Utility ran or did not yet complete when you run the recovery agent to check the status of the change-capture agent, the second archived log file is identified as an IMS log file that is needed for recovery. You run a DBRC LIST.LOG ALL report and determine that the second archived IMS log file was closed before the restart point reported by the recovery agent. Therefore, you need to input the online version of the log O-3 in the Log File Recovery batch job.

You decide to recover change data with the active online log or to wait until the IMS Log Archive job for online log file 3 is complete. When you run the recovery agent again and request status information for the change-capture agent that is in recovery, the recovery agent reports that archived IMS log file 3 is required. You could re-run the DBRC LIST.LOG ALL report and verify that archived IMS log 3 is in fact the first file that needs to be input into the recovery process, or you can skip this step because you know that archived IMS log file 2 does not contain the restart point and that you need to start with archived IMS log file 3.

4. When the third online log file is archived, you can run the recovery agent in Log File Recovery mode, supplying the name of the IMS archived log 3.
5. You then wait until the fourth online log file is archived. At this point, you run the recovery agent in Check Status mode, requesting the status of the change-capture agent in recovery mode. The recovery agent reports that archived log file 3 and archived IMS log 4 are required. When recovering change data for archived IMS logs that become available from a DB/DC or DBCTL subsystem, you will probably need to input the IMS log file before the first IMS log file is recovered because usually, when a log file switch occurs, one or more UORs are in-flight. As shown in Figure 2 on page 54, an agent's restart point is the first type 99 data capture IMS log record for the oldest living UOR when tracking begins for a new UOR. Therefore, the restart point will most likely exist somewhere in the previous log file when a switch occurs.

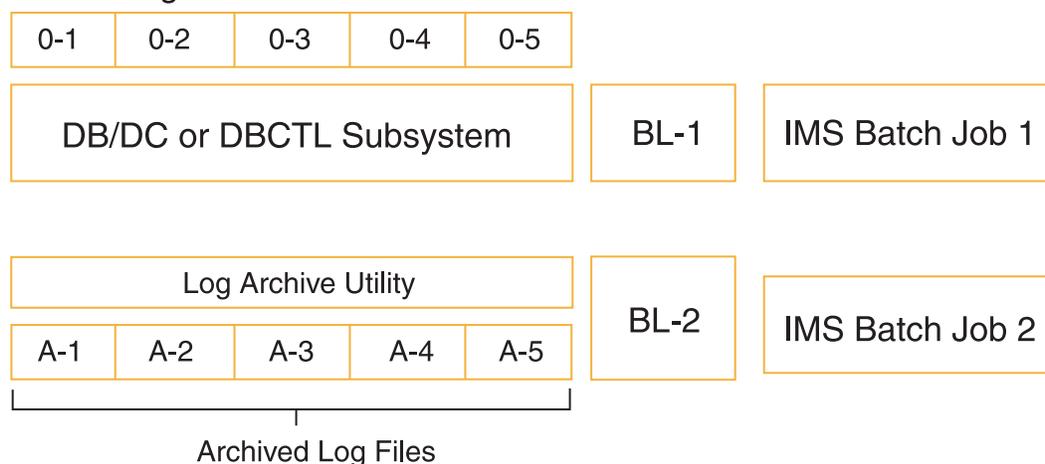
If you have IMS applications that run for extended periods of time that do not issue frequent checkpoints, you might find that the restart point is further in the past than you think. When you are feeding IMS logs into the recovery process for archived logs that become available, run the recovery agent in Check Status mode each time the archive job completes to determine whether there is a new restart point.

6. After the sixth online log file is archived, you stop the IMS DB/DC or DBCTL subsystem.
7. You run the recovery agent in Log File Recovery mode and you supply the names of archived log files 3 through 6.
8. You run the recovery agent in "Activate change-capture agent" mode.
9. You restart the IMS subsystem.

#### **Single DB/DC or DBCTL subsystem with two IMS batch jobs that are in recovery mode**

In this example, a DB/DC or DBCTL subsystem was active while two batch jobs were executed without a correlation service running. All three of these change-capture agents are now in recovery mode, although the correlation service does not know that they are.

## Online Log Files



Time

1. You create recovery data sets for the change-capture agents.
2. You set up IMS log file tracking.
3. While CEP is running, the three change-capture agents go into recovery mode.
4. You start the correlation service, which does not know about the three change-capture agents that are in recovery mode.
5. You run the recovery agent in Check Status mode. The output from the job identifies restart points for the three change-capture agents and identifies the IMS log files that are required for the recovery process.
6. You run the recovery agent in Set Agent To Recovery Mode mode. The recovery agent tells the correlation service that the three change-capture agents are in recovery mode. Part of the output from this operation is the list of IMS log files that must be recovered for each agent. For the DB/DC or DBCTL subsystem, the archived IMS log files A-2 through A-5 are identified as required IMS log files. Likewise, for the two batch jobs, IMS log file BL-1 and BL-2 are identified as potentially-necessary IMS log files.
7. You run a DBRC LIST.LOG ALL report and find an entry for archived IMS log file A-1. In the report, you see an entry for archived IMS log file A-1 and that this file was created either soon before or soon after the restart point for the DB/DC or DBCTL subsystem. In the report, you also see entries for IMS log files BL-1 and BL-2, which were created either soon before or soon after the restart points for these two IMS change-capture agents under DBRC control.
8. You run the recovery agent in Log File Recovery mode and you input all of the IMS log files that were identified by the recovery agent. In this example, you would supply seven control files, five for the DB/DC or DBCTL region identifying the names of archived log files A-1 through A-6 and a control card for each batch agent and their associated system log files (BL-1 and BL-2). After these IMS log files are successfully processed, WebSphere Classic Event Publisher for z/OS will be synchronized with IMS.

9. You run the recovery agent in Activate Change-Capture Agent mode, which puts the three change-capture agents back into active state. Assuming the correlation service remains running, then the next time the DB/DC or DBCTL subsystem is started, the IMS batch jobs are run again, or both, changes will be captured.

A second approach to this recovery situation is to recover the three agents separately.

If the two IMS batch jobs updated different IMS databases, then each of these agents' IMS log files can be recovered independently. However, if these batch jobs updated a shared database then these agents should be recovered together.

If the two IMS batch jobs updated the same IMS database record and these agents are recovered independently, you run the risk of propagating updates out-of-sequence. If the IMS batch jobs updated different IMS database records in the shared database, then the agents can be recovered independently, but it is difficult and time-consuming to determine whether an IMS application that updated a shared database updated the same database record that another IMS application updated.

The following table summarizes the previous two paragraphs:

*Table 17. Summary of options for the two IMS batch jobs*

<b>Records updated</b>	<b>In separate databases</b>	<b>In shared database</b>
<b>Updated same IMS records</b>	N/A	Recover the agents together or risk propagating updates out-of-sequence
<b>Updated different IMS records</b>	Can recover independently	Can recover independently, but it can be difficult to determine whether an IMS application that updated a shared database updated the same database record that another IMS application updated. It is recommended to recover the agents together.

Regardless of whether you are using a data-sharing environment, you can recover independently the five archived IMS log files that are associated with the DB/DC or DBCTL subsystem because no other IMS applications were running while the subsystem was operational. In other words, because you shut down the online region before running the two batch jobs, you can recover change data from that region's log files, activate the region's change-capture agent, and then recover the two batch jobs. Change-capture agents and the correlation service do not know if you are in a data-sharing environment. If you are in a data-sharing environment and an change-capture agent goes into recovery mode, immediately stop the correlation service. Any other change-capture agents that communicate with that correlation service will go into recovery mode and will automatically place any new IMS applications into recovery mode if they are run while the correlation service is down.

## Recovery agents for IMS databases

The recovery agent is a batch job that is used to synchronize one or more change-capture agents that are in recovery mode with the IMS control region where the change-capture agents are active.

Synchronization is performed by reading the IMS log files (created by the IMS control region that was being monitored) and forwarding the appropriate IMS log records to the correlation service for processing.

The recovery agent runs in one of four modes:

**Check status of change-capture agent**

You can run the recovery agent to request the status of a change-capture agent. If a change-capture agent is reported as being in recovery mode, either by the content of the restart data set or by the correlation service, the recovery agent identifies a restart point for the change-capture agent.

**Set change-capture agent to recovery mode**

An unknown agent is a change-capture agent that was activated when its corresponding correlation service was not active. If you are using recovery data sets, the output from the Check Agent Status job allows you to identify these agents. The change-capture agents will have recovery data set information and a restart point, but the correlation service reports that the agent is not in recovery mode.

Running the recovery agent in this mode informs the correlation service that the change-capture agent is in recovery mode and transfers the information from the recovery data set to the correlation service.

**Recover log files**

After you identify the IMS log files that need to be input into the recovery process, you can input these files into the recovery process and move the recovery restart point forward in time for one or more change-capture agents that are in recovery mode.

**Activate change-capture agent**

After you complete the recovery process for all of your change-capture agents that are in recovery mode, you use the recovery agent to place those change-capture agents back into active mode.

You supply a control file when you run the recovery agent. This control file determines the mode in which the recovery agent runs.

The recovery agent address space requires read access to IMS log files and read-write access to recovery data sets.

The JCL to run the IMS recovery agent is located in SCACSAMP member CACIMSRA. In the PARM parameter of the EXEC statement, supply one or more of the keywords in the following table:

Table 18. Keywords for the PARM value of the EXEC statement in the JCL for recovery agents

Keyword	Description
THROTTLE= <i>nnnn</i>	<p>Use when you run a recovery agent in Log File Recovery mode.</p> <p>Defines a throttling value to prevent the IMS recovery change-capture agent from overrunning the correlation service with change and synchpoint log records. The throttle value defines the maximum number of messages to be queued to the correlation service at any time. This maximum helps ensure available space in the message queue for communications with the correlation service during the recovery process.</p> <p>If this value is not specified, the throttle value defaults to 128. A value of 0 disables throttling of messages.</p> <p>The throttling applies only during IMS log file recovery processing. The throttle value is applied to each IMS change-capture agent in recovery mode for which IMS log files are supplied. For example, if you use the default value and you have two agents performing IMS log file recovery, a throttle message is issued for the first agent after 128 buffers are sent to the correlation service. Likewise, a throttle message is sent after 128 buffers are sent for the second agent in recovery.</p> <p>PARM=' THROTTLE=512 '</p>
TIMEOUT= <i>nnM S</i>	<p>Use when running a recovery agent in Log File Recovery mode.</p> <p>Defines a timeout value (in minutes or seconds) when throttling is used. Throttling relies on the correlation service responding to requests that messages are received. If the correlation service is unable to respond within the specified timeout, the recovery agent terminates with an error.</p> <p>If this parameter is not supplied, the default TIMEOUT value is 5 minutes.</p> <p>PARM=' THROTTLE=512, TIMEOUT=5M '</p>
SERVER= <i>name</i>	<p>Use when you run a recovery agent in any mode if you are using named correlation services.</p> <p>Defines the name of the correlation service to communicate with when you use named correlation services. If this parameter is not supplied, the default is to communicate with an unnamed correlation service.</p> <p>PARM=' SERVER=IMSTEST '</p>

## Sequence checking for log records for IMS databases

The correlation service and change-capture agent track information about the IMS log records that are passed to the change-capture agent by IMS and sent to the correlation service. The information is used to detect lost, out-of-sequence, or duplicated log record conditions. If lost or out-of-sequence log records are detected, the change-capture agent enters recovery mode.

Regardless of whether no data capture or synchpoint log records are in the buffer that is passed by IMS to a change-capture agent, the agent sends a data gram to the correlation service. In these cases, the data gram contains only information about the starting and ending log record sequence numbers of the log records that are contained in the buffer. The correlation service tracks each active

change-capture agent's log record sequence numbers, because DFSFLGX0 can be called with duplicated, lost, or out-of-sequence buffers. The correlation service automatically discards duplicated buffers, though when a lost or out-of-sequence condition is encountered, this condition forces a recovery situation.

If IMS makes a mistake because the log buffer is successfully written to the IMS log files, recovery is still possible because the IMS log records are guaranteed to be in the correct sequence in the IMS log files.

The correlation service is designed to detect and deal with duplicated, lost, or out-of-sequence log buffer conditions. The correlation service is also designed to track the state and health of each IMS control region. This tracking is done by sending additional log records to the correlation service that are not related to data capture or to tracking the progress of the currently in-flight units-of-work. The main log record of interest for tracking control region status is the type 06 IMS/VS Accounting Record.

Using the information provided in the IMS/VS Accounting Record, plus information provided by IMS to DFSFGLX0, the correlation service detects and handles the following situations:

- Start of an IMS control region.
- Normal termination of a batch job.
- Abnormal termination of a batch job. If a unit-of-work is in-flight, it is automatically rolled back. In this situation, you might need to run the IMS Batch Backout Utility, but this situation is not a WebSphere Classic Event Publisher for z/OS error condition.
- Normal termination of a DB/DC or DBCTL subsystem.
- Abnormal termination of a DB/DC or DBCTL subsystem. Any in-flight units-of-work that performed updates are retained until the control regions restart.
- Emergency restart of a DB/DC or DBCTL subsystem.

For example, in the case of an emergency restart of a DB/DC or DBCTL subsystem, the correlation service expects to see a series of synchpoint control records. These records report roll backs for the units-of-work that were updating a database when the system failed. If the correlation service remained operational between the termination and restart of the DB/DC or DBCTL subsystem, the correlation service retained tracking information about the subsystem that terminated and its in-flight units-of-work. In this situation, these units-of-work are rolled back as if the application issued a roll back request.

In recovery situations, the correlation service no longer performs IMS log file tracking and the recovery agent is responsible for ensuring that the proper IMS log records are sent to the correlation service. When you supply the names of multiple IMS log files during IMS log file recovery, the recovery agent potentially reads each IMS log file twice. Before processing the IMS log files, the recovery agent scans each input log file to determine whether the file is required, to ensure that the IMS log files are processed in the correct chronological order, and to ensure that an invalid or duplicated log file was not specified.

If no errors are encountered during this pre-scanning phase, the recovery agent discards any IMS log files that are not required and then processes only the required log files. This process is referred to as log-file sequence checking. The

recovery agent issues WTO messages that indicate when sequence checking begins, provides status messages, and indicates when log-file sequence checking is completed.

Even if a single IMS log file is supplied, WTO messages are issued that indicate that the IMS log-file sequence checking is occurring. However, because only a single log file exists, the IMS log file is not actually scanned, because no effective sequence checking can be performed, and so the IMS log-file recovery process begins almost immediately.

Regardless of whether you specify a single or multiple IMS log files, when sequence checking is completed and IMS log-file recovery begins, the recovery agents attempt to locate the restart point in the log files. If the restart point cannot be found, WTO message CACH036E is issued, which indicates that the IMS log files specified do not contain the restart point.

## **Situations in which data cannot be recovered from IMS databases**

There are four situations in which data cannot be recovered after a change-capture agent goes into recovery mode.

The four situations are:

- There is a media failure for an IMS log file that is needed in the IMS log file recovery process and dual logging was not in effect, or both log files have media problems.
- The IMS log or archived log files that are required for IMS log file recovery processing no longer exist. This situation occurs only when you defer the recovery process for an extended period of time.
- Archived log files are used in the recovery process and the IMS Log Archive Utility supplied an SLDS control statement that removed log records of interest to WebSphere Classic Event Publisher for z/OS from the archived log file.
- An IMS log file contains a type 99 Data Capture Log record that is marked “in error.” This situation occurs only when cascade delete information is being recorded.

## **How Cross Memory queue overruns affect the recovery of change data from IMS databases**

The recovery agent uses XM data grams to communicate with the correlation service. There are some conditions under which the default XM queue size might not be adequate, and might need to be increased.

If a single change-capture agent is being recovered, an XM data gram is posted when a 32-kilobyte buffer is filled up with IMS log records of interest. If IMS log files for multiple change-capture agents are being recovered, this behavior is modified and an XM data gram is posted when:

- A 32-kilobyte buffer is filled up with IMS log records of interest.
- A buffer is truncated due to system clock overlaps with another agent.

There are some conditions under which the default XM queue size might not be adequate, and might need to be increased. For example, if you have a BMP or batch job that performs a large number of updates to databases from which you are capturing changes, and the BMP or batch job does not issue frequent checkpoints, large amounts of data are written to the message store, which slows

down the correlation service's overall responsiveness. When a checkpoint is issued, the checkpoint causes the UOR to be committed, and the correlation service begins sending the data changes that are made by the UOR to the publication service. By default, the correlation service sends all changes that are associated with a committed UOR to the distribution service. Before checking the XM queue for more work, the correlation service waits for confirmation that the publication service received those changes.

The recovery agent is much more likely to encounter this situation because the recovery agent is directly reading the IMS log files and does not need to wait to be invoked by IMS.

If you have applications that generate large transactions but do not issue frequent checkpoints, you can use three ways to reduce the risk of filling up the XM queue that is used by a change-capture agent or recovery agent to send data to a correlation service:

#### **Increase the size of the XM queue**

Increasing the size of the XM queue helps reduce the possibility that the XM queue will fill up. Such a condition would put the change-capture agent in recovery mode and adversely affect any recovery agents that might be running.

By default, the size of the XM queue that is allocated is 8 megabytes. In the service information entry for the correlation service, you can change the size of the queue to be from 1 to 256 megabytes.

For a DB/DC subsystem that is primarily processing transactions, the default 8 megabytes of queue space should be sufficient to handle the traffic between the active change-capture agent and the correlation service. Likewise, for a DBCTL subsystem that is primarily processing DRA "transactions" (such as CICS or WebSphere Classic Federation Server for z/OS), the default setting should be adequate for normal operations.

You also need to increase the size of the XM queue when you have multiple change-capture agents monitoring different IMS control regions in a high-volume environment and those change-capture agents communicate with the same correlation service. All of the agents are sharing the same XM queue. A queue overrun can occur if the correlation service is busy processing committed UORs. A queue overrun can also occur if the correlation service in the process of analyzing the XM data gram from one change-capture agent and cannot retrieve the data grams fast enough to empty the queue.

There is no general rule for determining the optimum size of the XM queue for your site. The size depends on how your applications are written and how active your IMS control regions are. You need knowledge of the applications that are being monitored to determine whether they generate large UORs that contain many changes, or whether the majority of the work that is being monitored is transaction based. Likewise, you should know how active your system is.

#### **Specify an interrupt value**

To help alleviate problems with UORs that contain large numbers of changes, the correlation service allows you to specify an interrupt value. This value allows the correlation service to alternate between sending changes to the distribution service and checking for more work from change-capture agents and recovery agents.

### Specify a value for throttling message delivery

You can also specify a throttle value that causes the recovery agent to slow down when sending data to a correlation service.

## Control files for recovery agents for IMS databases

The recovery agent runs as a batch job. You use an 80-character text input file, called a control file, to tell the recovery agent the operations to perform.

The contents of the control file identify the operations that the recovery agent is to perform. The format of the control file is identified in the following table.

*Table 19. Format of control files for recovery agents*

Parameter name	Starting offset	Length	Description
Action identifier	1	1	Identifies the action that the recovery agent is to perform.
Name of change-capture agent	3	8	The 8-character job name of the change-capture agent.
Region type	12	1	Identifies the type of IMS control region type from which data is being recovered when an IMS log file is supplied.
Data set name	14	44	<ul style="list-style-type: none"><li>• If you are running the recovery agent in Log file recovery mode, specify the name of an IMS log file to use in the recovery process.</li><li>• For all other modes in which the recovery agent can run, specify the name of the recovery data set.</li></ul>

You use the first parameter of the control file to tell the recovery agent to perform one of the actions in the following table.

*Table 20. Actions that recovery agents can perform*

Identifier	Description	Action
C	Check status of change-capture agents	<p>Check the recovery data set of a change-capture agent to see if that change-capture agent is in recovery mode. If so, identify a restart point.</p> <p>The correlation service is also accessed to see if the change-capture agent is currently in recovery mode. The recovery agent compares the restart point in the recovery data set to the restart point that is known to the correlation service. If the two restart points differ, the recovery agent uses the restart point that is known to the correlation service.</p>

Table 20. Actions that recovery agents can perform (continued)

Identifier	Description	Action
S	Set change-capture agents in recovery mode	<p>Given the name of a recovery data set and the name of a change-capture agent, place that change-capture agent in recovery mode. The name of the change-capture agent must match the name of the agent recorded in the recovery data set. The recovery agent informs the correlation service that the change-capture agent is in recovery mode and passes the restart point from the recovery data set to the correlation service.</p> <p>Perform this action with the recovery agent only when the change-capture agent is unknown to the corresponding correlation service because the change-capture agent started when the correlation service was not active.</p>
L	Log file recovery	<p>Given the name of one or more IMS log files, the name of a change-capture agent, and the region type, the recovery agent attempts to move the recovery point forward in time. In other words, the recovery agent passes to the correlation service the change data that the change-capture agent could not capture after going into recovery mode.</p>
A	Place change-capture agents in active mode	<p>Take the change-capture agent out of recovery mode and put it into active mode.</p>

When you are recovering data from log files, you must also identify the type of IMS Control Region that contains the data that is being recovered. For the Region Type parameter in the control file, use one of the values in the following table.

Table 21. Region types

Identifier	Region type
1	Online DB/DC
2	IMS batch
4	DBCTL

## Determining the status of change-capture agents for IMS databases

You can run the recovery agent to request the status of a change-capture agent. If a change-capture agent is reported as being in recovery mode, the recovery agent identifies a restart point for the change-capture agent either by looking at the content of the restart data set or by querying the correlation service.

### About this task

If the change-capture agent started before the correlation service and is therefore not known to the correlation service, the recovery agent identifies the restart point

by looking in the recovery data set. If the change-capture agent started after the correlation service, the correlation service gives the restart point to the recovery agent.

In either of these situations, both the recovery data set and the correlation service can report restart points. The restart point that is reported by the correlation service always takes precedence.

### Procedure

To determine the status of a change-capture agent:

1. Create a control file.  
In the control file, create a control card for each change-capture agent that you want to check the status of. Input the following values in each control card:
  - a. For the action identifier, use "C".
  - b. For the name of the change-capture agent, specify the name of the change-capture agent for which you want to check the status.
  - c. For the region type, specify the type of IMS control region type in which the recovery data set for the agent is located.
  - d. For the data set name, specify the name of the recovery data set that you want to check for the existence of a restart point. If the change-capture agent started after the correlation service started, you can input a dummy name.
2. Run the recovery agent. The JCL to run this batch job is located in the member CACIMSRA in data set SCACSAMP.

### Example

See "Examples of determining the status of change-capture agents for IMS databases" on page 191.

After you run the recovery agent to get restart information, you can use one of these three methods to determine which log files you must supply to the recovery agent to bring WebSphere Classic Event Publisher for z/OS out of recovery mode:

- Manually track the IMS jobs and IMS log files that were created, and supply this information to the IMS recovery agent.
- Run a DBRC LIST.LOG ALL (or variant) report if the IMS jobs are registered with DBRC. Review the DBRC output to identify the IMS log files that are associated with the IMS control region for the change-capture agents that are in recovery mode.
- Have WebSphere Classic Event Publisher for z/OS track the IMS log files associated with an agent. When this tracking is implemented, the recovery change-capture agent automatically identifies the IMS log files that you need when you check the status of a change-capture agent.

Using either of the first two methods to identify IMS log files that are required for recovery is error prone (especially if one or more agents are in recovery mode for extended periods of time). The likelihood of missing a log file is greater than if you let WebSphere Classic Event Publisher for z/OS track the log files. However, even if you use log file tracking, it is possible to miss a log file if the control file is not updated properly.

## Examples of determining the status of change-capture agents for IMS databases

The following two examples demonstrate how you can use recovery agents to determine the status of change-capture agents for IMS databases.

These two examples use the same two change-capture agents. These change-capture agents are batch jobs, although the examples work with DB/DC or DBCTL regions that are in recovery mode.

