IBM InfoSphere Classic Change Data Capture for z/OS

Guide and Reference

Version 11 Release 3

IBM InfoSphere Classic Change Data Capture for z/OS

Guide and Reference

Version 11 Release 3

Note

Before using this information and the product it supports, read the information in "Notices" on page 295.

Edition notice

This edition applies to version 11, release 3 of IBM InfoSphere Classic Change Data Capture for z/OS (product number 5655-W29) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. About InfoSphere Classic Change Data Capture for z/OS

IBM® InfoSphere® Classic Change Data Capture for z/OS® (InfoSphere Classic CDC for z/OS) is a replication solution that captures changes to non-relational mainframe data and delivers them to relational databases, producing an accurate relational replica of your mainframe data on supported target databases in near-real time.

With the InfoSphere Classic CDC for z/OS solution, you can replicate changes to an InfoSphere Change Data Capture target replication engine and you can publish changes in a delimited format directly to WebSphere® MQ.

To capture changes to your IBM IMS™ and VSAM data sources, InfoSphere Classic CDC for z/OS reads data from database logs to capture the changes to your non-relational mainframe data. The source engine packages the changes as discrete change messages that contain a relational description of the insert, update, and delete operations on the data. It then sends the change messages to an InfoSphere CDC target engine or sends change messages in a delimited format to WebSphere MQ.

The CDC target engine delivers them in one of the following ways:

- Applies the changes to a supported target database
- Sends the changes to IBM InfoSphere DataStage® for transformation processing

The InfoSphere Classic CDC for z/OS solution requires the following software components:

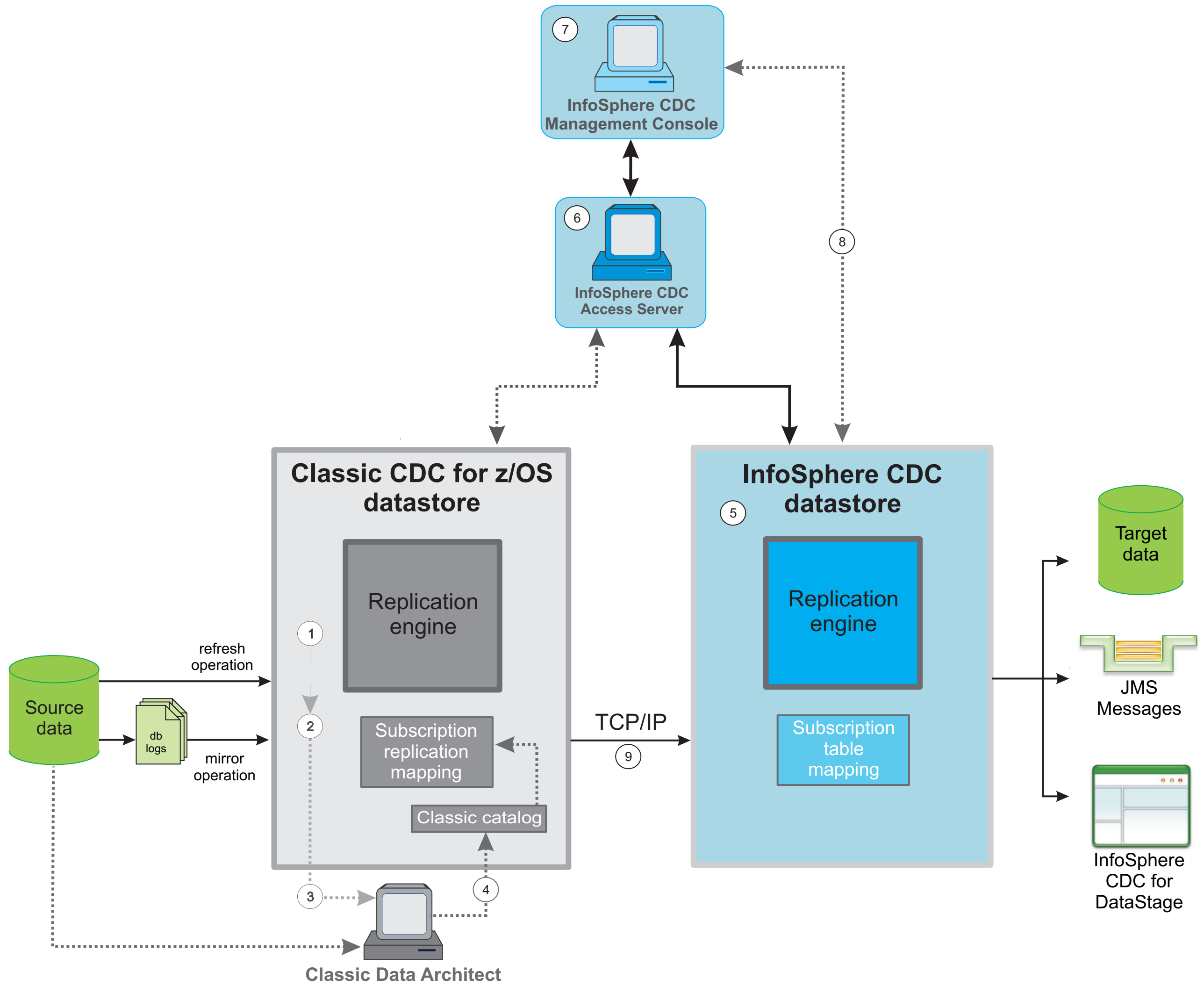
Component		Function
Source replication engine	InfoSphere Classic CDC for z/OS	The source engine is a Classic data server that runs in its own z/OS address space. The data server contains the services and other components that manage replication and process changes. The data server manages change data capture and refresh processing by coordinating the components and activities that the following sections describe in greater detail: • Maintains the metadata catalog which contains definitions for tables and underlying data source information as well as stored procedures. • Maintains metadata that describes subscriptions and replication mappings to ensure accurate replication • Processes commands from users or from remote configuration and administration tools • Transfers change messages to the target engine over a TCP/IP connection
Source replication engine configuration and administration tool	IBM InfoSphere Classic Data Architect	The GUI tool for InfoSphere Classic CDC for z/OS, which connects to the source engine from a client workstation to simplify configuration tasks and create table mappings.
Target replication engine	InfoSphere Change Data Capture	The target engine receives change messages from the source engine and applies the changes to the specified target tables in the correct order. It also performs these additional functions: • Tracks restart positions that specify where log reading resumes after replication stops and restarts • Performs code page conversions • Exchanges metadata about subscriptions and replication mappings with the source engine to validate your deployment • Processes commands from users or from remote configuration and administration tools

Component		Function
Target replication engine configuration and administration tools	• InfoSphere Data Replication Management Console • InfoSphere Data Replication Access Server • InfoSphere Data Replication Access Manager	Management Console configures and monitors replication from a client workstation. Access Server is a server that Management Console connects to for authentication purposes. Access Manager creates datastores and users, assigns roles and privileges to users, and authenticates and connects users to datastores.

Understanding the InfoSphere Classic CDC for z/OS workflow

The illustrations and the explanations that follow describe how the components of the InfoSphere Classic CDC for z/OS solution work together.

Illustration 1: Sending change messages to an InfoSphere CDC target engine. This illustration shows an environment that is set up for the InfoSphere Classic CDC for z/OS source replication engine to communicate with an InfoSphere CDC target replication engine.

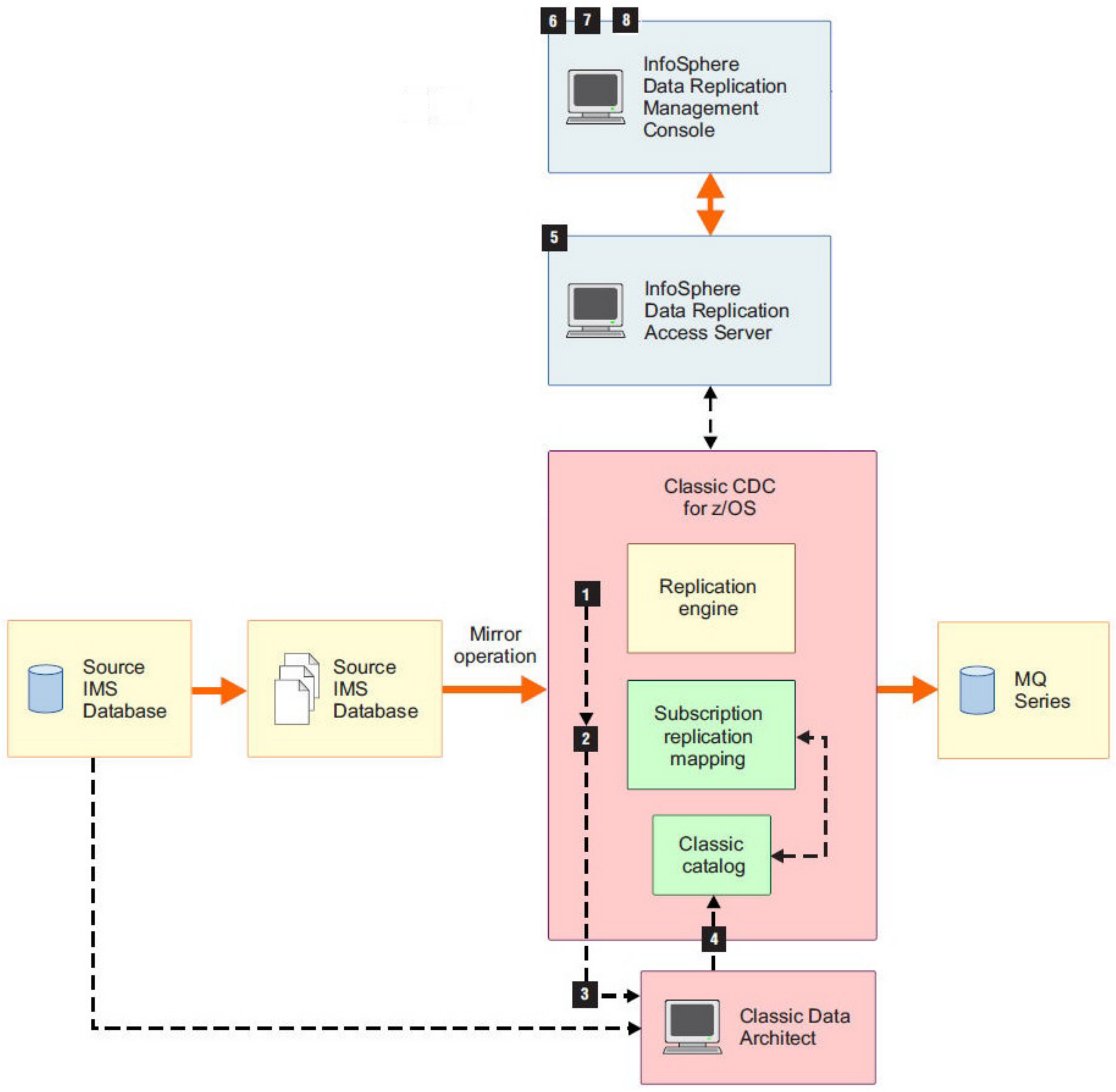


1. Install InfoSphere Classic CDC for z/OS Server.
2. Follow the InfoSphere Classic CDC for z/OS installer process for customizing the Classic address space and services to suit your business requirements.
3. Install Classic Data Architect.
4. Design and define the source tables in Classic Data Architect. This establishes a relational structure for the non-relational data in the source database within the InfoSphere Classic CDC for z/OS engine. Alter the tables for change capture. The tables and columns that you define in these mappings are later used to determine what data is replicated.
5. Install and configure the InfoSphere CDC target replication engine.
6. Install Access Server.
7. Install Management Console.
8. Set up the replication environment by using Management Console to complete the following associated tasks:
   a. Create the source InfoSphere Classic CDC for z/OS data source
   b. Create a target data source
   c. Create a subscription using the source InfoSphere Classic CDC for z/OS datastore and the target datastore you created

d. Map the InfoSphere Classic CDC for z/OS source datastore tables to the target datastore

9. Start refresh or replication using Management Console.

Illustration 2: Publishing changes to WebSphere MQ. This illustration shows an environment that is set up for the InfoSphere Classic CDC for z/OS source replication engine to publish changes in a delimited format directly to WebSphere MQ.



1. Install InfoSphere Classic CDC for z/OS Server.
2. Follow the InfoSphere Classic CDC for z/OS installer process for customizing the Classic address space and services to suit your business requirements.
3. Install Classic Data Architect.
4. Design and define the source tables in Classic Data Architect. This establishes a relational structure for the non-relational data in the source database within the

InfoSphere Classic CDC for z/OS engine. Alter the tables for change capture. The tables and columns that you define in these mappings are later used to determine what data is replicated.

5. Install Access Server.
6. Install Management Console.
7. Set up the replication environment by using Management Console to complete the following associated tasks:
   a. Create the InfoSphere Classic CDC for z/OS data source
   b. Create a subscription using the source InfoSphere Classic CDC for z/OS datastore as both the source and target datastore
   c. Add the InfoSphere Classic CDC for z/OS source datastore tables to the subscription
8. Start replication by using Management Console.

Comparing InfoSphere Classic CDC for z/OS and InfoSphere CDC terminology

InfoSphere Classic CDC for z/OS uses terms that have similar or synonymous meanings to terms that are used by previous Classic versions and InfoSphere CDC.

The following table provides past and present terms for equivalent functionality across the two products.

Table 1. Comparisons of current product terminology with previous terms

Current product term	Previous Classic terms	InfoSphere CDC
bookmark	restart point restart position	bookmark
change data capture	change capture	change data capture
column filtering*	table/view	column filtering
column mapping*	table/view	column mapping
controlled stop	quiesce	controlled stop
data source	data source	datastore source database target database
datastore credentials	data server credentials ODBC credentials JDBC credentials	datastore credentials
datastore user	user	datastore user
delimited format subscription (change messages in a delimited format published to WebSphere MQ)	publishing map send queue	(Not applicable)
immediate stop	N/A	immediate stop
log reader	log reader service	scraper
log reading	log reading	scraping
replication	replication	mirroring

Table 1. Comparisons of current product terminology with previous terms (continued)

Current product term	Previous Classic terms	InfoSphere CDC
replication mapping	(Not applicable)	table mapping
replication object	user table data source database file system	table
row filtering*	view	row filtering
source server	source data server capture server	source server
subscription (change messages sent to an InfoSphere CDC target engine)	subscription	subscription
table mapping	user table table mapping mapped table mapping publication	(Not applicable) In InfoSphere CDC, *table mapping* is synonymous with *replication mapping*.
target server	target server apply server	target server
* In InfoSphere Classic CDC for z/OS, you define tables or views to implement a static version of this function.		

Chapter 2. System requirements

Before you install your InfoSphere Classic CDC for z/OS source replication engine and your InfoSphere CDC target replication engine, ensure that the systems you choose meet the necessary operating system, hardware, software, communications, disk, and memory requirements.

Software requirements

The following list indicates the minimum software versions required to successfully replicate using InfoSphere Classic CDC for z/OS:

- InfoSphere Classic CDC for z/OS version 11.3
- Classic Data Architect version 11.3
- Management Console version 11.3.3
- Access Server version 11.3.3
- InfoSphere CDC version 11.3.3
- WebSphere MQ version 7.0.1

Note: The installed version of Management Console and Access Server must be equal to or greater than the highest version of the installed InfoSphere CDC replication engine or CDA.

System requirements for InfoSphere Classic CDC for z/OS and Classic Data Architect

Before you install IBM InfoSphere Classic Change Data Capture for z/OS , ensure that your system meets all of the system prerequisites and requirements.

System requirements: IMS

For replication from IMS data sources, your system must meet the following system prerequisites and requirements.

Prerequisites

- IBM z/OS Version 1.13 or later
- IBM IMS Version 12.1 or later
- IBM InfoSphere CDC, Version 6.5 or later

Supported IMS databases

Classic change data capture supports the following IMS database types:

- Direct entry (DEDB)
- High availability large (HALDB)
- Hierarchical direct access method (HDAM)
- Hierarchical indexed direct access method (HIDAM)

DL/I batch jobs (IMS)

You can capture changes that DL/I batch jobs make to IMS databases only under the following conditions:

- The batch job must run with `DBRC=Y`.
- The DFSFLGX0 exit must be in the STEPLIB of the batch job.
- IEFRDER must point to a permanent dataset, therefore cannot be `DUMMY`.

Other requirements (IMS)

- You must register subsystems whose changes you want to capture in a shared set of RECON data sets.
- You must catalog all IMS logs that you want to capture in a shared integrated catalog facility (ICF).
- DASD-based logs must reside on shared DASD that any image in the source sysplex can access.
- On the source IMS site, before you use any newly allocated online log data set (OLDS), you must ensure that the space used by the OLDS is filled to capacity by valid IMS log records at least once. This requirement is more stringent than the related recommendation that the space used by an OLDS allocated on newly initialized or reinitialized volumes should first be formatted for performance reasons.

System requirements: VSAM

For replication from VSAM data sources, your system must meet the following system prerequisites and requirements.

Prerequisites

- IBM z/OS Version 1.13 or later
- CICS® Transaction Server for z/OS (CICS TS) Version 5.1 or later
- CICS VSAM Recovery (CICS VR) Version 5.1 or later is required to replicate changes made by batch jobs outside of CICS Transaction Server.
- APAR OA39035, PTF UA65303 on z/OS Version 1.13
- APAR PM86451, PTF UK97129 on CICS TS Version 5.1
- APAR PM95393 on CICS TS Version 5.1
- APAR PM95451 on CICS VR Version 5.1
- APAR PM95661 on CICS TS Version 5.1
- APAR PM96728 on CICS TS Version 5.1
- APAR PI13825, PTF UI18777 on CICS TS Version 5.1
- APAR PI16561, PTF UI20138 on CICS TS Version 5.1, PTF UI20139 on CICS TS Version 5.2

Supported VSAM data set types

Classic CDC supports capturing changes from online environments under the control of the CICS TS and CICS VR.

Classic CDC supports the VSAM key-sequenced data set (KSDS) organization.

Other requirements (VSAM)

- You cannot define table mappings or views for VSAM data sets that do not exist.

- All VSAM data sets in a subscription must use the same replication log. All tables mappings in the same subscription must be mapped to VSAM data sets that use the same replication log. If you change the name of the log stream defined to any one VSAM data set within a subscription, all other VSAM data sets in the same subscription must also be updated to use the same replication log stream name. Otherwise you must move the changed VSAM data set to a new subscription.
- A source data set cannot be opened by any address space that will be logging updates when the log position is set.

System requirements for InfoSphere CDC, Management Console and Access Server

Click the following links to view hardware and software requirements for InfoSphere Data Replication, InfoSphere CDC, Management Console, and Access Server:

Table 2. System requirements for InfoSphere CDC versions 10.2.x

Version	Linux, UNIX and Windows	z/OS
Version 10.2.1	https://ibm.biz/BdxwE7	https://ibm.biz/BdxwXB
Version 10.2	https://ibm.biz/BdxyzE	https://ibm.biz/Bdxyd5

Table 3. System requirements for InfoSphere CDC versions 6.5.x

Version	Operating Systems	Databases
Version 6.5.2	https://ibm.biz/BdDGas	https://ibm.biz/BdDGaZ
Version 6.5.1	https://ibm.biz/BdDGaE	https://ibm.biz/BdDGaX
Version 6.5	https://ibm.biz/BdDGad	https://ibm.biz/BdDGaD

Chapter 3. Before you install

This section contains information about the tasks that should be performed prior to installing InfoSphere Classic CDC for z/OS, Classic Data Architect, and InfoSphere CDC.

Restrictions

Before you install IBM InfoSphere Classic Change Data Capture for z/OS, ensure that your data sources qualify for replication.

Restrictions for IMS data sources

Unsupported database types

Classic change data capture does not support the following IMS database types:

- GSAM (generalized sequential access method)
- HSAM (hierarchical sequential access method)
- MSDB (main storage databases)

Certain logically-related databases

Logically-related databases with multiple access paths can lead to conflicting conditions that make these databases ineligible for replication. When you consider them together, the following criteria for EXIT parameters, logical relationships, and delete rules create the conflict:

- Logical child segments with EXIT coding *must* have a virtual delete rule.
- Logical parent segments with EXIT coding *cannot* have a virtual delete rule.

If a segment acts as both a logical parent and logical child, you cannot include an EXIT statement on that segment or any of its children. Therefore you cannot enable a table or view for change data capture that is not augmented with an EXIT statement.

Unsupported subsystem and application types

You cannot capture changes from IMS subsystems that operate in an Extended Recovery Facility (XRF) complex.

Consider these additional restrictions when you implement Classic change data capture:

- Classic change data capture does not replicate data that you restore from image copy.
- Classic change data capture does not replicate data that you load into the database.
- You must register subsystems whose changes you want to capture in a shared set of RECON data sets.
- You must catalog all IMS logs that you want to capture in a shared integrated catalog facility (ICF).

- DASD-based logs must reside on shared DASD that any image in the source sysplex can access.
- Changes made by applications that issue FLD calls can be captured when using IMS version 13 and when the segment is augmented with the FLD option.
- You cannot capture changes to primary and secondary indexes.

Resource consumption

Classic change data capture from an IMS and VSAM data source increases resource consumption on the source logical partition (LPAR). Examples include logging and archiving activity, CPU consumption, network bandwidth, and memory.

Archive logs

- For IMS data sources

  IMS creates more archive logs, which increases the amount of direct access storage devices (DASD), virtual tape, or physical tape that your LPAR requires. If your LPAR runs DL/I batch workload that logs to DASD, you might have to increase the size of these logs by a factor of 2 or 3 to accommodate the additional volume of data.
- For VSAM data sources

  VSAM logging causes the z/OS System Logger to create offload log data sets. Depending on the RETPD setting and the amount of logging, there might be increased numbers of log data sets with replication logging.

Processing time

CPU time increases measurably, though not necessarily significantly, compared with processing the same workload without change capture. Batch processing produces the most measurable increase.

Network bandwidth

Classic change data capture uses significant network bandwidth, and requires TCP/IP links between the source and target engines to operate effectively at high speeds. All replicated data and the control messages necessary to maintain replication flow over the network connection.

Note: This type of network connection does not apply when publishing changes in delimited format to WebSphere MQ.

Service classes

Classic change data capture must run in high-priority service classes to maintain low latency. Use a workload manager (WLM) to give the Classic data servers the resources that they require to synchronize source and target databases in near-real time. Define service classes and assign the Classic data servers to these classes. Give the Classic data servers a higher priority to allocate sufficient cycles for the workload.

Typically, you give the target engine a higher relative dispatching priority than the source server so that it can offload work as fast as (or faster than) the source server sends it.

Note: Target servers do not apply to publishing changes in delimited format to WebSphere MQ.

Memory

The source Classic data server consumes system memory.

Source servers have a capture cache that can require considerable resources, especially as the number of subscriptions increase.

Subscription processing

Subscriptions use a cache to accommodate differences in the speed of the source and target servers. These subscription-level caches can improve performance when replication errors require your Classic data servers to catch up to current processing. You configure the cache independently by using a z/OS operator command after the subscription is created.

Classic data servers

A Classic data server runs in its own address space.

The source engine is a Classic data server that runs in its own z/OS address space. The target engine can be any environment that InfoSphere change data capture supports.

Classic data servers perform the following functions:

- Determine the type of data to access
- Maintain the environment
- Process changed data
- Administer replication
- Map tables and views and alter them for data capture

The architecture of a Classic data server is service-based. The Classic data server consists of several components, or *services*.

Services and their functions

When a Classic data server is created during the installation customization process, the services required for the Classic data server are pre-configured.

Configuration definitions include the following server-wide and individual service categories:

- Server-wide, or global, definitions that affect all services within the Classic data server
- Service definitions specific to change data capture that consist of unique configuration information that affects each service individually.

Each service is a member of a service class. Services also have a service name and a task name.

service class

The type of service, such as the capture service (CAP) or the administration service (PAA).

service name

The unique name that references a specific instance of a service in a Classic data server.

task name

The load module that is associated with services of a service class.

A service class contains a specific set of configuration parameters. The values of the configuration parameters define a service instance and the behavior of that service.

Critical services

A *critical service* is a service that is critical to the operation of the Classic data server. The Classic data server cannot continue running when one or more services that are critical to the operation of the server are stopped or stop abnormally.

The following services are critical services:

Administration service

The administration service manages activities such as creating and updating replication metadata and processing client requests.

Capture service

The capture service runs in the source server. This service manages the source side of replication, including the log reader service and change streams.

IMS log reader service

The IMS log reader service reads IMS log files and sends change messages to the capture service.

Logger service

The logger service receives messages from all services in the Classic data server and coordinates writing the messages to common logs.

You cannot stop a critical service. If you attempt to stop a critical service by issuing a STOP,SERVICE command or a STOP,TASKID command, a warning message is issued.

Detailed information describes each service that runs in the Classic data server, each service configuration parameter, and configuration methods for administrating the configurations for the Classic data server.

Optimizing memory consumption for a Classic data server (guidelines)

To optimize memory consumption, estimate initial memory settings in the job control language (JCL) for your Classic data server and then evaluate them in a test environment.

Procedure

1. Estimate initial values for **REGION** in the JCL for the Classic data server and for the **MESSAGEPOOLSIZE** configuration parameter.
   - For smaller environments, try the default values for the Classic data server:
     - REGION=96MB
     - MESSAGEPOOLSIZE=64MB
   - Consider larger values for **REGION** and **MESSAGEPOOLSIZE** for larger deployments that require more resources. For example, you can begin with values that are larger than needed at first. Then you can define your environment and work toward reducing these values by monitoring the environment and running reports such as the output from the DISPLAY,MEMORY command.