The following table shows the contents of the control file. The parameter for the region type is not used.

Table 22. Contents of control file

Mode	Agent	Name of data set
C	WCA008LA	WCA008.CEP.WCA008LA
C	WCA008DA	WCA008.CEP.WCA008DA

### Example one

Agent WCA008LA was run while no correlation services were active. Agent WCA008DA was never run. The following WTO messages are issued when the recovery agent is run in Check Status mode:

**CACH061I RECOVERY MODE: CHECK AGENT STATUS**

Identifies the mode in which the recovery agent is running. In this example, the recovery agent is running in Check Status mode.

**CACH048I RECOVERY DATASET WCA008.XSYNC.WCA008LA OPENED**

Indicates that the recovery agent successfully opened the recovery data set and found that the data set contains a record with the restart point in the IMS log where recovery needs to begin.

**CACH049I AGENT NAME IS 'WCA008LA'**

Identifies the name of the change-capture agent that is in recovery mode.

**CACH039I DB2 RESTART TIME 20040726 09294000**

Identifies the restart point date and time in DB2 format. The date and time that are identified are July 26th, 2004 at 9:29:40 am local time.

**CACH040I IMS RESTART TIME 04.208 09:29:40.0**

Identifies the restart point in the date and time format used by IMS in DBRC reports. The date and time that are identified are the 208th day of 2004 at 9:29:40 am local time.

**CACH037I RESTART SYSTEM CLOCK BB931C6945765200**

Identifies the system clock value from the IMS log record suffix where restart needs to begin. In this example, the system clock value is BB931C6945765200.

**CACH038I RESTART LOG SEQ. # 00000000-00000001**

Identifies the log record sequence number from the IMS log record suffix where restart needs to begin. In this example, the log record sequence number is 00000000-00000001.

**Note:** This example is for an IMS batch application. For these types of applications, when the batch job is run without a correlation service being active, the restart point is always the first IMS log record in the file.

**CACH030I AGENT 'WCA008LA' IS NOT IN RECOVERY MODE**

Identifies that the correlation service does not report the agent in recovery mode. In this example, because the change-capture agent recovery data set does contain restart information and the correlation service reports that the agent is not in recovery mode, then IMS was started without a correlation service active. You must set the agent in recovery mode using the procedure described in “Putting change-capture agents that are unknown to correlation services into recovery mode” on page 194.

**CACH050I RECOVERY DATASET WCA008.XSYNC.WCA008LA REPORTS AGENT IS NOT IN \ RECOVERY MODE**

Indicates that the recovery data set is empty, meaning that the change-capture agent is not in recovery mode.

**CACH030I AGENT 'WCA008LA' IS NOT IN RECOVERY MODE**

Indicates that the correlation service does not know about this change-capture agent. In this example, the reason is that the change-capture agent did not run yet.

**CACH071I 1 IMS AGENTS ARE IN RECOVERY MODE**

Identifies that there is one change-capture agent (WCA008DA) that is in recovery mode. This message is issued only when there are change-capture agents in recovery mode. The job ends with a return code value of 1 to indicate that one or more agents are in recovery.

**CACH062I RECOVERY PROCESSING COMPLETED SUCCESSFULLY**

Indicates that the recovery agent ran and ended without error. Other messages that indicate the success or failure of the agent are:

**CACH062W**

Processing completed successfully. However, the recovery agent detected a non-fatal error. Such errors can include attempts to put into recovery mode change-capture agents that are already in recovery mode, and attempts to activate change-capture agents that are already active.

**CACH062E**

Fatal error due to memory allocation failure, failures while communicating with the correlation service, invalid control card input, or invalid IMS log file input. One or two messages precede this message and identify the error condition.

## Example two

In this example, change-capture agent WCA008LA was started after the correlation service was activated and all changes were processed successfully. Change-capture agent WCA008DA also started after the correlation service was activated. However, because the correlation service was terminated before the committed UORs of change-capture agent WCA008DA were processed, the change-capture agent went into recovery mode.

The correlation service was then warm started so that recovery information was retained.

The following WTO messages are issued when you run the recovery agent in Check Status mode:

**CACH061I RECOVERY MODE: CHECK AGENT STATUS**

Identifies the mode in which the recovery agent is running. In this example, the recovery agent is running in Check Status mode.

**CACH050I RECOVERY DATASET WCA008.XSYNC.WCA008LA REPORTS AGENT IS NOT IN \ RECOVERY MODE**

Indicates that the recovery data set for the change-capture agent is empty.

**CACH030I AGENT 'WCA008LA' IS NOT IN RECOVERY MODE**

Indicates that the correlation service does not know about this change-capture agent.

**CACH050I RECOVERY DATASET WCA008.XSYNC.WCA008DA REPORTS AGENT IS NOT IN \ RECOVERY MODE**

Indicates that the recovery data set for the change-capture agent WCA008DA is empty.

**CACH051I CORRELATION SERVICE REPORTS AGENT IN RECOVERY MODE**

Indicates that the correlation service knows that the change-capture agent is in recovery mode.

**CACH049I AGENT NAME IS 'WCA008DA'**

Identifies the name of the change-capture agent that is in recovery mode.

**CACH039I DB2 RESTART TIME 20040726 13031403**

Identifies the restart point date and time in the format of a DB2 timestamp. The restart point identified is July 26th, 2004 at 13:03:14 pm.

**CACH040I IMS RESTART TIME 04.208 13:03:14.0**

Identifies the restart point in the date and time format that is used by IMS in DBRC reports. The restart point identified is the 206th day of 2004 at 13:03:14 pm.

**CACH037I RESTART SYSTEM CLOCK BB934C25B83B8E00**

Identifies the 8-byte system clock value that is contained in the log record suffix. The log record suffix identifies the log record where the recovery process needs to begin.

**CACH038I RESTART LOG SEQ. # 00000000-00000001**

Identifies the double-word log record sequence number that is contained in the log record suffix. The log record suffix identifies the log record where the recovery process needs to begin.

**CACH071I 1 IMS AGENTS ARE IN RECOVERY MODE**

Identifies that one change-capture agent (WCA008DA) is in recovery mode.

**CACH062I RECOVERY PROCESSING COMPLETED SUCCESSFULLY**

Indicates that the recovery agent encountered no errors while checking whether the change-capture agents were in recovery mode.

## Determining whether change data needs to be recovered from IMS databases

If any change-capture agents are in recovery mode, the recovery process is not required if the IMS applications for the change-capture agents that failed did not update any monitored databases.

### Procedure

To determine whether you need to recover data from the IMS log files that are associated with a change-capture agent that is in recovery mode:

1. Use member CACIMSLA in SCACSAMP to determine whether the IMS log files that are associated with the change-capture agent that is in recovery mode contain updates to any databases from which you are capturing changes.  
CACIMSLA executes the IMS File Select and Formatting Print Utility (DFSERA10). The input control cards search the IMS log files for type 99 data capture log records. If data capture log records exist in the IMS log files, their contents are printed to standard output and the utility ends. If you execute CACIMSLA and the IMS File Select and Formatting Print Utility does not display data capture log records, you do not need to follow the recovery process.
2. If you execute CACIMSLA and the IMS File Select and Formatting Print Utility displays data capture log records, you must follow the recovery process to recover data in the IMS log files associated with the change-capture agent.
  - a. Start the correlation service, if it is not already running.
  - b. Run the recovery agent:
    - If the correlation service reports that the change-capture agent is in recovery mode, run the recovery agent to activate the change-capture agent.
    - If the correlation service does not report that the change-capture agent is in recovery mode, run the recovery agent to set the change-capture agent in recovery mode. Then, run the recovery agent to activate the change-capture agent.

## Putting change-capture agents that are unknown to correlation services into recovery mode

An unknown agent is a change-capture agent that was started when its corresponding correlation service was not active. If you are using recovery data sets, the output from the Check Status job allows you to identify such change-capture agents.

### About this task

These change-capture agents will have recovery data set information and restart points, but the correlation service reports that these change-capture agents are not in recovery mode. You need to create a custom input control file and run the IMS recovery agent in Set Change-Capture Agent in Recovery Mode mode. The recovery agent communicates with the correlation service to notify the correlation service that a change-capture agent is in recovery mode and supplies the correlation service with the restart point for the agent.

### Procedure

To manually put a change-capture agent into recovery mode:

1. Create a control file.  
In the control file, create a control card for each change-capture agent that you want to put into recovery mode. Specify following values in each control card:
  - a. For the action identifier, use "S".
  - b. For the name of the change-capture agent, specify the name of the change-capture agent to put into recovery mode.

- c. For the region type, specify the type of IMS control region type from which the recovery agent will recover data.
  - d. For the data set name, specify the name of the recovery data set for the change-capture agent to put into recovery mode.
2. Run the recovery agent. The JCL to run this batch job is located in the member CACIMSRA in data set SCACSAMP.

## Example of putting into recovery mode a change-capture agent that is unknown to a correlation service

Read this example to understand how to put into recovery mode a change-capture agent that is not known to the corresponding correlation service.

Change-capture agent WCA008LA is in recovery mode. The correlation service needs to know:

- that the change-capture agent exists
- that the change-capture agent is in recovery mode
- what the restart point is for the change-capture agent

Supplying the following control file input and running the recovery agent notifies the correlation service that change-capture agent WCA008LA is in recovery mode. The parameter for the region type is not used.

*Table 23. Contents of the control file*

Mode	Agent	Name of data set
S	WCA008LA	WCA008.XSYNC.WCA008LA

Listed below are the WTO messages that are returned from the recovery agent.

```
CACH061I RECOVERY MODE: SET AGENT IN RECOVERY
CACH048I RECOVERY DATASET WCA008.XSYNC.WCA008LA OPENED
CACH052I AGENT 'WCA008LA' IS NOW IN RECOVERY MODE
CACH039I DB2 RESTART TIME 20040726 09294000
CACH040I IMS RESTART TIME 04.208 09:29:40.0
CACH037I RESTART SYSTEM CLOCK BB931C6945765200
CACH038I RESTART LOG SEQ. # 00000000-00000001
CACH062I RECOVERY PROCESSING COMPLETED SUCCESSFULLY
```

If you re-run this job, or supply a control card for a change-capture agent that has recovery data set information and is already in recovery mode, you receive the following WTO messages:

```
CACH061I RECOVERY MODE: SET AGENT IN RECOVERY
CACH048I RECOVERY DATASET WCA008.XSYNC.WCA008LA OPENED
CACH053I AGENT 'WCA008LA' ALREADY IN RECOVERY MODE
CACH062W RECOVERY PROCESSING ENDED WITH WARNINGS
```

Here is the output from the Check Status job after job WCA008LA is manually placed into recovery: mode

```
CACH061I RECOVERY MODE: CHECK AGENT STATUS
CACH048I RECOVERY DATASET WCA008.XSYNC.WCA008LA OPENED
CACH049I AGENT NAME IS 'WCA008LA'
CACH039I DB2 RESTART TIME 20040726 09294000
CACH040I IMS RESTART TIME 04.208 09:29:40.0
CACH037I RESTART SYSTEM CLOCK BB931C6945765200
CACH038I RESTART LOG SEQ. # 00000000-00000001
CACH051I CORRELATION SERVICE REPORTS AGENT IN RECOVERY MODE
CACH049I AGENT NAME IS 'WCA008LA'
```

```
CACH039I DB2 RESTART TIME 20040726 09294000
CACH040I IMS RESTART TIME 04.208 09:29:40.0
CACH037I RESTART SYSTEM CLOCK BB931C6945765200
CACH038I RESTART LOG SEQ. # 00000000-00000001
CACH050I RECOVERY DATASET WCA008.XSYNC.WCA008LA REPORTS AGENT IS NOT IN RECOVERY MODE
CACH030I AGENT 'WCA008LA' IS NOT IN RECOVERY MODE
CACH071I 1 IMS AGENTS ARE IN RECOVERY MODE
CACH062I RECOVERY PROCESSING COMPLETED SUCCESSFULLY
```

In addition to the restart information that is contained in the recovery data set, restart information is also obtained from the correlation service. The restart date and time and restart point are now the same values. If you run the recovery agent and it reports a restart point from both the change-capture agent's recovery data set and the correlation service, the correlation service's information should take precedence.

## Using DBRC to identify log files with change data that needs to be recovered from IMS databases

If you are using DBRC to track the IMS log files for a change-capture agent and that agent goes into recovery mode, you can use the output from the DBRC LIST.LOG ALL report to determine which IMS log files have change data to be recovered.

- You must have recovery data sets for the change-capture agents that are in recovery mode.
- You must run the recovery agent in Check Status mode both to determine which change-capture agents are in recovery mode and to determine the restart points for those change-capture agents.

### About this task

When a recovery agent finds that a change-capture agent is in recovery mode, the recovery agent provides you with this information:

- The date and time of the IMS log record where the recovery of change data must begin. This information appears in two formats: the first is a DB2 timestamp, the second is an IMS DBRC format.
- The system clock value that identifies the IMS log record where the recovery of change data must begin. This value is from the suffix of the identified IMS log record.
- The double-word IMS log record sequence number of the IMS log record where the recovery of change data must begin.

The combination of the system clock value and the log record sequence number identifies a unique position within an IMS log file where the recovery of change data must begin.

With the information that is provided by the recovery agent, you can use DBRC to determine the names of the IMS log files that are associated with the IMS control region that has data that needs to be recovered. You can obtain information about the IMS log files that are associated with an IMS subsystem using the DBRC LIST.LOG ALL command.

An IMS log file is required in the recovery process if the restart time is within the log file's start and stop times, and the restart log record sequence number is within

the range of the IMS log records that are contained in the file. All IMS log files that are created after the restart point must also be input into the IMS log file recovery process.

An IMS log file is not required in the recovery process if the IMS log file's start or stop time is before the restart time that is reported by the recovery agent.

### Procedure

To use DBRC to identify log files with data that needs to be recovered:

- For an IMS batch job with log files being tracked by DBRC, select the IMS log files that need to be input into the recovery process by identifying the IMS log file that was active at time of failure.

This file has a creation date that is on or before the IMS restart time that is displayed and has a starting log record sequence number that is less than or equal to the restart log record sequence number. The initial IMS log file and all other IMS log files that were created after it contain change data that needs to be recovered.

- For an IMS DB/DC or DBCTL subsystem with log files being tracked by DBRC, issue the following command to obtain a list of all of the IMS log files that DBRC knows about for the subsystem:

```
LIST.LOG ALL
```

In these cases the name of the change-capture agent that is in recovery mode will not match the subsystem ID that needs to be supplied, if the DB/DC or DBCTL started procedure name is not the same as the IMS subsystem ID.

If you have multiple DB/DC or DBCTL subsystems, you can restrict the DBRC report to a specific subsystem by specifying the 4-character ID of the subsystem in the SSID parameter.

```
LIST.LOG ALL SSID(4-character_ID_of_the_subsystem)
```

1. Open the report in an editor and search for instances of LOGALL.

Several of these instances will appear in the report. Here is an example:

```
LOGALL
START = 05.066 08:22:19.8 *
DBDS ALLOC=0
105.249 08:23:28.9 LISTING OF RECON
0-----
PRISLD
```

2. Find the instance that does not have any database names associated with it in the report header and that has a subheading of PRISLD. If dual logging is activated, the subheading can be PRISLD or SECSLD. When you are using dual logging, you can use either the primary or secondary log files as input to the recovery process.
3. Find the section of the report that matches your configuration:
  - For a batch application, find the section of the report that lists log file tracking information for all PRILOG files that are being tracked for the batch job. The SSID that is recorded by DBRC is the name of the batch job.
  - For a DB/DC or DBCTL subsystem (if you are planning to use archived log files as input), find the section of the report that lists PRILOG log file tracking information for the DB/DC or DBCTL subsystem ID (SSID) that is associated with the change-capture agent that is in recovery mode.
  - For a DB/DC or DBCTL subsystem (if you are planning to use online logs as input), find the section of the report that lists PRIOLD log file tracking

information for the DB/DC or DBCTL subsystem ID (SSID) that is associated with the change-capture agent that is in recovery mode.

After you find the correct section of the report, you will see a list of log files listed in ascending time sequence.

4. Decide whether to use archived or online log files:
  - Use non-archived online log files or the active log as input into the recovery process if the online logs or active log contain the record that is identified by the restart point, or were created after the restart point.
  - Use archived IMS log files as input into the recovery process if the following conditions are true:
    - The DB/DC or DBCTL subsystem is running.
    - The archived online logs contain the restart point reported by the recovery agent.
  - Use the active online log file if the following conditions are true:
    - The PIROL or SECOLD subheading is listed at the end of the DBRC.LIST LOG ALL report.
    - The log start time is before the restart point, and the first log record number is less than the log number that is identified as the restart point.
    - The log stop time and that last log record numbers are zeros.
5. Select the IMS log files that need to be input into the recovery process are selected by identifying the IMS log file that was active at time of failure. This file has a creation date that is on or before the IMS restart time displayed and has a starting log record sequence number that is less than or equal to the restart log record sequence number. The initial IMS log file and all other IMS log files created after it contain change data that needs to be recovered.

## Example

You are using log file tracking for a change-capture agent named IMSD. This name is identical to the IMS subsystem ID. The name and ID do not have to be identical. However, it is good practice at least to prefix the name of the change-capture agent with the ID of the IMS subsystem. For example, the name of the change-capture agent could be IMSDCR (or IMSDCR1) when the IMS subsystem ID is IMSD.

The recovery agent reports that change-capture agent IMSD is in recovery mode. The recovery agent also reports that IMS log file CXMAINT.IMSD.SLDSP.IMS3.LOGBKUP.G1161V00 is required for the recovery process and gives the following information about the restart point:

```
DB2 RESTART TIME 20020501 16341629
IMS RESTART TIME 02.121 16:34:16.2
RESTART STORE CLOCK B79054D35F0F7640
RESTART LOG SEQ. # 00000000-00140149
```

You run the DBRC LIST.LOG ALL report and find the PRILOG section of the report for SSID=IMSD. The IMS log file CXMAINT.IMSD.SLDSP.IMS3.LOGBKUP.G1161V00 is listed as one of the files being tracked by DBRC. The following information for the log file is in the DBRC report:

```
DSN=CXMAINT.IMSD.SLDSP.IMS3.LOGBKUP.G1161V00          UNIT=3390
START = 02.121 16:33:40.7                               FIRST DS LSN= 0000000000140121
STOP = 02.121 16:38:30.5                               LAST DS LSN= 000000000014BF10
FILE SEQ=0001 #VOLUMES=0001
```

VOLSER=CXA008 STOPTIME = 02.121 16:38:30.5  
CKPTCT=2 CHKPT ID = 02.121 16:38:19.7

This IMS log file is required in the recovery process because the restart time is between the log file's start and stop times, and the restart log record sequence number is within the range of the IMS log records that are contained in the file.

If the IMS log file start or stop time is before the restart time that is reported by the recovery agent, then the IMS log file is not required in the IMS log file recovery process. All IMS log files that are created after the restart point must also be input into the IMS log file recovery process.

## Creating recovery data sets for IMS databases when change-capture agents are in recovery mode

You can recover change data even if you forget to update an IMS job or started task and the change-capture agent is in recovery mode because it was started without a correlation service being active.

### About this task

You must know:

- The date and time that the IMS job or started task was started.
- The name of the first (or only) IMS log file created by IMS.

If a change-capture agent is in recovery mode and the correlation service knows that the agent is in recovery, you do not have to have a recovery data set for that agent. In this scenario, specify a non-existent recovery data set in the IMS recovery agent control file.

### Procedure

To create recovery data sets when change-capture agents are in recovery mode, follow these steps for each of your change-capture agents:

1. Determine when the IMS job or started task was started and identify the IMS log files that are required to recover data.

You can find this information in at least three ways:

- Search the system log for WTO messages that begin with a prefix of CAC. A change-capture agent issues a WTO message when it detects that no correlation service is running.
- Search the system log for the IMS job or started task name to determine when the IMS job or started task was started.
- Identify the creation date or time of the IMS log files that are associated with the IMS job or started task.

2. Allocate an 80-byte fixed-length physical sequential data set.

Define the block size as 80-bytes and allocate only one block. No secondary extents are required because the data set contains only a single record.

Use the following naming standard for recovery data sets:

`&HLVLQUAL.WSDCEP.&JOBNAME`

where `&HLVLQUAL` is a high-level data set name qualifier that the IMS control region job or started task name can access and update. The second-level qualifier is a constant (in the example above, `WSDCEP`). This constant can be any unique identifier up to eight characters. Its purpose is to group the

recovery data sets together so that you can focus on them. The file name suffix should be the job name or started task name of the IMS control region that is being monitored.

3. Edit the file in ISPF and insert one line into the file to define the restart point. The following table describes the content and format of the record.

Table 24. The content and format of a recovery data set

Name	Starting offset	Length	Description
Agent Type	1	4	Identifies the type of agent that is in recovery mode. Set the value to IMS_
Agent Name	5	12	Name of the IMS job or started task that is in recovery mode. The name must be padded with trailing blanks.
DB2 Time Stamp	17	10	Approximate restart time in DB2 timestamp format.
IMS UOR Identifier	27	16	Leave this field blank.
Current <sup>®</sup> UOR Starting Log Record Sequence Number	43	8	The IMS log sequence number of the first IMS log record in the file when the IMS subsystem was started.
Current UOR Starting System Clock	51	8	The system clock value from the log record suffix of the first IMS log record in the file when the IMS subsystem was started.
Oldest UOR Log Record Sequence Number	59	8	The value that must be supplied is the same value that is contained in the Current UOR Starting Log Record Sequence Number field.
Oldest UOR System Clock	67	8	The value that must be supplied is the same value that is contained in the Current UOR Starting System Clock field.

To identify the oldest UOR log record sequence number and the oldest UOR system clock, follow these steps:

- a. Open in an editor the first IMS log file that was created by the IMS region.
- b. Scroll to the right until you reach the end of the first log record.

The IMS log record suffix consists of the last 16-bytes of the log record. In the IMS log record, the suffix consists of the System Clock value followed by the IMS Log Record Sequence Number. Below is the hexadecimal representation of the IMS Log Record Suffix that corresponds to the information that is stored in the recovery data set for this example:

```
B4E9D414000000CF
BDAFF810000000EF
```

- c. Copy and paste this information from the IMS log file into the recovery data set.

The following example shows the content of a recovery data set for change-capture agent DBCD. The information is displayed in hexadecimal format and column numbers are on.

```
***** ***** Top of Data *****
-----
=COLS> 1-----2-----3-----4-----5-----6-----7--
        F-----F-----F-----F-----F-----F-----F--
```



The recovery agent detects duplicated log files and IMS log files that have overlapping system clock timestamps. Additional checks are performed for IMS batch applications to insure that IMS log files are not supplied from a different job. For online regions, log file switches are also tracked to ensure that a required online log file is not missed.

However, if you forget to supply one or more of the log files that need to be input into the recovery process, the changes in these "missed" log files are lost and can not be recovered because the restart point is "pushed" into the future and the correlation service will discard all changes from these IMS log files. If the recovery agent does not detect missing log files, the correlation service might eventually detect this fact. However, by that time the restart point will most likely be after the time when the missing log records were created and therefore, those changes will be unrecoverable.

If you have multiple change-capture agents in recovery mode, you must supply IMS log files for all of these change-capture agents each time you run the recovery agent to recover data from log files. Failure to supply these log files can cause changes to be inserted to the WebSphere MQ message queues in an incorrect chronological order.

Use the following guidelines for identifying IMS log files with data that needs to be recovered:

- Supply IMS log files that were created before the current restart point. The recovery agent reads these files and discards log records that were created before the current restart point.
- If you are supplying more than one IMS log file for a change-capture agent in recovery, you can identify the files in any sequence of control cards in the control file. The recovery agent sorts these files into the correct sequence based on the system clock values that are contained in each IMS log record.
- If you are recovering multiple change-capture agents, you can identify them in any sequence of control cards in the control file. The recovery agent merges IMS log files for multiple change-capture agents in system clock sequence and presents the log records to the correlation service in the correct time sequence.