Consider factors that contribute to resource consumption:

– Fixed overhead per Classic data server outside message pool storage:
  - C runtime library functions (LE)
  - Added threads
  - Added threads, such as the WebSphere MQ writer threads. One WebSphere MQ writer thread will be created for each active delimited publishing subscription.
– The number of subscriptions, and related metadata:
  - Subscriptions
  - Replication mappings
  - Replication objects

2. Experiment with different configurations in a test environment to verify that your Classic data server has sufficient resources for the size of your environment.
   a. Specify a region size that is at least 8 MB lower than the site limit, and use the greater of these values:
      - 8 MB higher than the message pool
      - 20% higher than the message pool

      If the 8 MB gap between the region and the message pool is still not sufficient, increase this difference in increments of 8 MB.
   b. Set the **MESSAGEPOOLSIZE** parameter to the greater of these values:
      - 20% less than the region size
      - 8 MB below the **REGION** value or 8 MB below any site limit imposed by exits.

      If you increase the value of the **MESSAGEPOOLSIZE** parameter, set the region size higher to maintain the 8 MB gap.
3. If possible, set the maximum cache size of 3 GB per subscription for the source server.

   Caches on the source server use storage areas outside of the message pool storage, but those storage areas can also be a factor in estimating region size.

   The size of the capture cache that you set with the CAPTURECACHE parameter represents the maximum size of one of the storage areas associated with the capture cache. A capture cache consists of multiple storage areas. As a result, the maximum storage that a capture cache uses is approximately 1.5 times the specified value. For example, if you specify 2048M as the capture cache size of a subscription, in a situation where the capture cache fills, the capture cache will occupy approximately 3072 megabytes of storage.

Latency and DL/I batch applications (IMS)

For IMS data sources, InfoSphere Classic CDC for z/OS can capture changes made by DL/I batch jobs. However, these jobs can take hours to complete, which introduces latency.

The logs are not available for change data capture until each batch job step finishes, because InfoSphere Classic CDC for z/OS cannot access the IMS log until it closes. When a DL/I batch job starts and identifies itself as a subsystem that affects ordering, the IMS log reader service must suspend ordering operations until each DL/I batch job step finishes processing. The process of ordering and merging resumes when the source server can access the log.

To help manage latency, you can specify a list of IMS subsystems (both DL/I batch and online) to exclude from ordering decisions.

If your business requirements cannot accept latency, convert any existing DL/I batch workload to run as a batch message processing (BMP) program under the control of a DB/DC or DBCTL subsystem.

Chapter 4. Installing InfoSphere Classic CDC for z/OS

Installing InfoSphere Classic CDC for z/OS consists of installing mainframe components and the Classic Data Architect, preparing your installation environment, and customizing the installation to create a functional runtime environment.

The following table lists the major tasks required for Classic change data capture installation with a link to where to find information about each task. Perform the tasks in the following recommended order.

Task		Reference
1	Perform the SMP/E tasks to install the components required for Classic change data capture on the mainframe.	"Obtaining installation instructions for the mainframe"
2	Prepare for the installation customization process by completing the tasks in the checklist for setting up the installation environment. Obtaining items such as required authorizations and port numbers will prepare you for the customization procedure.	"Setting up the installation environment" on page 20
3	Customize the installation environment by completing the tasks in the installation customization process for the type of Classic data server that you want to customize.	"Installation customization process" on page 20
4	Install the Classic Data Architect client application to manage server connections, tables, views and subscriptions, monitor metrics, and perform configuration tasks.	Chapter 5, "Installing the Classic Data Architect," on page 41

Obtaining installation instructions for the mainframe

The IBM InfoSphere Classic Change Data Capture for z/OS product is included on tape and the installation instructions are provided in the product program directory.

The program directory details the system requirements and provides installation instructions that supplement the installation information in the product library. See InfoSphere Classic Change Data Capture for z/OS Program Directory V11.3.

Setting up the installation environment

After you complete the mainframe SMP/E installation, the next step in the installation process is to set up the installation environment. Setting up the installation environment is a prerequisite to the installation customization process.

The following table provides a checklist of tasks needed to set up the installation environment for source servers.

Table 4. Checklist of installation environment setup for source servers

Task	Reference
Obtain APF library authorizations for the installation load library SCACLOAD	Obtaining library authorizations for the authorized program facility (APF)
Assign port numbers for communication for Classic data servers and log reader notification.	Obtaining ports for replication
Set up resources profiles and security classes for security for Classic data servers.	Securing a Classic data server
Ensure that you have the authorization required to run the Administrative Data Utility (IXCMIAPU). You need this authorization before you run the utility to define the Classic event log and the diagnostic z/OS log streams.	Administrative Data Utility
If you plan to publish changes in delimited format to WebSphere MQ on z/OS, perform the setup requirements.	Configuring publishing to WebSphere MQ
Augment data for change data capture.	For IMS data sources: Augmenting DBDs for change data capture For VSAM data sources: Augmenting VSAM data sets for change data capture

Installation customization process

Installation customization is a process that allows you to provide setup and configuration information to create a customized installation environment for a source server.

The installation customization process involves a set of steps that you perform after you complete the mainframe SMP/E installation. You provide setup and configuration information that is used to generate all of the sample JCL and configuration members in the *USERHLQ*.USERSAMP data set that require edits. You then run installation customization jobs that are generated based on the parameters that you specify to create a customized installation environment.

The installation customization process is based on the role of the Classic data server. You specify the role of a Classic data server with the SERVERROLE parameter for the installation customization utilities. This parameter controls the installation components that you customize. You create installation data sets (*USERHLQ*.USERSAMP and *USERHLQ*.USERCONF) that contain the required components for the type of installation that you choose, and you customize only the parameters needed for that environment.

When you complete the installation customization process, an operational environment is established that you can build upon as needed. The environment includes a functional Classic data server and all of the services required for the specified role. All services are pre-configured during the customization process.

Overview of installation customization procedure

The installation customization process consists of the following basic steps:

1. The user samples allocation utility creates a working set of the SCACSAMP and SCACCONF data sets that contain all customized JCL and configuration members. This working set is referred to as the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets.
2. You gather site-specific configuration information needed to customize the environment and enter that information in the customization parameters file.
3. The installation customization utility generates customized JCL and configuration members and stores them in the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets that were created in the first step.
4. You allocate and initialize the following components by using generated customization jobs:
   - z/OS log streams
   - Configuration files
   - A zFS aggregate that you define and format
   - Metadata catalogs
   - Data sets to store your subscriptions and replication mappings
5. You use the generated JCL to create the WebSphere MQ bookmark queue for delimited publishing.
6. You use the generated JCL and configuration members to start the runtime environment.

Installation customization components

The following table lists the components and sample JCL members that you use during the installation customization process.

Table 5. Summary of installation customization components distributed in the SCACSAMP data set.

Component name	**Description**
User samples allocation utility	Allocates the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets. Populates the *USERHLQ*.USERSAMP data set with a copy of the customization parameters file (CECCUSPC) and the installation customization utility JCL (CECCUSC2). The SCACSAMP(CECCUSC1) JCL runs this utility. SCACSAMP(CECCUSC1) is the JCL that runs the user samples allocation utility. CECCUSC1 is the only member in the distributed SCACSAMP data set that you edit. The JCL comments provide editing instructions.

Table 5. Summary of installation customization components distributed in the SCACSAMP data set. (continued)

Component name	Description
Installation customization utility	Reads the customization parameters file *USERHLQ*.USERSAMP(CECCUSPC) and generates the necessary JCL and configuration members in the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF partitioned data sets. The *USERHLQ*.USERSAMP(CECCUSC2) JCL runs this utility. *USERHLQ*.USERSAMP(CECCUSC2) is the generated JCL that submits the installation customization utility and generates all necessary JCL and configuration members.
Customization parameters file	Contains the installation and customization information that you specify in the form of parameter and value pairs to complete an installation and establish an initial functioning environment. This file is located in *USERHLQ*.USERSAMP(CECCUSPC).
The following components are specific to the server role for the source server.	
USERHLQ.USERSAMP(CECCDSLS)	Generated JCL that runs the Administrative Data Utility (IXCMIAPU) to define the z/OS event log stream and a log stream for the diagnostic log for the source server.
USERHLQ.USERSAMP(CECCDCFG)	Generated JCL that allocates and initializes the configuration files for the source server.
USERHLQ.USERSAMP(CECCRZCT)	Generated JCL to define and format a new zFS aggregate to use for a USS file system resident version of the system catalogs. Although you can continue using sequential data set resident system catalogs, zFS file system resident catalogs are recommended.
USERHLQ.USERSAMP(CECCDCAT)	Generated JCL that allocates and initializes the metadata catalog for the source server.
USERHLQ.USERSAMP(CECCDSUB)	Generated JCL that allocates and initializes the VSAM data sets for the source server that store metadata for subscriptions and replication mappings.
USERHLQ.USERSAMP(CECPBKLQ)	Generated JCL that runs the CSQUTIL WebSphere MQ utility that defines the bookmark queue that stores restart information when changes are published in a delimited format to WebSphere MQ.
USERHLQ.USERSAMP(CECPSUBQ)	Generated JCL that runs the CSQUTIL WebSphere MQ utility to define a local queue that a subscription can reference when changes are published in a delimited format to WebSphere MQ.
USERHLQ.USERSAMP(CECCDSRC)	Generated JCL to start the source server.
USERHLQ.USERSAMP(CECCDVCD)	Generated JCL to validate the installation.

User samples allocation utility

The user samples allocation utility allocates the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets and populates the *USERHLQ*.USERSAMP data set with a copy of the customization parameters file and the installation customization utility.

The user samples allocation utility performs these functions:

- Allocates the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets. These data sets are created with the same characteristics as the distributed SCACSAMP and SCACCONF data sets.

  If you run the utility again, the *USERHLQ*.USERSAMP or *USERHLQ*.USERCONF data sets that already exist are reused. The utility replaces the customization parameters file and customization utility JCL members. All other members remain the same.

- Generates the customization parameters file *USERHLQ*.USERSAMP(CECCUSPC) and the installation customization utility JCL *USERHLQ*.USERSAMP(CECCUSC2). All input parameters specified for the samples allocation utility are populated in the generated CECCUSPC and CECCUSC2 members.

You use the SCACSAMP(CECCUSC1) job to run the allocation utility. The JCL contains comments with editing instructions. You specify the following input as parameters:

CACINHLQ=CEC.V11R3M00
The value specified for the CACINHLQ keyword must be the high-level qualifier of the installation data sets that the SMP/E installation produces.

CACUSHLQ=USER.V11R3M00.CDCSRC
The value specified for the CACUSHLQ keyword must be the high-level qualifier for the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets that the samples allocation utility creates or updates.

CACDUNIT=SYSALLDA
The value specified for the CACDUNIT keyword identifies the disk unit that is used when allocating the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets. This is an optional parameter.

CACDVOLM=
The value specified for the CACDVOLM keyword identifies the disk volume that is used when allocating the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets. This is an optional parameter.

CACSTGCL=
The value specified for the CACSTGCL keyword identifies the SMS storage class that is used when allocating the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets. This is an optional parameter.

CACMGTCL=
The value specified for the CACMGTCL keyword identifies the SMS management class that is used when allocating the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets. This is an optional parameter.

ISPFHLQ=ISP
The value specified for the ISPFHLQ keyword identifies the high-level qualifier for ISPF installation. The samples allocation utility runs a TSO batch application and uses TSO functions.

ISPFLANG=ENU
The value specified for the ISPFLANG keyword identifies the language prefix for the ISPF installation.

SERVERROLE=(*role-name*, ...)
The value of the SERVERROLE keyword specifies that the server environment being installed and customized contains the components required for a source server environment.

The samples allocation utility produces a summary report that is written to the SYSTSPRT DD specified in the JCL. The report lists the status for allocating the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets and lists the members updated in the *USERHLQ*.USERSAMP data set.

The following figure shows sample output written to SYSTSPRT.

```
                                  **************** Samples Allocation ****************
                                                    Summary Report


CACCUSX1 compiled on 2012-09-13 15:31:02 by REXX370 3.48
Execution timestamp: 2012-09-13 15:31:02  MVS Product ID: z/OS 01.10.00  SMF ID: ABC System ID: ABC
---------------------------------------------------------------------------------------------------


 Effective Parameters:

   CACDUNIT:    SYSALLDA
   CACDVOLM:
   CACINHLQ:    CEC.V11R3M00
   CACMGTCL:
   CACSTGCL:
   CACUSHLQ:    USER.V11R3M00.CDCSRC
   ISPFHLQ:     ISP
   ISPFLANG:    ENU
   SERVERROLE:  CDC_IMS_SRC

Member     Status
--------   --------
CECCUSC2   Processed successfully in DDN(USERSAMP)
CECCUSPC   Processed successfully in DDN(USERSAMP)


Summary:
   Members Successful:       2
   Members in Error:         0
   Members Not Replaced:     0
   Members Processed:        2


Return Status: 0
```

Figure 1. Sample output for the samples allocation summary report

Installation customization utility

The installation customization utility generates the JCL and configuration members needed in the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets based on the values that you provide in the customization parameters file.

The installation customization utility performs these functions:

- Captures the customization settings that you provide in the customization parameters file *USERHLQ*.USERSAMP(CECCUSPC).
- Applies the customization parameters to all JCL members associated with the specified SERVERROLE parameter and places the customized members in the *USERHLQ*.USERSAMP data set.
- Applies the customization parameters to all configuration members associated with the specified SERVERROLE parameter and places the customized members in the *USERHLQ*.USERCONF data set.

You use the *USERHLQ*.USERSAMP(CECCUSC2) job to run the installation customization utility. You specify the following input as parameters:

CACINHLQ=CEC.V11R3M00
: The value specified for the CACINHLQ keyword must be the high-level qualifier for Classic distribution data sets produced by the SMP/E

installation. This value is automatically populated with the value previously specified as input to the user samples allocation utility.

CACUSHLQ=USER.V11R3M00.CDCSRC
The value specified for the CACUSHLQ keyword must be the high-level qualifier for the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets that were created or updated by the user samples allocation utility. This value is automatically populated with the value previously specified as input to the user samples allocation utility.

MEMBER=(*member-name***, ...)**
This is an optional parameter. The value specified for the MEMBER keyword identifies a list of one or more member names to process. Only a subset of the members associated with the specified SERVERROLE parameter is processed. If you specify multiple member names, you must separate the names with commas and enclose the names in parentheses.

All members are processed when this parameter is not specified.

OVERWRITE=YES|NO
The value specified for the OVERWRITE keyword indicates how to process existing members of target data sets, for example the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets.

- When you specify OVERWRITE=NO, existing members of the target data sets are not replaced. OVERWRITE=NO is the default.
- When you specify OVERWRITE=YES, existing members of the target data sets are replaced.

Example: OVERWRITE=NO is in effect. Members CACCFGDS and CACCFGUT already exist in the target data set. Member CACSX04 does not exist in the target data set.

```
Member   Status
-------- --------
CACCFGDS CACCFGDS not replaced in DDN(USERSAMP)
CACCFGUT CACCFGUT not replaced in DDN(USERSAMP)
CACSX04  Processed successfully in DDN(USERSAMP)
```

Example: OVERWRITE=YES is in effect. The processing status for the same members shown in the previous example appear as follows (whether or not any of these members previously existed in the target data set):

```
Member   Status
-------- --------
CACCFGDS Processed successfully in DDN(USERSAMP)
CACCFGUT Processed successfully in DDN(USERSAMP)
CACSX04  Processed successfully in DDN(USERSAMP)
```

The processing summary information produced at the bottom of the report identifies the number of members that were stored successfully and the number of members that were not replaced. For example:

```
Summary:
   Members Successful:      90
   Members in Error:         0
   Members Not Replaced:     0
   Members Processed:       90
```

SERVERROLE=(*role-name***, ...)**
The value of the SERVERROLE keyword specifies that the server environment being installed and customized contains the components required for a source server environment.

The utility produces a summary report that is written to the SYSTSPRT DD that is specified in the JCL. The report lists the partitioned data sets and the data set members that were processed. The final summary lists the total number of members processed, the number that were successful, and the number with errors.

The following figure shows sample output written to SYSTSPRT.

```
********************     Installation Customization    ***********************
                             Summary Report

CACCUSX2 compiled on 2012-08-15 08:46:51 by REXXC370 3.48
Execution timestamp: 2012-08-15 08:49:39  MVS Product ID: z/OS 01.10.00 SMF ID: SYE9  System ID:
------------------------------------------------------------------------------------------------

Effective Parameters:

  CACINHLQ:    CEC.V11R3M00
  CACUSHLQ:    USER.V11R3M00.CDCSRC
  OVERWRITE:   No
  SERVERROLE:  CDC_IMS_SRC


Processing parameters file: USER.V11R3M00.CDCSRC.USERSAMP  Member: CECCUSPC

Processing Members for Product: All  Role: Common
Member     Status
--------   --------
CACCFGDS   Processed successfully in DDN(USERSAMP)
CACCFGUT   Processed successfully in DDN(USERSAMP)
CACPRTLS   Processed successfully in DDN(USERSAMP)
CACLGFLT   Processed successfully in DDN(USERSAMP)
CACSX04    Processed successfully in DDN(USERSAMP)

Processing Members for Product: Classic Change Data Capture for z/OS Role: Common
Member     Status
--------   --------
CECCDCFG   Processed successfully in DDN(USERSAMP)
CECCDCAT   Processed successfully in DDN(USERSAMP)
CECCDSLS   Processed successfully in DDN(USERSAMP)
CECCDSUB   Processed successfully in DDN(USERSAMP)
CECCDSRC   Processed successfully in DDN(USERSAMP)
CECCDVCD   Processed successfully in DDN(USERSAMP)
. . .
. . .


Processing Members for Product: Classic Change Data Capture for z/OS Role: CDC_IMS_SRC
Member     Status
--------   --------


Summary:
   Members Successful:       24
   Members in Error:          0
   Members Not Replaced:      0
   Members Processed:        24

Return Status: 0
```

Figure 2. Sample output for the installation customization summary report.

Working with the customization parameters file

These guidelines describe how to enter values in the customization parameters file.

The customization parameters file, *USERHLQ*.USERSAMP(CECCUSPC), contains pairs of keyword and value settings used to customize JCL and configuration files in the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets.

The following sections provide guidelines for entering input into the customization parameters file, describe how the file is organized, and list the keyword and value settings that the customization parameters file contains. Other considerations include the use of job cards, pre-defined variables, and STEPLIB concatenations.

Input guidelines

The following guidelines describe how to enter values in the customization parameters file:

- Keyword and value pairs:
  - You cannot change the keyword component.
  - You must delimit the value component with double quotes ("").
  - Spaces are allowed before and after the keyword and value.
  - Values cannot span multiple lines.
- The minimum required parameters that you must change for a successful installation are denoted by an asterisk within parentheses at the end of the comment for that parameter. For example: CACINHLQ="&CACINHLQ" HLQ of Classic product(*)
- Comments:
  - An asterisk (*) in column 1 defines the line as a comment line.
  - Any input that you include after the first space after the value component is treated as comments.

File organization

The following table describes the organization of the customization parameters file.

Table 6. Organization of the customization parameters file.

Section name	Section content
Common installation section	Parameters that apply to all installations, such as the high-level qualifier for the Classic product installation
Data server files parameters	Parameters associated with the metadata catalog and configuration files needed for the data server.
Source server parameters	Parameters that apply to the source server
Source catalog migration parameters	Migration parameters that control catalog migration for the source server.
Source security parameters	Security parameters that control user connections for the source server
Source log stream parameters	Log stream parameters for the source server
Source IMS parameters	IMS-specific parameters for the source server
Source WebSphere MQ delimited publishing parameters	Publishing parameters for delimited format subscriptions for the source server
Source WebSphere MQ delimited publishing parameters	Publishing parameters for delimited format subscriptions for the source server

Use of job cards

Job card information is defined in the common installation section of the customization parameters file. The following two-line job card information is used as a template when generating JCL members:

```
CACDJOB1="JOB (CLASSIC),'CLASSIC JOB',CLASS=A,"
CACDJOB2="MSGCLASS=X,NOTIFY=&SYSUID"
```

The CACDJOB1 value is placed after the job name in each generated JCL member. The CACDJOB2 value is provided on the second line of the job card in each JCL member.

The initial value for the job card keywords is populated from the job card that is specified on the JCL member.

Use of pre-defined variables

Many of the data set values in the customization parameters file contain pre-defined variables such as &CEC to reference previously defined high-level qualifiers. Most of the generated JCL members make use of inline PROC definitions. These variables reference the actual PROC variables. The following table describes what each variable defines:

Table 7. Pre-defined variables.

Variable	**Description**
&CEC	Classic product installation high-level qualifier
&USERHLQ	User SCACSAMP high-level qualifier
&IMS	IMS installation high-level qualifier

Library concatenations

For Classic data server parameters that require specific DD data set concatenation customization such as STEPLIB, parameters are provided for concatenation. You can specify the same parameter keyword multiple times. The order specified for the parameter keywords is the order in which the data sets will be included in the data set concatenation.

Customization parameters file: Common section

The common section of the customization parameters file contains parameters that are common to all installations.

The following table lists the parameters in the common section of the customization parameters file, the parameter default values, and a description of each parameter.

Table 8. Common parameter and default settings for USERSAMP(CECCUSPC)

Parameters	**Default value**	**Description**
Common section		
CACINHLQ	CEC.V11R3M00	High-level qualifier of the installation data sets for the Classic product. This value is populated with the value specified for the CACINHLQ input parameter of the CECCUSC1 job.

Table 8. Common parameter and default settings for USERSAMP(CECCUSPC) (continued)

Parameters	Default value	Description
CACUSHLQ	USER.V11R3M00	High-level qualifier for the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets. This value is populated with the value specified for the CACUSHLQ input parameter of the CECCUSC1 job.
CACDUNIT	SYSALLDA	Disk unit that is used for the generated jobs that create data sets such as the configuration, subscription, and replication mapping files. This value is populated with the value specified for the CACDUNIT input parameter of the CECCUSC1 job. If the value is "", it is assumed that the site SMS rules will determine the data set allocation.
CACDVOLM	""	Disk volume that is used for the generated jobs that create data sets such as the configuration, subscription, and replication mapping files. This value is populated with the value specified for the CACDVOLM input parameter of the CECCUSC1 job. If the value is "", it is assumed that the site SMS rules will determine the data set allocation.
CACSTGCL	""	SMS storage class that is used for the generated jobs that create data sets such as the configuration, subscription, and replication mapping files. This value is populated with the value specified for the CACSTGCL input parameter of the CECCUSC1 job. If the value is "", it is assumed that the site SMS rules will determine the data set allocation.
CACMGTCL	""	SMS management class that is used for the generated jobs that create data sets such as the configuration, subscription, and replication mapping files. This value is populated with the value specified for the CACMGTCL input parameter of the CECCUSC1 job. If the value is "", it is assumed that the site SMS rules will determine the data set allocation.

Customization parameters file: Source server section

The source server section of the customization parameters file contains parameters that are specific to a source server.

The following table lists the parameters in the source server section of the customization parameters file, the parameter default values, and a description of each parameter.

Table 9. Source server parameters and default settings for USERSAMP(CECCUSPC)

Parameters	Default value	Description
Common server parameters for change data capture		
CDCADMUS	CACUSER	User ID to which to SYSADM privileges are granted in the metadata catalog.
CDCPHLQD	USER.V11R3M00.CDCSRC	High-level qualifier (HLQ) of data sets created for the source server: • Binary configuration data sets • Metadata catalog • Subscription data set • Replication mapping data set This value replaces the references to &CPHLQ.. in other keyword values such as CPCPCFGD, CDCCPCFGX, and CDCPSUBS.
CATLGMB	150MB	Size of the primary catalog allocation.

Table 9. Source server parameters and default settings for USERSAMP(CECCUSPC) (continued)

Parameters	Default value	Description
CATPATH	/opt/IBM/isclassic113/ catalog	USS file system path to the metadata catalog files. The named directory contains the following file names: **caccat** Data component **cacindx** Index component zFS file system resident catalogs are recommended (rather than other file systems, such as physical sequential). When this parameter is specified, it supersedes CDCCATNM and CDCIDXNM.
CDCCATNM	&CPHLQ..CATALOG	Name of the metadata catalog data file for the source server. The name is prefixed with the CDCPHLQD high-level qualifier specified above. When the CATPATH parameter is specified, it supersedes CDCCATNM.
CDCIDXNM	&CPHLQ..CATINDX	Name of the metadata catalog index file for the source server. The name is prefixed with the CDCPHLQD high-level qualifier specified above. When the CATPATH parameter is specified, it supersedes CDCIDXNM.
CDCPCFGD	&CPHLQ..CACCFGD	Name of the configuration data file for the source server. This value is prefixed with the CDCPHLQD high-level qualifier.
CDCPCFGX	&CPHLQ..CACCFGX	Name of the configuration index file for the source server. This value is prefixed with the CDCPHLQD high-level qualifier.
CDCPHOST	0.0.0.0	Host name or IP address where the source server will run. This value is used in the definition of the COMMSTRING configuration parameter of the connection handler for the source server.
CDCPPORT	9087	Port number that the source server connection handler service listens on. This listen port communicates with the Classic Data Architect to monitor and manage the source server. This value is used in the definition of the COMMSTRING configuration parameter of the connection handler for the source server.
CDCDSRCE	SAMPLEDS	Data source name for the query processor service.
CDCPSUBS	&CPHLQ..SUB	Data file name for source server subscriptions. This value is prefixed with the CDCPHLQD high-level qualifier.
CDCSCHMC	0	Schema change behavior for the source server: • 0: Stop replication • 1: Park replication mapping
CDCTIMEZ	""	Name of the IANA Time Zone Database (also referred to as the tz database).
HOSTCDPG	""	Host code page that the site uses.
Migration and upgrade parameters: The parameters in this section apply to installations that you need to migrate from a version 10.1 or later release of Classic CDC to the current version. If a migration is not required, use the default values.		
OLDCAHLQ	CAC.V11R1M00	High-level qualifier for the previous product installation.
OLDCTHLQ	CAC.V11R1M00	High-level qualifier for the previous product level.
OLDCATNM	&OLDCAT..CATALOG	Suffix for the metadata catalog data file at the previous product level. The &OLDCAT value will be replaced in the generated JCL PROC with the high-level qualifier specified for the OLDCTHLQ parameter.
OLDCATIX	&OLDCAT..CATINDX	Suffix for the metadata catalog index file at the previous product level. The &OLDCAT value will be replaced in the generated JCL PROC with the high-level qualifier specified for the OLDCTHLQ parameter.

Table 9. Source server parameters and default settings for USERSAMP(CECCUSPC) (continued)

Parameters	Default value	Description
OLDCPATH	"" (null)	If upgrading an existing zFS catalog, use this parameter. When defined, this parameter supersedes OLDCATNM and OLDIDXNM. Specifies the USS file system path to the metadata catalog files. (for example, /opt/IBM/isclassic113/catalog). The named directory contains the following file names: **caccat** Data component **cacindx** Index component
NEWCTHLQ	&CACUSHLQ	High-level qualifier of new catalog.
NEWCATNM	&&NEWCAT..CATALOG	New data set for catalog data.
NEWCATIX	&&NEWCAT..CATINDX	New catalog data set for catalog index.
OLDCFHLQ	CAC.V11R1M00	High-level qualifier of pre-version 11.3 configuration.
OLDCFGNV	&&OLDCFG..SCACCONF	Pre-version 11.3 configuration PDS.
OLDCFGNM	&&OLDCFG..CACCFGD	Previous data set for configuration data.
OLDCFGIX	&&OLDCFG..CACCFGX	Previous data set for configuration index.
NEWCFGNM	&&NEWCFG..CACCFGD	New data set for configuration data.
NEWCFGIX	&&NEWCFG..CACCFGX	New data set for configuration index.
Security parameters for the source server		
CDCPSQLS	Y	Indicates if the operator service commands will be secured by using security definitions in the metadata catalog.
CDCPSAFX	CACSX04	SAFEXIT load module that enables security for the source server. A value of "" disables security for the source server. This value defines the SAFEXIT configuration parameter of the operator, administration, and monitor services for the source server. For more information, see Securing a Classic data server.
For more information about the following parameter settings for the SAFEXIT, see Security.		
CDCADMVL	N	SAF administration service validation (Y/N). This value instructs the administration service to use SAF to validate the access authority of a user. The SAF resource profile is specified through the CDCADMPR parameter and the SAF security class is specified through the CDCADMCL parameter. This parameter only applies when the CDCPSAFX parameter (the SAF exit name) is not null.
CDCADMPR	""	Resource profile name for the SAF administration service. This parameter only applies when the CDCADMVL parameter is Y.
CDCADMCL	""	Security class name for the SAF administration service. This parameter only applies when the CDCADMVL parameter is Y.

Table 9. Source server parameters and default settings for USERSAMP(CECCUSPC) (continued)

Parameters	Default value	Description
CDCOPRVL	N	SAF operator service validation (Y/N). This value instructs the operator service to use SAF to validate the access authority of a user. The SAF resource profile is specified through the CDCOPRPR parameter and the SAF security class is specified through the CDCOPRCL parameter. This parameter only applies when the CDCPSAFX parameter (the SAF exit name) is not null.
CDCOPRPR	""	Resource profile name for the SAF operator service. This parameter only applies when the CDCOPRVL parameter is Y.
CDCOPRCL	""	Security class name for the SAF operator service. This parameter only applies when the CDCOPRVL parameter is Y.
CDCMONVL	N	SAF monitor service validation (Y/N). This value instructs the monitor service to use SAF to validate the access authority of a user. The SAF resource profile is specified through the CDCMONPR parameter and the SAF security class is specified through the CDCMONCL parameter. This parameter only applies when the CDCPSAFX parameter (the SAF exit name) is not null.
CDCMONPR	""	Resource profile name for the SAF monitor service. This parameter only applies when the CDCMONVL parameter is Y.
CDCMONCL	""	Security class name for the SAF monitor service. This parameter only applies when the CDCMONVL parameter is Y.
CDCQPSVL	N	SAF query processor service validation (Y/N). This parameter only applies when the CDCPSAFX parameter (the SAF exit name) is not null.
CDCQPIMC	""	IMS PSB schedule class parameter on the query processor service for the SAFEXIT, when the exit is enabled.
CDCQPIMP	""	IMS PSB prefix parameter on the query processor service for the SAFEXIT, when the exit is enabled.
Log stream parameters for the source server		
CDCPLGST	CDCSRC.DIAGLOG	z/OS log stream name for the source server diagnostic log. If a log stream name is not specified, the log service is configured to write to the CACLOG DD data set. This value defines the STREAMNAME configuration parameter of the log service for the source server.
CDCPLGDS	Y	Identifies whether the z/OS log stream should use DASD or the coupling facility: • Y: DASD • N: Coupling facility This value is valid when CDCPLGST is specified.
CDCPLGRT	7	Retention period, in days, to retain the log records before they for they are eligible to be deleted. This value is valid when CDCPLGST is specified.

Table 9. Source server parameters and default settings for USERSAMP(CECCUSPC) (continued)

Parameters	Default value	Description
CDCPLGSC	STG1	Storage class (STG_DATACLAS) for the log stream. This value is valid when CDCPLGST is specified.
CDCPLGSR	CCL1	Coupling facility structure name (STRUCTNAME). This value is valid when CDCPLGST is specified and the coupling facility is chosen (LGSTRDASD="N").
CDCPEVST	CDCSRC.EVENTS	z/OS log stream for the Classic event log. This value defines the EVENTLOG configuration parameter of the log service for the source server.
CDCPEVDS	Y	Identifies whether the z/OS log stream should use DASD or the coupling facility: • Y: DASD • N: Coupling facility This value is valid when CDCPEVST is specified.
CDCPEGRT	14	Retention period, in days, to retain the log records before they for they are eligible to be deleted. This value is valid when CDCPEVST is specified.
CDCPEVSC	STG1	Storage class (STG_DATACLAS) for the log stream. This value is valid when CDCPEVST is specified.
CDCPEVSR	CCP1	Name of the coupling facility structure (STRUCTNAME). This value is valid when CDCPEVST is specified and the coupling facility is chosen (LGSTRDASD="N").
IMS source parameters for Classic change data capture		
IMSINHLQ	IMS	High-level qualifier (HLQ) assigned to the IMS libraries that the source server uses to access IMS. This HLQ is applied to the IMS data sets in the STEPLIB sections of the generated JCL. This value replaces the references to &IMS.. in other keyword values such as IMSSTEPL.
IMSSTEPL	&IMS..SDFSRESL	STEPLIB concatenation. You can specify this keyword multiple times. The order of the multiple IMSSTEPL keywords defines the order in which the files are included in the STEPLIB concatenation for the generated JCL members.
IMSDBDLB	&IMS..DBDLIB	DBDLIB concatenation. You can specify this keyword multiple times. The order of the multiple IMSDBDLB keywords defines the order in which the files are included in the DBDLIB concatenation for the generated JCL members.
IMCPNPRT	5003	Port that the IMS log reader service listens on to receive notifications from the IMS environment.
IMCPEXCL	""	IMS log reader exclusion list. If specified, this list includes the IMS SSIDs separated by commas. For example: "SSID1, SSID2". For more information, see SSIDEXCLUDELIST.
IMSDRAUS	DRAUSER	User ID that the DRA initialization services uses to access IMS.
IMSDRASX	00	Suffix for customizing the IMS PZP module.
VSAM source parameters for Classic change data capture		
Parameters for publishing to WebSphere MQ		

Table 9. Source server parameters and default settings for USERSAMP(CECCUSPC) (continued)

Parameters	Default value	Description
CDCMQHLQ	None	High-level qualifier for the WebSphere MQ installation. Only supply a value if you plan to publish changes to WebSphere MQ on z/OS or if the WebSphere MQ load library and message library are loaded in the system link pack area or included in the link list.
CDCMQMGR	None	Identifies the name of the z/OS queue manager or queue sharing group that the capture service connects to when InfoSphere Classic CDC for z/OS publishes changes in a delimited format to WebSphere MQ.
CDCBKMKQ	None	Identifies the name of the bookmark queue that stores restart information when InfoSphere Classic CDC for z/OS publishes changes in a delimited format to WebSphere MQ. The bookmark queue must be a local queue definition that you can create by running the *USERHLQ*.USERSAMP(CECPBKLQ) JCL.
CDCSUBLQ	None	Tailors a local queue definition in the *USERHLQ*.USERSAMP(CECPSUBQ) JCL. You can use this sample as a model for the local queue definitions that can be referenced by a subscription that publishes delimited messages to WebSphere MQ. Only supply a value if you plan to use local queues when capturing changes and publishing them to WebSphere MQ in a delimited message format.

Installing Classic data servers

After you set up the installation environment, follow the customization steps for installing source servers for change data capture to customize and validate your environment.

Installing Classic CDC source servers

You can follow the installation customization process to install and customize a source server for Classic change data capture.

Before you begin

Before you begin the installation customization process, you must complete the SMP/E installation and the steps required to prepare the installation environment.

About this task

When setting up the metadata catalog, the default configuration creates and initializes the metadata catalog as a zSeries File System (zFS) file. The z/FS file provides significant performance and capacity improvements compared to the sequential or linear data set formats used in releases prior to V11.1.

The topic *Creating and initializing zFS metadata catalogs* provides more information about the use of zFS metadata catalogs.

Procedure

1. Edit the user samples allocation utility JCL in the installation samples member SCACSAMP(CECCUSC1). Follow the instructions in the JCL to edit the job card and procedure variables and to specify the following input parameters:

   CACINHLQ=CAC.V11R3M00
   The value specified for the CACINHLQ keyword must match the high-level qualifier of the installation data sets that the SMP/E installation produces for Classic change data capture.

   CACUSHLQ=USER.V11R3M00.CDCSRC
   The value specified for the CACUSHLQ keyword is the high-level qualifier for the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets that the samples allocation utility creates or updates.

   CACDUNIT=SYSALLDA
   The value specified for the CACDUNIT keyword identifies the disk unit that is used when allocating the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets. This is an optional parameter. You do not need to specify a value for CACDUNIT if SMS manages the data sets.

   CACDVOLM=
   The value specified for the CACDVOLM keyword identifies the disk volume that is used when allocating the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets. This is an optional parameter. You do not need to specify a value for CACDVOLM if SMS manages the data sets.

   CACSTGCL=
   The value specified for the CACSTGCL keyword identifies the SMS storage class that is used when allocating the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets. This is an optional parameter. Specify a value for CACSTGCL only when a specific storage class is required.

   CACMGTCL=
   The value specified for the CACMGTCL keyword identifies the SMS management class that is used when allocating the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets. This is an optional parameter. Specify a value for CACMGTCL only when a specific management class is required.

   ISPFHLQ=ISP
   The value specified for the ISPFHLQ keyword identifies the high-level qualifier for the ISPF installation. The samples allocation utility runs a TSO batch application and uses TSO functions.

   ISPFLANG=ENU
   The value specified for the ISPFLANG keyword identifies the language prefix for the ISPF installation.

   SERVERROLE=*role-name*
   The value specified for the SERVERROLE keyword identifies the server environment to install and customize. Specify one of the following values to generate the JCL members and components required for a source server environment for Classic change data capture.

   - To install components for IMS access, specify SERVERROLE=CDC_IMS_SRC.

- To install components for VSAM access, specify SERVERROLE=CDC_VSAM_SRC.

2. Submit SCACSAMP(CECCUSC1) to allocate the *USERHLQ*.USERSAMP and *USERHLQ*.USERCONF data sets. Verify that all job steps result in a return code <= 4.

   This job populates the *USERHLQ*.USERSAMP data set with the necessary objects and the customization parameters file for Classic change data capture, CECCUSPC.
3. Edit the Classic change data capture customization parameters file *USERHLQ*.USERSAMP(CECCUSPC) to provide customization parameters.

   This file will contain only the parameters that apply to a source server environment. See the customization parameters file settings for details.
4. Submit the generated *USERHLQ*.USERSAMP(CECCUSC2) customization utility JCL. Verify that all job steps result in a return code <= 4.

   The *USERHLQ*.USERSAMP data set is populated with the customized JCL and objects needed to run a source server.
5. Optional: Define the z/OS log stream for the diagnostic log for the data server. This step is only needed when the DIAGLGST, EVENLGST, or CDCRLGST parameter in the CECCUSPC files specifies a log stream name.
   a. Verify that you have the authority required to run the Administrative Data Utility (IXCMIAPU). The job that defines the logs runs this utility.
   b. Submit the generated *USERHLQ*.USERSAMP(CECCDSLS) JCL to define the log streams. A log streams is defined for the diagnostic log and the event log. For VSAM data sources, a log stream is defined for the IVP replication log.
   c. Verify that all job steps result in a return code = 0.
6. Optional: If the CATPATH value specified that the metadata catalog files will reside in a new zFS file system aggregate, you must define and format the new zFS aggregate.
   a. Verify that you have the authority required to run the following jobs:
      - SAF READ-access level authorization is required for the SUPERUSER.FILESYS.PFSCTL resource in the UNIXPRIV class to run the zFS administration command, IOEZADM.
      - SAF READ-access level authorization is required for the SUPERUSER.FILESYS.MOUNT resource in the UNIXPRIV class to perform MOUNT and UNMOUNT operations against USS file systems.
   b. View and edit *USERHLQ*.USERSAMP(CECCRZCT) job cards.
   c. Submit *USERHLQ*.USERSAMP(CECCRZCT).
   d. Verify that all job steps result in a return code = 0.
   e. Mount the file systems using the sample MOUNT command provided in *USERHLQ*.USERSAMP(CECCRZCT).
7. Submit the generated JCL to create a new source server configuration or to migrate a source server configuration.
   - If you are creating a new configuration data set for the source server, submit the generated *USERHLQ*.USERSAMP(CECCDCFG) JCL to allocate and initialize the source server configuration. This job populates the configuration with the service definitions needed for a source server environment.
   - If you are migrating the source server from V10R1M00 or later to V11R3M00, submit the generated *USERHLQ*.USERSAMP(CECCDMCF) JCL

to allocate new configuration data sets for the source server and populate the new source server configuration based on the previous source server configuration.

Verify that all job steps result in a return code = 0.

8. Submit the generated *USERHLQ*.USERSAMP(CECCDCAT) JCL to allocate and initialize the metadata catalog for the source server.
9. Submit the generated JCL to create new subscription and replication mapping data sets or to migrate subscription and replication mapping data sets.
   - If you are creating new subscription data sets, submit the generated *USERHLQ*.USERSAMP(CECCDSUB) JCL to allocate the subscription and replication mapping data sets for the source server.
   - If you are migrating the source server from V10R1M00 or later to V11R3M00, submit the generated *USERHLQ*.USERSAMP(CECCDSUB) JCL to allocate the subscription and replication mapping data sets for the source server. Then submit the generated *USERHLQ*.USERSAMP(CECCDMSU) JCL to copy the subscription and replication mapping data sets from the previous version of the data sets to the V11R3M00 data sets before starting the source server.
10. Optional: Submit the generated USERHLQ.USERSAMP(CECPBKLQ) member to create the bookmark queue. This step is only needed when:
    - You plan to publish changes to WebSphere MQ in delimited format
    - You supplied values for the CDCMQMGR and the CDCBKMKQ parameters in the customization parameters file
11. Submit the generated *USERHLQ*.USERSAMP(CECCDSRC) JCL to start the source server. When the Classic data server starts, the following services are customized and running in the source server based on the SERVERROLE specified.

Table 10. Services customized for IMS and VSAM data sources

For IMS data sources (SERVERROLE=CDC_IMS_SRC)	For VSAM data sources (SERVERROLE=CDC_VSAM_SRC)
• Administration service • Capture service • Connection handler service • Database resource adapter (DRA) initialization service The DRA service is required when you access IMS data by using the Classic Data Architect (CDA) for table and view validation and for subscription refresh processing. • IMS log reader service • Logger service • Operator service • Query processor service • Monitor service • Refresh query processor service If you define subscriptions that publish delimited changes to WebSphere MQ only, the refresh query processor service is not used. • Region controller service	• Administration service • Capture service • Connection handler service • Logger service • Operator service • Query processor service • Monitor service • Refresh query processor service If you define subscriptions that publish delimited changes to WebSphere MQ only, the refresh query processor service is not used. • Region controller service • VSAM access service • VSAM log reader service

12. Run the validation job in the next step if security is enabled for the source server for change data capture. If security is not enabled, continue with the next step.

    Security is enabled for the source server by setting the CDCPSAFX parameter in the installation customization parameters file. If security is enabled, the following steps are required before you run the validation job in the final step.

    Enabling security requires providing a password for all user access. The utilities used in the validation job require encrypted passwords to access the Classic data server. Set up the encrypted password for the validation job by following these steps:

    a. Edit the generated *USERHLQ*.USERCONF(CACPWDIN) member. This member provides a PASSWORD=value parameter for the password generator utility. Set the value to the TSO password for the User ID that you use to run the customization jobs.
    b. Submit the generated *USERHLQ*.USERSAMP(CACENCRP) JCL to run the password generator utility. This JCL updates the *USERHLQ*.USERCONF(CACPWDIN) with the encrypted value of the password provided in the previous step.
    c. Edit the *USERHLQ*.USERCONF(CACPWDIN) member and copy the hex string value (x'<16-byte hexadecimal value>') for the ENCRYPTED= keyword.
    d. Edit the generated *USERHLQ*.USERCONF(CACMUCON) member and replace the X'.ENCRYP PASSWD..' string with the hex string copied in the previous step.
    e. Edit the generated *USERHLQ*.USERSAMP(CACQRSYS) member and replace the second line of the member with the hex string that you copied to ensure that the hex string starts in the first column

13. Submit the generated *USERHLQ*.USERSAMP(CECCDVCD) validation job JCL. Verify that all job steps result in a return code <= 4. This job ensures that the metadata catalog is set up properly and that communication to the source server is successful.

    For VSAM data sources, if you plan to use *USERHLQ*.USERSAMP(CECCDVCD) to validate replication you must provide a value for CDCRLGST. Specifying CDCRLGST controls whether the EMPLOYEE VSAM file is defined for change capture.

Results

The source server is now operational. When the corresponding target engine is running, you can configure replication by using the management console to connect the source server and target engine.

If you plan to define delimited subscriptions that publish to WebSphere MQ, the source server is operational and a target server is not required.

Chapter 5. Installing the Classic Data Architect

To install the Classic Data Architect you extract an installation zip package and run IBM Installation Manager.

Before you begin

If you have installed an earlier beta version of the Classic Data Architect (CDA) Version 11.3, you must uninstall it first. See "Uninstalling the Classic Data Architect" on page 42.

Ensure that your client computer meets the following minimum system requirements:

Operating system:
- Microsoft Windows 8.1, 8, and 7

Memory
1024 MB.

Disk space
800 MB for both IBM Installation Manager and the Classic Data Architect.

Procedure

Start the CDA installation as a non-Administrator or as an Administrator.

Method	Description
To start the CDA installation as a non-Administrator	1. Unzip the installation package to a temporary directory. 2. From the command line, change to the temporary directory and take one of the following actions: • On Windows: Run `userinst.exe`. • On Linux: Run `userinst`. 3. Proceed through the Installation Manager wizard.
To start the CDA installation as an Administrator	1. Unzip the installation package to a temporary directory. 2. From the command line, change to the temporary directory and take one of the following actions: • On Windows: Run `install.exe`. • On Linux: Run `install`. 3. Proceed through the Installation Manager wizard.

Results

When the `userinst` or `install` program starts, Installation Manager is installed if it is not already on your computer and automatically started. Installation Manager is configured with the location of the repository (installation files) for IBM InfoSphere Classic Data Architect V11.3.

What to do next

You can launch the product:

- From Windows: **Start** > **Programs** > **IBM Classic Data Architect** > **IBM InfoSphere Classic Data Architect V11.3**. Alternatively, from a command line run `<installRoot>/eclipse.exe`.
- From Linux: **Applications** > **IBM Classic Data Architect** > **IBM InfoSphere Classic Data Architect V11.3**. Alternatively, from a command line run `<installRoot>/eclipse`.

Starting IBM Installation Manager

If you start the Classic Data Architect installation from the downloadable image, IBM Installation Manager starts automatically.

About this task

When you start the installation of Classic Data Architect from the downloadable image, either by running the install or userinst program, IBM Installation Manager automatically starts. If Installation Manager is not installed, it will be installed and automatically started.

If you already installed Installation Manager, you can start it by using one of the methods in the following procedure.

Procedure

Start the Installation Manager from Windows or Linux.

Method	Description
To start the Installation Manager from Windows	Click **Start** > **All Programs** > **IBM Installation Manager** > **IBM Installation Manager**.
To start the Installation Manager from Linux	Click **Applications** > **IBM Installation Manager** > **IBM Installation Manager** or alternatively change to *`Installation Manager directory`* `/eclipse` and run `IBMIM`.

Uninstalling the Classic Data Architect

You can uninstall the Classic Data Architect by using the IBM Installation Manager.

Before you begin

To uninstall the IBM InfoSphere Classic Data Architect product package, you must log in to the system by using the same user account that you used to install the product package. You must close the programs that you installed by using IBM Installation Manager.

About this task

You can use the **Uninstall** option in IBM Installation Manager to uninstall the IBM InfoSphere Classic Data Architect product package from a single installation location. You can also uninstall all of the installed packages from every installation location.

To uninstall the CDA product package, complete the steps in the following procedure.

Procedure

1. Start IBM Installation Manager.
2. On the Start page, click the **Uninstall** button.
3. On the Uninstall Packages page, from the **Installation Packages** list, select IBM InfoSphere Classic Data Architect version 11.3.0, and click **Next**.
4. On the Summary page, review the list of packages that will be uninstalled. The Complete page displays after the packages are removed.
5. Click **Finish**.

Chapter 6. Installing InfoSphere CDC

InfoSphere CDC has a variety of target engines for use with InfoSphere Classic CDC for z/OS, each having their own unique installation and configuration information.

See the end user documentation for your chosen target replication engine for instructions on planning and implementing your target engine.

Chapter 7. Installing Access Server and Management Console

This section outlines how to install Access Server and Management Console.

Installing Access Server

Access Server is a background process on a Windows or UNIX workstation that implements the InfoSphere CDC security model

This section outlines how to install Access Server.

See also:

To install Access Server (Windows)

Procedure

1. Double-click on the installation file.

   The IBM InfoSphere Data Replication Access Server installation wizard opens.
2. Click **Next**.
3. Read the license agreement and select **I accept the terms in the license agreement** and then click **Next**.
4. Select the folder where you want to install Access Server and click **Next**.
5. Select the location for the product icons and click **Next**.
6. Specify the port number to communicate with Management Console and click **Next**.
7. Enter the username and password for an Access Server account that will be used to perform the following actions:
   - Log in to Access Server from Management Console.
   - Manage users and datastores in Management Console.
8. Review the installation summary and click **Install**.
9. Click **Done** to exit the installation.

To install Access Server (UNIX and Linux)

Procedure

1. Log on to your UNIX or Linux machine with the account you are using for the Access Server installation.
2. Copy the installation file for your platform from the CD-ROM or download it from the IBM Web site.
3. Make the installation program executable.
4. Start the installation with the following command:
   ```
   ./<installation_binary_name>
   ```
   where `<installation_binary_name>` is the name of the installation file.
5. Press **Enter** on the **Introduction** screen to display the license agreement. Follow the instructions on the screen to navigate through the license agreement.
6. Enter 1 to accept the license agreement.
7. Enter the absolute path to your installation directory or press **Enter** to accept the default directory.

The directory that you specify must be owned by the account you are using for the installation. If the installation program cannot create the directory, you are prompted to specify a different directory.

8. Review the installation summary and press **Enter** to start the installation.
9. Press **Enter** to exit the installation after the installation is complete.

   After you have installed Access Server, you must complete both of the following tasks in the order they are listed before you can log in to Management Console for the first time:
   - Start Access Server.
   - Create an Access Server user account.

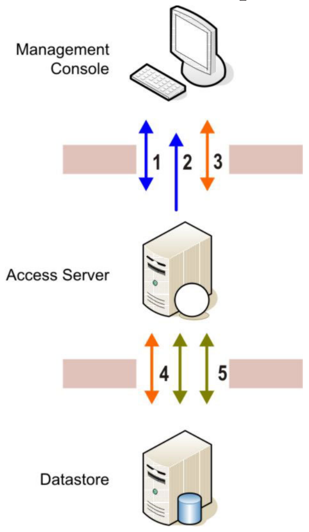
Configuring firewall settings for outbound (static) ports

If your network uses a firewall or other security mechanism that requires static ports for communication, then you must specify the ports that other computers can use to communicate with Access Server services.

Note: In addition to a network firewall, you might have personal firewall software installed and enabled on client machines. This firewall may cause a problem when connecting to Management Console from Access Server.

To calculate the number of Access Server ports to open, use this formula: `number of ports to open = 2 * (number of users + (number of users * number of datastores) + number of datastores)` where a datastore refers to an InfoSphere CDC instance.

The following figure highlights the ports you can configure for Management Console and Access Server components. You can configure static port numbers for all or some of these ports, depending on your network requirements.