If you supply an IMS log file that contains log records that were created before the recovery restart point, no IMS log records are sent to the correlation service is zero. The IMS recovery agent automatically filters these types of record outs.

## Procedure

To recover data from IMS log files:

1. Create a control file.

Create a control card for each log file with data that you want to recover. Input the following values in each control card:

- a. For the action identifier, use "L".
- b. For the name of the change-capture agent, specify the name of the change-capture agent that is in recovery mode.
- c. For the region type, specify the type of IMS control region type from which the recovery agent will recover data.
- d. For the data set name, specify the name of the IMS log file with change data that needs to be recovered.

2. Run the recovery agent. The JCL to run this batch job is located in the member CACIMSRA in data set SCACSAMP.
3. Continue performing incremental recoveries until either or both of the following conditions is true: you reach a quiesce point in all control regions when changes are no longer occurring the databases from which you are capturing changes, all IMS control regions can be shut down, or both.
  - A quiesce point exists when no changes are occurring to your databases.
  - All IMS control regions can be shut down.

## Example

Change-capture agent WCA008DA went into recovery mode due to the correlation service being shutdown before the committed UORs that were created by WCA008DA were processed. There were three steps in job WCA008DA that updated IMS. Also, the correlation service was shutdown while executing the first job step that updated IMS data.

In this scenario, three log files need to be input into the recovery process. The IMS log files are being written to generation data set groups. When using generation data set groups, specify the fully qualified data set name and do not use relative generation numbers. Using the fully-qualified name ensures that log files are not processed out of sequence due to the creation of a new generation. Also, in general, when using generation data set groups you need to perform the recovery process up to the latest (current) generation. The following table describes the control cards that contain log file recovery information for change-capture agent WCA008DA.

*Table 25. Contents of control cards for the "Log file recovery" batch job*

Mode	Agent	Region type	Name of data set
L	WCA008DA	2	WCA008.IMS.WCA008DA.LOG..G0002V00
L	WCA008DA	2	WCA008.IMS.WCA008DA.LOG..G0001V00
L	WCA008DA	2	WCA008.IMS.WCA008DA.LOG..G0003V00

In this example, you are requesting a full log recovery and you identified change-capture agent WCA008DA as an IMS batch region. Also, note that the log files were not listed in sequence in the control file. The listing below shows the WTO messages that are issued by the recovery agent when it processes the log files:

```

CACH061I RECOVERY MODE: IMS LOG FILE RECOVERY
CACH055I STARTING LOG FILE SEQUENCE CHECKING
CACH058I LOG FILE SEQUENCE CHECKING COMPLETED
CACH044I AGENT 'WCA008DA' LOG OPENED: WCA008.IMS.WCA008DA.LOG.G0001V00
CACH045I AGENT 'WCA008DA' LOG CLOSED: WCA008.IMS.WCA008DA.LOG.G0001V00
CACH044I AGENT 'WCA008DA' LOG OPENED: WCA008.IMS.WCA008DA.LOG.G0002V00
CACH045I AGENT 'WCA008DA' LOG CLOSED: WCA008.IMS.WCA008DA.LOG.G0002V00
CACH044I AGENT 'WCA008DA' LOG OPENED: WCA008.IMS.WCA008DA.LOG.G0003V00
CACH045I AGENT 'WCA008DA' LOG CLOSED: WCA008.IMS.WCA008DA.LOG.G0003V00
CACH072I BUFFERS SENT 1 RECORDS 4 THROTTLES 1
CACH062I RECOVERY
PROCESSING COMPLETED SUCCESSFULLY

```

Message CACH061I states that the recovery agent is performing IMS log file recovery.

Message CACH055I is issued after the recovery agent reads the control file, verifies that the change-capture agent is in recovery mode, and establishes communications with the correlation service. The message indicates that the recovery agent is verifying that you supplied the correct set of IMS log files. During this process, the recovery agent opens the IMS log files and reads their contents.

Message CACH058I states that the recovery agent successfully sequenced the IMS log files and that it is about to start recovering data. If you supplied an invalid IMS log file, the recovery agent issues one of the following WTO messages and immediately stops with a non-zero return code:

**CACH056E AGENT 'Name' DUPLICATE LOG: Name**

The IMS log file identified in the message has the same starting or ending system time stamp values as another IMS log file that is associated with the agent.

**CACH057E AGENT 'Name' INVALID LOG: Name**

This message is issued when an overlap in system time stamps is discovered with the identified IMS log file and another IMS log file that is associated with the change-capture agent. This message indicates that you specified the name of an IMS log file that is associated with some other IMS control region.

**CACH075E AGENT 'Name' MISSING ONLINE LOG BEFORE: Name**

Message CACH075E is issued when you are recovering changes for a DB/DC or DBCTL subsystem. For these types of subsystems, the IMS recovery agent monitors the IMS type 43 (log data set control) records in the IMS log files that you input into the recovery process. These records identify when log file switches occur. Message CACH075E is issued if the IMS recovery agent found one of these records, which indicates that a switch occurred. However, a log file that identifies the name of the new log was not supplied, but another online log was supplied that was created after the switch. Message CACH0H75E identifies the name of the IMS log file that was active when the switch occurred.

The recovery agent issues message CACH077I when an IMS log file was supplied that is not required. This message identifies the name of the extraneous log file that was discarded.

Message CACH044I is issued each time that an IMS log file is opened for recovery processing. After the IMS log is opened, its contents are read and IMS log records of interest are sent to the correlation service for processing.

When the end of the file is reached, WTO message CACH045I is issued. If the recovery agent encounters an error when opening an IMS log file or processing its contents, the recovery agent issues a WTO message and immediately terminates processing. In this example, you can see three sets of the CACH0044I/CACH045I messages, one for each of the IMS log files that are processed. Also notice that the sequence in which these messages are issued indicates that the recovery agent resequenced the IMS log file processing order.

After all log files are processed, message CACH072I is issued. This message provides summary information about the communications that the recovery agent had with the correlation service. The message identifies the number of buffers sent (XM data grams), the total number of IMS log records that were sent to the correlation service, and the number of times throttle requests were sent to and responses were received from the correlation service.

The last message issued is CACH062I or CACH062E. These messages tell whether the data in the IMS log files were recovered successfully or whether errors were encountered. In this example, the recovery agent processed all IMS log files successfully and reported no errors.

## Returning change-capture agents for IMS databases to active mode

After you complete the recovery process for all of your change-capture agents that are in recovery mode, you use the recovery agent to place those change-capture agents back into active mode.

### About this task

Change-capture agents can successfully be placed back into active mode when the following conditions are met:

- The IMS control region for the failed agent is not active, or the databases that are being monitored are stopped so that changes cannot occur, or you have reached a logical quiesce point and you know that changes to the monitored databases will not occur.
- The IMS log file recovery process is completed for each IMS log file (or archived log file for DB/DC or DBCTL subsystems) that was created by the IMS control region from the failure point up to and including the last IMS log file that was created by the IMS control region where the change-capture agent failed.

When you run the recovery agent to activate change-capture agents, the recovery agent informs the correlation service that the change-capture agents are no longer in recovery mode. The recovery agent informs the correlation service that the recovery agent completed recovery operations and that the recovery agent terminated. This activation sequence causes the correlation service to issue WTO messages that indicate that the recovery agent started recovery, completed recovery, and then terminated. After this activation process ends and the change-capture agent starts sending data to the correlation service, the correlation service issues a WTO message to indicate that the user is receiving change data. The recovery agent also deletes the contents of the recovery data sets for the change-capture agents.

If you activate a change-capture agent for a DB/DC or DBCTL subsystem and the IMS control region is running, the recovery agent receives an error when it attempts to delete the contents of the recovery data set. The recovery agent stops with a return code of 4 and the job output contains the message "LSCX872 File in use by another job." The change-capture agent is nevertheless active.

### Procedure

To activate change-capture agents:

1. Create a control file.

In the control file, create a control card for each change-capture agent that you want to activate. Specify the following entries in each control card:

- a. For the action identifier, use "A".
- b. For the name of the change-capture agent, specify the name of the change-capture agent that you want to activate.
- c. For the data set name, specify the name of the recovery data set that is used by the change-capture agent.

- Run the recovery agent. The JCL to run this batch job is located in the member CACIMSRA in data set SCACSAMP.

### Example

Assume that agent WCA008DA is in recovery mode and that you recovered the three required IMS log files. If you did not run job WCA008DA again during the recovery process, change-capture agent WCA008DA meets the requirements for activation. You supply the following control file to reactivate the change-capture agent.

Table 26. Sample control file for reactivating a change-capture agent

Mode	Change-capture agent	Name of data set
A	WCA008DA	WCA008.XSYNC.WCA008DA

The recovery agent returns the messages listed below:

```
CACH061I RECOVERY MODE: ACTIVATE AGENT
CACH054I PREPARING TO ACTIVATE AGENT WCA008DA
CACH031I AGENT WCA008DA SWITCHING TO ACTIVE MODE
CACH062I RECOVERY PROCESSING COMPLETED SUCCESSFULLY
```

The first message indicates that the recovery agent is preparing to place one or more change-capture agents back into active mode. For each change-capture agent that is being activated, the recovery agent issues message CACH054I, which informs you that the recovery agent is about to attempt to reactivate the change-capture agent. If the change-capture agent is successfully activated, the recovery agent issues message CACH031I. If the agent is reported as not being in recovery or there are problems communicating with the correlation service, message CACH031I is not issued and you will see other error messages that identify the problem.

As part of activating a change-capture agent, the recovery agent attempts to delete the contents of the recovery data set. If you are activating a change-capture agent for a DB/DC or DBCTL subsystem and the control region is active, the recovery agent is not able to delete the contents of the recovery data set. The recovery agent cannot delete the contents because the recovery data set is still open in the IMS control region. In these instances, the change-capture agent is still activated. However, the recovery agent issues these two messages:

```
CACH047E RECOVERY DATASET OPEN ERROR FOR recovery-data-set-name ERRNO: 19
CACH062W RECOVERY PROCESSING ENDED WITH WARNINGS
```

The recovery agent then ends with a return code of 4.

---

## Recovering change data from CICS VSAM files

The process of recovering change data from CICS VSAM files depends on how you are capturing that data.

- If you are using a file control exit, see “Recovering change data when capturing changes with file control agents” on page 207.
- If you are using an auto-journal agent, see “Recovering change data when capturing changes with auto-journal agents” on page 207
- If you are using a log-reading agent, see “Recovering change data when capturing changes with log-reading agents” on page 207

## Recovering change data when capturing changes with auto-journal agents

Because the auto-journal agent reads from the auto-journals, there is no need for any recovery process.

## Recovering change data when capturing changes with log-reading agents

The log-reading change-capture agent can also recover change data. When the agent starts, it determines whether it is in recovery mode or active mode. If the agent is in recovery mode, the agent queries the correlation service for the restart point and starts reading the log streams at that restart point. The agent also indicates that it is in recovery mode by writing a message in the change capture log.

To stop and start the log-reading agent, see “Starting and stopping log-reading agents when the data server where they are configured is running” on page 131.

The log-reading agent automatically switches to active mode after reaching the end of file on the log streams. The agent can reach the end of the log streams when CICS is shut down or quiesced, or when transaction activity decreases enough.

## Recovering change data when capturing changes with file control agents

If you are capturing data with a file control exit, you can use the auto-journal agent or your own process for recovering change data if your configuration goes into recovery mode.

When errors occur in the change-capture agent, correlation service, or distribution service, the system enters recovery mode. In this mode, the capturing of changes stops and messages are no longer sent on the WebSphere MQ message queues.

However, database updates continue to be made while change capture is stopped. To capture those changes, you must first correct the problem that caused change capture to stop. Then, you must recover the changes, using either your own recovery process or the auto-journal recovery agent to begin reading the change data in the log stream from the point in time at which the error occurred and continue up to the current point in the log stream.

### Recovering data with the auto-journal agent

To recover the changes that are made to your file while change capture is in recovery mode, you can use the auto-journal agent. If you want to use this agent, you must define auto-journals for your VSAM files before you begin capturing changes that are made to those files.

You configure the auto-journal agent with a service information entry in the configuration file for the data server. (The sample member for the configuration file is CACCSCF in the SCACCONF data set.) When change capture enters recovery mode, you issue a command to start the auto-journal agent, which runs in the same address space as the correlation service. The auto-journal agent reads the auto-journals that you specify from the restart point passed to it by the correlation

service. The auto-journals agent reads the auto-journals and processes the logs until the end of the log stream, all the while sending change data to the correlation service.

The typical steps in the process of recovering change data are:

1. When configuring change capture, you define auto-journals for the VSAM files that you want to capture changes from.
2. A failure occurs while changes are being captured and you are notified that change capture is in recovery mode.
3. You quiesce CICS activity.
4. You issue the command to start the auto-journal agent.
5. The auto-journal agent processes the specified auto-journals.
6. The auto-journal agent reads to the end of the log stream, reactivates the change-capture agent, and terminates.
7. The file control agent resumes capturing changes when you resume normal CICS transaction activity.

### **User-defined recovery process**

You can use your own restore utility to recover change data, if you do not want to define auto-journals for your VSAM files. You can also choose not to recover any change data, if you do not need that data.

The following steps describe the recovery process where no auto-journals have been defined:

1. A failure occurs and you are notified that change capture is in recovery mode.
2. (Optional) You quiesce CICS activity to prevent applications from making further updates to your file.
3. (Optional) You recover change data to the restart point that you want using your own restore utility.
4. You restart the correlation service, specifying COLDSTART on the correlation service's service information entry, to reactivate change capture. When normal CICS transaction activity resumes, change capture progresses as it did before the error occurred.

## **Starting the auto-journal agent to recover change data**

Use an MTO command to start the auto-journal agent when you need to recover change data.

### **Before you begin**

You must have a service information entry defined for the auto-journal agent in the configuration file (member CACCSCF in the SCACCONF data set) for the data server. The service information entry must contain the RECOVERY keyword in field ten.

### **Procedure**

To start the auto-journal agent:

In an MTO command interface, issue the following command:

```
F CACCS,START,SERVICE=name
```

where *name* is the name specified for the auto-journal agent in the service information entry.

---

## Administering NVA instances with the NVA SVC Manager

Use the NVA SVC Manager to load NVA load modules into DLPA and to remove these load modules from DLPA when it is safe to do so. Use the NVA SVC Manager also to install and uninstall, enable and disable, and set filters on NVA instances. You can also find out the NVA SVC intercepts that are in an SVC chain.

The NVA SVC Manager is distributed as SCACLOAD load module name CACNVAIN. Sample JCL to execute the NVA SVC Manager is located in member CACNVASM in the SCACSAMP data set. The NVA SVC Manager accepts input from the command line parameter. Before running the JCL you need to update it to provide a valid job card and modify the CAC parameter to identify the high-level qualifier for the Classic libraries.

### Filters to restrict the change data that NVA instances capture

Both the Enable (EN) and the Set Filter (SF) commands let you configure filters that restrict the changes that are captured by an NVA instance.

A filter can restrict an NVA instance to capturing changes that are due to a job, a user ID, or both a job and user ID.

The value that you specify for a job or a user ID can be in one of the following formats:

**A name of one to eight characters.**

The name must exactly match the user ID or name of the job that is available when the NVA instance processes an open request.

You can use an asterisk (\*) as a wildcard character at the end of the name or as the only character in the name. Using the asterisk alone is equivalent to turning filtering off.

@ You can specify this symbol for either the user ID or for the job name. This symbol specifies that you want to use the default filtering value from the options module that is linked to the NVA load module.

**Null value**

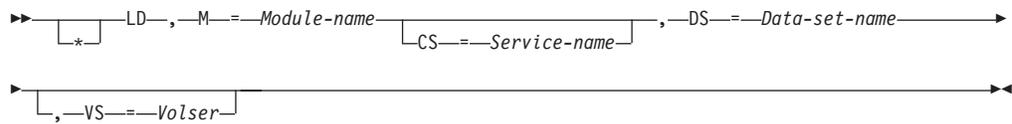
A null value turns off filtering. For example, you could specify U= to turn off filtering by username or J= to turn off filtering by job name.

### Loading NVA load modules into DLPA and installing them

The load (LD) command copies the NVA load module and the extended SVC load module into DLPA. This command also installs the NVA instance into the SVC chain, if the load is successful.

You must load the extended SVC load module IGX00060 once into DLPA. You must load the load module for each NVA SVC intercept instance (CACNVA00, CACNVA01,..., CACNVA $n$ ) into DLPA before installing the NVA SVC intercept on the z/OS LPAR.

### Command line syntax



## Parameters

- \* Optional parameter that runs the NVA SVC Manager in test mode. The NVA SVC Manager validates the parameters and verifies that your environment is properly configured for loading the load modules. However, the NVA SVC Manager does not load the load modules.

### LD

Required parameter that specifies the load command.

### M=*Module-name*

Required parameter that identifies the name of the NVA load module that the NVA SVC Manager is to load and install. The name of the shipped NVA load module is CACNVA00.

### CS=*Service-name*

Optional parameter that identifies the name of the correlation service name that must exist in the CACE1NVA module that is linked into the NVA load module that is identified by the M parameter.

You can include this parameter as a means of verifying that the correlation service is in fact associated with the NVA load module. For example, you can say M=X,CS=Y and the NVA SVC manager will verify that X in fact is associated with Y.

### DS=*Data-set-name*

Required parameter that identifies the data set that contains the NVA load module to load and install and the NVA extended SVC load module (IGX00060). The data set must be APF-authorized.

### VS=*Volser*

Optional parameter that identifies the volser of the data set that is specified by the DS parameter. The volser is not required if the NVA SVC Manager can locate the data set without using the volser.

## Results

If any errors are detected during load processing, the LD command fails and DLPA is updated with any modules that were successfully loaded up to the failure point.

If no errors are detected, the requested NVA load module is loaded into DLPA. If it does not already exist in DLPA, the NVA extended SVC load module is also loaded into DLPA.

During a successful load operation, the following WTO messages are issued for each load module that is loaded into DLPA:

```

CACH860I csect LOADING MODULE <Module-name>
CACH862I csect MODULE <Module-name> LOADED AT EP=<Entry-point-address> LOADED NAME =<CSECT-name>

```

If the loads are successful, the NVA open and close intercepts are installed into the SVC chain. The NVA instance is disabled and must be enabled before capture operations can begin.

## Example

```
LD,M=CACNVA00,DS=CAC.V910.LOADLIB
```

This example loads NVA module CACNVA00 from authorized data set CAC.V910.LOADLIB into dynamic LPA and installs CACNVA00 as an NVA instance into the SVC chain.

If CACNVA00 is already in LPA, or the NVA instance represented by CACNVA00 is already installed in the SVC chain, the LD command fails and the load and installation of this module does not succeed.

## Uninstalling NVA instances from SVC chains and removing those instances from DLPA

Use the remove (RM) command to uninstall an NVA instance from the SVC chain and delete the NVA load module from DLPA. Use this command only if you loaded and installed the module with the load (LD) command.

The specified NVA instance must be uninstallable. In other words, it must be the first (or head) NVA instance in the SVC chain. If the NVA instance is not first in the SVC chain, then it will not be uninstalled and removed unless the force option (F=Y) is specified.

If the NVA instance is uninstallable, then it is first uninstalled (as would be done with the UN command), which means changing the SVC table to point to the next SVC handler in the chain. (The NVA instance maintains the information about the next SVC handler in the chain). If no uninstall errors are encountered, then NVA removes the NVA instance from DLPA. In addition, if this was the last NVA instance to be removed from the chain, then module IGX00060 is also removed from DLPA.

If the NVA instance is successfully uninstalled but errors are detected when attempting to remove the NVA instance from DLPA, error messages will indicate the cause of error, and the NVA instance will not be removed from DLPA.

## Command line syntax

```
RM, M=Module-name, CS=Service-name, F=Y
```

## Parameters

- \* Optional parameter that runs the NVA SVC Manager in test mode. The NVA SVC Manager validates the command line parameters and verifies that your environment is properly configured for removing an NVA instance. However, the NVA SVC Manager does not remove the instance.

### RM

Required parameter that specifies the remove command.

**M**=*Module-name*

Identifies the name of the NVA load module that the NVA SVC Manager is to uninstall and remove. The name of the shipped NVA load module is CACNVA00. When you remove an NVA instance, provide the name of the load module that you specified in the NAME statement when you re-linked NVA after customizing the NVA options module.

The NVA load module that you specify must be the last NVA instance in the SVC chain. Otherwise, the NVA SVC Manager will not remove and uninstall the NVA instance.

If you specify both the M and the CS parameters, the NVA SVC Manager uses the value of the M parameter.

**CS**=*Service-name*

Identifies the name of the correlation service name that must exist in the CACE1NVA module that is linked into the NVA load module.

When you use this keyword, the NVA SVC Manager verifies that the specified correlation service matches the value for SrvNam in the CACE1NVA options module that is linked to the NVA instance. The RM command runs only if the names match.

**F=Y**

Indicates force mode. Use this option to replace an existing NVA instance with an updated or newer version without having to first uninstall NVA instance and then re-install it.

## Results

The NVA SVC Manager uninstalls the NVA load module, changing the SVC table to point to the next SVC handler in the chain. The NVA SVC Manager next removes the NVA load module from DLPA.

If you remove the last NVA load module in DLPA, the extended SVC module (IGX00060) is also removed from DLPA.

## Example

```
RM,M=CACNVA00
```

This example uninstalls NVA module CACNVA00 from the SVC chain and removes CACNVA00 from dynamic LPA.

If NVA module CACNVA00 is not installed as an NVA instance in the SVC chain, the RM command fails. If CACNVA00 is installed, it is uninstalled from the SVC chain. If CACNVA00 is installed in dynamic LPA, it is removed from dynamic LPA. In addition, if this is the last NVA instance to be uninstalled from the SVC chain, module IGX00060 is uninstalled from the SVC chain and is also removed from dynamic LPA if it exists there.

## Installing NVA instances into SVC chains when not using DLPA

The install (IN) command installs an NVA instance into the SVC chain if you are using FLPA, MLPA, or PLPA.

Each NVA SVC intercept instance is a load module (CACNVA00, CACNVA01, and so on) that is linked with a CACE1NVA module which contains the name of a unique correlation service.

Use this command if the NVA instance was loaded into LPA either at the time of the initial program load (IPL) or manually.

NVA is disabled after you install it. You must enable NVA with the enable command (EN).

## Command line syntax

► `[*] IN, -M=Module-name [, -CS=Service-name] [-F=Y]` ►

### Parameters

\* Optional parameter that runs the NVA SVC Manager in test mode. The NVA SVC Manager validates the command line parameters and verifies that your environment is properly configured for installing the load modules. However, the NVA SVC Manager does not install the load modules.

**IN** Required parameter that specifies the install command.

**M**=*Module-name*

Required parameter that identifies the name of the NVA load module that the NVA SVC Manager is to install. The name of the shipped NVA load module is CACNVA00. When you install an NVA instance, provide the name of the load module that you specified in the NAME statement when you re-linked NVA after customizing the NVA options module (CACE1NVA).

**CS**=*Service-name*

Optional parameter that identifies the name of the correlation service name that must exist in the CACE1NVA module that is linked into the NVA load module that is identified by the M parameter.

You can include this parameter as a means of verifying that the correlation service is in fact associated with the NVA load module. For example, you can say M=X,CS=Y and the NVA SVC manager will verify that X in fact is associated with Y.

**F**=Y

Indicates force mode. Use this option to replace an existing NVA instance with an updated or newer version without having to first uninstall NVA instance and then re-install it.

### Example

```
IN,M=CACNVA00
```

This example installs CACNVA00 as an NVA instance into the SVC chain.