The labels in the figure correspond to the following groups of ports:

- 1—Communication from Management Console to the Access Server service. You specify this port when you install Access Server and when you log in to Management Console. The default port is 10101 and you can set this value in Management Console.
- 2—Communication from Access Server back to Management Console for monitor updates.

- 3—Communication from Management Console to the Access Server service, per datastore (that is, per InfoSphere CDC instance). This requires two ports for each InfoSphere CDC instance.
- 4—Communication from the Access Server service to the datastore, listen process. This is established for each Management Console connection.
- 5—Communication from the Access Server service to the datastore, monitor process. This is a shared connection between all Management Console connections on the same datastore. This requires two ports for each datastore.

You must also configure your routers and firewalls to allow communication through the configured ports. For more information, contact your network administrator.

Management Console requires:
- One input and output port to the Access Server.
- One input port from the Access Server
- One input and output port per datastore (regardless of whether you connect to the datastore)

The Access Server requires:
- One input and output port per datastore, per installation of Management Console
- Two input and output ports, per datastore

Additionally, you can have more than one datastore, or more than one installation of Management Console; for example:
- One installation of Management Console and one datastore
- One installation of Management Console and two datastores
- Two installations of Management Console and one datastore
- Two installations of Management Console and two datastores

Example: calculating ports required

To help determine the number of ports required, take a scenario where there are ten concurrent users and three datastores.

To calculate the number of Access Server ports to open, use this formula: `number of ports to open = 2 * (number of users + (number of users * number of datastores) + number of datastores)` where a datastore refers to an InfoSphere CDC instance.

Using the previously mentioned scenario of ten concurrent users and three datastores, the number of Access Server ports required is 86. Here is the breakdown of the calculation, following the order in the figure illustrating the ports you can configure for Management Console and Access Server components:
- Number of concurrent users that will log into Access Server = 10
- One port per user to connect to and deliver unsolicited message = 10
- Number of possible concurrent connections from Management Console to connect to datastores); that is, 10 users * 3 datastores = 10 * 3
- Number of possible concurrent connections to datastore, listen process; that is, 10 users * 3 datastores)
- Two ports required to connect to each datastore, monitor process = 2 * 3

Therefore, 10 + 10 + (10 *3) + (10 *3) + (2 *3) = 86

To calculate the number of ports to open Management Console, use this formula: `number of ports to open = 2 + number of datastores`

Using the previously mentioned scenario of ten concurrent users and three datastores, the number of ports required is 5 for each Management Console. This is the breakdown of the calculation for each Management Console:

- Connection to Access Server = 1
- Connection for unsolicited updates from Access Server = 1
- One port for each connection to a datastore, listen process = 1 * 3

Therefore, 1 + 1 + (1 *3) = 5

See also:

To configure static ports

Procedure

1. Open the `dmaccessserver.vmargs` file in a text editor. This file is located in the `conf` directory in your Access Server installation directory.
2. Replace the entry in this file with the following text:

   ```
   -jar lib/server.jar local_port:<first_port>
   local_port_count:<number_available_ports> <Access_Server_listener_port>
   ```

   where:

   - *<first_port>* is the first port in the range that you want the Access Server service to use when sending messages or establishing connections.
   - *<number_available_ports>* is the number of ports you want to reserve for this use.

     To calculate the number of Access Server ports to open, use this formula: `number of ports to open = 2 * (number of users + (number of users * number of datastores) + number of datastores)` where a datastore refers to an InfoSphere CDC instance.
   - *<Access_Server_listener_port>* is the port number that Access Server listens on and is set during the Access Server installation. You do not have to specify a value here if you are using the default port number of `10101`.

   For example, if the number of available ports for communication is 500 and you want Access Server to listen for connections on port 10101, then the entry would be as follows:

   ```
   -jar lib/server.jar local_port:10102 local_port_count:500 10101
   ```

   This enables Access Server to listen for connections on port 10101 and restricts it to using ports in the range of 10102 to 10601.

   These changes will take effect after you restart the Access Server service.

Installing Management Console

Management Console is the administrator application that provide the necessary support to configure, manage, and monitor an entire replication configuration from a central location.

This section outlines how to install Management Console.

To install Management Console

Procedure

1. Double-click on the installation file.

   The IBM InfoSphere Data Replication Management Console installation wizard opens.
2. Click **Next**.
3. Accept the terms of the license agreement by selecting **I accept the terms in the license agreement** and click **Next**.
4. Select the folder where you want to install Management Console and click **Next**.
5. Select the location for the product icons and click **Next**.
6. Review the installation summary and click **Install**.
7. Click **Done** to exit the installation.

Chapter 8. Configuring InfoSphere Classic CDC for z/OS

Configuring InfoSphere Classic CDC for z/OS requires setup tasks that include obtaining library authorizations and ports for replication, preparing the data source environment, deploying and securing data servers, setting up IMS and VSAM for replication, and setting up the source server for InfoSphere Classic CDC for z/OS.

Gathering information and securing your environment

Classic change data capture requires initial setup tasks, such as obtaining library authorizations and port assignments and securing your Classic data servers.

Obtaining library authorizations for the APF

InfoSphere Classic CDC for z/OS requires authorized program facility (APF) authorization for the installation load library on the source logical partition (LPAR).

About this task

The source servers and utility programs use z/OS facilities and services that require APF authorization. The load library data sets must be defined to z/OS as APF authorized.

Procedure

Before setting up and configuring the source servers, ensure that the installation load library SCACLOAD is APF authorized.

Obtaining ports for replication

InfoSphere Classic CDC for z/OS uses TCP/IP to communicate between various components, which requires multiple port numbers.

Before you begin

Obtain the port assignments that you need from your network administrator to ensure that you are using dedicated ports for the exclusive use of Classic change data capture.

About this task

Each subscription that replicates change data to the target server uses a TCP/IP connection. Work with your network administrator to ensure that source servers have authorization to use the assigned ports.

Procedure

You must obtain a port assignment for each of the following communication channels on the source server.

1. A port for connections with the Classic Data Architect (CDA), Management Console. You define this port number as part of the COMMSTRING configuration parameter for the INIT service.

- CDA uses this port for table mappings and optionally for configuration.
- Management Console use this port for subscription definition and operations.

9087 is the well-known and reserved port for connections between the these components and a connection handler service (INIT) that runs in the source server.

2. For IMS, define a listening port for the IMS log reader service. You define this port number as part of the NOTIFICATIONURL configuration parameter for the LRSI service.

Securing your replication environment

To implement security, work with your Security Administrator to define any required classes and profiles, and then secure your Classic data servers and catalogs.

Use these different layers to secure your replication environment:
- "z/OS-level security"
- "Server security"
- "Catalog security" on page 56

z/OS-level security

The System Authorization Facility (SAF) is a z/OS interface that programs use to communicate with an external security manager (ESM), such as the Resource Access Control Facility (RACF®). SAF and your ESM work together to grant access rights to system resources, such as the following:
- Classic data servers
- Services
- Tables
- Catalogs

RACF *classes* organize profiles into groupings of related system resources. *Profiles* define security for specific users, groups, and protected resources. Your security administrator creates classes and profiles, then grants users or groups READ or CONTROL access to the resources in the profiles.

For more information about z/OS security, see the *z/OS Security Server RACF Security Administrator's Guide*.

Server security

The following table describes specific tasks on a Classic data server and the required SAF access that a user needs to perform them. A user with CONTROL access automatically has READ access.

Table 11. Tasks and required user access

Task	Service to configure	Minimum required access
View subscriptions	Administration service	READ
Create, update, and delete subscriptions	Administration service	CONTROL
Manage replication (start, stop, change state)	Administration or Operator service	CONTROL

Table 11. Tasks and required user access (continued)

Task	Service to configure	Minimum required access
Monitor metrics	Monitoring service	READ
Issue remote console commands	Operator service	READ

The SAFEXIT service parameter

Secure your Classic data server by using the SAFEXIT service parameter. The following services have a SAFEXIT parameter:

- Administration service (PAA)
- Monitoring service (MAA)
- Operator service (OPER)
- Query processor service (QP)

SAFEXIT and protected resources

If you define the SAFEXIT parameter by specifying the SAFEXIT value `CACSX04,VALIDATE=N` the Classic data server performs user ID and password authentication only. No other user validation takes place.

Use the SAFEXIT value `CACSX04,VALIDATE=Y` to grant multiple users different levels of access to system resources based on classes and profiles. For each of these services except the monitoring service, you can override the default class and profile by specifying different z/OS class or profile names as values for service parameters. Your ESM then authenticates user access by checking the specified profiles.

Table 12. Default classes and profiles per service, with override service parameters

Service name	Default class	Default profile	Override class parameter	Override profile parameter
Administration service	SERVAUTH	CEC.ADMIN	ADMCLASS	ADMPROF
Monitoring service*	Not applicable	Not applicable	Not applicable	Not applicable
Operator service	SERVAUTH	CEC.OPER	OPRCLASS	OPRPROF
Query processor service	Not applicable	Not applicable	Not applicable	Not applicable
* See the 'Monitoring service' section below for additional information.				

You can supply values for these parameters during the installation customization process or by setting them in the Console Explorer in the Classic Data Architect. For example:

```
SAFEXIT="CACSX04,VALIDATE=Y,ADMCLASS=xxxxxxxx,ADMPROF=yyyy.yyyyy"
```

Administration service

The administration service secures user connections to the Classic data server by checking z/OS credentials and access rights to protected resources.

The Classic data server also restricts access rights for InfoSphere CDC users based on catalog privileges. See the section about catalog security.

Monitoring service

The monitoring service secures access to subscription states, statuses, and metrics.

A Classic data server provides different ways of accessing monitoring information, depending on your solution. Data Replication for InfoSphere Classic CDC for z/OS sends monitoring information to the InfoSphere CDC Management Console for display.

InfoSphere CDC Access Server maintains information about user accounts and access rights. In Classic change data capture, users of IBM InfoSphere Change Data Capture Access Server who connect to the monitoring service automatically have monitoring privileges. For this reason, the **`SAFEXIT`**, **`MONPROF`**, and **`MONCLASS`** parameters have no effect on access to monitoring information.

Query processor service

The query processor service uses the SAF exit to authenticate users who map tables and run test queries.

Operator service

The operator service authenticates users who run remote console commands on the Classic data server, including the MTO command files run from the Classic Data Architect. Console users are generally system operators who make z/OS system console requests to a Classic data server.

The operator service does not secure operator commands that you enter from a z/OS console or equivalent interface, such as the System Display and Search Facility (SDSF). When you issue commands to the console you have implied authority to issue commands to the Classic data server, so you must secure these command interfaces to prevent unrestricted access. Ensure that users who run remote operator commands have READ access to connect the data server and to issue the DISPLAY command, and CONTROL access for all other commands.

Note: You can also use the **`SQLSECURITY`** parameter in the operator service to specify whether to secure operator commands at the user level. When set to `TRUE`, the Classic data server checks the system catalog for DISPLAY and SYSOPER privileges for users who run commands on the Classic data server.

Catalog security

Catalog security manages access to objects in the catalog, such as tables, procedures, and views. The Classic data server maintains security information in a table in the catalog.

Catalog security also extends to the subscriptions and replication mappings that reference these tables and views. If you map a table, you own it and automatically have full access rights to perform replication tasks. If different users administer replication, then you must run data description language (DDL) on the Classic data server that contains the appropriate GRANT statements.

Note: Ensure that users have the required access to all tables and views in the subscriptions that they manage. Catalog security prohibits access to subscriptions unless you have SELECT access to all the tables in the subscription. You can unexpectedly remove a subscription from a user's view by adding a new table to which the user does not have access.

The table describes specific actions against objects in the catalog and the required SQL-based access to perform them.

Table 13. Actions against catalog objects and required access

	SELECT	INSERT	UPDATE	DELETE
Read tables and views	✓			
Modify tables and views		✓	✓	✓
Read subscriptions	✓ (every table or view)			
Modify subscriptions	✓ (every table or view)		✓ (every table)	
Read replication mappings	✓			
Modify replication mappings	✓			

A user with SYSADM access can read or modify all tables in the catalog. A user with DBADM access can read or modify all tables in a specific database class, such as $IMS or $VSAM.

If your catalog has many tables, using catalog security can affect performance, especially when you define subscriptions or start replication. If your site has multiple administrative users who require access to different tables, then you must implement table-level security. If catalog security is not necessary, specify GRANT SYSADM TO PUBLIC in the catalog so that all authorized users of the Classic data server can access all the tables and views.

Securing a Classic data server

To secure your Classic data server, create security classes and profiles in your external security manager (ESM) and configure service-level parameters in the server.

About this task

Work with your Security Administrator to determine the appropriate levels of security for your site. For best results, secure as many operations as you can while maintaining site standards and performance. You can secure administrative connections to your Classic data server, remote operator commands, monitoring, table mapping, catalogs, and the replication process itself.

When you secure the configuration data sets and the VSAM files for subscriptions and replication mappings, use the following table to locate them. You specified high level qualifiers for these data sets in the customization parameters file when you performed the installation customization process.

Table 14. High level qualifiers and corresponding data sets.

High level qualifier (HLQ)	Function

CDCPCFGD="&&CPHLQ..CACCFGD"
 	
CDCPCFGX="&&CPHLQ..CACCFGX"
	Configuration data sets

Table 14. High level qualifiers and corresponding data sets. (continued)

High level qualifier (HLQ)	Function

CDCCATNM="&CPHLQ..CATALOG"
 	
CDCIDXNM="&CPHLQ..CATINDX"
	Catalog data sets

CDCPSUBS="&CPHLQ..SUB"	Subscription data sets
CDCPRMDS="&CPHLQ..RM"	Replication mapping data sets

Procedure

1. Determine which users require access to your Classic data servers and the level of access that each user requires.

   Decide whether each user requires READ or CONTROL access to subscriptions, administrative functions, and remote console commands.
2. Ensure that each user has a valid z/OS user account.
3. If you grant different levels of access to system resources for multiple users, follow these steps.
   a. Ask your System Administrator to define the profiles and classes that you require by using the System Authorization Facility (SAF).

      Follow the remaining steps for each applicable service:
      - Administration service (PAA)
      - Operator service (OPER)
      - Query processor service (QP)

   b. Ensure that VALIDATE=Y for the **SAFEXIT** parameter.

      VALIDATE=Y is the default setting, so omitting the keyword has the same effect as specifying VALIDATE=Y.

      If you use SAFEXIT with VALIDATE=N then the Classic data server only validates that each user has a valid user account and password. Users have full access to objects on the Classic data server except the catalog, where SQL privileges determine access.

   c. Optional: Override the default class and profile names in the *xxx***CLASS** and *xxx***PROF** parameters.

      Exception: You cannot specify class and profile overrides for the monitoring service or query processor service.
4. Run data description language (DDL) on the server that contains the appropriate GRANT statements to secure access to catalog objects.
5. Perform the following security tasks for the source server.
   a. For IMS:
      - Grant UPDATE access to the IMS RECON data sets.
      - If you use IMS DBRC command authorization, the Classic data server needs LIST.* authority for DBRC.
      - Ensure that the data server has READ authority for the IMS DBDLIB data sets that the data server references. Grant READ authority for all IMS log data sets that need to be accessed.
        - For a DB/DC or DBCTL subsystem these data sets include the primary and secondary online log data sets and the corresponding primary and secondary system log data sets.

  - For logs produced by DL/I batch jobs one of these types of logs will only be accessed if it contains data capture log records. When processing historical changes, the log can potentially contain data capture log records.

b. Grant READ access for CLASS(FACILITY) RESOURCE(BPX.CONSOLE) to the job or started task for the Classic data server.
c. Grant UPDATE access to the configuration data sets and the VSAM files for subscriptions and replication mappings.
d. The user ID associated with the job or started task needs an OMVS segment for TCP/IP access.

Using the SAF exit for security validations

The SAF exit verifies access for user and client connections and controls the operations that a user can perform.

The SAF exit controls access to system resources based on classes and profiles. The exit authenticates user passwords at connection time. In addition, you can configure the exit to perform the following functions:

- Validate user authorization to access system resources, access metrics data, or run remote console commands.
- Authenticate the TCP/IP address of client connections at connection time

Recommendation: Use the supplied sample exit (CACSX04) in the SAMPLIB data set. When you use the sample exit, you do not need to re-assemble or re-linkedit the exit.

If you choose to modify or replace the supplied sample exit, you must assemble and bind the module as described in the overview of the SAF exit API.

Activating the SAF exit:

To activate the SAF exit, you set up security resources and define configuration parameters during the installation customization process.

Before you begin

Ensure that the STEPLIB DD statement in the JCL for the Classic data server references the library where the SAF exit load module (CACSX04) is located. The library must be APF-authorized

About this task

Ensure that only valid z/OS users can access resources by setting SAFEXIT=CACSX04 for the administration service, monitoring service and operator service.

You can configure the supplied SAF exit CACSX04 by using the following configuration parameters. You can supply configuration parameters only when you use the IBM-supplied version of the SAF exit CACSX04.

VALIDATE=*Y/N*
: Indicates whether the exit should validate that the user ID has authority to access system resources, access metrics data, or run remote console commands.

Specify an operand of **N** to indicate that the exit routine should not perform access rights validation.
The exit routine is invoked for access rights checking regardless of the **Y** or **N** value of the operand. The default value for VALIDATE is **Y**.

This parameter helps you to control access with greater precision:

- Ensure that only valid users can access resources by setting VALIDATE=Y against the administration, monitor, or operator service.

  VALIDATE=Y authenticates each individual resource.
- Eliminate the overhead of verifying that the user has authority to access a resource by setting VALIDATE=N against the administration, monitor, or operator service. (The Classic data server performs user ID and password authentication only.)

NETACCESS=*Y/N*
Indicates whether the exit should validate the IP address of the connected client to authenticate access to the Classic data server.

Set the value to Y when the IP address of the connected client is known and the SERVAUTH parameter of the RACROUTE REQUEST=VERIFY invocation is supplied. The RACROUTE operation is successful when the associated user ID has at least READ-level access rights to the network security zone resource. If the security system indicates that it cannot make a decision in response to the request because a corresponding network security zone resource profile does not exist, the SAF exit regards the response as Access Denied.

A value of N indicates that the SERVAUTH parameter is omitted from the RACROUTE REQUEST=VERIFY invocation. This is the default.

ADMCLASS=*administrator-class-name*
Indicates the name of the security class that contains a profile that provides access authentication.

This parameter is valid if VALIDATE=Y on the administration service for the SAF exit. If this parameter is not specified, SERVAUTH is the name of the default security class.

ADMPROF=*administrator-profile-name*
Indicates the name of the resource profile that provides access authentication.

This parameter is valid if VALIDATE=Y on the administration service for the SAF exit. If this parameter is not specified, CEC.ADMIN is the default profile name.

OPERCLASS=*operator-class-name*
Indicates the name of the security class that contains a profile that provides access authentication.

This parameter is valid if VALIDATE=Y on the operator service for the SAF exit. If this parameter is not specified, SERVAUTH is the default name of the operator security class.

OPERPROF=*operator-profile-name*
Indicates the name of the resource profile that provides access authentication.

This parameter is valid if VALIDATE=Y on the operator service for the SAF exit. If this parameter is not specified, CEC.OPER is the default profile name.

If you want to modify or replace the supplied sample exit, you must assemble and bind the module as described in the overview of the SAF exit API.

To configure the SAF exit and verify that it is working:

Procedure

1. Edit the sample SAF exit (SCACSAMP member CACSX04) if you need to customize CACSX04.
2. Modify any required SAFEXIT configuration parameters to the service definition for the administration, monitor, or operator service classes (PAA, MAA, or OPER) by using the Classic Data Architect or MTO SET,CONFIG command.
3. After changing the SAF definition for any services that you need to secure, stop and restart the Classic data server.

Example

Ensure that only a valid z/OS user can connect to the operator, administration, and monitoring services by using the following command. This command runs the supplied SAF exit with the default values:

```
F <Data-Server-Name>,SET,CONFIG,SERVICE=<Service-Name>,SAFEXIT=CACSX04
```

If you want to take advantage of the flexibility of the exit by using SAF exit parameters against a monitoring service, add name/value pairs to the command, as follows:

```
F <Data-Server-Name>,SET,CONFIG,SERVICE=<Monitor-Service-Name>,
SAFEXIT="CACSX04,VALIDATE=Y"
```

The first value of the SAFEXIT parameter must be the exit name. You must enclose the SAFEXIT values in double quotes.

SAF exit: API overview:

The parameters passed to the SAF exit are defined by the CACSXPL4 member located in the SCACMAC library. The SAF exit is called for one of three functions: initialization, validation, or termination.

The CACSXPL4 macro describes the interface to the SAF exit. The comments provided in the CACSXPL4 macro describe the SAF structure fields and their intended usage. Comments contained within the CACSX04 source code describe the interface to and the intended behavior of the SAF exit.

Assembling and binding the SAF exit

Member CACALX04 in the USERSAMP data set contains a JCL stream that you can use to assemble and bind the sample SAF exit, CACSX04.

Requirements for assembling and binding CACSX04:

- You must direct the assembler to produce Extended Format (GOFF) object code. Specifying the assembler options GOFF,OBJ,NODECK satisfies this requirement and directs the assembler to write the object code to the file referenced by the SYSLIN DD statement.
- The CACSX04 executable module (a program object) must reside in a PDSE that is included in the list of APF-authorized libraries.

- The traditional load module format is not supported.
- You must direct the z/OS Program Management Binder to produce a program object at the z/OS V1R10 level or higher with support for mixed-case external symbol names. The resultant program object must contain the re-entrant attribute and must not contain the AC(1) attribute.

Specifying the binder options RENT,CASE=MIXED,COMPAT=ZOSV1R10 meets these requirements.

For more information about assembler options, see the High Level Assembler documentation in the z/OS information center.

For more information about the z/OS binder options see the z/OS MVS™ Program Management documentation in the z/VM® information center.

Preparing the replication environment

Preparing the replication environment requires setting up access to IMS and VSAM data sources and setup tasks for the source server.

Preparing a source IMS subsystem for replication

On the source IMS subsystem, augment the database descriptions (DBDs) for each database that you want to replicate.

Augmenting DBDs for IMS change data capture

To generate IMS records for change data capture, you must augment each DBD, typically by adding an **EXIT** parameter to the SEGM statement for each segment that you want to capture.

About this task

By adding the **EXIT** parameter to selected segments instead of the DBD control statement, you generate fewer log records for change data capture, thereby reducing the volume of log records in the IMS subsystem. Choose the EXIT options that work best with your data to keep the size of your log records to a minimum.

To augment a DBD to capture changes from all required segments:

Procedure

1. Using the examples as a guide, add an EXIT keyword to the SEGM statement for each segment in the DBD for which you want to capture log records for change data capture.
   a. Always include the `KEY` option to write concatenated key information to the log record that describes the physical path to the changed segment.
   b. Always include the `DATA` option to place before and after image data for the changed segment in the log records.
   c. If the database and table definitions do not follow the design principles of third normal form, include the `PATH` option to add physical segment data to the log records for parents of the changed segment.

      If they do follow third normal form, specify the `NOPATH` option. In a case like this, you must also design the target tables in third normal form.
   d. Include the appropriate cascade option for your source and target databases.

If you design the target database with the appropriate referential integrity and delete rules to support cascading, then you don't have to code an IMS CASCADE option. Otherwise, specify CASCADE.

2. Run DBDGEN for the updated DBD.
3. Run the ACBGEN utility to update all program specification blocks (PSBs) that reference the DBD.
4. Add the updated DBD and PSB members to your production libraries for access control blocks (ACBLIB).

Examples

In the first example, the database and related table definitions follow the design principles of third normal form. In this case, you can reduce the size of the log records by specifying the `NOPATH` option:

```
EXIT=(*,KEY,DATA,NOPATH,(CASCADE,KEY,DATA,NOPATH),LOG)
```

In the next example, the database and table definitions do not follow third normal form, and you must specify the `PATH` option.

```
EXIT=(*,KEY,DATA,PATH,(CASCADE,KEY,DATA,PATH),LOG)
```

If the insert and delete rules for the source IMS database correspond with the referential integrity constraints at the target database, you can specify the `NOCASCADE` option to reduce log record size:

```
EXIT=(*,KEY,DATA,PATH,(NOCASCADE),LOG)
```

Augmentation of DBD and segment statements:

Follow these guidelines when you augment database descriptions (DBDs) for Classic change data capture.

The DBD modifications in this section affect only the DBD definition in the DBD and ACB libraries and do not affect the physical database.

Augmentation concepts

The log records that IMS generates for recovery purposes do not contain enough contextual information for change data capture. Therefore, you must augment each replicated database to generate special log records. When you supply an **`EXIT`** parameter on a SEGM statement, IMS creates this type of log record for that segment. Augment all segments that meet these criteria:

- You reference the segment as a leaf segment in an IMS Classic table mapping
- You alter that table for change data capture

If you define a view on the table to capture redefined data, alter the view instead of the table.

Choose the right level of augmentation for your DBD. You generate too much data in the records and place an unnecessary load on your system if, for example, you include unnecessary path information. If you add fewer **`EXIT`** options than you require, you might not capture all of the changes that you want to process.

The minimum requirement for Classic change data capture is to capture any deletes, inserts, or updates to a segment and concatenated key information. Classic change data capture also requires additional before image (PATH) data for parent segments that either do not have a sequence field or have a non-unique sequence field.

Choose the EXIT options that work best with your data to keep the size of your log records to a minimum.

PATH option

If your database and its related table definitions follow the design principles of third normal form, all the columns in the table definition map to parent segments that reference sequence fields for all non-leaf child segments. In this case, specify the `NOPATH` option to reduce the size of your log records.

CASCADE option

You typically include a `CASCADE` option to capture information about cascade deletes. You can make an exception if your target tables have the appropriate referential integrity and delete rules to support the deletion of children when you delete a parent. In this case, specify the `NOCASCADE` option instead to reduce the size of your log records.

If you cannot capture all the primary key columns in your table definitions by using only the `KEY` and `DATA` options, you must also specify the `PATH` sub-option with the `CASCADE` option so that delete operations at the target database can include related rows in child tables.

Augmentation of DBDs for child segments

When you are augmenting a DBD with child segments, consider whether your database is designed in third normal form, and the number of child segments that physically exist. When your IMS database follows the design principles of third normal form, each table that you alter for change data capture maps to fields of only one child segment, 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)
```

Presumably, an IMS database that follows third normal form has the required foreign key information in the IMS concatenated key. Code the following EXIT statement for the child segment that you are replicating:

```
EXIT=(*,KEY,DATA,NOPATH,(?),LOG)
```

The question mark represents inherited cascade delete options that are already in effect.

Your IMS database might be in third normal form, but the target engine might not recognize the segment and concatenated key data as sufficient to create a foreign key for its own use. In these situations, you must capture path data:

```
EXIT=(*,KEY,DATA,PATH,(?),LOG)
```

EXIT parameter in DBD and SEGM statements:

To augment the DBD for which you want to capture changes, specify the information that you want to capture in the **`EXIT`** parameter.

Purpose

IMS supports an **EXIT** parameter for the DBD control statement and the SEGM control statement. An **EXIT** parameter on the DBD statement defines default values for all segments in the DBD. An **EXIT** parameter on a SEGM statement overrides the default values.

Format

The format of the **EXIT** parameter is as follows:

```
EXIT=(Exit-Name,KEY|NOKEY,DATA|NODATA,PATH|NOPATH,FLD|NOFLD,INPOS|NOINPOS,SSPCMD|
NOSSPCMD(CASCADE|NOCASCADE,KEY|NOKEY,DATA|NODATA,PATH|NOPATH),LOG|NOLOG)
```

For more information about the format of the EXIT parameter and the CASCADE operand, see the documentation for the IMS system utility *Database Description (DBD) Generation utility*.

Keywords

Table 15. Keywords for the EXIT parameter

Keyword	Purpose
Exit-Name	Indicates the name of a DPropNR synchronous data capture exit. Specify an asterisk (*) to indicate that there is no exit. Specify NONE to deactivate an exit routine on a SEGM statement. Classic change data capture does 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 an asterisk (*) for the **Exit-Name** keyword.
KEY | NOKEY	Indicates whether you want the log records to contain concatenated key information about the physical path to deleted, inserted, or updated segments. The default is KEY.
DATA | NODATA	Indicates whether you want to include before images and after images of the changed segment in the log records for deleted, inserted, or updated segments. The default is DATA.
PATH | NOPATH	Indicates whether you want to include physical segment data in the log records for the parents of deleted, inserted, or updated segments. The default is NOPATH.
FLD | NOFLD	Identifies whether IMS should generate a data capture log record when an application updates a DEDB database using a FLD call. The default is NOFLD. If you are unsure whether any of your applications use FLD calls, include the FLD option on a segment level EXIT specification. Enabling this option does not create additional overhead from an IMS perspective, unless FLD updates are encountered. In that case you will want the changes to be captured and replicated to the target.
Note: Specification of the NOBEFORE and NODLET keywords is not allowed. Specification of the INPOS and SSPCMD keywords is allowed, but the additional information produced in these log records is not used for standard Classic CDC subscriptions or delimited format subscriptions.	
CASCADE | NOCASCADE	Indicates whether you want log records to contain cascade delete information for deleted segments that have child segments.

Table 15. Keywords for the EXIT parameter (continued)

Keyword	Purpose
LOG|NOLOG	Indicates whether you want to generate log records for data capture in the IMS log files. If you specify an asterisk (*) for **Exit-Name**, the default is LOG. For Classic change data capture, specify LOG.

Example

```
EXIT=(*,KEY,DATA,PATH,(CASCADE,KEY,DATA,PATH),LOG)
```

Setting up IMS notification exits

InfoSphere Classic CDC for z/OS uses its own versions of two IMS notification exits (DFSPPUE0 and DFSFLGX0) to inform the IMS log reader service about the start of IMS subsystems and the start and stop of DL/I batch jobs.

About this task

You must perform some tasks to set up these Classic versions of the IMS exits:

The partner program exit (DFSPPUE0)

Informs the IMS log reader service about the start of IMS subsystems, enabling change data capture to include participating subsystems when they start during replication.

The logger exit routine (DFSFLGX0)

Informs the IMS log reader service about the start and stop of DL/I batch jobs, enabling change data capture to suspend ordering operations for participating subsystems for the duration of each job step that issues DL/I calls and has the Classic logger exit installed.

Customizing the configuration table module for IMS notification exits:

To use the notification exits, the configuration table module (CECE1OPT) must be customized for your environment.

About this task

The notification exits communicate their event notification to IMS change-capture through TCP/IP. The system administrator must specify the TCP/IP host name and port information for this exit by customizing, compiling, and linking the assembler module named CECE1OPT. CECE1OPT requires customization because it contains default values that may not be applicable to your specific environment. Samples of the configuration module CECE1OPT are provided in the sample library *USERHLQ*.SAMPLIB.

The address specified in CECE1OPT must be the same as the configuration parameter NOTIFICATIONURL in the log reader service.

Procedure

1. Edit a copy of the CECE1OPT module specification (in the *USERHLQ*.*. SAMPLIB library)., specifying the following parameters:

   IPVSN

   Indicates whether to use Internet Protocol Version 4 (IPv4) or Internet Protocol Version 6 (IPv6). If 4 is specified, the exit connects to the

Classic data server by using the Ipv4 protocol. If 6 is specified, the exit connects to the server by using the IPv6 protocol. Values other than 4 or 6 result in an error.

TCPADDR4
The IP address to be used by the IMS exits for event notification if an IPVSN of 4 is specified. (Ignored if an IPVSN of 6 is specified.) The address is specified as a four byte hexadecimal value.

TCPADDR6
The IP address to be used by the IMS exits for event notification if an IPVSN of 6 is specified. (Ignored if an IPVSN of 4 is specified.) The address is specified as a sixteen byte hexadecimal value.

Port The port number to be used by the IMS exits for event notification. The port is specified as a two byte hexadecimal value.

```
IPVSN    DC    AL1(4)             | 4=IPv4, 6=IPv6
         DC    AL1(0)             | Reserved
TCPADDR4 DC    AL1(192),AL1(168),AL1(4),AL1(37)   IPv4 addr
TCPADDR6 DC    AL2(9)               IPv6 addr - 1st 16 bits
         DC    AL2(3155)            IPv6 addr - 2nd 16 bits
         DC    AL2(3155)            IPv6 addr - 3rd 16 bits
         DC    AL2(3155)            IPv6 addr - 4th 16 bits
         DC    AL2(0)               IPv6 addr - 5th 16 bits
         DC    AL2(0)               IPv6 addr - 6th 16 bits
         DC    AL2(0)               IPv6 addr - 7th 16 bits
         DC    AL2(0)               IPv6 addr - 8th 16 bits
PORT     DC    AL2(1234)                  TCP PORT
```

. Example of a CECE1OPT module specification with an IPv4 address of 192.168.4.37, and a port of 1234. The IPv6 address ignored.

2. Assemble and link the module. The module must be made available in any authorized data set concatenated in the IMS STEPLIB DD statement.

Installing the partner program exit routine:

You must install the partner program exit routine (CECPPUE0) to enable change data capture for IMS to identify and manage IMS subsystems start events that might occur during replication.

Before you begin

Customize the configuration table module.

Procedure

- If you did implement your own IMS partner program exit or are using a different exit, complete the following steps. If you did not, go to the next bullet. When linking the modules, you can link them directly into any authorized data set concatenated in the IMS STEPLIB DD statement.
  1. Edit the sample *USERHLQ*.USERSAMP(CECLRIL6) following the instructions in the file.
  2. Run the sample job *USERHLQ*.USERSAMP(CECLRIL6) to create a backup of the exit that you are using and link the IMS partner program exit CECPPUE0 object with your pre-existing partner program exit object. The new composite exit must be named DFSPPUE0 for the call made by IMS to succeed.
- If you did not implement your own IMS partner program exit and you are not using a different exit, use the provided DFSPPUE0 as is.

What to do next

Verify the correct exit installation by starting the DB/DC or DBCTL region where you installed the notification exit. You must also start the source server and start replication for at least one subscription. Look for the following operator messages in the IMS region JES messages:

```
CECZ0751I Subsystem start notification issued for <subsystem-name>
CECZ0757I TCP/IP connection established for subsystem <subsystem-name>
to IP address <ip-address> on port number <port-number>
CECZ0991I SUBSYSTEM START NOTIFICATION RECEIVED FOR <subsystem-name>.
```

Excluding IMS subsystems from ordering decisions

You can specify a list of IMS subsystems to exclude from change capture operations.

About this task

You might choose to exclude selected subsystems from processing if they are no longer active, or if they change no databases or records that affect ordering decisions.

Exclude subsystems from processing by adding them to the list parameter **SSIDEXCLUDELIST** in the IMS log reader service. You can add values to the list during installation customization or by adding values in the Console Explorer view of the Classic Data Architect.

You cannot replicate any changes that a subsystem makes if you exclude it from ordering decisions. Make sure that each subsystem that you include on the list updates databases that you do not replicate.

Procedure

Add the subsystem id (SSID) of each IMS subsystem that you want to exclude from processing to the **SSIDEXCLUDELIST** parameter. **Tip:** If the Classic data server is running when you set this parameter, stop and restart the server to enable your change to take effect.

Examples

When you use operator commands to exclude or include IMS subsystems from ordering decisions, you must add or remove each subsystem from the list separately.

Use the ADD,CONFIG command to add an IMS subsystem to the list parameter **SSIDEXCLUDELIST**:

```
F server-name,ADD,CONFIG,SERVICELIST=SSIDEXCLUDELIST,SERVICE=IMSLRS,VALUE=IMSA
```

Use the DELETE,CONFIG command to remove an IMS subsystem from the list parameter **SSIDEXCLUDELIST**:

```
F server-name,DELETE,CONFIG,SERVICELIST=SSIDEXCLUDELIST,SERVICE=IMSLRS,VALUE=IMSA
```

Using a BMP program to manage IMS subsystem inactivity automatically

To avoid halting change data capture or increasing latency in a data sharing environment when an IMS subsystem becomes inactive, generate activity in that subsystem to flush the log buffers for the online data sets (OLDS).

About this task

To replicate data in the correct order across multiple DB/DC or DBCTL subsystems, all subsystems that are active at the source site must generate continuous activity. Otherwise, your deployment is unable to track whether an inactive subsystem has records to capture in the buffers for the online data sets (OLDS).

Use these approaches to manage subsystem inactivity automatically:
- Run a batch message processing (BMP) program to flush the log buffers when a subsystem becomes inactive.
- If you want to save space in your system log data sets (SLDS), ensure that the IMS Log Archive utility does not archive the records that the BMP generates.

Notification of inactive subsystems

The CECZ0400W message provides information about IMS subsystems that become inactive:

```
CECZ0400W IMS change capture halted due to lack of activity from
sub-system subsystem-id. No new log data has been received since:
time-stamp. Current inactivity message threshold value: threshold-value.
```

Exception: If all participating subsystems are inactive, the source server does not issue the CECZ0400W message.

Threshold-value indicates the time interval at which the source server issues the CECZ0400W message. By default, the CECZ0400W message is displayed at 30-second intervals, but you can adjust this by changing the value of the **INACTTHRESHOLD** parameter in the IMS log reader service.

Tip: If a participating IMS subsystem fails and you are not running fast database recovery (FDBR), you must manually close the log for the failed subsystem. A failed IMS subsystem with an open log is equivalent to an inactive IMS subsystem.

The BMP program

Using the first sample in the Examples section as a guide, run the IMS-supplied DFSDDLT0 utility to issue LOG DL/I calls that generate logging activity and flush the OLDS buffers. Create enough copies of the DATA lines to generate sufficient data to flush the buffers. The logged data must equal or exceed the size of the OLDS buffer that you specified in the **BLKSZ DCB** parameter when you allocated the OLDS data sets.

A single LOG call in the program DFSDDLT0 can never be greater than 9999 bytes, and cannot exceed the **IOASIZE** parameter in the IMS Program Specification Block (PSB) generation utility. If the IOAREA for this job is too large, IMS issues an AT status code.

Tip: The log records that the BMP program writes have an ID in the first two bytes of data (`x'D3D6'` in these examples). If your applications or tools access IMS log records, ensure that this log record ID does not conflict with these operations.

Save SLDS space

Use the second sample in the Examples section to help you customize the job control language (JCL) for the IMS Log Archive utility to exclude records that the BMP generates from archiving activities.

Set the **NOLOG** parameter to the hexadecimal value of the log record ID by using this syntax:

```
SLDS NOLOG (D3)
```

Use comma-separated values to specify additional IDs to exclude from logging:

```
SLDS NOLOG (D3,10,45,5F,67,69)
```

Restriction: Do not exclude log records for data capture from the archive logs by adding '99' to this list. Classic change data capture requires these type 0x99 log records for data capture.

For more information about the IMS resource configuration parameters in the first example, see the *IMS System Administration Guide*. For more information about the Log Archive utility, see the IMS documentation for system utilities.

Procedure

1. Create automation that intercepts the message CECZ0400W.
2. Set up the automation to run a BMP program that flushes the OLDS buffers.
3. Optional: Customize the JCL for the Log Archive utility (DFSUARC0).

Examples

In this example, you run the utility DFSDDLT0 as an IMS BMP. The control statements use LOG calls to generate 35224 bytes of log data by using a GPSB with the name STLGPSBB.

```
//* CECLRIFS PROVIDE VALID JOB CARD
//*******************************************************************
//**        EXAMPLE NOT INTENDED FOR EXECUTION
//**
//**  Licensed Materials - Property of IBM
//**
//**  5655-R54, 5655-R55, 5655-R56, 5655-R57
//**
//**      Copyright IBM Corp. 2008,2010 All Rights Reserved
//**
//**  US Government Users Restricted Rights - Use, duplication or
//**  disclosure restricted by GSA ADP Schedule contract with
//**  IBM Corp.
//**
//*******************************************************************
//**
//** JOB CONTROL STATEMENTS TO GENERATE SUFFICIENT ACTIVITY AND    *
//** LOG RECORDS TO FLUSH THE OLDS BUFFERS.                        *
//** THIS WILL TEMPORARILY RESUME Data Replication for IMS LOG ORDERING     *
//** OPERATIONS FOR THE SUBSYSTEM IDENTIFIED IN THE IMSID FIELD.   *
//** SPECIFY THE FOLLOWING PARAMETERS ACCORDING TO SITE STANDARDS: *
//** o VALID JOBCARD                                               *
//** o IMSID = IMS CONTROL REGION SUBSYSTEM ID                     *
//** o EXEC BMP PROCEDURE NAME                                     *
//** o MBR = GPSBNAME                                              *
//**                                                               *
//** APAR...  ID  PREREQ.  DATE....  DESCRIPTION..................*
//**                                                               *
//*******************************************************************
//BMP      EXEC BMP8CSAM,MBR=DFSDDLT0,IMSID=IMS1,PSB=STLGPSBB
//BMP.SYSIN DD *
S22 2 2 2 2    TP     2
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 1
L        LOG
```

```
L  Z2516 DATA  LOG FILL BUFF PART 2
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 3
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 4
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 5
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 6
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 7
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 8
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 9
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 10
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 11
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 12
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 13
L        LOG
L  Z2516 DATA  LOG FILL BUFF PART 14
/*
```

The next example shows customized JCL for the Log Archive utility DFSUARC0. The **NOLOG** parameter specifies that the utility does not write log records to your SLDS if the log record ID begins with x'D3'.

```
/ARCHIVE2 JOB MSGCLASS=A,CLASS=A,MSGLEVEL=(1,1)
//*
//ARC2    EXEC PGM=DFSUARC0,PARM='SYSA'
//STEPLIB   DD DSN=IMS.&SYS2..SDFSRESL,DISP=SHR
//*   COPY FROM 2 OLDS TO DUAL SLDS                  */
//DFSOLP02  DD DSN=OLP902,DISP=SHR
//DFSOLP00  DD DSN=OLP900,DISP=SHR
//DFSOLS00  DD DSN=OLS900,DISP=SHR
//DFSOLS02  DD DSN=OLS902,DISP=SHR
//DFSSLOGP  DD DSN=SLDSP.D82001.N001,DISP=(,KEEP),
//             UNIT=TAPE,VOL=(,,,99),LABEL=(,SL)
//DFSSLOGS  DD DSN=SLDSS.D82001.N001,DISP=(,KEEP),
//             UNIT=TAPE,VOL=(,,,99),LABEL=(,SL)
//RECON1   DD  DSN=RECON1,DISP=SHR
//RECON2   DD  DSN=RECON2,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
//SYSIN    DD  *
  SLDS NOLOG (D3) FEOV (08000)
  /* DFSUARC0 FORCES AN EOV FOR A DATA SET            */
  /* AFTER WRITING 8000 BLOCKS, AND WRITES NO         */
  /* D3 LOG RECORDS TO THE DATA SET THAT THE          */
  /* DFSLOGP DFSLOGS DD POINTS TO.                    */
/*
```

What to do next

The SYSPRINT DD statement in DFSUARC0 points to the data set that contains a list of excluded records. Verify that records beginning with your log record ID are on the list.

Managing IMS subsystem inactivity manually

Follow these manual steps to prevent inactive IMS subsystems from affecting the performance of your replication environment.

About this task

InfoSphere Classic CDC for z/OS provides manual and automated options to manage the effects of inactive subsystems on latency or message activity.

Ideally, use the CECZ0400W message as a trigger to generate new activity automatically. You can do this by running a prepared BMP or WIFI region that generates a sufficient number of records to flush the buffers.

You can easily manage subsystem inactivity if your site can tolerate latency:
- Wait for new activity on inactive subsystems to flush the buffers naturally and permit ordering operations to continue.
- Set a higher value for the **INACTTHRESHOLD** parameter in the IMS log reader service (IMSL) to reduce message activity on the system console.

  If you do not expect update activity on inactive subsystems for an extended period of time, increasing this value reduces the frequency of the CECZ0400W message.

Use the following steps to manage inactive subsystems manually in response to the CECZ0400W message:

Procedure

- If you do not expect workload on the identified subsystem, shut it down.

  The source server removes this subsystem from ordering decisions.
- Issue the IMS command **/CHECKPOINT** for the identified subsystem.

  This is a temporary measure that flushes the buffers and writes their log records to the online data sets (OLDS). Ordering operations can continue to merge changes from all participating subsystems until one of the following events occurs:
  - The period of time elapses that you specified in the **INACTTHRESHOLD** parameter
  - A subsystem becomes inactive

Setting up access to VSAM data sets for replication

To replicate VSAM data sets, you must *augment* your source files for change data capture and define replication log streams to store the log records.

Different environments perform replication logging, depending on the type of file access:

CICS Transaction Server (Customer Information Control System)

The CICS address space performs replication logging for CICS access.

CICS VR (CICS VSAM Recovery)

The CICS VR address space performs replication logging for batch job access.

Augmenting VSAM data sets for change data capture

To perform change data capture from a VSAM data set, augment the file by using the IDCAMS utility to set the **LOGREPLICATE** parameter and set the value for the **LOGSTREAMID** parameter in the integrated catalog facility (ICF).

About this task

For Classic change data capture, use IDCAMS **DEFINE CLUSTER** or **ALTER** commands to define replication options in the ICF rather than in CICS resource definitions. For more information about the IDCAMS utility, see the IBM documentation for z/OS basic skills.

Set these parameters to activate replication logging for a VSAM data set and specify its related log stream.

LOGREPLICATE|NOLOGREPLICATE

A LOGREPLICATE value activates replication logging for the VSAM data set. A NOLOGREPLICATE value disables replication logging. Classic change data capture requires **LOGREPLICATE**.

LOGSTREAMID(*log-name*)

Identifies the name of the log stream that stores the log records that describe changes to the VSAM data set. The log streams that you specify on the LOGSTREAMID parameter are the same as the replication log streams that you define.

Note: Both replication logging and forward recovery logging use the **LOGSTREAMID** parameter, and both operations share the same log stream. For information about recovery options for CICS VSAM data sets, see the optional DEFINE CLUSTER parameters.

Procedure

1. Run the IDCAMS utility to specify **LOGREPLICATE** in the ICF catalog entry for the VSAM data set.
2. Specify the name of the log stream in the **LOGSTREAMID** parameter.

Example

The following example shows the job control language (JCL) for an IDCAMS ALTER command that sets the **LOGREPLICATE** parameter and adds the logstream CICSD.CICSD.DFJ02 to an existing VSAM data set USER1.SAMPLE.VSAMFILE.

```
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  ALTER -
  USER1.SAMPLE.VSAMFILE -
  LOGSTREAMID(CICSD.CICSD.DFJ02) -
  LOGREPLICATE
/*
```

Defining replication log streams

You need to define a replication log stream for the VSAM data sets that you activate for replication logging.

About this task

A replication log is treated the same as a CICS general log.

When you define a replication log stream, you need to determine the type of log stream to define and consider how the options that you specify affect replication logging.

Procedure

Define a replication log stream by using the Administrative Data Utility (IXCMIAPU).
The replication log stream name that you specify for the NAME parameter must match the LOGSTREAMID parameter that you define when augmenting VSAM data sets for change data capture.
See the z/OS product documentation for detailed information about using this utility.

Example

The following example shows sample job control language (JCL) for defining a replication log stream.

```
//DEFLOG EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *

DATA TYPE(LOGR) REPORT(NO)

DEFINE LOGSTREAM NAME(STREAM1.VSMLOG1)
 DASDONLY(YES)
 LS_SIZE(1024)
 STG_DATACLAS(STGCLS)
 STG_SIZE(512)
 RETPD(14) AUTODELETE(YES)
/*
```

Coordinating tieup records

TBA

Switching between batch and online (CICS)

TBA.

Setting up a source server for change capture

Setting up a source server for change capture requires preparation tasks to customize, start, and validate the server.

Meeting requirements for universal character set (UCS) support

To enable the administration service and the monitoring service to use the z/OS Unicode environment, ensure that you load the required conversion tables for universal character set (UCS) support and that they are available.

About this task

The z/OS Unicode environment exists on every logical partition (LPAR) either because you explicitly configured the environment or because z/OS loaded the default environment. Confirm that the z/OS Unicode environment is available by issuing the z/OS operator command **D UNI**.

You specify the codepage for a Classic data server by setting the value of the **HOSTCODEPAGE** parameter for the region controller service. The default value is 37.

The administration service and monitoring service send and receive data in the following code pages:

Table 16. Code page conversions by service

Service	**Code page**
Administration service	UTF-8
Monitoring service	UCS-2

The z/OS operating system supports multiple techniques for codepage conversion, and a Classic data server uses the LER technique for search order (Language environment, Enforced subset, Round trip). If UCS conversion tables are not available in your z/OS Unicode environment, ask your system programmer to define them.

The following conversion tables must be available:

- *Server codepage* to UTF-8 (1208)

  Note: This conversion table is used for delimited publishing when the value of the PUBUTF8 configuration parameter is set to TRUE.
- UTF-8 (1208) to *server codepage*
- *Server codepage* to UCS-2 (1200)
- UCS-2 (1200) to *server codepage*
- 1047 to UTF-8 (1208)

  This conversion table is also needed if the value of the PUBUTF8 configuration parameter is set to TRUE.

For information about codepage conversion techniques and z/OS handling for codepage 1200, see *z/OS Support for Using Unicode: Using Unicode Services*. In z/OS version 12, see *Unicode Services User's Guide and Reference*

For information about updating conversion tables by using the **SETUNI** command or the **D UNI** command, see *MVS System Commands*.

Procedure

1. Issue the **D UNI** command to determine whether the z/OS Unicode environment is available on your LPAR.

   See the example for sample output.
2. Issue the following commands to verify whether the required conversion tables exist.

   If you changed the value of the **HOSTCODEPAGE** parameter, use that value in the command instead of 37.
   a. `D UNI,FROMID=37,TOID=1208`
   b. `D UNI,FROMID=1208,TOID=37`
   c. `D UNI,FROMID=37,TOID=1200`
   d. `D UNI,FROMID=1200,TOID=37`
3. If the required conversion tables are not available, ask your system programmer to define them by using one of the LER techniques.

Examples

This example shows one possible output for the **D UNI** command when you issue it without arguments:

```
D UNI
CUN3000I 08.20.58 UNI DISPLAY 398
 ENVIRONMENT: CREATED       03/02/2009 AT 06.10.27
              MODIFIED      04/29/2009 AT 14.54.15
              IMAGE CREATED --/--/---- AT --.--.--
```

The remaining examples show sample output for the **D UNI** command when you issue it with the `FROMID` and `TOID` arguments. The CONVERSION value indicates that the conversion table exists, and the trailing L, E, or R indicates the conversion technique.

Command processing substitutes 1200 for codepage 13488 because the z/OS Unicode environment has special handling for codepage 1200.