If the NVA instance represented by CACNVA00 is already installed in the SVC chain, the IN command fails and the installation of this module does not succeed.

## Uninstalling NVA instances from SVC chains when not using DLPA

The uninstall (UN) command removes an NVA instance from the SVC chain if you loaded the NVA load module into FLPA, MLPA, or PLPA.

After you successfully run this command, you can safely remove the NVA load module and extended SVC load module (provided that there are no other NVA instances installed) from LPA.

You can uninstall an NVA instance only if it is the last SVC intercept in both the open and close SVC chains. If you installed any SVC intercepts after installing the NVA instance that you want to uninstall, you must first uninstall those subsequent SVC intercepts. If these subsequent SVC intercepts are other NVA instances, you

can use the NVA SVC Manager to uninstall them. If these subsequent SVC intercepts were created by third-party software or other IBM software, refer to the appropriate documentation for instructions on removing those SVC intercepts.

If you cannot uninstall an NVA instance, you can deactivate it with the DI command in the NVA SVC Manager. A deactivated NVA instance is still operational; however, it immediately transfers control to the next SVC in the chain.

## Command line syntax

```
► UN, [M=Module-name | CS=Service-name] ►
```

## Parameters

### UN

Required parameter that specifies the uninstall command.

### M=*Module-name*

Identifies the name of the NVA load module that the NVA SVC Manager is to uninstall. The name of the shipped NVA load module is CACNVA00. When you uninstall an NVA instance, provide the name of the load module that you specified in the NAME statement when you re-linked NVA after customizing the NVA options module (CACE1NVA).

If you specify both the M and the CS parameters, the NVA SVC Manager uses the value of the M parameter.

### CS=*Service-name*

Identifies the name of the correlation service name that must exist in the CACE1NVA module that is linked into the NVA load module.

## Example

```
UN,M=CACNVA00
```

This example uninstalls loads NVA module CACNVA00 from the SVC chain.

If NVA module CACNVA00 is not installed as an NVA instance in the SVC chain, the UN command fails. If CACNVA00 is installed, it is uninstalled from the SVC chain. Note that if CACNVA00 is installed in dynamic LPA (DLPA), it is not removed from DLPA.

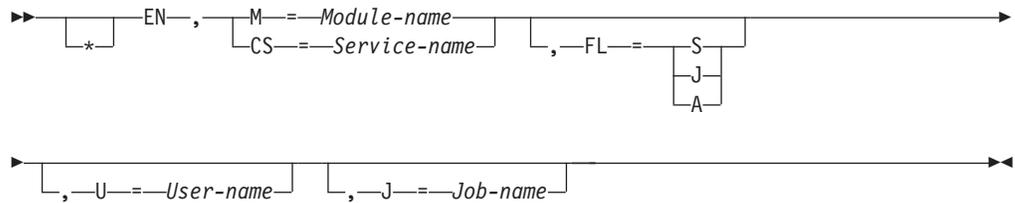
## Enabling NVA instances to capture changes

Use the enable (EN) command to enable an NVA instance to capture changes.

After you enable an NVA instance, that instance processes all SVC open and close requests before passing them to the next handler in the SVC chain.

You can also configure a filter to restrict the changes that the NVA instance captures. The filter persists until you change it or remove it. For example, if you set a filter when you enable an NVA instance that you set a filter for, disable the instance, and then re-enable the instance, the filter remains in effect. You can show the filters that are active for NVA instances by using the query SVC chain (QU) command.

## Command line syntax



## Parameters

- \* Optional parameter that runs the NVA SVC Manager in test mode. The NVA SVC Manager validates the command line parameters and verifies that your environment is properly configured for enabling the load modules. However, the NVA SVC Manager does not enable the load modules.

### EN

Required parameter that specifies the enable command.

### M=*Module-name*

Identifies the name of the NVA load module that the NVA SVC Manager is to enable. The name of the shipped NVA load module is CACNVA00.

### CS=*Service-name*

Identifies the name of the correlation service name that must exist in the CACE1NVA module that is linked into the NVA load module that is identified by the M parameter.

### FL=S | J | A

Indicates the type of enablement.

**S** Enables SVC intercepts only.

**J** Enables JRNAD processing only.

**A** Enables both SVC intercepts and JRNAD processing.

### U=*User-name*

Optional parameter that identifies the user ID that must be associated with the job that opens an ACB for a VSAM file to consider the file eligible for change capture.

### J=*Job-name*

Optional parameter that identifies the job or started task name that must be associated with the ACB for a VSAM file to consider the file eligible for change capture.

## Examples

### Example 1

```
EN,M=CACNVA00,FL=A,U=,J=ABC*
```

Enables the NVA instance that is represented by module CACNVA00, and sets *User-name* filter to blanks and *Job-name* filter to ABC\*.

### Example 2

```
EN,M=CACNVA00
```

Outputs the current status of the NVA instance that is represented by module CACNVA00, but no enablement occurs.

### Example 3

```
EN,M=CACNVA00,FL=A
```

Enables the NVA instance that is represented by module CACNVA00.

## Disabling NVA instances from capturing changes

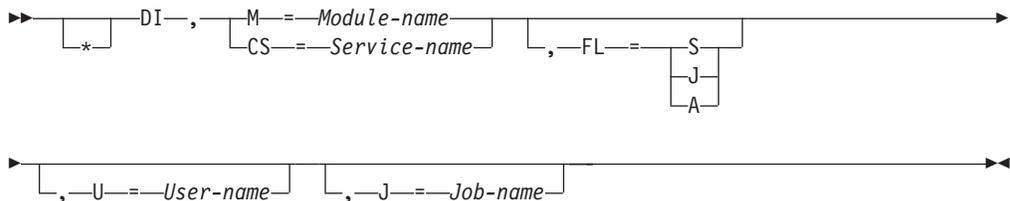
Use the disable (DI) command to disable an NVA instance, preventing the instance from capturing changes to any VSAM files that are on the same z/OS LPAR.

### Results

If a disabled NVA instance receives an open or close requests, the instance passes the requests to the next SVC in the SVC chain.

If an NVA instance is capturing changes for a VSAM file at the time that you issue the disable command, the instance stops change capture immediately.

### Command line syntax



### Parameters

\* Optional parameter that runs the NVA SVC Manager in test mode. The NVA SVC Manager validates the command line parameters and verifies that your environment is properly configured for disabling the load modules. However, the NVA SVC Manager does not disable the load modules.

**DI** Required parameter that specifies the disable command.

**M=Module-name**

Identifies the name of the NVA load module that the NVA SVC Manager is to disable. The name of the shipped NVA load module is CACNVA00.

**CS=Service-name**

Identifies the name of the correlation service name that must exist in the CACE1NVA module that is linked into the NVA load module that is identified by the M parameter.

**FL=S | J | A**

Indicates the type of disablement.

**S** Disables SVC intercepts only.

**J** Disables JRNAD processing only.

**A** Disables both SVC intercepts and JRNAD processing.

**U=User-name**

If you want to set a filter for the NVA instance, this parameter identifies the new username filter specification. The filter is activated when you enable the NVA instance.

J=*Job-name*

If you want to set a filter for the NVA instance, this parameter identifies the new job name filter specification. The filter is activated when you enable the NVA instance.

## Examples

### Example 1

```
DI,M=CACNVA00,FL=A,U=,J=ABC*
```

Disables the NVA instance that is represented by module CACNVA00, and sets *User-name* filter to blanks and *Job-name* filter to ABC\*.

### Example 2

```
DI,M=CACNVA00
```

Outputs the current status of the NVA instance that is represented by module CACNVA00, but no disablement occurs.

### Example 3

```
DI,M=CACNVA00,FL=A
```

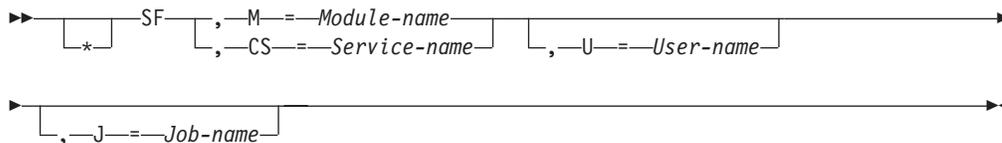
Disables the NVA instance that is represented by module CACNVA00.

## Setting filters to restrict the changes that NVA instances capture

Use the set filter (SF) command to set, update, or remove filtering information that tells an NVA instance which changes to capture. The changes take effect immediately.

The filter persists until you change it or remove it. For example, if you set a filter for an NVA instance, disable the instance, and then re-enable the instance, the filter remains in effect. You can show the filters that are active for NVA instances by using the query SVC chain (QU) command.

### Command line syntax



### Parameters

- \* Optional parameter that runs the NVA SVC Manager in test mode. The NVA SVC Manager validates the command line parameters and verifies that your environment is properly configured for setting filters. However, the NVA SVC Manager does not actually set filters.

**SF** Required parameter that specifies the set filter command.

**M=***Module-name*

Identifies the name of the NVA load module that the NVA SVC Manager is to set a filter for. The name of the shipped NVA load module is CACNVA00.

**CS=***Service-name*

Identifies the name of the correlation service communicates with the NVA

instance that you want to set the filter for. The name of the correlation service must exist in the CACE1NVA module that is linked into the NVA load module that is identified by the M parameter.

**U=***User-name*

Optional parameter that filters VSAM files by the user ID that is associated with the job that opens them. Only these files will be eligible for change capture.

**J=***Job-name*

Optional parameter that filters VSAM files by the job or started task that opens them. Only these files will be eligible for change capture.

## Examples

### Example 1

```
SF,M=CACNVA00,U=,J=ABC*
```

This example sets the *User-name* filter to blanks and the *Job-name* filter to ABC\*.

### Example 2

```
SF,M=CACNVA00,U=@,J=JOB1
```

This example sets the *User-name* filter to the original *User-name* filter in CACE1NVA and the *Job-name* filter to JOB1.

### Example 3

```
SF,M=CACNVA00,J=*
```

Sets the *Job-name* filter to \* and leaves the *User-name* filter unchanged.

## Querying the NVA instances that are in SVC chains

Use the query SVC chain (QU) command to display information about one or all NVA instances in the SVC chain.

### Command line syntax

```
➔ QU —, —M —= —Module-name —, —CS —= —Service-name —, —ALL —➔
```

### Parameters

**QU**

Required parameter that specifies the query command.

**M=***Module-name*

Identifies the name of the NVA load module that the NVA SVC Manager is to query. The name of the shipped NVA load module is CACNVA00.

**CS=***Service-name*

Identifies the name of the correlation service that must exist in the CACE1OPT module that is linked into the NVA load module that is identified by the M parameter.

**ALL**

If specified without M and CS, this parameter displays all NVA instances in the SVC chain, beginning with the most recently installed NVA instance.

If specified with either M or CS, this parameter displays all NVA instances in the SVC chain, beginning with the specified instance.

## Examples

### Example 1

```
QU,M=CACNVA00
```

This example queries and displays information for the installed NVA instance that is represented by module CACNVA00.

### Example 2

```
QU,M=CACNVA00,ALL
```

This example queries and displays information for the installed NVA instance that is represented by module CACNVA00, and then queries and displays information for the remaining installed NVA instances in the SVC chain.

### Example 3

```
QU,ALL
```

This example queries and displays information for all of the installed NVA instances in the SVC chain, beginning with the first installed NVA instance.

## Setting trace levels for NVA instances

Use the Trace (TR) command to set the trace level for various trace options for an NVA instance. You can also use this command to display the current trace options and trace level for an NVA instance.

**Note:** Use this command only when working with IBM support.

### Command line syntax

```
TR [ * ] [ , -M - - - Module-name ] [ , -CS - - - Service-name ] [ , -O - C - J - A - - Trace-value ]
```

### Parameters

- \* Optional parameter that runs the NVA SVC Manager in test mode. The NVA SVC Manager validates the command line parameters and verifies that your environment is properly configured for setting trace values. However, the NVA SVC Manager does not set trace values.

#### TR

Required parameter that specifies the trace command.

#### M=Module-name

Identifies the name of the NVA load module that the NVA SVC Manager is to set a trace value for. The name of the shipped NVA load module is CACNVA00. When you install an NVA instance, provide the name of the load module that you specified in the NAME statement when you re-linked NVA after customizing the NVA options module (CACE1NVA).

#### CS=Service-name

Identifies the NVA instance by means of the name of the correlation service that the NVA instance is configured to communicate with. The name must exist in the CACE1NVA module that is linked into the NVA load module.

**OCJA**=*Trace-value*

Optional parameter that sets trace options and the trace level for those options. If you do not specify this parameter, the trace command displays the current trace options and trace level for the specified NVA instance.

The trace options are:

- O** Open SVC processing
- C** Close SVC processing
- J** JRNAD processing
- A** All (Open, Close, JRNAD processing)

You can specify any combination of the trace options O, C, and J.

*Trace-value* is a number from 0-9 that determines the trace level for the trace options. The trace level determines the amount of information that is recorded in the log when the NVA instance performs any of the actions that are designated by the trace options that you select.

---

## Displaying information about NVA instances and associated correlation services

Use NVA commands from the standard z/OS modify MTO command interface (*/f job-name*) to display information about NVA instances and correlation services that receive change data from NVA instances.

The following example shows the general format of the commands:

```
/f Job-Name,CMD,Service-Name,'NVA,...'
```

*Job-Name*

The name of the data server job or started task.

*Service-Name*

The Cross Memory data queue that the NVA instance uses to communicate with the correlation service.

## Generating reports when capturing changes with NVA instances

You can generate several different types of reports that can help you to administer your NVA instances.

### Generating reports on the jobs that change VSAM files that are monitored by NVA instances

Use the REPORT CCA command to view information about running jobs that are updating files that are being monitored for changes by NVA instances that communicate with a single correlation service.

The following information is displayed for each job:

- How much activity the job has generated for the NVA instance that is capturing changes to the VSAM files that the job is updating.
- Which VSAM files the job is updating that an NVA instance is monitoring for changes.

For each VSAM file that is listed, the following information is displayed:

- The number of changes that the job made to the file and that were captured

- The time that the last change was received from the NVA instance.

## Syntax

```
▶▶ CMD, Service-name, NVA, REPORT, CCA ALL Job-name ▶▶
```

## Parameters

### *Service-name*

Required parameter that names the XM queue that the NVA instance uses to communicate with a correlation service.

### ALL

Shows information about all active NVA instances.

### Name=*Job-name*

Shows information about a particular job.

## Generating reports on correlation services that are associated with NVA instances

Use this command to display information about all correlation services on a z/OS LPAR that share a filter table that is maintained by a name service.

When you use Classic Data Architect to map a relational table to a VSAM file, you specify a Cross Memory (XM) queue. The NVA instance that capture changes for the table sends change data on that queue to a correlation service. Multiple VSAM files can share the same XM queue, with the change data for each file going to the same correlation service.

You can use this command to find out the status of the correlation services that match these two criteria:

- The correlation services receive the change data on the XM queues for the tables that you mapped to your VSAM files.
- The correlation services share a filter table that is maintained by a name service.

This command is useful for checking the status of correlation services before you start a nightly batch run that updates your VSAM files, for example.

After you issue this command, WTO message CACG780I is displayed, followed by a series of CACG781I WTO messages that display identity information for each active correlation service that shares a filter table.

## Syntax

```
▶▶ CMD, Service-name, NVA, REPORT, CS ▶▶
```

## Parameters

### *Service-name*

Required parameter that names the XM queue that an NVA instance uses to communicate with a correlation service.

## Generating reports about VSAM files

Use the REPORT FILE command to view information about VSAM files that NVA instances are monitoring for changes.

## Syntax

```
►►—CMD—,—Service-name—,—NVA—,—REPORT—,—FILE—STATUS—,—NAME—=—Name—”—————►►  
                                          └—ALL—,—NAME—=—Name—┘
```

## Parameters

### *Service-name*

Required parameter that names the XM queue that the NVA instance uses to communicate with a correlation service.

### STATUS

Causes the command to report information that is maintained in the filter table that is used by the correlation service.

### NAME=*Name*

Use the NAME keyword to restrict the output to a specific file or group of files. To see a group of files, specify a wildcard at the end of the name. For example, NAME=ABC\* displays information about all file names that begin with ABC. NAME=\* displays information about all files that NVA instances are monitoring for changes.

### ALL

Requests information about what is stored in the filter table and any processing statistics that the correlation service might have for the file. If the correlation service is not responsible for processing the changes for a file, the correlation service displays only filter table information for the file.

### NAME=*Name*

Use the NAME keyword to restrict the output to a specific file or group of files. To see a group of files, specify a wildcard at the end of the name. For example, NAME=ABC\* displays information about all file names that begin with ABC. NAME=\* displays information about all files that NVA instances are monitoring for changes.

## Results

The output of the REPORT FILE command is displayed in columnar format by a series of messages. Figure 1 shows an example of the output for the REPORT FILE,STATUS command.

```
CACG770I FILE NAME                               XM URL      ORDINAL # B E S TAB SUB MESSAGE #  
CACG771I FILE1                                  XM1/NVA1/NVA1 000000001 N N A 001 001  
CACG771I 1234567890123456789012345678901234 XM1/XXXX/XXXX NNNNNNNNN Y N A NNN NNN  
CACG772I Number FILES LISTED
```

Figure 9. Sample output for the REPORT FILE STATUS command

Message CACG770I is the header row for the output and displays the headers for the columns in the output. Message CACG771I displays the values for each VSAM file.

Figure 2 shows an example of the output for the REPORT FILE, ALL command.

```

CACG770I FILE NAME                                XM URL      ORDINAL # B E S TAB SUB MESSAGE #
CACG773I      OPENS      CLOSES      DELETES      INSERTS      UPDATES      READS
CACG771I 1234567890123456789012345678901234 XM1/XXXX/XXXX NNNNNNNNNN Y N A NNN NNN
CACG774I nnnnnnnnnn nnnnnnnnnn NNNNNNNNNNNNNN NNNNNNNNNNNNNN NNNNNNNNNNNNNN NNNNNNNNNNNNNN
CACG772I Number FILES LISTED

```

Figure 10. Sample output for the REPORT FILE, ALL command

Message CACG773I is the header row for the additional output and displays the headers for the columns in this additional output. Message CACG774I displays the values for each VSAM file.

Table 1 describes the information that appears when you use either the STATUS keyword or the ALL keyword.

Table 27. Information that appears when you use either the STATUS or ALL keywords

CACG770I Header	CACG771I Value
FILE NAME	Data set name for a file that is eligible for change capture.
XM URL	The name of the XM queue that identifies the correlation service that receives change data for the file
ORDINAL #	Current ordinal number assigned to a file. This number indicates how many times the file was updated since an NVA instance started capturing the changes.
B	Flag to indicate whether before images are being captured. The values are Y and N.
E	The Capture Failure Indicator, which indicates whether capture failed for the file. The values are Y and N. If the value is Y, look in the Message # column for the system message number that identifies the cause of the failure.
S	Aggregate status of Q subscriptions (in Classic replication) and publications (in Classic event publishing) that reference the file. <ul style="list-style-type: none"> <li>A One or more of the Q subscriptions, publications, or both are active.</li> <li>D All of the Q subscriptions, publications, or both are inactive.</li> </ul>
TAB	Number of tables altered for data capture that reference the file.
SUB	Number of Q subscriptions (in Classic replication) and publications (in Classic event publishing) that reference the file.
MESSAGE #	Hexadecimal number that identifies the system message that gives the reason why capture failed for the file. This column is populated with information only when the Capture Failure Indicator (the column named "E") contains a Y.

Table 2 describes the information that is displayed for a file when you use the ALL keyword. When the counts displayed for a file are all zeros, either no changes have occurred to the file while the correlation service has been active or the changes to the file are being processed by another correlation service.

Table 28. Additional information that appears when you use the ALL keyword

CACG773I Header	CACG774I Value
OPENS	Number of open notification messages that have been received for the file.

Table 28. Additional information that appears when you use the ALL keyword (continued)

CACG773I Header	CACG774I Value
CLOSES	Number of close notification messages that have been received for the file.
DELETES	Number of erase messages that have been received for the file.
INSERTS	Number of insert messages that have been received for the file.
UPDATES	Number of update messages that have been received for the file.
READS	Number of read-for-update messages that have been received for the file.

## Generating reports on the activity of NVA instances

Use the REPORT STATUS command to display current operational statistics for an NVA instance on a z/OS LPAR.

The command generates WTO messages CACG740I and CACG732I. Together, these messages list the following information:

- The number of NVA instances that have communicated with a correlation service.
- The number of NVA instances that are currently being tracked.
- Statistics that summarize the activity of the currently tracked NVA instances.
  - Number of files that are being tracked
  - Number of deletes that have been received
  - Number of inserts that have been received
  - Number of updates that have been received
  - Number of read-for-updates that have been received

## Syntax

►►—CMD—,—*Service-name*—,—NVA—,—REPORT—,—STATUS—”—————►◄

## Parameters

### *Service-name*

Required parameter that names the XM queue that the NVA instance uses to communicate with a correlation service.

## Stopping the capture of changes that are made to single VSAM files

Use the STOP command to deactivate change capture for a VSAM file. You can use this command when you do not want to capture changes that are being made to a VSAM file.

You can resume capturing changes for the VSAM file by activating the publications (in Classic event publishing) or Q subscriptions (in Classic replication) that are associated with the file. You should start all of the publications or Q subscriptions at the same time. Change capture resumes when the next OPEN request is made for the file.

You do not have to issue the STOP command before issuing either the disable (DI) or remove (RM) commands in the NVA SVC Manager.

## Syntax

```
►►—CMD—,—,"—NVA—,—STOP—,—FILE—==—File-name—"—————►◄
```

## Parameters

*File-name*

Required parameter that names the VSAM file that you want to stop capturing changes for.

## Resetting change capture for single VSAM files

Use the RESET command to start change capture for a VSAM file if you issued the STOP command for the file, and later activated a publication or Q subscription whose source maps to the file, but change capture did not start again.

## Syntax

```
►►—CMD—,—,"—NVA—,—RESET—,—FILE—==—File-name—"—————►◄
```

## Parameters

*File-name*

Required parameter that names the VSAM file that you want to start capturing changes for.



---

## Chapter 4. Reference for event publishing

The reference documentation for Classic event publishing includes descriptions of the configuration parameters for the configuration files for data servers, descriptions of the structures in XML messages, and a description of the format for delimited messages.

---

### Configuration parameters for change capture

Configuration parameters modify the behavior of the services that run within a data server.

#### Format of configuration parameters

Configuration parameters consist of fixed-length 80-byte records containing either a parameter starting in column 1 or a comment, represented as an asterisk (\*), in column 1.

The parameter syntax follows.

```
parameter_name = value
```

In this example,

- *parameter name* is one or more keywords that begin in the first column of the record.
- A blank must exist on both sides of the equal sign.
- *value* is any number of characters up to the end of the record.
- String values are not surrounded by delimiters.
- Comments after the value are not allowed.

The maximum parameter length is 255 characters, but parameters can be continued across 80-byte records by using the backslash (\) as a continuation character. The continuation character cannot be used until after the equal sign, and must be the last non-blank character of the record. The backslash character and any leading blanks on the continued record will be discarded. Comment lines can be inserted between the continued records.

When editing configuration data sets, do not insert sequence numbers at the end of the records because they become part of the value that is assigned to the corresponding keyword. If ISPF is used, make sure that NUM OFF is set in the edit profile when you edit configuration members.

If a configuration file contains invalid parameters, then none of that configuration image is loaded into memory. If the configuration file cannot be loaded, the data server will terminate.

#### CLIENT CODEPAGE configuration parameter

Optional parameter that changes the code page that is used for XML publications to 1208 (UTF-8).

## Description

By default, distribution services publish messages that use the code page that is specified by the `SERVER CODEPAGE` configuration parameter. If you want the distribution service to publish messages that use the 1208 (UTF-8) code page, set the value of `CLIENT CODEPAGE` to 1208.

## COLUMN DELIMITER configuration parameter

Optional parameter if you are using Classic event publishing to send messages in a delimited format.

### Description

This parameter specifies which character to use for delimiting columns. Specify this parameter in the configuration file of a data server where a distribution service is configured.

Specify the value as a hexadecimal number or enclose the value in single quotation marks.

Default value if you do not explicitly set a value: ,

## COMMON FILTER TABLE NAME configuration parameter

Required parameter in configuration files in which a correlation service is defined.

### Description

Use this parameter to identify the name of a filter table where the correlation service maintains information about databases or VSAM files that you are capturing changes from. Change-capture agents also access the information that is maintained in the filter table, with the exception of the change-capture agent for IMS databases.

### Specifications

- Allowable value type: string
- Maximum string length: 16
- Default: none

### Example

```
COMMON FILTER TABLE NAME = FILTER_TABLE
```

## CONTROL TABLE NAME configuration parameter

Required parameter in the configuration file where you configure the name service.

### Description

If you plan to run the name service in a data server where one or more correlation services are configured, this configuration parameter must exist in the configuration file for that data server.

If you plan to run the name service in a separate data server, this configuration parameter must exist in the configuration file for that data server.

This parameter identifies the names of one or more tables that are used to record recovery status about change-capture agents. You determine whether to locate control tables in a coupling facility as list structures or in the data server's message pool as local tables.

The number of control tables must match the number of correlation services that you plan to run in data servers on the z/OS LPAR where the name service will run. For the value of the CONTROL TABLE NAME configuration parameter, specify a comma-separated list of names for each control table. Each name can be from 1 to 8 characters long.

If you are running an unnamed correlation service, use the name (NONAME) for its control table.

If you plan to use local tables rather than list structures in a coupling facility, specify each control table in this format:

*table-name/number-of-change-capture-agents*

**table-name**

The 1 to 8 characters in the name of the control table. This name matches the name of a correlation service or is (NONAME) if you plan to run only one correlation service.

**number-of-change-capture-agents**

The number of change-capture agents that will send change data to the correlation service.

**Specifications**

- Allowable value type: string
- Maximum length: 255
- Default: none

**Example**

```
CONTROL TABLE NAME = (NONAME)/32
```

*Figure 11. Example 1*

In this example, only one correlation service will run on the z/OS LPAR. It is not necessary to name the only running correlation service on an LPAR. However, if two or more correlation services run on the same LPAR, only one correlation service can be unnamed.

32 change-capture agents will send change data to this correlation service. A local control table will track the recovery status for those 32 agents.

```
CONTROL TABLE NAME = TEST/32,PROD
```

*Figure 12. Example 1*

In this example, a local control table that is named TEST will track the recovery status for 32 change-capture agents.

A second control table that is named PROD is associated with a list structure in a coupling facility.

## **DATASOURCE configuration parameter**

Required parameter that matches the name of the query processor and the TCP/IP string that is used for the TCP/IP connection handler.

### **Description**

If you change the name of the query processor that is configured in a data server where you are also configuring a correlation service, you must change the value of the DATASOURCE parameter to that name.

Also, if you change the TCP/IP string for the connection handler, you must change the TCP/IP string for the DATASOURCE parameter.

## **DEFLOC configuration parameter**

Required parameter that matches the name of the query processor.

### **Description**

If you change the name of the query processor that is configured in a data server where you are also configuring a correlation service, you must change the value of the DEFLOC parameter to that name.

## **LD TEMP SPACE configuration parameter**

Optional parameter that defines a temporary data set that is dynamically allocated by a correlation service to store uncommitted data changes in addition to committed data changes that are not yet published. The default LD TEMP SPACE definition should be adequate under all conditions. The default uses HiperSpace for performance and allows the HiperSpace to grow to the largest supported value in 16 megabyte increments.

### **Description**

The correlation service stores uncommitted raw changes in a message store. A store for pending commits contains information about the committed UORs that were received from the change-capture agents and that need to be sent to the distribution service for processing. These stores are implemented as B-tree data sets and are configured using the LD TEMP SPACE parameter.

Both the message store and the pending commit store are temporary data sets. Any information contained in these stores is automatically deleted during initialization of the correlation service. For performance reasons, it is recommended that you use HiperSpace for these stores rather than temporary DASD. HiperSpace is a feature that allows the placement of temporary data files, such as temporary files, spill files, and so on, in expanded storage. Using this feature results in improved performance. Note, however, that the supplied LD TEMP SPACE definition uses DASD.

If you use HiperSpace, begin with a fairly small initial allocation (perhaps 16 megabytes) and allow the HiperSpace to grow slowly up to the maximum configured size. The maximum size is 2 gigabytes. Use a small initial allocation because the LD TEMP SPACE parameter is used for both the message store and the pending commit store. Ideally, there will be a single UOR or a small number of UORs that are waiting to be sent to the distribution service for processing in a normal operational environment.

The maximum size of the message store depends on the number and size of the UORs that are being generated by your client applications and captured by the change-capture agents. If a UOR is not committed or rolled back, the changes that are generated are stored in the message store. Therefore, you must allocate enough space to hold all of these changes. The message store must be configured to be large enough to hold all of the changes for any committed UORs. These committed UORs include those that are being sent, or are waiting to be sent, to the distribution service while new changes are arriving from the change-capture agents.

Setting a moderate initial allocation size and moderate secondary allocation size and specifying a large maximum size is recommended because doing so allows the system to dynamically grow based on load. This approach is viable when your site has sufficient auxiliary storage available to handle the peak load.

### Parameters when using DASD

**ALCUNIT = BLOCK|TRK|CYL**

Unit of space allocation that specifies block, track, or cylinder. The default value is TRK.

**SPACE = nnn**

Primary amount of space to allocate. The default value is 15.

**EXTEND = nnn**

Secondary amount of space to allocate. The default value is 5.

**VOL = VOLSER**

(Volume serial number). The default is the z/OS default for your site.

**UNIT = unit name**

The value can specify a DASD allocation group name or the VIO group name, if it exists. The default unit name is the z/OS default for your site.

**RECFM = F|V|U**

Record format to allocate. This format corresponds to z/OS RECFM of FB|VB|U. The default is V.

**RECLLEN = nnn**

Record length. If variable format record, z/OS LRECL will be set to RECLLEN +4. The default is 255.

**BLKSIZE = nnn**

Block size. The default is 6144.

Here are three example entries for LD TEMP SPACE using DASD:

```
LD TEMP SPACE = ALCUNIT=TRK,SPACE=15,VOL=CACVOL
LD TEMP SPACE = ALCUNIT=CYL,SPACE=2
LD TEMP SPACE = ALCUNIT=CYL,SPACE=2,EXTEND=1,UNIT=VIO
```

### Parameters when using Hiperspace

Hiperspace requires APF authorization.

**INIT** Initial region size for the Hiperspace.

**MAX** Maximum region size for the Hiperspace.

**EXTEND**

Unit of growth when INIT is exceeded.

Here is an example entry for LD TEMP SPACE using Hiperspace:

LD TEMP SPACE = HIPERSPACE,INIT=16M,MAX=2G,EXTEND=8M

## MESSAGE POOL SIZE configuration parameter

Required parameter that specifies the size of the memory region that is used for all memory allocation.

### Description

The number is specified in bytes. The actual workable maximum value should be set to 2MB less than the region size. The region size is the maximum size of the address space, as specified in the started task or job that allocates the address space.

If the value specified is less than 1MB, 1MB is used. If the amount of storage that can be obtained is less than the value specified, the maximum amount available is obtained.

### Specifications

- Allowable value type: numeric
- Representation: decimal
- Maximum permitted value: 2147483648 (2GB)
- Minimum permitted value: 1048576 (1MB)
- Default value: 1048575 (1MB)

### Example

MESSAGE POOL SIZE = 16777216

## NL configuration parameter

Required parameter that specifies the language used for text messages (errors, warnings, and informational messages) that are produced by IBM WebSphere Classic Data Event Publisher for z/OS and IBM WebSphere Classic Replication Server for z/OS (except for messages that are issued for a Q Apply program).

### Specifications

- Allowable value type: string
- Default value: US ENGLISH

### Example

NL = US ENGLISH

## NL CAT configuration parameter

Required parameter that points to the language catalog that contains IBM WebSphere Classic Data Event Publisher for z/OS or IBM WebSphere Classic Replication Server for z/OS messages in a specified language.

### Description

The value of this parameter is defined by a DD statement in the start-up procedure.

## Specifications

- Allowable value type: string DD:, followed by a character string or string DSN, followed by a data set name.
- Representation: string
- Default value: DD:ENGCAT

## Example

NL CAT = DD:ENGCAT

## ROW DELIMITER configuration parameter

Optional parameter if you are using Classic event publishing to send messages in a delimited format.

### Description

This parameter specifies which character to use for delimiting rows. Specify this parameter in the configuration file of a data server where a distribution service is configured.

Specify the value as a hexadecimal number or enclose the value in single quotation marks.

Default value: \n

## SERVER CODEPAGE configuration parameter

Required parameter that is used in server configuration files to specify the code page that is used by the data server.

### Description

If the value SERVER CODEPAGE is different from the value of CLIENT CODEPAGE, XML or delimited messages are published in codepage 1208 (UTF-8).

If the value SERVER CODEPAGE is the same as the value of CLIENT CODEPAGE, XML or delimited messages in EBCDIC. There is no codepage conversion.

By default, the data server uses code page 1047.

## SERVICE INFO ENTRY configuration parameter

Required parameter used in data server configuration files to inform the region controller task that a service should be activated and how that service should be controlled. This parameter is valid only in data server and enterprise server configuration files.

### Description

Multiple service information entry parameters are required to activate multiple instances of a service if different subparameter values are needed. A single service information entry parameter is used if only a single instance is needed, or if multiple instances are using the same subparameter values. The restrictions and allowable combinations of multiple instances for a service are discussed in the specific descriptions for that service. Mutually-exclusive services are also noted in these descriptions.

The service information entry parameter consists of ten subparameters, each delimited by at least one space. The format of the first nine of these subfields is consistent across all services and is not service dependent. The format for the tenth subfield and all valid values of all ten fields are service dependent.

## STRING DELIMITER configuration parameter

Optional parameter if you are using Classic event publishing to send messages in a delimited format.

### Description

This parameter specifies which character to use for delimiting strings. Specify this parameter in the configuration file of a data server where a distribution service is configured.

Specify the value as a hexadecimal number or enclose the value in single quotation marks.

Default value: "

---

## XML messages for Classic event publishing

There are two types of XML messages: row-level messages and transaction messages. The root element is identical for both types.

The structure of the XML messages follows the XML Schema Language (Part 1: Structure and Part 2: Datatypes), W3C Recommendation, 2 May 2001.

The XML declaration in every message describes the code page that is used for the message. The CLIENT CODEPAGE configuration parameter determines the code page that is used. The default code page is 1208 (UTF-8), as specified in the XML 1.0 (2nd edition), W3C Recommendation, 6 October 2000. Changes from the source database are converted into messages using z/OS Unicode Conversion Services (UCS).

See the description of the CLIENT CODEPAGE configuration parameter in Appendix B for an explanation of why you might change the code page used for XML messages. Throughout the rest of this appendix, the example XML declarations use the default encoding of 1208.

## How XML delimiters are handled in character data

In XML messages, the values from mapped columns appear between XML tags that describe the column data type.

For example, the values 222 and Hello from a source table would be encoded as `<integer>222</integer>` and `<varchar>Hello</varchar>`.

Because the angle bracket (< or >) and ampersand (&) characters are predefined XML delimiters, these characters appear in column values as follows:

- < to &lt;
- > to &gt;
- & to &amp;

Also, when the apostrophe (') or double quotation mark (") appear in attribute values, these characters appear as follows:

- ' to &apos;
- " to &quot;

The resulting messages are valid XML document instances.

## msg: Root element for XML messages

The msg element is the root element for XML messages.

Table 29. Description of the msg element

Name	Properties
msg	Not empty, complex type, complex content

### Structure:

```
<?xml version="1.0"?>
<msg xmlns:xsi="XML_schema_instance"
      xsi:noNamespaceSchemaLocation="schema_document"
      version="version" dbName="database_name">

    elements

</msg>
```

### Details:

**<?xml version="1.0"?>**

The XML declaration defaults to this format. If you set the CLIENT CODEPAGE configuration parameter to 1208, the encoding attribute of the declaration is set to UTF-8.

*XML\_schema\_instance*

The URL of the XML schema instance. For event publishing, the URL is [www.w3.org/2001/XMLSchema-instance](http://www.w3.org/2001/XMLSchema-instance). XML data type: string.

*schema\_document*

The file name of the XML schema document. XML namespace is not supported in event publishing because messages refer to one XML schema only. Messages from a distribution service to a user application refer to the mqcap.xsd schema document. XML data type: string.

*version*

The version of the XML message schema. For Classic Event Publisher Version 8.2, the version is 1.0.0. XML data type: string.

*database\_name*

The type of data source. XML data type: string.

*elements*

One of the elements that the msg element contains. Only one of these elements appears in each message:

- trans
- rowOp

### Example:

The following example shows a message element.

```
<?xml version="1.0" encoding="UTF-8"?>
<msg xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="mqcap.xsd">
```

```

        version="1.0.0" dbName="DB1">
            elements
    </msg>

```

Where *elements* represents one of the following elements: trans or rowOpt.

## Transaction message

A transaction message contains one or more insert, update, or delete row operations on the source table. The transaction message also contains information about the time that the transaction was committed at the source database, and a time-based log sequence number.

- “Transaction element (trans)”
- “Row operation elements (insertRow, updateRow, and deleteRow)” on page 237
- “Column element (col)” on page 238
- “Elements for a single-column value” on page 240
- “Elements for a double-column value” on page 241
- “Before-value and after-value elements (beforeVal and afterVal)” on page 242

### Transaction element (trans)

The transaction element (trans) is contained by the msg element, and it contains one of the three row operation elements (insertRow, updateRow, or deleteRow).

Table 30. Element description for trans

Name	Properties
trans	Not empty, complex type, complex content

#### Structure:

```

<trans isLast="is_last_indicator" segmentNum="segment_number"
    cmitLSN="commit_logical_sequence_number" cmitTime="commit_time" authID="authID"
    planName="psb">
        elements
</trans>

```

#### Details:

##### *is\_last\_indicator*

A boolean value that indicates whether the transaction message is the last message in a database transaction. If it is the last message, the value is 1 (true). If it is not the last message, the value is 0 (false). XML data type: boolean.

##### *segment\_number*

A positive integer that indicate the message’s segment number in a divided transaction message. XML data type: positiveInteger.

##### *commit\_logical\_sequence\_number*

The commit logical sequence number (a time-based log sequence number) of the COMMIT statement for the transaction. XML data type: string.

##### *commit\_time*

The timestamp of the COMMIT statement for the transaction using Greenwich mean time (GMT), formatted in microseconds. XML data type: dateTime.

### *authID*

The authorization ID of the user responsible for the changes included in the XML message. This attribute is published only for IMS databases if the authorization ID is available in the IMS log records.

### *elements*

Each trans element contains one or more of these elements:

- insertRow
- updateRow
- deleteRow

### **Example:**

The following example shows a transaction element that contains one or more of the insert row, update row, or delete row elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<msg xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="mqcap.xsd"
  version="1.0.0" dbName="DB1">
  <trans isLast="1" segmentNum="1" cmitLSN="0000:0000::0000:06d6:87ab"
    cmitTime="2003-10-31T12:12:12.000122">
    insertRow, updateRow, or deleteRow
  </trans>
</msg>
```

Where *insertRow*, *updateRow*, or *deleteRow* represents the elements that are explained in “Row operation elements (insertRow, updateRow, and deleteRow).”

## **Row operation elements (insertRow, updateRow, and deleteRow)**

Within a transaction element, the row operation elements (insertRow, updateRow, and deleteRow) describe the type of operation that is performed on a row of the source table. Each of these elements contains one or more column elements (col) that describe changes to subscribed columns.

*Table 31. Element description for insertRow, deleteRow, and updateRow*

<b>Name</b>	<b>Properties</b>
insertRow	Not empty, complex type, complex content
deleteRow	Not empty, complex type, complex content
updateRow	Not empty, complex type, complex content

### **Structure:**

```
<insertRow subName="XML_publication_name" srcOwner="source_owner"
  srcName="source_name" rowNum="row_number">
```

*elements*

```
</insertRow>
```

```
<deleteRow subName="XML_publication_name" srcOwner="source_owner"
  srcName="source_name" rowNum="row_number">
```

*elements*

```
</deleteRow>
```

```
<updateRow subName="XML_publication_name" srcOwner="source_owner"  
  srcName="source_name" rowNum="row_number">
```

*elements*

```
</updateRow>
```

### Details:

*XML\_publication\_name*

The name of the XML publication to which this row operation belongs. XML data type: string.

*source\_owner*

The schema of the source table where the row operation originated. XML data type: string.

*source\_name*

The name of the source table. XML data type: string.

*elements*

One or more column elements (col) contained by the insertRow, updateRow, or deleteRow element.

### Example:

The following example shows insertRow, updateRow, and deleteRow elements within a transaction message.

```
<?xml version="1.0" encoding="UTF-8"?>  
<msg xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:noNamespaceSchemaLocation="mqcap.xsd" version="1.0.0"  
  dbName="DB1">  
  <trans isLast="1" segmentNum="1" cmitLSN="0000:0000::0000:06d6:87ab"  
    cmitTime="2003-10-31T12:12:12.000122">  
    <insertRow subName="S1" srcOwner="USER1" srcName="T1">  
      column_element  
    </insertRow>  
    <deleteRow subName="S1" srcOwner="USER1" srcName="T1">  
      column_element  
    </deleteRow>  
    <updateRow subName="S1" srcOwner="USER1" srcName="T1">  
      column_element  
    </updateRow>    </trans>  
</msg>
```

Where *column\_element* represents the column element that is explained in “Column element (col).”

## Column element (col)

The column element (col) describes the name of a mapped column in the source table, and it also tells whether the column is part of the key to be used for

publishing. A col element within an insert or delete operation contains a single value only. Within an update operation, the col element can contain a before value and an after value, depending on the options for sending data that you specified for the XML publication.

Table 32. Element description for col

Name	Properties
col	Not empty, complex type, complex content

**Structure:**

```
<col name="column_name" isKey="key_indicator">
    single_or_double_column_value
</col>
```

**Details:**

*column\_name*

The name of a subscribed column in the source table. XML data type: string.

*key\_indicator*

- 0 The column is not a key column.
- 1 The column is a key column.

*single\_or\_double\_column\_value*

If the column element is part of an insert or delete operation at the source table, it will contain one of the single-column- value elements. For update operations, the column element can contain a double-column value, which includes both a before value and an after value.

**Example:**

The following example shows an insert operation that contains single column values, and an update operation that contains double column values.

```
<?xml version="1.0" encoding="UTF-8"?>
<msg xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="mqcap.xsd" version="1.0.0"
    dbName="DB1">
    <trans isLast="1" segmentNum="1" cmitLSN="0000:0000::0000:06d6:87ab"
        cmitTime="2003-10-31T12:12:12.000122">
        <insertRow subName="S1" srcOwner="USER1" srcName="T1">
            <col name="COL1" isKey="0">
                single_column_value
            </col>
            <col name="COL2">
                single_column_value
            </col>
        </insertRow>
        <updateRow subName="S1" srcOwner="USER1" srcName="T1">
            <col name="COL1" isKey="0">
                double_column_value
            </col>
            <col name="COL2">
```

```

                double_column_value
            </col>
        </updateRow>
    </trans>
</msg>

```

Where *single\_column\_value* represents the elements that are explained in “Elements for a single-column value,” and *double\_column\_value* represents the elements that are explained in “Elements for a double-column value” on page 241.

## Elements for a single-column value

A single-column-value element contains an actual value from the source table. The distribution service uses single-column-value elements for insert and delete operations. These elements are named for the data type of the source column, and do not contain other elements. If the value from the source table is NULL, the element is empty and the `xsi:nil` attribute is set to 1 (true).

Table 33 describes the single-column-value elements. All of the elements are of complex type, and simple content.

Table 33. Element descriptions for single column value

Name	XML data type	Value’s data format
smallint	short	
integer	integer	
bigint	long	
float	float (32 bits)	[-]d.dddde[- +]dd
	double (64 bits)	[-]d.dddde[- +]dd
real	float	
double	double	
decimal	decimal	
date	date	YYYY-MM-DD
time	time	HH:MM:SS.SSS
timestamp	dateTime	YYYY-MM-DDTHH:MM:SS.SSS
char	string	
varchar	string	
long varchar	string	
bitchar	hexBinary	
bitvarchar	hexBinary	
bitlongvarchar	hexBinary	
graphic	string	
vargraphic	string	
longvargraphic	string	
rowid	hexBinary	

### Structure:

```
<data_type xsi:nil="null_indicator">value</data_type>
```

**Details:***data\_type*

The data type of the column in the source table. This data type is used to name the element.

*null\_indicator*

Optional: An integer that indicates whether the source column contains a NULL value. The default is 0 (false). If the source column contains a NULL value, the value of this attribute is 1 (true). XML data type: boolean.

*value*

The actual value in the source column. If the source value is NULL, the element is empty.

**Example:**

The following example shows an insert operation with single column values of 222 in a key column with an integer data type and Hello in a nonkey column with a varchar data type. The example also shows a delete operation of the row with a single-column value of 222 in a key column with an integer data type.

```
<?xml version="1.0" encoding="UTF-8"?>
<msg xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="mqcap.xsd" version="1.0.0"
      dbName="DB1">
  <trans isLast="1" segmentNum="1" cmitLSN="0000:0000::0000:06d6:87ab"
        cmitTime="2003-10-31T12:12:12.000122">
    <insertRow subName="S1" srcOwner="USER1" srcName="T1">
      <col name="COL1" isKey="1">
        <integer>222</integer>
      </col>
      <col name="COL2">
        <varchar>Hello</varchar>
      </col>
    </insertRow>
    <deleteRow subName="S1" srcOwner="USER1" srcName="T1">
      <col name="COL1" isKey="1">
        <integer>222</integer>
      </col>
    </deleteRow>
  </trans>
</msg>
```

**Elements for a double-column value**

Double-column-value elements are used in update operations when the distribution service needs to send both before and after values from source columns. In XML messages, the distribution service sends before values of key columns that have changed. It sends before values of nonkey columns that have changed if the BEFORE\_VALUES data-sending option is set to "Yes" for the XML publication. If the before and after values are the same, only the after-value element (afterValue) is used.

All double-column-value elements are not empty, have a complex type, and have complex content. Double-column-value elements have no attributes. For a description of the double-column-value elements, see "Elements for a single-column value" on page 240.

**Structure:**

```
<data_type>
  elements
</data_type>
```

**Details:**

*data\_type*

The data type of the column in the source table. This data type is used to name the element.

*elements*

One or both of the beforeValue or afterValue elements.

**Example:**

The following example shows double-column-value elements for:

- A key column (integer data type) that has changed.
- A nonkey column (varchar data type) that has changed, but the BEFORE\_VALUES data-sending option for the XML publication is set to "No."