```
D UNI,FROMID=37,TOID=1208
CUN3000I 08.03.18 UNI DISPLAY 750
CONVERSION: 00037-01208-R 00037-01208-L

D UNI,FROMID=1208,TOID=37
CUN3000I 08.03.28 UNI DISPLAY 752
CONVERSION: 01208-00037-E 01208-00037-L
D UNI,FROMID=37,TOID=1200
CUN3000I 15.11.37 UNI DISPLAY 558
  CONVERSION: 00037-01200(13488)-ER

D UNI,FROMID=1200,TOID=37
CUN3000I 15.11.56 UNI DISPLAY 560
  CONVERSION: 01200(13488)-00037-ER
```

If the conversion table does not exist, the output is similar to the following example:

```
D UNI,FROMID=1200,TOID=99999
CUN3000I 08.07.14 UNI DISPLAY 846
CONVERSION: NO CONVERSIONS FOUND
```

Setting up IMS access

Setting up access to IMS data sources requires configuring IMS DBRC access for the source server and setting up the database resource adapter (DRA).

Setting up IMS DBRC access for a source data server

The source server requires access to the recovery control (RECON) data sets in the source IMS subsystem. IMS Database Recovery Control (DBRC) provides this access.

About this task

DBRC cannot access the IMS RECON1, RECON2, and RECON3 data sets on behalf of the Classic data server unless the user ID associated with the address space has RACF UPDATE or equivalent authority to the dataset names.

You can use the application programming interface (API) for DBRC to grant installation control to individual requests that users issue. If your site uses DBRC command authorization, you might also have to authorize the following query requests:

- STARTDBRC

- STOPDBRC
- QUERY RECON
- QUERY LOG STARTTIME
- QUERY OLDS SSID=

For information about the security features of the DBRC API and how these requests correspond with SAF resources, see the IMS system programming documentation.

Procedure

1. Ask your Security Administrator to grant the user associated with the address space for the Classic data server RACF UDPATE authority to the IMS RECON1, RECON2, and RECON3 dataset names.
2. If your site uses DBRC command authorization, authorize any required query requests.

Setting up the database resource adapter (DRA) for IMS access

InfoSphere Classic CDC for z/OS requires the use of the database resource adapter (DRA) interface on the source server for IMS access.

About this task

Work with your IMS system administrator or system programmer to set up the DRA. The source server uses the IMS DRA interface to communicate with the source IMS subsystem for refresh processing. The DRA is also used for mapping verification of delimited and non-delimited subscriptions.

The default load module for the DRA, DFSPZP00, is in the IMS.SDFSRESL library. The remainder of the DRA modules reside in a load library that is dynamically allocated by DFSPRRC0 (the startup). The default DDNAME and DSNAME of the load library (IMS.SDFSRESL) are specified in the default startup table DFSPZP00. DFSPZP00 contains default values for the DRA initialization parameters.

You must also configure a DRA service that references the DRA startup table in the source server.

Note: This service does not apply to publishing changes for delimited format subscriptions to WebSphere MQ.

Procedure

1. Code a new start-up table. Name it, for example, DFSPZP01. You might want to use DFSPZP00 as an example.
2. Specify the required values.
3. Copy any unchanged values from the default table.
   a. Set a value of at least 3 for MINTHRD.
   b. Set a value of 5 for MAXTHRD.
4. Assemble and link the new module into a load library. If the new module is named DFSPZP01, the IMS DRA initialization, DRATABLESUFFIX configuration parameter, specifies a value of 01 in the DRA startup-table suffix.

What to do next

For information about DFSPZP00 see the IMS documentation about installation and customization. For information about using the DRA start-up table to optimize performance, see the CICS documentation about performance.

Setting up VSAM access

Setting up access to VSAM data sources requires configuring the VSAM access service.

Setting up the VSAM access service

InfoSphere Classic CDC for z/OS requires the use of the VSAM access service on the source server for VSAM access.

About this task

Work with your system administrator or system programmer to set up the VSAM access service.

Procedure

Important: Set the value of CLOSEONIDLE to FALSE to ensure that the data set is always closed after each access.

Defining VTAM and CICS resources

TBA

Understanding Classic data server configuration methods

After completing the installation customization process, you can use Classic Data Architect to update the definitions of the configurations for the Classic data server.

The initial configuration of a Classic data server is accomplished with the installation customization process. When you complete the installation customization process, the installation environment includes an operational Classic data server and all required services. The services are pre-configured during the customization process. You can then build upon the operational environment as needed.

You can use tools to administer the configurations for the Classic data server. In most cases, you need to stop and then restart the data server for the configuration updates to take effect.

The tools that you can use include:

- **Classic Data Architect—**Use Classic Data Architect to modify configurations while the Classic data server is running.

  By using the wizards that Classic Data Architect provides, you can make updates to a configuration for a Classic data server, including modifying global configuration parameters that affect server-wide configuration; and adding, modifying, or deleting configuration settings for services.

  Classic Data Architect provides a master terminal operator (MTO) command interface that you can use to issue MTO commands remotely from the Console Explorer within Classic Data Architect after you establish an operator connection to the Classic data server.

- **Configuration-related operator commands—**Issue the configuration-related commands by using the z/OS MTO interface. You can use the following configuration-related MTO commands to administer data server configurations against a running Classic data server:
  - `IMPORT` and `EXPORT` commands that you can combine into a command file. This capability enables you to import and export configurations for a Classic data server for backup and recovery purposes. Another use of these commands is to export an existing definition, modify the exported definition, and then import that definition to another Classic data server. You can also use these commands to perform migration functions.
  - `ADD`, `SET`, `DELETE`, and `DISPLAY` commands that you can use to update and display configuration data for a Classic data server.

    With these commands, you can update configuration information for global configuration parameters, services, service classes, and service lists.
- **Configuration migration and maintenance utility—**Use the CACCFGUT configuration migration and maintenance utility to update configurations for a Classic data server offline, when the Classic data server is not running.

  The main purpose of the utility is for backup and recovery of configuration information. You can use the `EXPORT` and `IMPORT` commands to restore a configuration environment to a previous point in time.

Mapping tables

Use the Classic Data Architect to create relational tables and views that map to data sources in supported nonrelational database management systems. Classic Data Architect connects to the source engine from a client workstation to simplify configuration tasks and create table mappings using InfoSphere Classic CDC for z/OS as the replication engine.

Before you begin

You must perform the following tasks on the data server where the query processor will run:
- Create and initialize a metadata catalog.
- Set up the configuration file.
- Start the data server.

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

1. Configure Classic Data Architect by creating prerequisite objects, creating connections to data servers, setting preferences, importing reference files, and granting privileges.
2. Create tables and views, by mapping data for InfoSphere Classic CDC for z/OS.
3. Optional: Modify your tables or views.
4. Generate and run DDL to promote your tables, views, indexes, and stored procedures to a data server.
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.

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

You must perform the following tasks on the data server where the capture service will run:
- Set up and configure a data server.
- Start the data server.

Procedure

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. 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.
3. Set various preferences. See "Setting preferences for change data capture" on page 82.
4. Import Classic reference files into a data design project. See "Importing data definitions into projects" on page 82.
5. Grant privileges to users to create objects or to run commands on a data server. See "Granting privileges for performing actions on data servers" on page 83.

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. This workspace is located on your local workstation, and you do not share it with others. 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

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**. Select Classic Integration as the database type, and select the appropriate version. 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.

Creating connections to the data server operator service

Several of the tasks that you perform in Classic Data Architect require connections to the data server operator. You can create new operator connections, edit operator connections, and import existing operator connections.

Before you begin

To use the configuration support in Classic Data Architect, the operator service must be running on the data server.

To view server and service configuration information and active services, you must create a connection to the command operator service on the data server. When you create an operator connection, Classic Data Architect attempts to connect to the data server.

- If the connection succeeds, the new connection is added to the **Connections** folder in the Console Explorer. You can view connection details in the Properties view.
- If the connection fails, an explanation for the failure displays in the Properties View for the connection.

Procedure

- Create a new operator connection. Open the New Operator Connection wizard by using either of the following options:
  - Right-click the **Connections** folder and select **New connection**.
  - Right-click the New Connection icon in the Console Explorer.
- Edit an existing operator connection. Open the Edit Operator Connection wizard. Right-click a connection in the **Connections** folder in the Console Explorer and select **Edit connection**.

- Import an existing Version 9.5 or greater operator connection from the Data Source Explorer. Open the Import Operator Connections wizard. Right-click the **Connections** folder and select **Import Connections**.

Setting preferences for change data capture

You can set various preferences in Classic Data Architect related to change data capture.

Procedure

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. Set these preferences in the Classic Data Architect pane as needed:
   - Setting an FTP subcommand for DBCS data when you import Classic Data Architect references or exporting files that contain SQL statements.
   - Mapping PIC9(n) USAGE DISPLAY data to the character or decimal SQL data type.
3. Specify the locale that the COBOL parser uses 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.

   Tip: For more information about COBOL and PL/I import preferences supported by Eclipse, see the preference pages associated with the importer.

Importing data definitions into projects

Before you can create tables with Classic Data Architect, you must import data definition files into a data design project. For IMS, you can import COBOL copybooks, PL/I include files, and DBDs.

About this task

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**.
- PL/I include files are placed into a subfolder that is called **PL/I Include Files**.
- 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 your 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.fd` is imported into the project as `mycopybook.cpy`. Imported files are given the following extensions:

Table 17. File types and extensions given when files are imported.

File type	Extension
COBOL copybook	cpy

Table 17. File types and extensions given when files are imported. (continued)

File type	Extension
PL/I include file	inc
DBD	dbd

Procedure

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 data definition 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 or include files, you can validate the data definitions one at a time. In the appropriate folder for the file type, such as COBOL Copybook or PL/I Include File, right-click a data definition file and select **Validate <*file type*>**. If Classic Data Architect detects errors, an error dialog directs you to the Problems view, which is located on a tab in the lower right quarter of the window. Double click an error to open the file in the Editor with the cursor positioned on the line containing the error.

Granting privileges for performing actions on data servers

If you want change data capture 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

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:

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

   $IMS Allows grantees to create, drop, and view IMS tables and views 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:
   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 will 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

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 will 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.

Creating IMS tables and views

You must create a relational table that maps to the IMS database. You can also create a view on the table to filter record types or to filter rows and columns. Use the New IMS Table wizard to create the table and optionally the view.

Before you begin

- Configure the data server where you plan to run the capture service that will process change data from your IMS database.
- Create a metadata catalog.
- Decide which segment you want 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.
- Ensure that the **IMS DBDs** folder in your project has a database definition file (DBD) that lists the segments from which you want to select the fields to map to columns.
- Ensure that a COBOL copybook or (on the Windows platform) PL/I include file for each IMS segment you want to map to is in the appropriate folder in your data design project.

  COBOL copybooks are stored in the **COBOL Copybooks** folder, and PL/I include files are stored in the **PL/I Include Files** folder.

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

1. Map your IMS database to a relational table and optionally 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 that you want to base the table on.
   c. Specify that the table is for change capture.
   d. Choose to use the table for updates.
   e. Choose whether to create a view on the table.
   f. Provide information about how to access the IMS database.
   g. For the each segment that is in the path, specify a COBOL copybook or include file, select the desired 01 level if there is more than one, and then select the elements that you want to map as columns.
   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, the view also appears under the selected schema.

2. Modify the table properties to add a primary key and optionally add privileges. Select the table and make any changes in the Properties view. Find the primary key specification in the **Columns** tab of 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 97.
   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. You can also choose to generate DDL for indexes.
      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 Data Source 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 97.
   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 IMS database. Be sure that the data server is connected to that database.
   a. In the Data Source 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.

Modifying the PCB selection for IMS tables

You can modify the PCB selection for an IMS table for refresh processing.

Procedure

1. Open the Modify PCB Selection wizard by right-clicking the IMS table and selecting **Modify PCB Selection Method**.
2. Select how you want Classic to select the PCBs that are needed to access the table.

Altering tables to support change data capture

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

Before you begin

Restrictions

- You cannot alter a table for change data capture if the table contains record arrays.
- A primary key must be defined.
- 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 data set cannot reference an alternate index.
  - If you want to capture changes from native VSAM, the VSAM data set 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 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 Data Source 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.

Mapping tables and views for redefined data

Redefined data uses alternate record layouts for the same storage area, based on record types.

About this task

To read redefined data, define a table with an associated view for each type of record. To insert, update, delete, or capture changes to redefined data, define a separate table for each record type.

Each table you map for redefined data must contain columns that identify common key information and a column for the type code field. These columns are followed by type-specific columns.

In the following PL/I example, the UNION attribute defines two alternative record mappings for the same storage area. The RECORD_TYPE variable specifies whether the data that follows describes employee information or address information.

```
DCL  01  EMPLOYEE_ADDRESS_RECORD,
         05 EMP_ID           CHAR(6),
         05 RECORD_TYPE      CHAR(1),
         05 RECORD_INFO UNION,
            10 EMPLOYEE_INFORMATION,
              15 LAST_NAME      CHAR(20),
              15 FIRST_NAME     CHAR(20),
              15 DATE_OF_BIRTH  PIC '(8)9',
              15 MONTHLY_SALARY DECIMAL(7,2),
              15 FILLER         CHAR(48),
            10 ADDRESS_INFORMATION,
              15 ADDRESS_LINE_1 CHAR(30),
              15 ADDRESS_LINE_2 CHAR(30),
              15 ADDRESS_CITY   CHAR(20),
              15 ADDRESS_STATE  CHAR(2),
              15 ADDRESS_ZIP    PIC '(5)9';
```

Procedure

1. Map two tables, each with an associated view.

   In the example, you define one base table and view for employee information, and another base table and view for address information.
2. Use the view or the table.
   a. To query redefined data, supply filtering information in the view definition when you map the table. This approach simplifies queries in client applications, because the queries do not require WHERE clause filtering.
   b. To insert, update, or delete redefined data, your application must use the base table name and provide WHERE filtering.

      Supply a WHERE clause and use the type code value to filter the records.

What to do next

Tip: To perform change capture on redefined data, mark the view for change capture when you map the table.

For additional examples of working with redefined data, see Record types in data definition examples.

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 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.

Before you begin

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

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 table 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 table that is 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 Data Source 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 Data Source 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.

Before you begin

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
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 table that is 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 Data Source 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 Data Source 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.

Before you begin

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
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 Data Source 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 Data Source 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 object properties

You can view and modify the properties of the different objects that you create in Classic Data Architect for InfoSphere Classic CDC for z/OS. You can also change the selection of columns in 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**.

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 all arrays the column belongs to.

Documentation
Lets you add comments to a column.

Database properties:

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

You can view settings in the Data Source Explorer and modify them in a physical data model in the Data Project Explorer. Select the icon for the database and properties that you want to manage.

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 V11.3.

Publishing Target page

Not applicable for change data capture.

Privileges page

This page lists the role-based privileges that apply to different grantees. The following information is provided for each grantee:

Type The catalog database object to which you are applying security, such as SYSTEM, $IMS or $VSAM.

Privilege
The ID associated with the authority level that you are granting, such as DISPLAY, SYSADM, or DBADM.

Grantable
If checked, indicates that the grantee can grant the privilege to others.

Grantor
The ID of the user or role that grants the privilege for the specified type.

Revoke
If checked, indicates that the grantee can revoke the privilege from others.

Documentation page

Optionally, this page provides additional information about the database.

Annotation page

This page, which is found only in the Data Project Explorer, enables you to add property and value pairs.

Table properties:

Table properties are shown in the Properties view within Classic Data Architect. 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
Name	Type the name of the table.
Schema	Displays the schema in the two-part name for the table.
Source DBMS	Displays the type of DBMS in which the source data is located.
Change capture	**Changes** Select this value for change data capture.

Columns page
Lists the columns of the table.

Source information page
The Source information page contains source information for:

Table 18. Source information for IMS

Property	Description
DBD name	Displays the name of the IMS DBD (database definition) that the table references.
DBD type	Displays the name of the IMS DBD (database definition) that the table references.
Leaf segment	Displays the name of the leaf segment.
Index root	Not applicable for change data capture.
IMS subsystem	Not applicable for change data capture.

Table 18. Source information for IMS (continued)

Property	Description
PSB name	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.
Join PSB name	Not applicable for change data capture.

Source columns page
Lists the columns in the table.

Path information page
Not applicable for IMS change data capture.

Source elements page
Not applicable for IMS change data capture.

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

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

PCB selection page
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 of these criteria:

- The view references only one table.
- The view references all of the columns in the base 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 data definition files

Use the Append Column wizard to add or replace columns in user tables that are based on data definition files, such as COBOL copybooks or PL/I include files.

Before you begin

The data definition file that contains the columns that you want to use must be in the appropriate folder in your project, for example, the **COBOL Copybooks** folder.

About this task

You can use columns from the data definition file on which the table is based, or you can use columns from a different file.

Procedure

1. Right-click the table and select **Modify Table** > **Update Columns**.
2. On page one of the wizard, select the data definition file that contains the columns that you want to use.

   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.

Populating metadata catalogs

Classic Data Architect can generate the SQL DDL statements that describe the tables, views, 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

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.

Procedure

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.

Results

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

Data type support in SQL operations

The following topics detail data type support in SQL operations in InfoSphere Classic CDC for z/OS.

Binary strings

A binary string is a sequence of bytes. The length of a binary string is the number of bytes in the sequence.

Binary strings are not associated with any CCSID. The supported binary string data types are BINARY and VARBINARY (BINARY VARYING).

The following subtypes of binary strings are supported:

Fixed-length binary strings
The type of fixed-length binary strings is BINARY. When fixed-length binary string distinct types, columns, and variables are defined, the length attribute is specified, and all values have the same length. For a fixed-length binary string, the length attribute must be between 1 and 32704 inclusive.

Varying-length binary strings
All values of varying-length string columns have the same maximum length, which is determined by the length attribute.

The varying-length binary string is VARBINARY (BINARY VARYING).

When varying-length binary strings, distinct types, columns, and variables are defined, the maximum length is specified and this length becomes the length attribute. Actual length values might have a smaller value than the length attribute value. For varying-length binary strings, the actual length specifies the number of bytes in the string.

For a VARBINARY string, the length attribute must be between 1 and 32704. Like a varying-length character string, varying-length binary string could be an empty string.

A binary string column is useful for storing non-character data, such as encoded or compressed data, pictures, voice, and mixed media. Another use is to hold structured data for exploitation by distinct types, user-defined functions, and stored procedures.

COBOL data types supported by the data server

When you map user tables in the metadata catalogs on the data server, Classic Data Architect can convert COBOL data types to native types that the server supports and their corresponding SQL types.

Data types in a CREATE TABLE statement

When you map a new table, you can import data definitions in COBOL copybooks that describe the layout of the source database and the types associated with the data items. The information in the following table shows how Classic Data Architect converts COBOL data types to native and SQL types.

When you create a new user table in the metadata catalogs on a data server, you run a CREATE TABLE statement. Typically, this statement is found within Data Definition Language (DDL) that Classic Data Architect creates automatically when you run the Generate DDL wizard. The CREATE TABLE statement assigns a native data type to each column by using a DATATYPE keyword and assigns an SQL data type in a USE AS clause.

You can also use the information shown here to override the data types that are assigned to columns in a user table. One approach is to edit the DDL in an SQL Editor window. With VSAM or CICS VSAM sources, you can also change the native data type in the Classic Column Info properties window. Data types that you assign manually must be appropriate for the data in the source database.

Supported COBOL data types

The following table describes COBOL data types that the data server supports. The table also provides information about native and SQL types that correspond with each COBOL type.

COBOL element type	COBOL usage	Sign	Native data type	Length	SQL data type	Example
COBOLAlphabeticType COBOLAlphaNumeric Type			C	<= 254	CHAR scale = 0	PIC X(4)
COBOLNumericType	Packed decimal	T	P		DECIMAL	PIC S9(3) COMP-3
COBOLNumericType	Packed decimal	F	UP		DECIMAL	PIC 9(3) COMP-3
COBOLNumericType	Display	T	C		DECIMAL	PIC S99
COBOLNumericType	Display	F	UC		CHAR scale = 0 DECIMAL scale > 0	PIC 99

COBOL element type	COBOL usage	Sign	Native data type	Length	SQL data type	Example
COBOLNumericType	Binary	T	H		SMALLINT	PIC S99 COMP-4
COBOLNumericType	Binary	F	UH		SMALLINT	PIC 99 COMP-4
COBOLNumericType	Binary	T	F		INTEGER	PIC S9(8) COMP-4
COBOLNumericType	Binary	F	UF		INTEGER	PIC 9(8) COMP-4
COBOLNumericType	Binary		D		DECIMAL	PIC 9(18) COMP-4
COBOLInternalFloatType	Float		F		REAL	COMP-1
COBOLInternalFloatType	Double		D		DOUBLE	COMP-2
COBOLDBCSType	DBCS		C	<= 127	GRAPHIC	PIC G (or PIC N if NSymbol parser setting to DBCS)
COBOLAlphaNumeric EditedType COBOLNumericEdited Type COBOLExternalFloat Type			C	<= 254	CHAR	PIC clause is a display format that might contain B,P,Z,.,+,-,CR,DB,E

Native data types

The following table describes Classic data types and their equivalent SQL types. A CREATE TABLE statement specifies the native data type by using the DATATYPE parameter.

The Other SQL data types column lists alternative SQL types that you can specify as overrides by editing DDL in the SQL Editor window.

Table 19. Native data types

DATATYPE value	Contents	Standard SQL data type	Other SQL data types
C	Mixed mode character data. When the SQL data type is DECIMAL, the data is assumed to consist wholly of numbers with the right most number identifying the sign.	CHAR DECIMAL	DECIMAL, VARCHAR, GRAPHIC, or VARGRAPHIC, BINARY, VARBINARY
P	Packed decimal data where the sign is stored in the far right aggregation of four bits.	DECIMAL	BINARY

Table 19. Native data types (continued)

DATATYPE value	Contents	Standard SQL data type	Other SQL data types
D[1]	Floating point data. The columns length or precision determines whether the SQL data type is DOUBLE PRECISION or DECIMAL. 64-bit data is mapped as DECIMAL.	DOUBLE PRECISION	FLOAT(*precision*) DECIMAL, BINARY
F[2]	32-bit signed binary value where the sign is in the high order bit.	DECIMAL INTEGER REAL	FLOAT(*precision*) BINARY
H	16-bit signed binary value where the sign is in the high order bit.	DECIMAL SMALLINT	BINARY
V	Variable mixed-mode character data, where the actual data is preceded by a 16-bit signed binary number that identifies the actual length of the data.	VARCHAR	LONG VARCHAR, VARGRAPHIC, or LONG VARGRAPHIC, VARBINARY
UC	Unsigned zoned-decimal data where the last character does not identify the sign. The value is always a positive value.	DECIMAL	CHAR, BINARY, VARBINARY
UP	Packed decimal data where the sign bit is ignored. The value is always positive.	DECIMAL	BINARY
UF	Unsigned 32-bit binary value.	INTEGER	BINARY
UH	Unsigned 16-bit binary value.	SMALLINT	BINARY
B[3]	Fixed length binary data.	BINARY	DECIMAL, INTEGER, SMALLINT, VARBINARY
VB[3]	Variable length binary data, where the actual data is preceded by a 16-bit signed binary number that identifies the actual length of the data.	VARBINARY	N/A

Table 19. Native data types (continued)

DATATYPE value	Contents	Standard SQL data type	Other SQL data types

[1]The SQL data type is DOUBLE PRECISION or FLOAT. DOUBLE PRECISION is shorthand for an 8-byte floating point number. In the USE AS clause, you can identify this data type as DOUBLE PRECISION or FLOAT(precision). If the precision value is in the range 22–53, the column represents an 8-byte floating point number. If you assign FLOAT, specify the maximum precision. The column length is based on the precision.

[2]The SQL data type is REAL or FLOAT. REAL is shorthand for a 4-byte floating point number. In the USE AS clause, you can identify this data type as REAL or FLOAT(precision). If the precision value is in the range 1–21, the column represents a 4-byte floating point number. If you assign FLOAT, specify the maximum precision. The column length is based on the precision.

[3] Although Cobol does not support native binary types, you can manually edit the DDL to use binary data types.

PL/I data types supported by the data server

When you map user tables in the metadata catalogs on the data server, Classic Data Architect can convert PL/I data types to native types that the server supports and their corresponding SQL types.

Data types in a CREATE TABLE statement

When you map a new table, you can import data definitions in PL/I include files that describe the layout of the source database and the types associated with the data items. The information in the following tables shows how the data server converts PL/I data types to native and SQL types.

When you create a new user table in the metadata catalogs on a data server, you run a CREATE TABLE statement. Typically, this statement is found in the Data Definition Language (DDL) that Classic Data Architect creates automatically when you run the Generate DDL wizard. The CREATE TABLE statement assigns a native data type to each column using a DATATYPE keyword and assigns an SQL data type in a USE AS clause.

You can also use the information shown here to modify the data types that are assigned to columns in a user table. One approach is to edit the DDL in an SQL Editor window. With VSAM, CICS VSAM, CA-Datacom, or sequential data sources, you can also change the native data type in the Classic Column Info properties window. Data types that you assign manually must be appropriate for the data in the source database.

Supported PL/I data types

The following table describes PL/I data types that the data server supports. The table also provides information about native and SQL types that correspond with PL/I.

Computational data type	Precision and range	Scale	Native data type	SQL data type
CHAR(n)	n <= 254		C	CHAR(n)
CHAR(n) VARYING	n <= 32704		V	VARCHAR(n)

Computational data type	Precision and range	Scale	Native data type	SQL data type
GRAPHIC(n)	n <= 127		C	GRAPHIC(n)
GRAPHIC(n) VARYING	n <= 16352		V	VARGRAPHIC(n)
PICTURE (non-numeric)	size <= 254		C	CHAR(n)
PICTURE (V,9)[1]			UC	DECIMAL(p,q)
PICTURE (all other numeric formats)[1]	size <= 254		C	CHAR(n)
BIN FIXED(p,q) UNSIGNED	9 <= p <= 16	q=0	UH	SMALLINT
BIN FIXED(p,q) UNSIGNED	17 <= p <= 32	q=0	UF	INTEGER
BIN FIXED(p,q) UNSIGNED	33 <= p <= 64	q=0	D	DECIMAL(21,0)
BIN FIXED(p,q) SIGNED[2]	8 <= p <= 15	q=0	H	SMALLINT
BIN FIXED(p,q) SIGNED[2]	16 <= p <= 32	q=0	F	INTEGER
BIN FIXED(p,q) SIGNED[2]	32 <= p <= 63	q=0	D	DECIMAL(19,0)
DECIMAL FIXED(p,q)		q>=0	P	DECIMAL(p,q)
DECIMAL FLOAT(p)	1 <= p <= 6		F	REAL
DECIMAL FLOAT(p)	7 <= p <= 16		D	DOUBLE
BIN FLOAT(p)	1 <= p <= 21		F	REAL
BIN FLOAT(p)	22 <= p <= 53		D	DOUBLE

[1]PL/I supports many different types of arithmetic character data, such as 9, V, S, +, -, Z, /, R, CR, DB, T, R, K, and F. Values that contain character strings are typically used for report formatting purposes, and the data server maps them as character data. With PL/I picture strings, S, +, and - denote a character, not a sign bit. The data server does not support them as DECIMAL.

[2]The SIGNED attribute is the default.

Native data types

The following table describes the contents of native data types and their equivalent SQL types. A CREATE TABLE statement specifies the native data type by using the DATATYPE parameter.

The Other SQL data types column lists alternative SQL data types that you can specify as overrides by editing DDL in the SQL Editor window.

Table 20. Native data types

DATATYPE value	Contents	Standard SQL data type	Other SQL data types
C	Mixed mode character data. When the SQL data type is DECIMAL, the data is assumed to consist wholly of numbers with the right most number identifying the sign.	CHAR DECIMAL	DECIMAL, VARCHAR, GRAPHIC, or VARGRAPHIC, BINARY, VARBINARY
P	Packed decimal data where the sign is stored in the far right aggregation of four bits.	DECIMAL	BINARY
D[1]	Floating point data. The columns length or precision determines whether the SQL data type is DOUBLE PRECISION or DECIMAL. 64-bit data is mapped as DECIMAL.	DOUBLE PRECISION	FLOAT(*precision*) DECIMAL, BINARY
F[2]	32-bit signed binary value where the sign is in the high order bit.	INTEGER REAL	FLOAT(*precision*) BINARY
H	16-bit signed binary value where the sign is in the high order bit.	SMALLINT	BINARY
V	Variable mixed-mode character data, where the actual data is preceded by a 16-bit signed binary number that identifies the actual length of the data.	VARCHAR	LONG VARCHAR, VARGRAPHIC, or LONG VARGRAPHIC, VARBINARY
UC	Unsigned zoned-decimal data where the last character does not identify the sign. The value is always a positive value.	DECIMAL	CHAR, BINARY, VARBINARY
UP	Packed decimal data where the sign bit is ignored. The value is always positive.	DECIMAL	BINARY
UF	Unsigned 32-bit binary value.	INTEGER	BINARY

Table 20. Native data types (continued)

DATATYPE value	Contents	Standard SQL data type	Other SQL data types
UH	Unsigned 16-bit binary value.	SMALLINT	BINARY
B[3]	Fixed length binary data.	BINARY	DECIMAL, INTEGER, SMALLINT, VARBINARY
VB[3]	Variable length binary data, where the actual data is preceded by a 16-bit signed binary number that identifies the actual length of the data.	VARBINARY	N/A

[1]The SQL data type is DOUBLE PRECISION or FLOAT. DOUBLE PRECISION is shorthand for an 8-byte floating point number. In the USE AS clause, you can identify this data type as DOUBLE PRECISION or FLOAT(precision). If the precision value is in the range 22–53, the column represents an 8-byte floating point number. If you assign FLOAT, specify the maximum precision. The column length is based on the precision.

[2]The SQL data type is REAL or FLOAT. REAL is shorthand for a 4-byte floating point number. In the USE AS clause, you can identify this data type as REAL or FLOAT(precision). If the precision value is in the range 1–21, the column represents a 4-byte floating point number. If you assign FLOAT, specify the maximum precision. The column length is based on the precision.

[3] Although PL/I does not support native binary types, you can manually edit the DDL to use binary data types.

Constants

A constant (also called a literal) specifies a value. Constants are classified as string constants or numeric constants. Numeric constants are further classified as integer, floating-point, or decimal. String constants are classified as character or graphic. All constants have the attribute NOT NULL. A negative sign in a numeric constant with a value of zero is ignored.

The types of constants are as follows:

Integer constants
Specifies a binary integer as a signed or unsigned number that has a maximum of 10 significant digits and no decimal point. If the value is not within the range of a large integer, the constant is interpreted as a decimal constant. The data type of an integer constant is large integer.

In syntax diagrams, the term *integer* is used for an integer constant that must not include a sign.

Floating-point constants
Specifies a floating-point number as two numbers separated by an E. The first number can include a sign and a decimal point. The second number can include a sign but not a decimal point. The value of the constant is the product of the first number and the power of 10 that is specified by the second number. The value must be within the range of floating-point numbers. The number of characters in the constant must not exceed 30. Excluding leading zeros, the number of digits in the first number must not

exceed 17, and the number of digits in the second must not exceed 2. The data type of a floating-point constant is double precision floating-point.

```
15E1     2.E5    -2.2E-1     +5.E+2
```

Figure 3. Floating-point constants that represent the numbers 150, 200000, -0.22, and 500

Decimal constants

Specifies a decimal number as a signed or unsigned number of no more than 31 digits and either includes a decimal point or is not within the range of binary integers. The precision is the total number of digits, including any to the right of the decimal point. The total includes all leading and trailing zeros. The scale is the number of digits to the right of the decimal point, including trailing zeros.

```
025.50   1000.   -15.   +37589333333333333333.33
```

Figure 4. Decimal constants that have precisions and scales of 5 and 2, 4 and 0, 2 and 0, 23 and 2

Binary string constants

Specifies a varying-length binary string. The binary constant is valid for columns BINARY or VARBINARY data types.

A binary-string constant is formed by specifying a BX followed by a sequence of characters that starts and ends with a string delimiter. The characters between the string delimiters must be an even number of hexadecimal digits. The number of hexadecimal digits must not exceed 254.

A hexadecimal digit is a digit or any of the letters A through F (upper case or lower case). Under the conventions of hexadecimal notation, each pair of hexadecimal digits represents one byte. Note that this representation is similar to the representation of the character-constant that uses the X'' form; however binary-string constant and character-string constant are not compatible and the X'' form can not be used to specify a binary-string constant, just as the BX'' form cannot be used to specify a character-string constant.

Examples of binary-string constants:

```
BX'0000'    BX'C141C242'     BX'FF00FF01FF'
```

Character string constants

Specifies a varying-length character string. Character string constants have these forms:

A sequence of characters that starts and ends with an apostrophe (').

Specifies the character string that is contained between the string delimiters. The number of bytes between the delimiters must not be greater than 255. Two consecutive string delimiters are used to represent one string delimiter within the character string.

An X followed by a sequence of characters that starts and ends with a string delimiter.

Also called a hexadecimal constant. The characters between the string delimiters must be an even number of hexadecimal digits. The number of hexadecimal digits must not exceed 254. A hexadecimal digit is a digit or any of the letters A through F (uppercase or lowercase). Under the conventions of hexadecimal notation, each pair of hexadecimal digits represents a character. A hexadecimal constant allows you to specify characters that do not have a keyboard representation.

```
'12/14/1985'    '32'    'DON''T CHANGE'    X'FFFF'    ''
```

Figure 5. Examples of character string constants

The last string in the example ('') represents an empty character string constant, which is a string of zero length.

A character string constant is classified as mixed data if it includes a DBCS substring. In all other cases, a character string constant is classified as SBCS data.

Graphic string constants
Specifies a varying-length graphic string.

In EBCDIC environments, the forms of graphic string constants are:
- G'0x0e[dbcs-string]0x0f'
- N'0x0e[dbcs-string]0x0f'

In SQL statements, graphic string constants cannot be continued from one line to the next. The maximum number of DBCS characters in a graphic string constant is 124.

Columns for IMS

You can define a column that references an IMS database. For IMS column definitions, you must also identify the name of the segment where the column resides.

Syntax

```
►►──column-name──SOURCE DEFINITION ENTRY──segment-name──┬────────────────┬──────────►
                                                        └─IMS-field-name─┘

►─DATAMAP OFFSET──relative-offset──LENGTH──length──┬────────────────────┬──────────►
                                                   └─DATATYPE───┬─C──┬──┘
                                                                ├─P──┤
                                                                ├─D──┤
                                                                ├─F──┤
                                                                ├─H──┤
                                                                ├─V──┤
                                                                ├─UC─┤
                                                                ├─UP─┤
                                                                ├─UH─┤
                                                                ├─UF─┤
                                                                ├─B──┤
                                                                └─VB─┘
```

```
►─USE AS──┬─CHAR──(──length──)─────────────┬─────────────────────────────────►
          ├─VARCHAR──(──length──)──────────┤
          ├─LONG VARCHAR───────────────────┤
          ├─GRAPHIC──(──length──)──────────┤
          ├─VARGRAPHIC──(──length──)───────┤
          ├─LONG VARGRAPHIC────────────────┤
          ├─INTEGER────────────────────────┤
          ├─SMALLINT───────────────────────┤
          ├─DECIMAL──(──precision, scale──)┤
          ├─DECIMAL──(──precision──)───────┤
          ├─FLOAT──(──precision──)─────────┤
          ├─REAL───────────────────────────┤
          ├─DOUBLE PRECISION───────────────┤
          ├─BINARY──(──length──)───────────┤
          └─VARBINARY──(──length──)────────┘


   ┌─WITHOUT CONVERSION───────────────────┐
►──┴─WITH CONVERSION──field_procedure_name─┴──┬────────────────────┬─────────►
                                              └─NULL IS──null_value─┘

►──┬─────────────┬──────────────────────────────────────────────────────────►◄
   └─PRIMARY KEY─┘
```

Parameters

column-name
: Identifies the name of the column.

Specifies the column name that is a long identifier. Column names cannot be qualified with a CREATE TABLE statement.

segment-name
: Identifies the segment where the column is located. The name is a short native identifier.

The name must also exist in the DBD specified in the CREATE TABLE statement.

DATAMAP
: Specifies the relative offset for a column. If a LENGTH keyword is specified, information about the length of the column is also defined.

OFFSET *relative-offset*

Follows the DATAMAP keyword to set the offset for the column.

The relative offset identifies the relative zero offset of the starting position of the column within the object that the column is associated with. For VSAM, the offset is generally measured from the start of the record. For IMS, the relative offset is measured from the start of a segment.

Note: Columns do not need to be defined in ascending relative offset starting sequence.

LENGTH *length*
: The LENGTH clause is required for IMS column definitions.

Specifies the length of the column. Generally, inconsistencies between the length that is specified using the LENGTH keyword and the length that is obtained from the SQL data type definition on the USE AS clause are ignored.

For native DECIMAL data types the LENGTH must match the computed physical length of the column, based on the precision and scale specified in the USE AS clause when the USE AS precision is non-zero. For USE AS DECIMAL definitions the LENGTH must match the computed physical length of the column when the scale is non-zero and greater than the precision. Additionally, differences between the *length* and the SQL length specification for VARCHAR and VARGRAPHIC data types identify how to interpret the length attribute for the varying length column. The following rules are applied for VARCHAR data types:

- If the *length* and the VARCHAR lengths are identical, then the control length is assumed to include the length (2 bytes) that is taken up by the control length component.
- If the *length* is two bytes greater than the VARCHAR length, then the control length component is assumed to identify the actual length of the data.

For a varying length graphic string, the same kind of conventions are used. However, the lengths are expressed in DBCS characters. Therefore the rules are as follows:

- If the *length* and the VARGRAPHIC lengths are identical, then the control length is assumed to include the length (1 DBCS character, which is 2 bytes) that is taken up by the control length of a component.
- If the *length* is one character greater (two bytes) than the VARGRAPHIC length, then the control length component is assumed to identify the actual length of the data in characters.

Generally, the USE AS length must match the *length* value specified. However, for VARCHAR, the *length* can be two bytes off. For VARGRAPHIC, the length can be different by one. For these data types, the differences do not represent a conflict.

DATATYPE

Identifies the native format of the column.

The following table identifies the basic native data types

Table 21. Native data types

DATATYPE value	Contents	Standard SQL data type	Other SQL data types
C	Mixed mode character data. When the SQL data type is DECIMAL, the data is assumed to consist wholly of numbers with the right most number identifying the sign.	CHAR DECIMAL	DECIMAL, VARCHAR, GRAPHIC, or VARGRAPHIC, BINARY, VARBINARY
P	Packed decimal data where the sign is stored in the far right aggregation of four bits.	DECIMAL	BINARY

Table 21. Native data types (continued)

DATATYPE value	Contents	Standard SQL data type	Other SQL data types
D[1]	Floating point data. The columns length or precision determines whether the SQL data type is DOUBLE PRECISION or DECIMAL. 64-bit data is mapped as DECIMAL.	DOUBLE PRECISION	FLOAT(*precision*) DECIMAL, BINARY
F[2]	32-bit signed binary value where the sign is in the high order bit.	INTEGER REAL	FLOAT(*precision*) BINARY
H	16-bit signed binary value where the sign is in the high order bit.	SMALLINT	BINARY
V	Variable mixed-mode character data, where the actual data is preceded by a 16-bit signed binary number that identifies the actual length of the data.	VARCHAR	LONG VARCHAR, VARGRAPHIC, or LONG VARGRAPHIC, VARBINARY
UC	Unsigned zoned-decimal data where the last character does not identify the sign. The value is always a positive value.	DECIMAL	CHAR, BINARY, VARBINARY
UP	Packed decimal data where the sign bit is ignored. The value is always positive.	DECIMAL	BINARY
UF	Unsigned 32-bit binary value.	INTEGER	BINARY
UH	Unsigned 16-bit binary value.	SMALLINT	BINARY
B[3]	Fixed length binary data.	BINARY	DECIMAL, INTEGER, SMALLINT, VARBINARY
VB[3]	Variable length binary data, where the actual data is preceded by a 16-bit signed binary number that identifies the actual length of the data.	VARBINARY	N/A

Table 21. Native data types (continued)

DATATYPE value	Contents	Standard SQL data type	Other SQL data types
[1]The SQL data type is DOUBLE PRECISION or FLOAT. DOUBLE PRECISION is shorthand for an 8-byte floating point number. In the USE AS clause, you can identify this data type as DOUBLE PRECISION or FLOAT(precision). If the precision value is in the range 22–53, the column represents an 8-byte floating point number. If you assign FLOAT, specify the maximum precision. The column length is based on the precision.			
[2]The SQL data type is REAL or FLOAT. REAL is shorthand for a 4-byte floating point number. In the USE AS clause, you can identify this data type as REAL or FLOAT(precision). If the precision value is in the range 1–21, the column represents a 4-byte floating point number. If you assign FLOAT, specify the maximum precision. The column length is based on the precision.			

If DATATYPE information is not specified, the native data type is synthesized based on the column of the SQL data type or from the database system.

The SQL data type information in the USE AS clause identifies the value in the SIGNED column in the following instances:

- One character code is supplied.
- No DATATYPE information is specified.
- Native data type information is not obtained from the database system.

USE AS

Identifies the SQL data type for the column.

The following table describes the data types for columns. The non-null SQLTYPE identifies the data type in internal control blocks and diagnostic trace information.

Table 22. SQL data type descriptions

Keyword identifier	Description	Maximum length	SQLTYPE
CHAR(length)	Fixed-length character string that contains mixed mode data.	254	452
VARCHAR(length)	Variable length character string that contains mixed mode data. A half-word length component precedes the character string and identifies the actual length of the data. The VARCHAR length does not include the length of the LENGTH filed.	32704	448
LONG VARCHAR	Long character string that contains mixed mode data. A half-word length component precedes the character string and identifies the actual length of the data. The LONG VARCHAR length does not include the length of the LENGTH field.	32704	456
GRAPHIC(length)	Fixed-length graphic string that is assumed to contain pure DBCS data without shift codes. The length is expressed in DBCS characters, and not bytes.	127	468

Table 22. SQL data type descriptions (continued)

Keyword identifier	Description	Maximum length	SQLTYPE
VARGRAPHIC(length)	Varying-length graphic string that is assumed to contain pure DBCS data without shift codes. A half-word length component precedes the graphic string and identifies the actual length of the data. The length is expressed in DBCS characters, and not bytes.	16352	464
LONG VARGRAPHIC	Long graphic string that is assumed to contain pure DBCS data without shift codes. A half-word length component precedes the graphic string and identifies the actual length of the data. The length is expressed in DBCS characters, and not bytes.	16352	472
INTEGER	Large integer.	Exactly 4	496
SMALLINT	Small integer.	Exactly 2	500
DECIMAL (precision,scale) or DECIMAL(precision)	Packed decimal data. A valid precision value is between 1 and 31. A valid scale value is between zero and the precision value.	31	484
FLOAT(precision)	Floating point number. Depending upon the precision, a column with a FLOAT data type takes on the attributes of a REAL or DOUBLE PRECISION data type. When precision is in the range of 1 to 21, the column is treated as a REAL. For precisions between 22 and 53, the column takes on DOUBLE PRECISION attributes. A precision of zero or greater than 53 is nonvalid.	4 or 8	480
REAL	A single-precision, floating-point number that is a 32-bit approximation of a real number. The number can be zero or can range from -3.40282347E+38 to -1.17549435E-38, or from 1.17549435E-38 to 3.40282347E+38.	4	480
DOUBLE PRECISION	A floating-point number that is a 64-bit approximation of a real number. The number can be zero or can range from -1.797693134862315E+308 to -2.225073858507201E-308, or from 2.225073858507201E-308 to 1.797693134862315E+308.	8	480
BINARY(length)	Fixed-length binary data.	32704	912
VARBINARY(length)	Variable length binary data, where the actual data is preceded by a 16-bit signed binary number that identifies the actual length of the data. The VARBINARY length does not include the length of the LENGTH filed.	32704	908

When you supply a USE AS LONG VARCHAR or USE AS LONG VARGRAPHIC clause, you do not specify a length for the column. The length is based on the physical attributes that are obtained about the object record or segment where the column is.

When you supply a USE AS VARCHAR clause with a length greater than 254, that column is changed into a LONG VARCHAR column with the specified length. The same behavior occurs for a USE AS VARGRAPHIC clause when the length is greater than 127.

WITHOUT CONVERSION
Specifies that a field procedure does not exist for the column.

WITH CONVERSION
Identifies the name of the load module for a field procedure, if a field procedure is required. The field procedure name is a short identifier.

NULL IS *null_value*
Specifies a value that identifies when the contents of the column is null. By default, the data in a column never contains a null value. This situation is true even for variable length character or graphic columns where the length component indicates that there is no data in the variable length column.

A null value is a character string that can specify up to 16 characters of data. Specifies the value in hexadecimal format. The total length of the string is 35 characters.

A column contains null values based on *null-value*. If the start of the column data matches the null value, then that instance of the column is null. For example, if a column is defined as CHAR(10) and a null value of x'4040' is specified, the null value length is 2. If the first 2 bytes of the column contain spaces, the column is also null.

PRIMARY KEY
Identifies the column to be one of the columns that constitute the primary key for the table. This form of primary key identification is mutually exclusive with specifying a primary key at the end of the CREATE TABLE statement.

You can identify a primary key at the column level when the columns that make up the primary key for the table are defined in ordinal sequence within the table definition.

Columns for VSAM

Column definitions are a part of CREATE TABLE statements. You use column definitions to define the columns in a table that references a VSAM file. There are no differences between a sequential column definition and the generic column definitions.

Syntax

```
►►──column-name──SOURCE DEFINITION──DATAMAP OFFSET──relative-offset──────────────►
```

```
►─LENGTH──length──┬────────────────────┬──────────────────────────────────────►
                  └─DATATYPE──┬─C──┬───┘
                              ├─P──┤
                              ├─D──┤
                              ├─F──┤
                              ├─H──┤
                              ├─V──┤
                              ├─UC─┤
                              ├─UP─┤
                              ├─UH─┤
                              ├─UF─┤
                              ├─B──┤
                              └─VB─┘

►─┬─────────────────────────────────────┬─────────────────────────────────────►
  ├─INDEX───────────────────────────────┤
  └─SECONDARY──┬─DD──DD name──────────┬─┘
               └─DD──dataset name─────┘

►─USE AS──┬─CHAR──(──length──)─────────────────┬──┬─────────────────────┬─────►
          ├─VARCHAR──(──length──)──────────────┤  └─USE RECORD LENGTH───┘
          ├─LONG VARCHAR───────────────────────┤
          ├─GRAPHIC──(──length──)──────────────┤
          ├─VARGRAPHIC──(──length──)───────────┤
          ├─LONG VARGRAPHIC────────────────────┤
          ├─INTEGER────────────────────────────┤
          ├─SMALLINT───────────────────────────┤
          ├─DECIMAL──(──precision, scale──)────┤
          ├─DECIMAL──(──precision──)───────────┤
          ├─FLOAT──(──precision──)─────────────┤
          ├─REAL───────────────────────────────┤
          ├─DOUBLE PRECISION───────────────────┤
          ├─BINARY──(──length──)───────────────┤
          └─VARBINARY──(──length──)────────────┘

  ┌─WITHOUT CONVERSION─────────────────────┐
►─┼────────────────────────────────────────┼──┬────────────────────┬──────────►
  └─WITH CONVERSION──field procedure name──┘  └─NULL IS──null value─┘