```
<?xml version="1.0" encoding="UTF-8"?>
<msg xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="mqcap.xsd" version="1.0.0"
  dbName="DB1">
  <trans isLast="1" segmentNum="1" cmitLSN="0000:0000::0000:06d6:87ab"
    cmitTime="2003-10-31T12:12:12.000122">
    <updateRow subName="S1" srcOwner="USER1" srcName="T1">
      <col name="COL1" isKey="1">
        <integer>
          beforeValue
          afterValue
        </integer>
      </col>
      <col name="COL2">
        <varchar>
          afterValue
        </varchar>
      </col>
    </updateRow>
  </trans>
</msg>
```

Where *beforeValue* and *afterValue* represent the elements that are explained in "Before-value and after-value elements (beforeVal and afterVal)."

**Before-value and after-value elements (beforeVal and afterVal)**

Before-value and after-value elements (beforeVal and afterVal) contain actual values from the source table. These elements are used in update operations for key columns that have changed, and for changed nonkey columns when the BEFORE\_VALUES data-sending-option for the XML publication is set to "Yes." If the XML publication calls for before values to be sent and the value in the source column has not changed, only the afterVal element is used. If the value from the source table is NULL, the elements are empty and the xsi:null attribute is set to 1 (true).

Table 34. Element descriptions for *beforeVal* and *afterVal*

Name	Properties
<i>beforeVal</i>	Nullable, complex type, simple content
<i>afterVal</i>	Nullable, complex type, simple content, optional

**Structure:**

```
<beforeVal xsi:nil="null_indicator">value</beforeVal>
<afterVal xsi:nil="null_indicator">value</afterVal>
```

**Details:**

*null\_indicator*

Optional: An integer that indicates whether the value in the source column is NULL. The default is 0 (false). If the source column contains a NULL value, the value of this attribute is 1 (true). XML data type: boolean.

*value*

The actual value in the source column. If the source value is NULL, the element will be empty.

**Example:**

The following example shows an update operation where the key column's value of 222 did not change (only the *afterVal* element is used), and where a varchar column in the same row changed from "Hello" to NULL. In this case, the BEFORE\_VALUES option for the XML publication is set to "Yes."

```
<?xml version="1.0" encoding="UTF-8"?>
<msg xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="mqcap.xsd" version="1.0.0"
  dbName="DB1">
  <trans isLast="1" segmentNum="1" cmitLSN="0000:0000::0000:06d6:87ab"
    cmitTime="2003-10-31T12:12:12.000122">
    <updateRow subName="S1" srcOwner="USER1" srcName="T1">
      <col name="COL1" isKey="1">
        <integer>
          <afterVal>222</afterVal>
        </integer>
      </col>
      <col name="COL2">
        <varchar>
          <beforeVal>Hello</beforeVal>
          <afterVal xsi:nil="1"/>
        </varchar>
      </col>
    </updateRow>
  </trans>
</msg>
```

## Row operation message

A row operation message must not exceed the maximum message size that is defined for the send queue. Row operation messages that exceed this size cannot be divided into multiple messages. In row operation messages, any inserts, updates, or deletes that belong to a transaction have the same commit time and commit logical sequence number.

Table 35 describes the rowOp element.

Table 35. Element description for rowOp

Name	Properties
rowOp	Not empty, complex type, complex content

### Structure:

```
<rowOp cmitLSN="commit_logical_sequence_number"
  cmitTime="commit_time" isLast="is_last_indicator" authID="authID" planName="psb">
  elements
</rowOp>
```

### Details:

#### *commit\_logical\_sequence\_number*

The commit logical sequence number (a time-based log sequence number) of the COMMIT statement for the transaction. XML data type: string.

#### *commit\_time*

The timestamp of the COMMIT statement for the transaction using Greenwich mean time (GMT), formatted in microseconds. XML data type: dateTime.

#### *is\_last\_indicator*

Optional: A boolean value that indicates whether the row operation message is the last message in a row operation from the source database. This attribute has no default value. XML data type: boolean.

#### *authID*

The authorization ID of the user responsible for the change included in the XML message. This attribute is published only for IMS databases if the authorization ID is available in the IMS log records.

#### *elements*

Each rowOp element contains one of these elements:

- insertRow
- updateRow
- deleteRow

### Example:

The following example shows a row operation element that contains an insertRow, updateRow, or deleteRow element.

```
<?xml version="1.0" encoding="UTF-8"?>
<msg xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="mqcap.xsd" version="1.0.0" dbName="DB1">
  <rowOp cmitLSN="0000:0000::0000:06d6:87ab"
    cmitTime="2003-10-31T12:12:12.000122">
    insertRow, deleteRow, or updateRow
  </rowOp></msg>
```

Where *insertRow*, *updateRow*, or *deleteRow* represents the elements that are explained in “Transaction message” on page 236.

## Schema for XML messages

The XML messages in Classic event publishing use the following schema.

This schema is also used in IBM WebSphere Data Event Publisher.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:annotation>
    <xs:documentation xml:lang="en">
      XML Schema of messages sent by Q Capture to a subscriber.
      (C) Copyright IBM CORPORATION 2003
    </xs:documentation>
  </xs:annotation>

  <!-- Message type definition -->

  <xs:element name="msg" type="msgType"/>

  <xs:complexType name="msgType">
    <xs:choice>
      <xs:element name="trans" type="transType"/>
      <xs:element name="rowOp" type="rowOpType"/>
      <xs:element name="subDeactivated" type="subDeactivatedType"/>
      <xs:element name="loadDoneRcvd" type="loadDoneRcvdType"/>
      <xs:element name="heartbeat" type="heartbeatType"/>
      <xs:element name="errorRpt" type="errorRptType"/>
      <xs:element name="subSchema" type="subSchemaType"/>
      <xs:element name="lob" type="lobType"/>
      <xs:element name="addColumn" type="addColumnType"/>
    </xs:choice>
    <xs:attribute name="version" type="xs:string" use="required"/>
    <xs:attribute name="dbName" type="xs:string" use="required"/>
  </xs:complexType>

  <!-- Transaction type definition -->

  <xs:complexType name="transType">
    <xs:choice maxOccurs="unbounded">
      <xs:element name="insertRow" type="singleValRowType"/>
      <xs:element name="deleteRow" type="singleValRowType"/>
      <xs:element name="updateRow" type="updateRowType"/>
    </xs:choice>
    <xs:attribute name="isLast" type="xs:boolean" use="required"/>
    <xs:attribute name="segmentNum" type="xs:positiveInteger" use="required"/>
    <xs:attribute name="cmitLSN" type="xs:string" use="required"/>
    <xs:attribute name="cmitTime" type="xs:dateTime" use="required"/>
    <xs:attribute name="authID" type="xs:string"/>
    <xs:attribute name="correlationID" type="xs:string"/>
    <xs:attribute name="planName" type="xs:string"/>
  </xs:complexType>

  <!-- LOB type definition -->

  <xs:complexType name="lobType">
    <xs:choice>
      <xs:element name="blob" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="blob"/>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="clob" nillable="true">
```

```

    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="clob"/>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="dbclob" nillable="true">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="dbclob"/>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:choice>
<xs:attributeGroup ref="commonAttrGroup"/>
<xs:attribute name="isLast" type="xs:boolean" use="required"/>
<xs:attribute name="colName" type="xs:string" use="required"/>
<xs:attribute name="rowNum" type="xs:positiveInteger" use="required"/>
<xs:attribute name="totalDataLen" type="xs:nonNegativeInteger" use="required"/>
<xs:attribute name="dataLen" type="xs:nonNegativeInteger" use="required"/>
</xs:complexType>

```

```

<!-- Row operation type definition -->

```

```

<xs:complexType name="rowOpType">
  <xs:choice>
    <xs:element name="insertRow" type="singleValRowType"></xs:element>
    <xs:element name="deleteRow" type="singleValRowType"></xs:element>
    <xs:element name="updateRow" type="updateRowType"></xs:element>
  </xs:choice>
  <xs:attribute name="isLast" type="xs:boolean"/>
  <xs:attribute name="cmitLSN" type="xs:string" use="required"/>
  <xs:attribute name="cmitTime" type="xs:dateTime" use="required"/>
  <xs:attribute name="authID" type="xs:string"/>
  <xs:attribute name="correlationID" type="xs:string"/>
  <xs:attribute name="planName" type="xs:string"/>
</xs:complexType>

```

```

<!-- Row types and their common attribute group definition -->

```

```

<xs:complexType name="singleValRowType">
  <xs:sequence maxOccurs="unbounded">
    <xs:element name="col" type="singleValColType"></xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="commonAttrGroup"/>
  <xs:attributeGroup ref="opAttrGroup"/>
</xs:complexType>

```

```

<xs:complexType name="updateRowType">
  <xs:sequence maxOccurs="unbounded">
    <xs:element name="col" type="updateValColType"></xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="commonAttrGroup"/>
  <xs:attributeGroup ref="opAttrGroup"/>
</xs:complexType>

```

```

<!-- Column types and their common attribute group definition -->

```

```

<xs:complexType name="singleValColType">
  <xs:choice>
    <xs:element name="smallint" nillable="true">
      <xs:complexType>
        <xs:simpleContent>

```

```

        <xs:extension base="smallint"/>
    </xs:simpleContent>
</xs:complexType>
</xs:element>
<xs:element name="integer" nillable="true">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="integer"/>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="bigint" nillable="true">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="bigint"/>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="float" nillable="true">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="float"/>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="real" nillable="true">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="real"/>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="double" nillable="true">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="double"/>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="decimal" nillable="true">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="decimal"/>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="date" nillable="true">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="date"/>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="time" nillable="true">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="time"/>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="timestamp" nillable="true">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="timestamp"/>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>

```

```

<xs:element name="char" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="char"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="varchar" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="varchar"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="longvarchar" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="longvarchar"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="bitchar" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="bitchar"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="bitvarchar" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="bitvarchar"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="bitlongvarchar" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="bitlongvarchar"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="graphic" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="graphic"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="vargraphic" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="vargraphic"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="longvargraphic" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="longvargraphic"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="rowid" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="rowid"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

```

```

        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="blob">
    <xs:complexType>
    </xs:complexType>
</xs:element>
<xs:element name="clob">
    <xs:complexType>
    </xs:complexType>
</xs:element>
<xs:element name="dbclob">
    <xs:complexType>
    </xs:complexType>
</xs:element>
</xs:choice>
<xs:attributeGroup ref="colAttrGroup"/>
</xs:complexType>

<xs:complexType name="updateValColType">
<xs:choice>
<xs:element name="smallint">
    <xs:complexType>
    <xs:sequence>
    <xs:sequence minOccurs="0">
    <xs:element name="beforeVal" nillable="true">
    <xs:complexType>
    <xs:simpleContent>
    <xs:extension base="smallint">
    <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
    <xs:attribute name="rawData" type="xs:boolean" default="false"/>
    </xs:extension>
    </xs:simpleContent>
    </xs:complexType>
    </xs:element>
    </xs:sequence>
    <xs:element name="afterVal" nillable="true">
    <xs:complexType>
    <xs:simpleContent>
    <xs:extension base="smallint">
    <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
    <xs:attribute name="rawData" type="xs:boolean" default="false"/>
    </xs:extension>
    </xs:simpleContent>
    </xs:complexType>
    </xs:element>
    </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="integer">
    <xs:complexType>
    <xs:sequence>
    <xs:sequence minOccurs="0">
    <xs:element name="beforeVal" nillable="true">
    <xs:complexType>
    <xs:simpleContent>
    <xs:extension base="integer">
    <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
    <xs:attribute name="rawData" type="xs:boolean" default="false"/>
    </xs:extension>
    </xs:simpleContent>
    </xs:complexType>
    </xs:element>
    </xs:sequence>
    <xs:element name="afterVal" nillable="true">
    <xs:complexType>
    <xs:simpleContent>

```

```

        <xs:extension base="integer">
          <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
          <xs:attribute name="rawData" type="xs:boolean" default="false"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="bigint">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="bigint">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="bigint">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="float">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="float">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="float">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="real">

```

```

<xs:complexType>
  <xs:sequence>
    <xs:sequence minOccurs="0">
      <xs:element name="beforeVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="real">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:element name="afterVal" nillable="true">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="real">
            <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
            <xs:attribute name="rawData" type="xs:boolean" default="false"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="double">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="double">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="double">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="decimal">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="decimal">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

    </xs:complexType>
  </xs:element>
</xs:sequence>
<xs:element name="afterVal" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="decimal">
        <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
        <xs:attribute name="rawData" type="xs:boolean" default="false"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="date">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="date">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="date">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="time">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="time">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="time">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

        </xs:complexType>
    </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="timestamp">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="timestamp">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="timestamp">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="char">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="char">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="char">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="varchar">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>

```

```

    <xs:simpleContent>
      <xs:extension base="varchar">
        <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
        <xs:attribute name="rawData" type="xs:boolean" default="false"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
</xs:sequence>
<xs:element name="afterVal" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="varchar">
        <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
        <xs:attribute name="rawData" type="xs:boolean" default="false"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="longvarchar">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="longvarchar">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="longvarchar">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="bitchar">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="bitchar">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>

```

```

        <xs:simpleContent>
          <xs:extension base="bitchar">
            <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
            <xs:attribute name="rawData" type="xs:boolean" default="false"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="bitvarchar">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="bitvarchar">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="bitvarchar">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="bitlongvarchar">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="bitlongvarchar">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="bitlongvarchar">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

<xs:element name="graphic">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="graphic">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="graphic">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="vargraphic">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="vargraphic">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:element name="afterVal" nillable="true">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="vargraphic">
              <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
              <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="longvargraphic">
  <xs:complexType>
    <xs:sequence>
      <xs:sequence minOccurs="0">
        <xs:element name="beforeVal" nillable="true">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="longvargraphic">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

        </xs:simpleContent>
    </xs:complexType>
</xs:element>
</xs:sequence>
<xs:element name="afterVal" nillable="true">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="longvargraphic">
                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="rowid">
    <xs:complexType>
        <xs:sequence>
            <xs:sequence minOccurs="0">
                <xs:element name="beforeVal" nillable="true">
                    <xs:complexType>
                        <xs:simpleContent>
                            <xs:extension base="rowid">
                                <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                                <xs:attribute name="rawData" type="xs:boolean" default="false"/>
                            </xs:extension>
                        </xs:simpleContent>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:element name="afterVal" nillable="true">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="rowid">
                            <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
                            <xs:attribute name="rawData" type="xs:boolean" default="false"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="blob">
    <xs:complexType>
</xs:complexType>
</xs:element>
<xs:element name="clob">
    <xs:complexType>
</xs:complexType>
</xs:element>
<xs:element name="dbclob">
    <xs:complexType>
</xs:complexType>
</xs:element>
</xs:choice>
<xs:attributeGroup ref="colAttrGroup"/>
</xs:complexType>

<!-- Attribute group definition -->

<xs:attributeGroup name="commonAttrGroup">
    <xs:attribute name="subName" type="xs:string" use="required"/>
    <xs:attribute name="srcOwner" type="xs:string" use="required"/>

```

```

    <xs:attribute name="srcName" type="xs:string" use="required"/>
    <xs:attribute name="intentSEQ" type="xs:string"/>
  </xs:attributeGroup>

  <xs:attributeGroup name="colAttrGroup">
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="isKey" type="xs:boolean" default="false"/>
    <xs:attribute name="invalidData" type="xs:boolean" default="false"/>
    <xs:attribute name="rawData" type="xs:boolean" default="false"/>
  </xs:attributeGroup>

  <xs:attributeGroup name="opAttrGroup">
    <xs:attribute name="rowNum" type="xs:positiveInteger"/>
    <xs:attribute name="hasLOBCols" type="xs:boolean" default="false"/>
  </xs:attributeGroup>

  <!-- Subscription deactivated type definition -->

  <xs:complexType name="subDeactivatedType">
    <xs:attributeGroup ref="commonAttrGroup"></xs:attributeGroup>
    <xs:attribute name="stateInfo" type="xs:string" use="required"/>
  </xs:complexType>

  <!-- Load done received type definition -->

  <xs:complexType name="loadDoneRcvdType">
    <xs:attributeGroup ref="commonAttrGroup"></xs:attributeGroup>
    <xs:attribute name="stateInfo" type="xs:string" use="required"/>
  </xs:complexType>

  <!-- Heartbeat type definition -->

  <xs:complexType name="heartbeatType">
    <xs:attribute name="sendQName" type="xs:string" use="required"/>
    <xs:attribute name="lastCmitTime" type="xs:dateTime"/>
  </xs:complexType>

  <!-- Error Report type definition -->

  <xs:complexType name="errorRptType">
    <xs:attributeGroup ref="commonAttrGroup"></xs:attributeGroup>
    <xs:attribute name="errorMsg" type="xs:string" use="required"/>
  </xs:complexType>

  <!-- Schema type definition -->

  <xs:complexType name="subSchemaType">
    <xs:sequence maxOccurs="unbounded">
      <xs:element name="col" type="colSchemaType"></xs:element>
    </xs:sequence>
    <xs:attributeGroup ref="commonAttrGroup"></xs:attributeGroup>
      <xs:attribute name="sendQName" type="xs:string" use="required"/>
    <xs:attribute name="allChangedRows" type="xs:boolean" default="false"/>
    <xs:attribute name="beforeValues" type="xs:boolean" default="false"/>
    <xs:attribute name="changedColsOnly" type="xs:boolean" default="true"/>
    <xs:attribute name="hasLoadPhase" type="loadPhaseEnumType" default="none"/>
    <xs:attribute name="dbServerType" type="dbServerTypeEnumType" use="required"/>
    <xs:attribute name="dbRelease" type="xs:string" use="required"/>
    <xs:attribute name="dbInstance" type="xs:string" use="required"/>
    <xs:attribute name="capRelease" type="xs:string" use="required"/>
    <xs:attribute name="timestamp" type="xs:string"/>
  </xs:complexType>

```

```

<!-- Add column definition -->

<xs:complexType name="addColumnType">
  <xs:sequence maxOccurs="1">
    <xs:element name="col" type="colSchemaType"></xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="commonAttrGroup"></xs:attributeGroup>
</xs:complexType>

<!-- Load phase enumeration type definition -->

<xs:simpleType name="loadPhaseEnumType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="external"/>
  </xs:restriction>
</xs:simpleType>

<!-- DB2 server type enumeration type definition -->
<!--
  DB2 server type enum values based on DB server type values described in
  asnrib.h
-->

<xs:simpleType name="dbServerTypeEnumType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="QDB2"/>
    <xs:enumeration value="QDB2/6000"/>
    <xs:enumeration value="QDB2/HPUX"/>
    <xs:enumeration value="QDB2/NT"/>
    <xs:enumeration value="QDB2/SUN"/>
    <xs:enumeration value="QDB2/SUN64"/>
    <xs:enumeration value="QDB2/LINUX"/>
    <xs:enumeration value="QDB2/Windows"/>
    <xs:enumeration value="QDB2/AIX64"/>
  </xs:restriction>
</xs:simpleType>

<!-- Column schema type definition -->

<xs:complexType name="colSchemaType">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="type" type="dataTypeEnumType" use="required"/>
  <xs:attribute name="isKey" type="xs:boolean" default="false"/>
  <xs:attribute name="len" type="xs:unsignedInt"/>
  <xs:attribute name="precision" type="xs:unsignedShort"/>
  <xs:attribute name="scale" type="xs:unsignedShort"/>
  <xs:attribute name="codepage" type="xs:unsignedInt" default="0"/>
  <xs:attribute name="default" type="xs:string"/>
  <xs:attribute name="beforeColName" type="xs:string"/>
</xs:complexType>

<!-- Data type enumeration type definition -->
<!-- Data type names are used as tag name also. Both are the same. -->

<xs:simpleType name="dataTypeEnumType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="smallint"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="bigint"/>
    <xs:enumeration value="float"/>
    <xs:enumeration value="real"/>
  </xs:restriction>
</xs:simpleType>

```

```

<xs:enumeration value="double"/>
<xs:enumeration value="decimal"/>
<xs:enumeration value="char"/>
<xs:enumeration value="varchar"/>
<xs:enumeration value="longvarchar"/>
<xs:enumeration value="bitchar"/>
<xs:enumeration value="bitvarchar"/>
<xs:enumeration value="bitlongvarchar"/>
<xs:enumeration value="graphic"/>
<xs:enumeration value="vargraphic"/>
<xs:enumeration value="longvargraphic"/>
<xs:enumeration value="time"/>
<xs:enumeration value="timestamp"/>
<xs:enumeration value="date"/>
<xs:enumeration value="rowid"/>
<xs:enumeration value="blob"/>
<xs:enumeration value="clob"/>
<xs:enumeration value="dbclob"/>
</xs:restriction>
</xs:simpleType>

```

```

<!-- Data type definitions -->

```

```

<xs:simpleType name="smallint">
  <xs:restriction base="xs:short">
  </xs:restriction>
</xs:simpleType>

```

```

<xs:simpleType name="integer">
  <xs:restriction base="xs:integer">
  </xs:restriction>
</xs:simpleType>

```

```

<xs:simpleType name="bigint">
  <xs:restriction base="xs:long">
  </xs:restriction>
</xs:simpleType>

```

```

<xs:simpleType name="float">
  <xs:restriction base="xs:float">
  </xs:restriction>
</xs:simpleType>

```

```

<xs:simpleType name="real">
  <xs:restriction base="xs:float">
  </xs:restriction>
</xs:simpleType>

```

```

<xs:simpleType name="double">
  <xs:restriction base="xs:double">
  </xs:restriction>
</xs:simpleType>

```

```

<xs:simpleType name="decimal">
  <xs:restriction base="xs:decimal">
  </xs:restriction>
</xs:simpleType>

```

```

<xs:simpleType name="date">
  <xs:restriction base="xs:date">
  </xs:restriction>
</xs:simpleType>

```

```

<xs:simpleType name="time">
  <xs:restriction base="xs:time">
  </xs:restriction>

```

```

</xs:simpleType>

<xs:simpleType name="timestamp">
  <xs:restriction base="xs:dateTime">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="char">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="varchar">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="longvarchar">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="bitchar">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="bitvarchar">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="bitlongvarchar">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="graphic">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="vargraphic">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="longvargraphic">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="rowid">
  <xs:restriction base="xs:hexBinary">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="blob">
  <xs:restriction base="xs:hexBinary">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="clob">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

```

```
<xs:simpleType name="dbclob">
  <xs:restriction base="xs:string">
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

---

## The catalog initialization and maintenance utility (CACCATUT)

The catalog initialization and maintenance utility (CACCATUT) is a z/OS batch job that creates or performs operations on an offline metadata catalog. Offline means that no services can reference the metadata catalog while CACCATUT is running.

You can configure CACCATUT to perform one of the following operations when it executes:

- Create and initialize a version 9.1 sequential metadata catalog
- Create and initialize a version 9.1 linear metadata catalog
- Upgrade an existing pre-version 9.1 sequential metadata catalog to a version 9.1 sequential metadata catalog.
- Upgrade an existing pre-version 9.1 linear metadata catalog to a version 9.1 sequential metadata catalog.
- Report on space utilization, the contents of a sequential metadata catalog, and any corruptions that might exist within the sequential metadata catalog.
- Reorganize the contents of a sequential metadata catalog to reclaim unused space and, where possible, correct any corruptions that might exist.
- Create a version 9.1 sequential copy of a version 9.1 sequential or linear metadata catalog.

The catalog initialization and maintenance utility is distributed in library SCACLOAD as member name CACCATUT.

### Creating and initializing sequential metadata catalogs

The INIT operation of the catalog initialization and maintenance utility (CACCATUT) initializes data sets for a version 9.1 sequential metadata catalog and creates the SYSIBM and SYSCAC system tables that make up the metadata catalog.

#### About this task

Member CACCATLG in the SCACSAMP data set contains JCL to run CACCATUT with the INIT option.

#### Procedure

To create and initialize a version 9.1 sequential metadata catalog:

Customize and run member CACCATLG in the SCACSAMP data set for a sequential metadata catalog.

1. Provide a job card that is valid for your site.
2. Change the value of the CAC parameter to your own high level qualifier.
3. Change the value of the DISKU parameter to a valid DASD unit type for your site.
4. Change the value of the DISKVOL parameter to identify the volume where you want the system catalog located.

## Creating and initializing linear metadata catalogs

To use a version 9.1 linear metadata catalog, you first create a sequential metadata catalog, populate it with the tables and other objects that you create with Classic Data Architect or the metadata utility, and then copy the content to a version 9.1 linear metadata catalog.

### Before you begin

A version 9.1 sequential metadata catalog must exist on the z/OS LPAR where the data server is located. In Classic event publishing and replication, this data server is where the correlation service is configured. To create a version 9.1 sequential metadata catalog, see “Creating and initializing sequential metadata catalogs” on page 262.

The version 9.1 sequential metadata catalog must contain the final versions of all of the mapped tables and other objects that you plan to use for Classic federation, event publishing, or replication. You cannot directly update the content of a linear metadata catalog. To modify a mapped table or any other object in a linear metadata catalog, you must update the object in a sequential metadata catalog and then copy the content of the sequential metadata catalog into your linear metadata catalog.

### Procedure

To create and initialize a version 9.1 linear metadata catalog:

1. Stop the data server. See “Stopping data servers” on page 143.
2. Customize and run member CACLCAT in the SCACSAMP data set.
  - a. Provide a job card that is valid for your site.
  - b. Change the data set names and volsers in the IDCAMS definition to match your requirements.
3. Copy the content of your version 9.1 sequential metadata catalog into the version 9.1 linear metadata catalog. See “Copying metadata catalogs” on page 264.
4. In the configuration file for the data server, set the value of the configuration parameter `STATIC CATALOGS` to 1.
5. Update the data server JCL to point to the linear metadata catalog.
6. Start the data server. See “Starting data servers” on page 139.

## Upgrading metadata catalogs

You can upgrade a version 8.2 sequential metadata catalog to a version 9.1 sequential metadata catalog.

### About this task

The `UPGRADE` operation copies an existing metadata catalog to a new version of the metadata catalog. For an `UPGRADE` operation, the `CACCAT` and `CACINDX DD` statements refer to the new metadata catalog data sets. These data sets are referred to as the target metadata catalog. The `INCAT` and `ININDX DD` statements refer to the old version of the metadata catalog.

During the `UPGRADE` operation, the user objects in the source metadata catalog are copied to the target metadata catalog. All user tables, indexes, views, and stored procedure definitions are copied to the target metadata catalog. Also, all

security authorizations that exist in the source metadata catalog, including all security authorizations that apply to the system tables, are copied to the target metadata catalog.

After the UPGRADE operation completes, the CACCATUT utility generates a summary analysis report that identifies the contents of the metadata catalog and current space utilization.

**Note:** When you upgrade a metadata catalog that contains table mappings for CA-IDMS that are flagged for data capture, CACCATUP verifies that these tables have an explicit database name defined. If such a table does not have a database name defined, CACCATUP changes the value of the DATA CAPTURE flag to a space and issues warning message 0x00760022.

### Procedure

To upgrade a version 8.2 sequential metadata catalog to a version 9.1 sequential metadata catalog:

1. Customize and run member CACCATUP of the SCACSAMP data set.
  - a. Provide a job card that is valid for your site.
  - b. Change the CAC parameter to installed high level qualifier.
  - c. Change the OLDCAT parameter to identify the high level qualifier of the input metadata catalogs.
  - d. Change the NEWCAT parameter to identify the high level qualifier of the output metadata catalogs.
  - e. Change the DISKU parameter to a valid DASD unit type for your site.
  - f. Change the VOLSER parameter to identify the volume where you want the metadata catalog located.
2. Update the JCL for the data server to point to the new metadata catalog.

## Copying metadata catalogs

You can create a copy of a metadata catalog.

### About this task

The COPY operation copies the contents of an existing metadata catalog into a new version of that metadata catalog.

The INCAT and ININDEX DD statements are required and identify the metadata catalog that is to be copied into the target metadata catalog identified by the CACCAT and CACINDEX DD statements. The INCAT and ININDEX DD statements cannot refer to the same data sets names that CACCAT and CACINDEX DD statements refer to.

During the COPY operation, you can modify the size and organization of the target metadata catalog so that it differs from that of the input metadata catalog. For example, you can change the data set organization from sequential to linear or vice versa. You can also increase or decrease the size of the target metadata catalog.

After processing finishes, a summary report is generated. This report identifies the contents of the metadata catalog and current space utilization of the created metadata catalog.

Table 36. Supported COPY operations.

- S = Sequential
- L = Linear

	To V9.1 S	To V9.1 L	To V8.2 S	To V8.2 L
From V9.1 S	Yes	Yes	No	No
From V9.1 L	Yes	Yes	No	No
From V8.2 S	No	No	Yes	Yes
From V8.2 L	No	No	Yes	Yes

### Procedure

To copy a metadata catalog:

1. Customize and run member CACCATUT of the SCACSAMP data set.
  - a. Provide a job card that is valid for your site.
  - b. Change the CAC parameter to installed high level qualifier.
  - c. Change the OLDCAT parameter to identify the high level qualifier of the input system catalogs.
  - d. Change the NEWCAT parameter to identify the high level qualifier of the output system catalogs.
  - e. For the PARM keyword of the EXEC statement, specify COPY.
2. Update the JCL for the data server to point to the target metadata catalog.

## Reorganizing metadata catalogs

You can reorganize a version 9.1 sequential metadata catalog to reclaim wasted disk space.

### Restrictions

You can reorganize sequential metadata catalogs only. You cannot reorganize linear metadata catalogs.

### About this task

The REORG operation compresses the contents of an existing sequential metadata catalog by moving the metadata catalog. The new version of the sequential metadata catalog does not contain the fragmented space that occurs as tables and other objects are dropped from the metadata catalog.

In this operation, the INCAT and ININDX DD statements reference the metadata catalog that is being reorganized and the CACCAT and CACINDX DD statements refer to the new metadata catalog that the REORG operation will create. If these statements reference an existing metadata catalog, the REORG operation replaces the content of this catalog.

If corruptions are detected in the source metadata catalog, the REORG operation attempts to minimize the loss of data when transferring the corrupted definitions from the source metadata catalog into the target metadata catalog.

After processing finishes, a summary analysis report is generated. This report identifies the contents of and the current space utilization of the new metadata catalog.

### **Procedure**

To reorganize a sequential metadata catalog:

1. Customize and run member CACCATUT of the SCACSAMP data set.
  - a. Provide a job card that is valid for your site.
  - b. Change the CAC parameter to installed high level qualifier.
  - c. Change the OLDCAT parameter to identify the high level qualifier of the input metadata catalogs.
  - d. Change the NEWCAT parameter to identify the high level qualifier of the output metadata catalogs.
  - e. For the PARM keyword of the EXEC statement, specify REORG.
2. Update the JCL for the data server to point to the new metadata catalog.

## **Generating reports about metadata catalogs**

Use member CACCATRP in the SCACSAMP data set to generate an analysis report that identifies the contents of a sequential metadata catalog that is referenced by the CACCAT and CACINDX DD statements. You can specify an additional command line parameter that identifies the level of detail to include in the report and the amount of validation to be performed on the contents of the metadata catalog.

### **About this task**

These options are available for the reports that you generate:

#### **SUMMARY**

Generates a summary report that identifies general information about the metadata catalogs, space usage within the metadata catalog and summary information about the number and types of objects stored in the metadata catalog. This is the default type of report.

#### **DETAIL**

Generates a summary report and produces a detail report that identifies all objects stored in the metadata catalog and identifies the structural linkages between the different metadata catalog objects.

#### **VALIDATE**

Generates a detail report and validates the structural linkages for each metadata catalog object.

- For more information on summary reports, see “Summary reports” on page 267.
- For more information on detail reports, see “Object detail reports” on page 270.

### **Procedure**

To generate a report

Customize and run member CACCATRP in the SCACSAMP data set.

For the PARM keyword of the EXEC statement, specify either SUMMARY, DETAIL, or VALIDATE.

## Summary reports

After the catalog initialization and maintenance utility (CACCATUT) performs UPGRADE, REORG, REPORT, and COPY operations, it generates system catalog analysis reports. All such reports include a summary report.

Summary information is printed on a single page. The following example shows the contents of the summary report that is written to SYSPRINT when an UPGRADE, REORG or COPY operation is performed, or when any kind of REPORT is requested.

```
Date: yyyy/mm/dd      metadata catalog Analysis Report      Page 1
Time: hh:mm:ss              Summary
```

### Index Component Information

```
Dataset name:          Data-Set-Name
Version identifier:    Version
Date created:          yyyy/mm/dd hh:mm:ss
Last updated:          yyyy/mm/dd hh:mm:ss
Last copied:           yyyy/mm/dd hh:mm:ss
Space used:            number-of-bytes
Maximum node ID:      number
Deleted objects:      number
Data file size used:  number
```

### Data Component Information

```
Dataset name:          Data-Set-Name
Version identifier:    Version
Catalog identifier:    Identifier
Date created:          yyyy/mm/dd hh:mm:ss
Last updated:          yyyy/mm/dd hh:mm:ss
Last copied:           yyyy/mm/dd hh:mm:ss
End-of-file:           number-of-bytes
Highest valid RID:     number
```

### Space Utilization Summary Information

Object Type	Records	Space
Tables	189	314496
Columns	5245	4028160
Fragments	282	134144
Indexes	447	457728
Keys	552	141312
Views	2	768
View dependents	2	512
Routines	32	32768
Parameters	127	32512
DB authorizations	10	1280
Table authorizations	183	46848
Routine authorizations	32	8192
User authorizations	15	1920
Indexing	33	135168
Catalog identifier	1	128
Free space	2	14976
Total	7134	5350912

The summary report has two main categories:

- Information about the index component of the metadata catalog.
- Information about the data component and the different kinds of objects that are stored in the data component.

The following fields give information about the index component of the metadata catalog:

#### Dataset name

The name of the data set for the index component.

**Version identifier**

The version of the metadata catalog. For a V9.1 metadata catalog, the version identifier is reported as 09.01.00. For a V8.x metadata catalog the value that is displayed is V8.x.

**Date created**

The date and time that the index component of the metadata catalog was created. The date and time are displayed in US English format. The creation date and time displayed for the data component should match the date and time that the index component was created.

**Last updated**

The date and time the index component was last updated, which corresponds to the date and time the last DDL statement was successfully executed for this metadata catalog. The date and time are displayed in US English format. The last update date and time displayed for the data component should match the date and time that the index component was last updated.

**Last copied**

The date and time that the index components contents was last replaced by a copy operation, or N/A if the metadata catalog has never been the target of a copy operation. The date and time are displayed in US English format. The value displayed for the data component should match the value displayed for the index component.

**Space used**

Identifies how much of the index component has been used, in bytes. The number is not related to the data set allocation size or the current physical size of the index component (that is, primary space allocation and extents). The space used is computed based on the number of logical records that exist within the index component. Each logical record is 64-bytes long, so that total spaced used should be divisible by 64.

**Maximum node ID**

The maximum node ID value identifies how many logical records exist in the index component. Each logical record is referred to as a node.

**Deleted objects**

The index component is used to track the number of metadata catalog objects (tables, views, indexes, and so on) that have been deleted from the metadata catalog. The deleted objects count identifies how many deleted metadata catalogs objects exist in the metadata catalog. You can use the REORG function to physically reclaim this unused space.

**Data file size used**

The data file size used number identifies how many bytes in the data component are currently being used. Used in this context means the space is either being actively used for an existing object, or represents “free space” because the object has been deleted. Like the space used value, the data file size used value has no relationship to the current physical DASD allocation size of the data component.

The following fields give information about the data component of the metadata catalog:

**Dataset name**

The name of the data set for the data component.

**Version identifier**

The version of the metadata catalog. For a V9.1 metadata catalog, the version identifier is reported as 09.01.00. For a V8.x metadata catalog the value that is displayed is V8.x.

**Catalog identifier**

A string value that is stored in the first logical record of the data component that identifies which version of the software was used to create the metadata catalog.

The following values can appear:

- V9.1 metadata catalog – IBM WebSphere Classic V9.1 *build-date*
- V8.2 metadata catalog - eXadas Release V8.2 *build-date*
- V8.1 metadata catalog - eXadas Release V3.0 *build-date*

The build-date takes the form *mmdyyy* and is updated for major releases and roll-up PTFs.

**Date created**

The date and time that the date component of the metadata catalog was created. The date and time are displayed in US English format. The creation date and time displayed for the data component should match the date and time that the index component was created.

**Last updated**

The date and time the data component was last updated, which corresponds to the date and time the last DDL statement was successfully executed for this metadata catalog. The date and time are displayed in US English format. The last update date and time displayed for the data component should match the date and time that the index component was last updated.

**Last copied**

The date and time that the data components contents was last replaced by a copy operation, or N/A if the metadata catalog has never been the target of a copy operation. The date and time are displayed in US English format. The value displayed for the data component should match the value displayed for the index component.

**End-of-file**

Number of physical bytes stored in the data component. For V9.1 system catalogs this value should match the Data file size used value displayed in the index component section of the report. If it does not this implies the catalog has been corrupted. For a V8.x version of the metadata catalog the end-of-file value must not match the value for *Data file size used*. The End-of-file value is used to compute the value of the highest valid RID value.

**Highest valid RID**

Identifies the highest valid record identifier (RID) that can be used to reference a record in the metadata catalog. RIDs are used to establish inter-record relationships within the metadata catalog. When a RID reference value is larger than the *Highest valid RID number* that RID is identified as being corrupted. It is not valid because it references a record that does not exist in the metadata catalog.

**Space Utilization Summary Information**

The space utilization summary information section of the report displays a table listing the different kinds of objects that are stored in the system

catalog. For each object type, this section displays the number of records that exist and the total space used for each object type. A summary line is also printed that identifies the number of records that exist in the data component and the total space that is currently being used in the data component. The total size should match the *Data file size used* number printed in the index component section of the summary report.

Table 37.

Object type	Description	Code
Tables	System tables and user tables created by a CREATE TABLE statement.	TAB
Columns	Columns associated with a table or view definition.	COL
Fragments	Objects that exist within a table to manage record arrays; or for CA-IDMS tables the record(s) referenced by the table and for IMS tables the segment(s) referenced by the table definition.	FRG
Indexes	Indexes automatically created for the system tables and user indexes created by a CREATE INDEX statement.	IDX
Keys	SYSIBM.SYSKEYS rows created when a column is referenced in a CREATE INDEX statement.	KEY
Views	View definitions created using the CREATE VIEW statement.	VEW
View dependents	SYSIBM.SYSVIEWDEP rows created to manage a CREATE VIEW statement.	VDP
Routines	System stored procedure definitions automatically created in the metadata catalog and user stored procedure definitions created by a CREATE PROCEDURE statement.	RTN
Parameters	SYSIBM.SYSPARMS rows created when a parameter is defined in a CREATE PROCEDURE statement.	PRM
DB authorizations	SYSIBM.SYSDBAUTH rows that were created due to DBMS GRANT/REVOKE statements.	ADB
Table authorizations	SYSIBM.SYSTABAUTH rows that were created due to table GRANT/REVOKE statements.	ATB
Routine authorizations	SYSIBM.SYSROUTINEAUTH rows that were created due to routine GRANT/REVOKE statements.	ART
User authorizations	SYSIBM.SYSUSERAUTH rows that were created due to user GRANT/REVOKE statements.	AUS
Indexing	Internally created index records used to optimize access to the contents of the metadata catalog.	SIX
Catalog identifier	Catalog identifier record. This is the first record in the data component and is used to record creation and last updated information.	IRD
Free space	Free space records that are available for reuse.	FRE

### Object detail reports

When you run a REPORT operation and add to the PARM parameter the DETAIL, VALIDATE, or DEBUG keywords, the metadata catalog analysis report also includes an object detail report.

The report lists two lines of information for each logical record that is stored in the data component of the metadata catalog. Page breaks occur after 50 lines of information corresponding to about 25 metadata catalog objects.

The following example shows the format of the object details report:

Date: yyyy/mm/dd                      metadata catalog Analysis Report                      Page 2  
 Time: hh:mm:ss    Object Detail

Record	RID	Size	Type	Name	Next	Prev	Start	End	Column / Parm	Table Index	Frag. Other
					-----	-----	-----	-----	-----	-----	-----
1	0	128	IRD	IBM Websphere II Classic V9.1							
2	1	1664	TAB	SYSIBM.SYSTABLES	N/A	N/A	N/A	N/A	N/A	N/A	N/A
3	14	768	COL	SYSIBM.SYSTABLES.NAME	200	41135	0	0	14	2596	0
4	20	768	COL	SYSIBM.SYSTABLES.CREATOR	20	1	N/A	N/A	1	N/A	N/A
5	26	768	COL	SYSIBM.SYSTABLES.TYPE	26	14	N/A	N/A	1	N/A	N/A
6	32	768	COL	SYSIBM.SYSTABLES.DBNAME	32	20	N/A	N/A	1	N/A	N/A
					38	26	N/A	N/A	1	N/A	N/A

The data component of the metadata catalog is organized as a set of logical records. Each logical record has a record ID (RID) associated with it and a logical record number. The minimum logical record size is 128-bytes and all logical records are an integral multiple of 128. RIDs start at zero and represent 128-byte increments in the data file. Internally, for inter-object relationships the RID is used to identify the "source" or "target" record. Therefore, given a RID the physical starting offset and the logical record can be computed in the data component.

For each logical record two lines of information are displayed. The following information is displayed on the first line:

**Record**

Identifies the logical record number for the data component object.

**RID** Identifies the logical records computed record identifier (RID.) A 'C' can be appended after the RID to indicate that corruption has been detected in the metadata catalog for the record being listed.

**Size** Identifies the length of the logical record in bytes. This value must be an integral multiple of 128.

**Type** Identifies the type of logical record. One of the values shown in the Detail Type column above in Table 4.12-7.

**Name** Identifies the external or internal name of the object. The following table identifies the different types of object names that can be displayed and their formats.

Table 38.

Object type	Name
TAB	Qualified table name: <i>owner-name.table-name</i>
COL	Qualified column name: <i>owner-name.table-name.column-name</i>

Table 38. (continued)

Object type	Name
FRG	Depends on the type of fragment, as follows: <ul style="list-style-type: none"> <li>Record Array Fragment Definitions – Fragment ID and level number</li> <li>CA-IDMS tables – record name</li> <li>IMS tables – segment name</li> <li>Table-level fragments for tables other than CA-IDMS or IMS – Fragment ID and level number</li> <li>System fragments – 'SYSTEM'</li> </ul>
IDX	Qualified index name: <i>owner-name.index-name</i>
KEY	Qualified key name: <i>owner-name.index-name.column-name</i>
VEW	Qualified view name: <i>owner-name.view-name</i>
VDP	Qualified view name and view dependent number: <i>owner-name.view-name(number)</i>
RTN	Qualified stored procedure name: <i>owner-name.routine-name</i>
PRM	Qualified parameter name: <i>owner-name.routine-name.parameter-name</i>
ADB	User and database class: <i>user-name.database-class</i>
ATB	Qualified table name and user: <i>owner-name.table-name.authorization-ID</i>
ART	Qualified stored procedure name and user: <i>owner-name.routine-name.authorization-ID</i>
AUS	Qualified object name and user: <i>owner-name.table-name.authorization-ID</i>
SIX	Internal information
IRD	Catalog identifier display in Summary section of the report
FRE	N/A

The second line of information displayed for a metadata catalog object consists of structural RID information that is stored in the record. The report lists RID information that is associated with most types of catalog objects. When there is no corresponding RID information for the object, the value N/A is displayed. A 'C' can be appended to the end of a RID to indicate that the RID value is invalid or a corruption has been detected in the referenced object.

This information is primarily for use by support personnel. The following RID information is displayed for each logical record:

**Next** The next logical RID number that that the logical record points to. Most of the metadata catalog objects are maintained as some form of linked list structure.

**Prev** The previous logical RID number that that the logical record points to. Most of the metadata catalog objects are maintained as some form of linked list structure.

**Authorization Start & End**

Identifies the first and last authorization record RIDs that are used to manage access to the object.

**Column / Parm**

Identifies the first RID of the list of columns, keys, or parameters that make up the owning object.

**Table or Index**

Identifies the RID of a related object. For example, the first index associated with a table or the first view dependent record associated with a view.

**Frag or Other**

Identifies the RID of a related object. For example, the first fragment definition associated with a table, or the first indexing record (SIX) associated with a system index definition.



---

## Accessing information about IBM

IBM has several methods for you to learn about products and services.

You can find the latest information on the Web at [www.ibm.com/software/data/sw-bycategory/subcategory/SWB50.html](http://www.ibm.com/software/data/sw-bycategory/subcategory/SWB50.html)

- Product documentation in PDF and online information centers
- Product downloads and fix packs
- Release notes and other support documentation
- Web resources, such as white papers and IBM Redbooks™
- Newsgroups and user groups
- Book orders

To access product documentation, go to this site:

[publib.boulder.ibm.com/infocenter/iisclzos/v9r1/](http://publib.boulder.ibm.com/infocenter/iisclzos/v9r1/)

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at [www.ibm.com/shop/publications/order](http://www.ibm.com/shop/publications/order).
- To order publications by telephone in the United States, call 1-800-879-2755.

To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at [www.ibm.com/planetwide](http://www.ibm.com/planetwide).

---

## Contacting IBM

You can contact IBM by telephone for customer support, software services, and general information.

### Customer support

To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).

### Software services

To learn about available service options, call one of the following numbers:

- In the United States: 1-888-426-4343
- In Canada: 1-800-465-9600

### General information

To find general information in the United States, call 1-800-IBM-CALL (1-800-426-2255).

Go to [www.ibm.com](http://www.ibm.com) for a list of numbers outside of the United States.

---

## Accessible documentation

Documentation is provided in XHTML format, which is viewable in most Web browsers.

XHTML allows you to view documentation according to the display preferences that you set in your browser. It also allows you to use screen readers and other assistive technologies.

Syntax diagrams are provided in dotted decimal format. This format is available only if you are accessing the online documentation using a screen reader.

---

## Providing comments on the documentation

Please send any comments that you have about this information or other documentation.

Your feedback helps IBM to provide quality information. You can use any of the following methods to provide comments:

- Send your comments using the online readers' comment form at [www.ibm.com/software/awdtools/rcf/](http://www.ibm.com/software/awdtools/rcf/).
- Send your comments by e-mail to [comments@us.ibm.com](mailto:comments@us.ibm.com). Include the name of the product, the version number of the product, and the name and part number of the information (if applicable). If you are commenting on specific text, please include the location of the text (for example, a title, a table number, or a page number).

---

## Notices and trademarks

---

### Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing 2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
J46A/G4  
555 Bailey Avenue  
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. \_enter the year or years\_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Trademarks

IBM trademarks and certain non-IBM trademarks are marked at their first occurrence in this document.

See <http://www.ibm.com/legal/copytrade.shtml> for information about IBM trademarks.

The following terms are trademarks or registered trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft<sup>®</sup>, Windows, Windows NT<sup>®</sup>, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel<sup>®</sup>, Intel Inside<sup>®</sup> (logos), MMX and Pentium<sup>®</sup> are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX<sup>®</sup> is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names might be trademarks or service marks of others.



---

# Index

## A

- accessibility 276
- activating
  - publications 78
- activating change-capture agents
  - CA-IDMS databases 101
- active mode
  - Adabas 168
  - CA-IDMS 177
  - change-capture agents
    - IMS 205
- Adabas databases
  - change-capture agents
    - installing 94
    - overview 92
  - configuring change capture 92
  - configuring database access 92
  - configuring event publishing
    - one server 5
    - overview 5
    - two servers 6
  - correlation services
    - multiple 93
  - recovering change data
    - automatically 164
    - manually 165
    - overview 164
  - recovery agents 165
  - recovery mode to active mode 168
- administering Classic event publishing
  - overview 139
- administering distribution services
  - stopping 163
- administering event publishing
  - correlation services
    - metrics 160
    - overview 146
    - recycling 152
    - reinitializing 152
    - starting 146
    - stopping 151
  - distribution services
    - MTO commands 156
    - starting 155
  - NVA instances 209
  - publication services
    - reports 162
  - publications
    - activating 153
    - deactivating 153
    - deactivating for send queue 154
    - metrics 161
    - reinitializing 154
- administering NVA instances
  - disabling for change capture 216
  - displaying information 220
  - enabling for change capture 214
  - installing into SVC chains 212
  - loading NVA load modules 209
  - overview 209
  - querying 218

- administering NVA instances (*continued*)
  - reports
    - activity 224
    - correlation services 221
    - overview 220
    - VSAM files 220, 222
  - restricting change data
    - filters 209
    - setting 217
  - setting trace levels 219
  - uninstalling 211
  - uninstalling from SVC chains 213
  - VSAM files
    - resetting change capture 225
    - stopping change capture 224
- administering send queues
  - MTO commands 157
- administering WebSphere MQ
  - displaying configuration 160
- altering tables for change capture 58
- altering views for change capture 58
- archived central version journal files
  - data recovery 175
- augmentation 106
- augmenting DBDs
  - IMS 104
- authorizations
  - WebSphere MQ 83
- auto-journal agents
  - CICS VSAM 116
  - configuring event publishing
    - one server 19
    - overview 19
    - two servers 21
  - configuring VSAM files 125
  - creating service information
    - entries 125
  - recovering change data 207
  - starting 208
  - starting and stopping 128
  - starting change capture 127
- auto-journal logging
  - activating for VSAM files 121
- AUTODELETE specifications
  - log data, CICS VSAM 128
- automatic recovery
  - change data
    - Adabas databases 164

## C

- CA-IDMS databases
  - automatic recovery 100
  - change-capture agents
    - installing and activating 101
    - overview 95
  - change-data
    - automatic recovery 100
  - configuring change capture
    - overview 95
  - configuring connectivity 95

- CA-IDMS databases (*continued*)
  - configuring event publishing
    - one server 8
    - overview 8
    - two servers 10
  - journals for recovery 99
  - local and central mode 99
  - multiple central versions
    - accessing 97
    - multiple correlation services 98
  - recovering change data
    - active mode 177
    - archived central version journal
      - files 175
    - disk journal files 173
    - manually 171
    - overview 168
    - recovery agents 169
    - recovery agents, keywords 170
    - recovery mode 168
    - recovery point files 170
    - single tapes or disk journal
      - files 174
  - reports
    - recovery sequences 172
  - testing
    - active mode 177
- CA-IDMS paths
  - mapping for change capture 51
- CA-IDMS tables
  - changing selection of records 72
  - modifying selection of records 72
- catalog initialization and maintenance
  - utility 262
- copying metadata catalogs 264
- creating linear metadata catalogs 263
- creating sequential metadata
  - catalogs 262
  - generating reports 266
  - object detail reports 271
  - reorganizing sequential metadata
    - catalogs 265
    - summary reports 267
  - upgrading sequential metadata
    - catalogs 263
- central version journal files
  - archived, recovering data 175
- central version mode
  - CA-IDMS databases 99
- central versions
  - access to multiple 97
  - correlation service, multiple 98
- CEP for IMS
  - recovery process requirement
    - determination 193
- change capture
  - configuration parameters
    - CLIENT CODEPAGE 228
    - COLUMN DELIMITER 228
    - COMMON FILTER TABLE
      - NAME 228

- change capture *(continued)*
  - configuration parameters *(continued)*
    - CONTROL TABLE NAME 228
    - DATASOURCE 230
    - DEFLOC 230
    - format 227
    - LD TEMP SPACE 230
    - MESSAGE POOL SIZE 232
    - NL 232
    - NL CAT 232
    - overview 227
    - ROW DELIMITER 233
    - SERVER CODEPAGE 233
    - SERVICE INFO ENTRY 233
    - STRING DELIMITER 234
  - configuring Adabas 92
  - configuring CA-IDMS 95
  - configuring CICS VSAM
    - access to files 131
    - file control exits 117
    - log-reading change-capture agents 128
    - overview 116
    - resource definitions 133
    - VTAM resource definitions 132
  - configuring IMS 103
  - configuring VSAM files
    - auto-journal agents 125
    - NVA instances 134, 135
    - overview 134
  - creating Adabas tables and views 47
  - creating CA-IDMS tables and views 49
  - creating CICS VSAM tables and views 52
  - creating IMS tables and views 54
  - creating native VSAM tables and views 56
  - IMS
    - environments and database types 103
    - monitoring 155
  - NVA instances
    - disabling 216
    - enabling 214
  - starting from CICS VSAM
    - auto-journal agents 127
    - log-reading agents 130
  - VSAM files
    - resetting for NVA 225
    - stopping for NVA 224
- change data
  - NVA
    - filters 209
    - setting filters 217
  - recovering, Adabas
    - automatic 164
    - manual 165
    - overview 164
  - recovering, CA-IDMS
    - active mode 177
    - disk journal files 173
    - manually 171
    - overview 168
    - recovery agents 169
    - recovery agents, keywords 170
    - recovery mode 168
  - change data *(continued)*
    - recovering, CA-IDMS *(continued)*
      - recovery point files 170
      - single tapes or disk journal files 174
    - recovering, CICS VSAM
      - auto-journal agents 207
      - auto-journal agents, starting 208
      - file control agents 207
      - overview 206
      - user-defined process 207
    - recovering, IMS
      - overview 177
      - process 178
      - recovery agents 183
      - recovery agents, control files 188
      - restrictions 186
      - sequence checking 184
      - XM queue overruns 186
    - recovery agents
      - Adabas 165
    - change-capture agents
      - Adabas 92
      - CA-IDMS
        - exit 95
        - installing 101
        - reports 172
      - CICS VSAM files 116
      - IMS 196
        - active mode 205
        - determining recovery 193
        - IMS Logger exit 103
        - installing 109
        - recovery data sets 108, 199
        - recovery mode 194
        - recovery mode, DBRC 196
        - recovery mode, example 195
        - status 189
        - status, examples 191
      - installing
        - Adabas 94
        - IMS 109
        - recovery mode to active mode 168
    - changing objects for Classic event publishing 62
    - child segments
      - augmentation of DBDs 106
    - CICS
      - file control agents
        - programs for 118
        - transactions for 119
        - transactions to disable 120
      - programs
        - file control agents 118
      - resource definitions, configuring 133
      - transactions
        - file control agents 119
        - file control agents, disabling 120
    - CICS VSAM
      - correlation services
        - file control agents 123
      - file control agents
        - correlation services 123
        - enabling 124
    - CICS VSAM files
      - change-capture agents 116
      - configuring access to 131
  - CICS VSAM files *(continued)*
    - configuring change capture
      - file control exits 117
      - log-reading change-capture agents 128
      - overview 116
    - configuring CICS resource definitions 133
    - configuring event publishing
      - auto-journal agents, one server 19
      - auto-journal agents, overview 19
      - auto-journal agents, two servers 21
      - file control agents, one server 15
      - file control agents, overview 15
      - file control agents, two servers 17
      - log-reading agents, one server 23
      - log-reading agents, overview 23
      - log-reading agents, two servers 24
    - configuring VTAM resource definitions 132
    - defining file control agents 117
    - defining log-reading agents 128
    - log data
      - retention period 128
    - recovering change data
      - auto-journal agents 207
      - auto-journal agents, starting 208
      - file control agents 207
      - overview 206
      - user-defined process 207
    - starting change capture
      - auto-journal agents 127
      - log-reading agents 130
  - Classic event publishing
    - administering, overview 139
    - configuring, main tasks 5
    - creating Adabas tables and views 47
    - creating CA-IDMS tables and views 49
    - creating CICS VSAM tables and views 52
    - creating IMS tables and views 54
    - creating native VSAM tables and views 56
    - mapping data for change capture 46
  - Classic replication
    - creating Adabas tables and views 47
    - creating CA-IDMS tables and views 49
    - creating CICS VSAM tables and views 52
    - creating IMS tables and views 54
    - creating native VSAM tables and views 56
    - mapping data for change capture 46
  - CLIENT CODEPAGE configuration parameter 228
  - COLUMN DELIMITER configuration parameter 228
  - columns
    - adding to tables 72
    - properties 63
    - replacing 72
  - comments on documentation 276

- COMMON FILTER TABLE NAME
    - configuration parameter 228
  - configuration
    - WebSphere MQ, displaying 160
  - configuration parameters
    - change capture
      - CLIENT CODEPAGE 228
      - COLUMN DELIMITER 228
      - COMMON FILTER TABLE NAME 228
      - CONTROL TABLE NAME 228
      - DATASOURCE 230
      - DEFLOC 230
      - format 227
      - LD TEMP SPACE 230
      - MESSAGE POOL SIZE 232
      - NL 232
      - NL CAT 232
      - overview 227
      - ROW DELIMITER 233
      - SERVER CODEPAGE 233
      - SERVICE INFO ENTRY 233
      - STRING DELIMITER 234
    - configuring change capture
      - Adabas
        - multiple correlation services 93
        - overview 92
      - CA-IDMS
        - overview 95
      - CICS VSAM
        - file control exits 117
        - overview 116
      - CICS VSAM files
        - log-reading change-capture agents 128
      - IMS
        - overview 103
      - VSAM files
        - auto-journal agents 125
        - overview 134
    - configuring CICS resource definitions
      - CICS VSAM 133
    - configuring Classic event publishing
      - main tasks 5
    - configuring connectivity
      - CA-IDMS 95
    - configuring data servers
      - name services
        - data server 38
        - overview 36
      - TCP/IP connection handlers 35
    - configuring database access
      - Adabas 92
    - configuring event publishing
      - Adabas
        - one server 5
        - overview 5
        - two servers 6
      - CA-IDMS
        - one server 8
        - overview 8
        - two servers 10
      - CICS VSAM
        - auto-journal agents, one server 19
        - auto-journal agents, overview 19
        - auto-journal agents, two servers 21
  - configuring event publishing (*continued*)
    - CICS VSAM (*continued*)
      - file control agents, one server 15
      - file control agents, overview 15
      - file control agents, two servers 17
      - log-reading agents, one server 23
      - log-reading agents, overview 23
      - log-reading agents, two servers 24
    - IMS
      - one server 12
      - overview 12
      - two servers 13
    - name services
      - description 37
      - list structure 37
    - VSAM files
      - one server 26
      - overview 26
      - two servers 27
    - WebSphere MQ
      - authorization requirements 83
      - objects for local destinations 78
      - objects for remote destinations 80
      - overview 78
  - configuring file access, CICS VSAM 131
  - configuring log file tracking, IMS 110
  - configuring VTAM resource definitions 132
  - connectivity
    - CA-IDMS 95
  - contacting IBM 275
  - control files
    - IMS, recovery agents 188
  - CONTROL TABLE NAME configuration parameter 228
  - correlation services
    - Adabas, multiple 93
    - administering, overview 146
    - CICS VSAM, for file control agents 123
    - defining 85
    - displaying metrics 160
    - IMS, multiple 107
    - monitoring 147
    - NVA instances
      - displaying information 220
      - reports 221
    - record selection exit 89
    - recycling 152
    - reinitializing 152
    - reports 147
    - starting 146
    - stopping 151
    - unknown change-capture agents 194, 195
  - creating tables
    - Adabas 47
    - CA-IDMS 49
    - CICS VSAM 52
    - IMS 54
    - native VSAM 56
  - creating views
    - Adabas 47
    - CA-IDMS 49
    - CICS VSAM 52
    - IMS 54
  - creating views (*continued*)
    - native VSAM 56
  - cross memory queues
    - overruns, IMS 186
  - CSA reporting and maintenance utility 148
- ## D
- DATA CAPTURE flag
    - tables 58
    - views 58
  - data recovery
    - CA-IDMS
      - disk journal files 173
      - single tape or disk journal files 174
    - IMS
      - log files 201
      - whether to recover 193
  - data servers
    - configuring name services
      - data server 38
      - overview 36
    - configuring TCP/IP connection handlers 35
    - name services
      - configuring 36
  - database access
    - configuring, Adabas 92
  - database types
    - IMS, supported for change capture 103
  - databases
    - properties 63
  - DATASOURCE configuration parameter 230
  - DBD statements
    - augmenting 104
    - augmenting, root only 106
    - EXIT parameter 105
  - DBRC
    - IMS log files 196
  - DDL
    - generating 73
  - deactivating
    - publications 78
  - DEFLOC configuration parameter 230
  - delimiters in character data 234
  - distribution services
    - defining 84
    - monitoring 158
    - MTO commands 156
    - starting 155
    - stopping 163
    - WebSphere MQ authorization requirements 83
  - DLPA
    - loading NVA load modules 209
    - removing NVA instances 211
  - documentation
    - accessible 276
    - ordering 275
    - Web site 275

## E

- event publishing
  - configuring Adabas
    - one server 5
    - overview 5
    - two servers 6
  - configuring CA-IDMS
    - one server 8
    - overview 8
    - two servers 10
  - configuring CICS VSAM
    - auto-journal agents, one server 19
    - auto-journal agents, overview 19
    - auto-journal agents, two servers 21
    - file control agents, one server 15
    - file control agents, overview 15
    - file control agents, two servers 17
    - log-reading agents, one server 23
    - log-reading agents, overview 23
    - log-reading agents, two servers 24
  - configuring IMS
    - one server 12
    - overview 12
    - two servers 13
  - configuring name services
    - list structures 37
  - configuring VSAM files
    - one server 26
    - overview 26
    - two servers 27
  - configuring WebSphere MQ
    - authorization requirements 83
    - objects for local destinations 78
    - objects for remote destinations 80
    - overview 78
  - correlation services
    - metrics 160
    - recycling 152
    - reinitializing 152
    - starting 146
    - stopping 151
  - distribution services
    - MTO commands 156
    - starting 155
  - overview 1
  - publication services
    - reports 162
  - publications
    - activating 153
    - deactivating 153
    - deactivating for send queue 154
    - metrics 161
    - reinitializing 154
  - XML messages
    - delimiters in character data 234
    - msg: root element 235
    - overview 234
    - row operation messages 243
    - schema 245
    - transaction messages 236

EXIT parameter
  - DBD and SEGM statements 105

exits
  - record selection 89

exporting SQL scripts 74

## F

- file access
  - configuring for CICS VSAM 131
- file control agents
  - CICS
    - programs to enable and disable 118
    - transactions for file control agents 119
    - transactions to disable 120
  - CICS VSAM 116
    - correlation services, configuring 123
    - enabling 124
  - configuring event publishing
    - one data server 15
    - overview 15
    - two data servers 17
  - defining to CICS 117
  - recovering change data 207
- file control exits
  - configuring CICS VSAM 117
- filters
  - NVA 209
  - NVA, setting 217

## G

generating DDL 73

## I

IMS databases

- augmentation of DBDs 106
- augmenting DBDs 104
- change-capture agents
  - determining recovery 193
  - installation 109
  - overview 103
  - recovery data sets 199
  - recovery mode 194
  - recovery mode, DBRC 196
  - recovery mode, example 195
  - status 189
  - status, examples 191
- configuring change capture 103
- configuring event publishing
  - one server 12
  - overview 12
  - two servers 13
- correlation services
  - multiple 107
- database types and environments 103
- EXIT parameter 105
- log files 201
- recovering change data
  - overview 177
  - process 178
  - recover agents 183
  - recovery agents, control files 188
  - restrictions 186
  - sequence checking 184
  - XM queue overruns 186
- recovery data sets 108
- restart points 113

- IMS Logger exit 103
- installing change-capture agents
  - Adabas 94
  - IMS 109

## J

journals 99

## L

- LD TEMP SPACE configuration
  - parameter 230
- legal notices 277
- linear metadata catalogs
  - creating 263
- list structures
  - configuring 37
- local mode
  - CA-IDMS databases 99
- log data
  - CICS VSAM files 128
- log file tracking
  - IMS 110, 112
- log files
  - DBRC 196
  - recovering change data 201
- log files and DBRC 196
- log-reading agents
  - CICS VSAM 116
  - configuring event publishing
    - one data server 23
    - overview 23
    - two data servers 24
- log-reading change-capture agents
  - configuring CICS VSAM files 128
  - defining for CICS VSAM files 128
  - starting and stopping 131
  - starting change capture, CICS VSAM 130
- logging
  - auto-journal
    - activating for VSAM files 121

## M

- manual recovery
  - change data, Adabas 165
- MESSAGE POOL SIZE configuration
  - parameter 232
- messages
  - XML
    - delimiters in character data 234
    - msg: root element 235
    - overview 234
    - row operation messages 243
    - schema 245
    - transaction messages 236
- metadata catalog
  - catalog initialization and maintenance
    - utility 262
  - create 262
  - reorganize 262
  - upgrade 262
- metadata catalogs
  - copying 264

- metadata catalogs (*continued*)
  - generating reports 266
  - object detail reports 271
  - populating 73
  - summary reports 267
- metrics
  - correlation services 160
  - publications 161
- modes
  - local and central version 99
- modifying objects for Classic event publishing 62
- monitoring
  - change capture 155
  - correlation services 147
  - distribution services 158
  - publication services 158
- msg: root element for XML messages 235
- MTO commands
  - displaying WebSphere MQ configuration 160
  - distribution services, reports 158
  - metrics for correlation services 160
  - metrics for publications 161
  - publication and Q subscription configuration 156
  - publication services, reports 158
  - reports for publication services 162
  - send queues 157
- multiple record layouts
  - record selection exit 89

**N**

- name services
  - configuring
    - data server to run 38
    - list structures 37
    - overview 36
  - description 37
- NL CAT configuration parameter 232
- NL configuration parameter 232
- NVA instances
  - administering 209
  - customizing 135
  - disabling for change capture 216
  - displaying information 220
  - enabling for change capture 214
  - installing into SVC chains 212
  - querying 218
  - removing from DLPA 211
  - reports
    - activity 224
    - correlation services 221
    - overview 220
    - VSAM files 220, 222
  - restricting change data
    - filters 209
    - setting 217
  - setting trace levels 219
  - uninstalling from SVC chains 213
  - VSAM files 134
    - resetting change capture 225
    - stopping change capture 224
- NVA load modules
  - loading into DLPA 209

- NVA SVC Manager
  - administering NVA instances 209

**P**

- populating metadata catalogs 73
- properties
  - columns 63
  - databases 63
  - publications 64
  - publishing queue maps 64
  - tables 65
  - views 71
- publication services
  - reports 162
- publications
  - activating 78, 153
  - creating 77
  - deactivating 78, 153
  - deactivating for send queue 154
  - displaying metrics 161
  - modifying 77
  - properties 64
  - reinitializing 78, 154
- publishing environments 1
- publishing messages
  - local destinations 78
  - remote destinations 80
  - required WebSphere MQ objects
    - local 78
    - remote 80
  - WebSphere MQ authorizations 83
- publishing queue maps
  - properties 64

**Q**

- queries
  - NVA instances 218

**R**

- readers' comment form 276
- recovering change data
  - Adabas
    - automatic 164
    - manually 165
    - overview 164
  - CA-IDMS
    - active mode 177
    - archived central version journal files 175
    - disk journal files 173
    - manually 171
    - overview 168
    - recovery agents 169
    - recovery agents, keywords 170
    - recovery mode 168
    - recovery point files 170
    - single tapes or disk journal files 174
  - CICS VSAM
    - auto-journal agents 207
    - file control agents 207
    - overview 206
    - user-defined process 207

- recovering change data (*continued*)
  - IMS
    - change-capture agents, status 189, 191
    - determining need 193
    - overview 177
    - process 178
    - recovery agents 183
    - recovery agents, control files 188
    - restrictions 186
    - sequence checking 184
    - XM queue overruns 186
  - recovery agents
    - Adabas 165
    - CA-IDMS
      - keywords 170
      - overview 169
    - IMS 183
      - control files 188
    - journals 99
  - recovery availability 99
  - recovery data sets
    - IMS
      - creating 108
      - recovery mode 199
  - recovery mode
    - Adabas
      - moving to active mode 168
    - CA-IDMS 168
    - change-capture agents, IMS 194, 195
    - IMS
      - DBRC 196
      - example 195
      - unknown to correlation services 194
    - restrictions for recovering data 186
  - recovery point files
    - CA-IDMS 170
  - recovery process
    - IMS 178
  - recovery, automatic
    - CA-IDMS 100
  - reinitializing
    - publications 78
  - reinitializing correlation services 152
  - reports
    - CA-IDMS
      - recovery sequences 172
    - correlation services 147
    - NVA instances
      - activity 224
      - correlation services 221
      - overview 220
      - VSAM files 220, 222
    - publication services 162
    - publication services and distribution services 158
  - resource definitions
    - CICS, configuring 133
    - VTAM, configuring 132
  - restart points
    - IMS databases 113
  - restore utility
    - recovering change data 207
  - retention period
    - log data
      - CICS VSAM files 128

- root-only databases
  - augmentation of DBDs 106
- ROW DELIMITER configuration
  - parameter 233
- row operation XML messages 243

## S

- schema for XML messages 245
- screen readers 276
- SEGM statement
  - EXIT parameter 105
- send queues
  - MTO commands 157
- sequence checking for log records
  - IMS 184
- sequential metadata catalogs
  - creating 262
  - reorganizing 265
  - upgrading 263
- SERVER CODEPAGE configuration
  - parameter 233
- SERVICE INFO ENTRY configuration
  - parameter 233
- service information entries
  - creating for auto-journal agents 125
- services
  - correlation
    - defining 85
    - displaying metrics 160
    - file control agents 123
    - monitoring 147
    - multiple in IMS 107
    - NVA instances 220
    - overview 146
    - record selection exit 89
    - recycling 152
    - reinitializing 152
    - reports, NVA instances 221
    - starting 146
    - stopping 151
  - distribution
    - defining 84
    - monitoring 158
    - MTO commands 156
    - starting 155
    - stopping 163
  - publication
    - monitoring 158
- source tables 77
- source views 77
- SQL scripts
  - exporting 74
- STRING DELIMITER configuration
  - parameter 234
- SVC chains
  - querying NVA instances 218
  - uninstalling NVA instances 211, 212, 213

## T

- tables
  - adding or replacing columns 72
  - altering for change capture 58

- tables (*continued*)
  - mapping
    - to CICS VSAM files 52
    - to native VSAM files 56
  - mapping to Adabas databases 47
  - mapping to CA-IDMS databases 49
  - mapping to IMS databases 54
  - properties 65
- TCP/IP connection handlers
  - configuring 35
- trace levels
  - NVA instances 219
- tracking log files
  - IMS 110, 112
- trademarks 279
- transaction XML messages 236

## U

- utilities
  - catalog initialization and maintenance
    - utility 262
  - CSA reporting and maintenance 148

## V

- viewing objects for Classic event
  - publishing 62
- views
  - altering for change capture 58
  - creating on existing tables 59
  - creating with SQL editor 60
  - creating with the SQL builder 59
  - properties 71
- VSAM files
  - auto-journal logging
    - activating 121
  - configuring change capture
    - auto-journal agents 125
    - overview 134
  - configuring event publishing
    - one server 26
    - overview 26
    - two servers 27
  - NVA instances 134, 135
  - reports 222
  - resetting change capture 225
  - stopping change capture 224
  - reports, NVA instances 220
- VTAM resource definitions
  - configuring for CICS VSAM 132

## W

- WebSphere MQ
  - authorization requirements 83
  - configuring for event publishing
    - objects for local destinations 78
    - objects for remote destinations 80
    - overview 78
  - displaying configuration 160
- WebSphere MQ Java Messaging Service
  - local 78
  - remote 80
- WebSphere MQ objects
  - required for local destinations 78

- WebSphere MQ objects (*continued*)
  - required for remote destinations 80

## X

- XM queues
  - distribution services 84
  - IMS, overruns 186
- XML delimiters
  - in character data 234
- XML messages
  - delimiters in character data 234
  - event publishing 234
  - msg: root element 235
  - row operation messages 243
  - schema 245
  - transaction messages 236





Printed in USA

SC19-1121-00



Spine information:

IBM Information Integration

Version 9.1

Classic Event Publishing Guide and Reference