►─┬───────────────┬───────────────────────────────────────────────────────────►◄
  └─PRIMARY KEY───┘
```

Parameters

column_name
: Identifies the name of the column.

  Specifies the column name that is a long identifier. Column names cannot be qualified with a CREATE TABLE statement.

DATAMAP
: Specifies the relative offset for a column. If a LENGTH keyword is specified, information about the length of the column is also defined.

OFFSET *relative_offset*

  Follows the DATAMAP keyword to set the offset for the column.

The relative offset identifies the relative zero offset of the starting position of the column within the object that the column is associated with. For VSAM, the offset is generally measured from the start of the record. For IMS, the relative offset is measured from the start of a segment.

Note: Columns do not need to be defined in ascending relative offset starting sequence.

LENGTH *length*

Specifies the length of the column. Generally, inconsistencies between the length that is specified using the LENGTH keyword and the length that is obtained from the SQL data type definition on the USE AS clause are ignored. For native DECIMAL data types the LENGTH must match the computed physical length of the column, based on the precision and scale specified in the USE AS clause when the USE AS precision is non-zero. For USE AS DECIMAL definitions the LENGTH must match the computed physical length of the column when the scale is non-zero and greater than the precision.

Additionally, differences between the *length* and the SQL length specification for VARCHAR and VARGRAPHIC data types identify how to interpret the length attribute for the varying length column. The following rules are applied for VARCHAR data types:

- If the *length* and the VARCHAR lengths are identical, then the control length is assumed to include the length (2 bytes) that is taken up by the control length component.
- If the *length* is two bytes greater than the VARCHAR length, then the control length component is assumed to identify the actual length of the data.

For a varying length graphic string, the same kind of conventions are used. However, the lengths are expressed in DBCS characters. Therefore the rules are as follows:

- If the *length* and the VARGRAPHIC lengths are identical, then the control length is assumed to include the length (1 DBCS character, which is 2 bytes) that is taken up by the control length of a component.
- If the *length* is one character greater (two bytes) than the VARGRAPHIC length, then the control length component is assumed to identify the actual length of the data in characters.

Generally, the USE AS length must match the *length* value specified. However, for VARCHAR, the *length* can be two bytes off. For VARGRAPHIC, the length can be different by one. For these data types, the differences do not represent a conflict.

DATATYPE

Identifies the native format of the column.

The following table identifies the basic native data types

Table 23. Native data types

DATATYPE value	Contents	Standard SQL data type	Other SQL data types
C	Mixed mode character data. When the SQL data type is DECIMAL, the data is assumed to consist wholly of numbers with the right most number identifying the sign.	CHAR DECIMAL	DECIMAL, VARCHAR, GRAPHIC, or VARGRAPHIC, BINARY, VARBINARY
P	Packed decimal data where the sign is stored in the far right aggregation of four bits.	DECIMAL	BINARY
D[1]	Floating point data. The columns length or precision determines whether the SQL data type is DOUBLE PRECISION or DECIMAL. 64-bit data is mapped as DECIMAL.	DOUBLE PRECISION	FLOAT(*precision*) DECIMAL, BINARY
F[2]	32-bit signed binary value where the sign is in the high order bit.	INTEGER REAL	FLOAT(*precision*) BINARY
H	16-bit signed binary value where the sign is in the high order bit.	SMALLINT	BINARY
V	Variable mixed-mode character data, where the actual data is preceded by a 16-bit signed binary number that identifies the actual length of the data.	VARCHAR	LONG VARCHAR, VARGRAPHIC, or LONG VARGRAPHIC, VARBINARY
UC	Unsigned zoned-decimal data where the last character does not identify the sign. The value is always a positive value.	DECIMAL	CHAR, BINARY, VARBINARY
UP	Packed decimal data where the sign bit is ignored. The value is always positive.	DECIMAL	BINARY
UF	Unsigned 32-bit binary value.	INTEGER	BINARY

Table 23. Native data types (continued)

DATATYPE value	Contents	Standard SQL data type	Other SQL data types
UH	Unsigned 16-bit binary value.	SMALLINT	BINARY
B[3]	Fixed length binary data.	BINARY	DECIMAL, INTEGER, SMALLINT, VARBINARY
VB[3]	Variable length binary data, where the actual data is preceded by a 16-bit signed binary number that identifies the actual length of the data.	VARBINARY	N/A

[1]The SQL data type is DOUBLE PRECISION or FLOAT. DOUBLE PRECISION is shorthand for an 8-byte floating point number. In the USE AS clause, you can identify this data type as DOUBLE PRECISION or FLOAT(precision). If the precision value is in the range 22–53, the column represents an 8-byte floating point number. If you assign FLOAT, specify the maximum precision. The column length is based on the precision.

[2]The SQL data type is REAL or FLOAT. REAL is shorthand for a 4-byte floating point number. In the USE AS clause, you can identify this data type as REAL or FLOAT(precision). If the precision value is in the range 1–21, the column represents a 4-byte floating point number. If you assign FLOAT, specify the maximum precision. The column length is based on the precision.

If DATATYPE information is not specified, the native data type is synthesized based on the column of the SQL data type or from the database system.

The SQL data type information in the USE AS clause identifies the value in the SIGNED column in the following instances:

- One character code is supplied.
- No DATATYPE information is specified.
- Native data type information is not obtained from the database system.

USE AS

Identifies the SQL data type for the column.

The following table describes the data types for columns. The non-null SQLTYPE identifies the data type in internal control blocks and diagnostic trace information.

Table 24. SQL data type descriptions

Keyword identifier	Description	Maximum length	SQLTYPE
CHAR(length)	Fixed-length character string that contains mixed mode data.	254	452
VARCHAR(length)	Variable length character string that contains mixed mode data. A half-word length component precedes the character string and identifies the actual length of the data. The VARCHAR length does not include the length of the LENGTH filed.	32704	448

Table 24. SQL data type descriptions (continued)

Keyword identifier	Description	Maximum length	SQLTYPE
LONG VARCHAR	Long character string that contains mixed mode data. A half-word length component precedes the character string and identifies the actual length of the data. The LONG VARCHAR length does not include the length of the LENGTH field.	32704	456
GRAPHIC(length)	Fixed-length graphic string that is assumed to contain pure DBCS data without shift codes. The length is expressed in DBCS characters, and not bytes.	127	468
VARGRAPHIC(length)	Varying-length graphic string that is assumed to contain pure DBCS data without shift codes. A half-word length component precedes the graphic string and identifies the actual length of the data. The length is expressed in DBCS characters, and not bytes.	16352	464
LONG VARGRAPHIC	Long graphic string that is assumed to contain pure DBCS data without shift codes. A half-word length component precedes the graphic string and identifies the actual length of the data. The length is expressed in DBCS characters, and not bytes.	16352	472
INTEGER	Large integer.	Exactly 4	496
SMALLINT	Small integer.	Exactly 2	500
DECIMAL (precision,scale) or DECIMAL(precision)	Packed decimal data. A valid precision value is between 1 and 31. A valid scale value is between zero and the precision value.	31	484
FLOAT(precision)	Floating point number. Depending upon the precision, a column with a FLOAT data type takes on the attributes of a REAL or DOUBLE PRECISION data type. When precision is in the range of 1 to 21, the column is treated as a REAL. For precisions between 22 and 53, the column takes on DOUBLE PRECISION attributes. A precision of zero or greater than 53 is nonvalid.	4 or 8	480
REAL	A single-precision, floating-point number that is a 32-bit approximation of a real number. The number can be zero or can range from -3.40282347E+38 to -1.17549435E-38, or from 1.17549435E-38 to 3.40282347E+38.	4	480

Table 24. SQL data type descriptions (continued)

Keyword identifier	Description	Maximum length	SQLTYPE
DOUBLE PRECISION	A floating-point number that is a 64-bit approximation of a real number. The number can be zero or can range from -1.797693134862315E+308 to -2.225073858507201E-308, or from 2.225073858507201E-308 to 1.797693134862315E+308.	8	480
BINARY(length)	Fixed-length binary data.	32704	912
VARBINARY(length)	Variable length binary data, where the actual data is preceded by a 16-bit signed binary number that identifies the actual length of the data. The VARBINARY length does not include the length of the LENGTH filed.	32704	908

When you supply a USE AS LONG VARCHAR or USE AS LONG VARGRAPHIC clause, you do not specify a length for the column. The length is based on the physical attributes that are obtained about the object record or segment where the column is.

When you supply a USE AS VARCHAR clause with a length greater than 254, that column is changed into a LONG VARCHAR column with the specified length. The same behavior occurs for a USE AS VARGRAPHIC clause when the length is greater than 127.

USE RECORD LENGTH

The USE RECORD LENGTH clause is specified after the USE AS clause for a VARCHAR, LONG VARCHAR, VARGRAPHIC or LONG VARGRAPHIC data type specification.

Indicates the varying string or graphic column contains the entire contents of the record that the column is associated with. The entire record content contains character data. The actual data contents were not important to any client application that accessed one of these columns that ran on an MVS system where no code page conversion was performed.

WITHOUT CONVERSION
Specifies that a field procedure does not exist for the column.

WITH CONVERSION
Identifies the name of the load module for a field procedure, if a field procedure is required. The field procedure name is a short identifier.

NULL IS *null_value*
Specifies a value that identifies when the contents of the column is null. By default, the data in a column never contains a null value. This situation is true even for variable length character or graphic columns where the length component indicates that there is no data in the variable length column.

A null value is a character string that can specify up to 16 characters of data. Specifies the value in hexadecimal format. The total length of the string is 35 characters.

A column contains null values based on *null-value*. If the start of the column data matches the null value, then that instance of the column is null. For example, if a column is defined as CHAR(10) and a null value of x'4040' is specified, the null value length is 2. If the first 2 bytes of the column contain spaces, the column is also null.

PRIMARY KEY

Identifies the column to be one of the columns that constitute the primary key for the table. This form of primary key identification is mutually exclusive with specifying a primary key at the end of the CREATE TABLE statement.

You can identify a primary key at the column level when the columns that make up the primary key for the table are defined in ordinal sequence within the table definition.

KSDS or files that are accessed through an alternate index of columns that map to the key offset and length information, which are determined during validation, are good candidate columns to identify as primary keys. VSAM ESDS or RRDS files that are not accessed through an alternate index are always poor candidates for primary keys.
You can use the PRIMARY KEY clause on the column definition to identify the primary key columns when the keys are defined in the same order.

Chapter 9. Configuring publishing to WebSphere MQ

You can define subscriptions that contain changes that are written to WebSphere MQ in a delimited message format. Messages in a delimited format use specified characters as separators between fields.

InfoSphere Classic CDC publishes changes in a delimited format to WebSphere MQ. When the changes are published, the capture services connects to a z/OS queue manager or to a queue sharing group. The WebSphere MQ queue manager must run on the same z/OS image as the InfoSphere Classic CDC data server.

Each delimited format subscription must reference a definition of the required WebSphere MQ objects. The required objects include the subscription's own local or remote queue and a bookmark queue definition.

- Local or remote queues

  You need to direct the changes that are captured for a delimited format subscription to a local queue or to a remote queue.
  - Use a local queue for consumption by local applications or by a local instance of the WebSphere MQ Java™ Messaging Service. Sample member *USERHLQ*.USERSAMP (CECPSUBQ) is provided for the bookmark queue definition.
  - Use a remote queue to target consumption of applications that run on another operating system.
- Bookmark queues

  The *bookmark queue* is another local queue that stores restart information for the subscriptions that target WebSphere MQ. Sample member *USERHLQ*.USERSAMP (CECPBKLQ) is provided for the bookmark queue definition.

You must ensure that the necessary authorizations are assigned to the source server and administrator for working with MQ objects and running sample jobs.

Defining a bookmark queue

The bookmark queue stores restart information for subscriptions that target WebSphere MQ.

About this task

You define a bookmark queue by using sample member *USERHLQ*.USERSAMP (CECPBKLQ). The sample member contains a DEFINE QLOCAL definition that specifies the attributes to define the bookmark queue that are required for delimited publishing. The sample member also contains JCL to run the WebSphere MQ CSQUTIL program and uses the tailored definition as input.

During the installation customization process sample CECPBKLQ is tailored based on the values provided for delimited publishing in the installation customization parameters file *USERHLQ*.USERSAMP(CECCUSPC).

Procedure

- Specify the BOOKMARKQUEUE configuration parameter for the capture service. Define the bookmark queue as a local queue that includes the following attributes:

  DEFPSIST(YES)
  : Specifies that messages are logged and survive a restart of the queue manager.

  GET(ENABLED)
  : Allows the capture service to get messages from the queue.

  INDXTYPE(MSGID)
  : Instructs the queue manager to maintain an index based on message identifiers.

  PUT(ENABLED)
  : Allows the capture service to put messages on the queue.

  DEFSOPT(SHARED)
  : Specifies that, by default, multiple reader tasks can read the queue concurrently.

  SHARE
  : Identifies to the WebSphere MQ queue manager that multiple reader tasks can read this queue concurrently.
- Submit *USERHLQ*.USERSAMP (CECPBKLQ) after it is tailored.
- Ensure that you have the authority to run the WebSphere MQ CSQUTIL utility program to define queues.

Defining a local queue

You can write the changes captured for subscriptions that target WebSphere MQ to a local queue.

About this task

You define a local queue by using sample member *USERHLQ*.USERSAMP (CECPSUBQ). The sample member contains a DEFINE QLOCAL definition that specifies the attributes to define the local queue that are required for delimited publishing. The sample member also contains JCL to run the WebSphere MQ CSQUTIL program and uses the tailored definition as input.

During the installation customization process sample CECPSUBQ is tailored based on the values provided for delimited publishing in the installation customization parameters file *USERHLQ*.USERSAMP(CECCUSPC).

Procedure

- Define a local queue that includes the following attributes:

  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.

  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 source server to put messages on the queue.

SHARE
Identifies to the WebSphere MQ queue manager that multiple reader tasks can read this queue concurrently.

- Submit *USERHLQ*.USERSAMP (CECPSUBQ) after it is tailored.
- Ensure that you have the authority to run the WebSphere MQ CSQUTIL utility program to define queues.

Configuring WebSphere MQ objects for publishing to remote destinations

You can direct messages for delimited format subscriptions to destinations that are remote from InfoSphere Classic CDC.

Defining a remote queue

You can define an environment in which the changes captured for a delimited format subscription are sent to another queue manager for processing.

About this task

No samples are provided for the WebSphere MQ definition of this kind of configuration. You need to define the objects that the following describes.

Procedure

- The subscription needs to reference a remote queue definition with the following attributes:

  DEFPSIST(YES)
  Specifies that messages are logged and survive a restart of the queue manager.

  PUT(ENABLED)
  Allows the capture service to put messages on the queue.
- The z/OS WebSphere MQ queue manager needs at least one transmission queue which requires the following attributes:

  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.
- SYSTEM.CHANNEL.INITQ transmission queue.
- 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.

CONNAME(*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 25. 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 Classic CDC for z/OS source server.

    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.

  CONNAME(*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.

- Ensure that you have the authority to run the WebSphere MQ CSQUTIL utility program to define queues.

Configuring direct access by a remote client application

A remote client application can directly access the changes for a delimited format subscription by connecting to the z/OS queue manager.

About this task

No samples are provided for the WebSphere MQ definition of this kind of configuration. To set up direct access from a remote client application, you need to define a local queue for the subscription to use and you need the objects that the following procedure describes.

Procedure

- Define a local queue for the subscription. See “Defining a local queue” on page 122.
- Define the following objects at the WebSphere MQ server:
  - 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 the following values:

    CHLTYPE(CLNTCOMM)
    : Specifies that the MQI channel is a client-connection channel.

CONNAME(*string*)
There are many possible values for this attribute. See the WebSphere MQ command documentation for details.

- Define 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.

    CONNAME(*string*)
    There are many possible values for this attribute. See See the WebSphere MQ command documentation for details.
  - 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.
- Ensure that you have the authority to run the WebSphere MQ CSQUTIL utility program to define queues.

Configuring a remote WebSphere MQ Java Messaging Service as a WebSphere MQ client

You can configure a remote WebSphere MQ Java Messaging service to act as a WebSphere MQ client.

No samples are provided for the WebSphere MQ definition of this kind of configuration. See the WebSphere MQ documentation about using Java.

WebSphere MQ authorization requirements

The user ID associated with the source server and administrator require authorization privileges in WebSphere MQ.

Procedure

- Authorize user IDs associated with the source server to perform the following actions:
  - Connect to the queue manager (MQCONN or MQCONNX) on the system where the source server runs.
  - Commit messages (MQCMIT)
  - Roll back messages (MQBACK)
  - Open (MQOPEN)
  - Inquire about attributes (MQINQ)
  - Put messages (MQPUT)
  - Get messages (MQGET)
- Authorize an administrator to perform the following actions:
  - Run sample member *USERHLQ*.USERSAMP CECPBKLQ to define the bookmark queue (CECPBKLQ).
  - Run sample member *USERHLQ*.USERSAMP CECPSUBQ to define a local queue that a subscription can be reference (CECPSUBQ).
  - Use the command server queue (SYSTEM.COMMAND)
  - Use the WebSphere MQ commands that you want to issue.

Chapter 10. Configuring replication in Management Console

You can use Management Console to complete the replication configuration and to administer and monitor replication.

In Management Console, you will need to complete the following tasks:

- Create datastores for the source and target data sources

  When publishing changes in a delimited format to WebSphere MQ, you specify the source server as the target server when you define a new subscription. You do not need to create a target data source for delimited format subscriptions.
- Set up and configure subscriptions
- Map tables and columns

Only the relevant information will be displayed in the interface and only the eligible features and options will be available for use.

After configuring replication, you are able to perform the following actions in Management Console:

- Start replication
- Stop replication
- Monitor replication

Be aware that the information, features, and options displayed in the Management Console interface is determined by several factors:

- Your role as a user
- The datastores to which you have access
- The type of the datastore
- The version of the datastore

Only the functionality that is applicable to InfoSphere Classic CDC for z/OS will be enabled. Features that are not visible or enabled on menus are not available to InfoSphere Classic CDC for z/OS datastores.

For more information, see the *Management Console Administration Guide*.

Chapter 11. After you install and configure InfoSphere Classic CDC for z/OS

Once you have installed and configured InfoSphere Classic CDC for z/OS, you can start using it.

In this section, you will learn:

Connecting a source data server to a target engine

Before you connect the source and target, ensure you have set up a source server and used Classic Data Architect to map tables.

Note: When publishing changes in a delimited format to WebSphere MQ, you specify the source server as the target server when you define a new subscription. You do not need to connect a source data server to a target engine for delimited format subscriptions.

See also:

To connect a source data server to a target engine

Procedure

1. Start Management Console and log into Access Server.
2. Create a subscription between a Classic source server and your InfoSphere CDC target engine. You can create multiple subscriptions, each with a different target engine.

Starting a data server

When you start a data server, you start those services that are configured to start in the configuration file for the server.

About this task

You can start Classic data servers using either of the methods listed.

Note:

- For IMS data sources, you can run a source server despite whether IMS is running on the source LPAR.
- For VSAM data sources, you can run a source server on another LPAR if the replication log is backed by a coupling facility. If the replication log is backed by DASD, all address spaces that access the replication log stream must be on the same system.

Procedure

Start a data server by using one of the following methods:

- Issue a console command to start the JCL procedure for the data server.

  ```
  S procname
  ```

procname
: 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 the JCL as a batch job.

Stopping a data server

Stopping a data server stops all of the services that are running within it.

Procedure

Stop a data server by using one of the following methods:

- Issue the following command in an MTO interface:

  ```
  F name,STOP,ALL
  ```

  name Indicates the name of the started task or batch job of the data server.

  This form of the stop command requests a controlled shutdown of the data server. The controlled shutdown waits for all connected users to disconnect, ends replication for all actively replicating subscriptions, does an orderly shutdown of all services, and stops the source server. For persistent subscriptions, replication automatically restarts when the source server is restarted.
- Issue the z/OS **STOP** command:

  ```
  STOP procname
  ```

  procname
  : Indicates the 1-8 character PROCLIB member name that represents the server that you want to stop.

IMS-specific subscription processing

Before you create subscriptions, you must understand basic replication concepts such as log reading and validation. Then plan your subscription deployment by using design principles that optimize replication for your site

Releasing IMS RECON data sets for reorganization

When your IMS administrator reorganizes RECON data sets, use the RELEASE,RECON command to release any hold that the IMS log reader service has on the data sets.

About this task

Your IMS administrator reorganizes your IMS RECON data sets periodically as part of routine maintenance. As part of the process, the administrator might have to release a discarded data set from any IMS jobs that allocated it, including instances of the IMS log reader service. After release, the administrator can then delete and reallocate the data set.

Note: If you enable automatic RECON loss notification in your IMS system, you do not have to release the discarded data set.

Issue the **RELEASE,RECON** command to DBRC to release RECON data sets from the log reader service.

For information about reorganizing IMS RECON data sets, see "Maintaining the RECON Data Set" in the IMS documentation for the Database Recovery Control Facility (DBRC).

Procedure

1. Your IMS administrator issues the IMS command /RML to release the discarded RECON data set:

   ```
   /R nn,/RML DBRC='RECON STATUS'
   ```

   IMS releases the hold that any jobs have on the discarded RECON data set.
2. Release the discarded data set from the IMS log reader service by issuing the **RELEASE,RECON** command against the Classic data server that hosts the log reader service.

   ```
   /f Data-Server-Name,RELEASE,RECON
   ```

Results

After you issue the **RELEASE,RECON** command, the following console message appears:

```
CECZ0929I DBRC release RECON completed.
```

Reading IMS logs

The IMS log reader service, which runs in the source server, captures change data from any registered DB/DC or DBCTL subsystem in the RECON data sets that the source server accesses.

It does this by processing active and archived IMS logs. The log reader service can also capture changes from registered DL/I batch subsystems in the RECON. The service then sends change messages to the capture service for processing. The change messages include log records that contain both change data and sync points.

Overview

The following diagram shows how IMS writes log data first to recent online logs, then to archive logs.

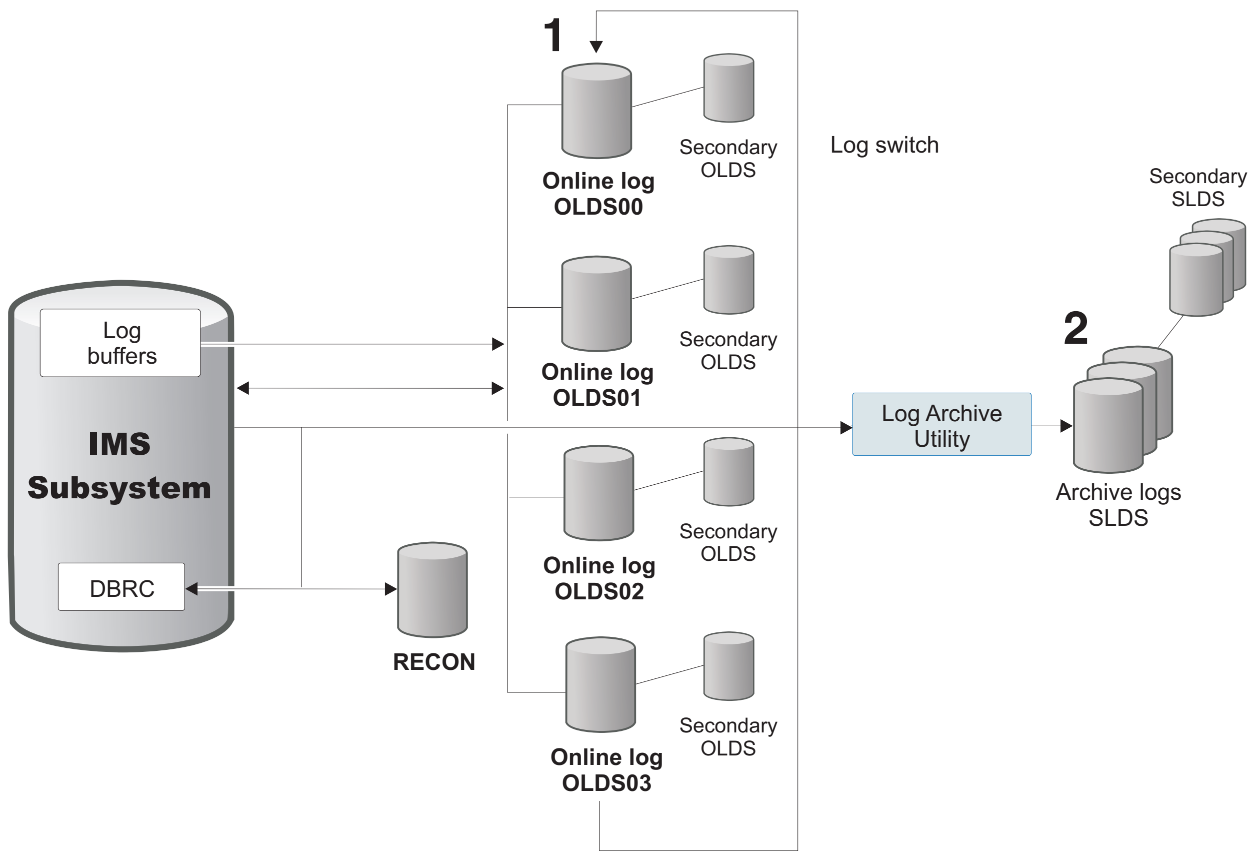


Figure 6. An IMS logging environment

Your IMS database administrator designs a logging environment that can sustain the logging rates and recovery requirements of the subsystem. The design deploys these sets of logs:

- Online data sets (OLDS) called *online logs*, one of which is the *active log* to which IMS is currently writing data
- Secondary OLDS, to which IMS writes data simultaneously with the OLDS
- System log data sets (SLDS) called *archive logs*, where IMS stores recovery data
- Secondary SLDS, to which IMS writes data simultaneously with the SLDS
- Recovery data sets that the IMS recovery utilities use

  The IMS log reader service does not process the recovery data sets, because they do not contain data capture log records.

The database administrator creates a fixed number of online logs for temporary storage of recovery data. When the active log is full, IMS automatically switches to the next online log in sequence. You can also force a log switch manually by using a command. Typically, the IMS Log Archive Utility transfers the data from the active log that is full to the archive logs when the log switch occurs.

If IMS marks a primary OLDS or SLDS in error, the log reader switches to reading the secondary logs.

Subscription processing overview

Before you create subscriptions in the IBM InfoSphere Change Data Capture Management Console, you must understand basic replication concepts such as log reading, validation, and refresh processing.

Understanding IMS units of recovery

The Classic data server sends only committed work to the target engine. If commit control is in effect at the target database, the target engine applies all changes within a transaction as a unit.

A unit of recovery (UOR) represents an IMS transaction, such as an online transaction, or changes that applications generate. Examples of applications that modify IMS databases include the following:

- Batch message processing (BMP) applications
- The IMS Database Resource Adapter (DRA)
- Open Database Access (ODBA) applications
- DL/I batch applications

Typically, a BMP application generates multiple UORs, one for each checkpoint that the application issues. Ideally, a DL/I batch application also issues checkpoints and generates multiple UORs, but this is not a requirement.

VSAM-specific subscription processing

Before you create subscriptions, you must understand basic replication concepts such as log reading and transaction processing.

VSAM log reading

The VSAM log reader service that runs in the source server captures change data from a z/OS System Logger log stream, referred to as the *replication log*.

The replication log is specified in the VSAM cluster definition (LOGSTREAMID) for each data set mapped in a subscription. Each subscription has a single replication log. A subscription cannot be associated with more than one replication log.

Multiple subscriptions can share the same replication log. However, this is not optimal because each subscription must read all blocks in the replication log.

Types of log streams

You must define log streams in data sharing and non-data sharing environments.

The type of log stream that you define depends on whether you access the log stream from one LPAR or from multiple LPARs.

- If you only access a log stream from one LPAR, you can use a DASD-only log stream or you can use a coupling facility log stream.
- If you access a log stream from more than one LPAR, you need a data sharing environment and a coupling facility log stream.

In a data sharing environment, you must define log streams in a coupling facility to enable logging operations to access the same log stream across all participating logical partitions (LPARs) in the sysplex. Multiple LPARs can generate inserts, updates, and deletes to the same VSAM data set. The target CICS region can write changes from all LPARs in the correct order and maintain transactional consistency.

For example: CICS updates all source files from one LPAR (LPAR1). The source server runs on a different LPAR (LPAR2). In this case, you need a coupling facility log stream.

A data sharing environment also requires that all the systems that access the source files share an ICF catalog.

For DASD-only log streams, all applications and the source server must run on the same image. If you have to apply maintenance, you might incur a replication outage.

For additional guidance about setting up log streams in a coupling facility, which type of log stream to define, and sizing the log streams, see the CICS product documentation about defining the logger environment.

Replication logging considerations

When you define a replication log stream, in addition to determining the type of log stream to define you need to consider factors that affect recovery and the volume of data that is generated.

Retention periods

When you define a replication log stream, you need to consider how long you want to keep data in the log.

The RETPD and AUTODELETE parameters that you set when you define a log stream determine how long data is retained and when data can be deleted. Manage the values that you set for these parameters to ensure that the required log replication data is available when replication is started.

You must retain log data long enough that you can recover change data if your replication configuration goes into recovery mode. For example, if you set the replication start time to 10 days ago and set the retention period to 5 days ago, the necessary data is not available and your replication configuration is in recovery.

Log volumes

When you activate replication logging, the types and volume of records written to the log is based on the recovery and replication options that you define. The number of updates and the number of files assigned to a given log stream also impact the volume of data that replication logging generates.

For more information about log record types and the effects of recovery and replication options, see the CICS product documentation.

Reference

The following table provides resources for additional information about defining replication log streams.

Table 26. Replication log stream considerations

Information resource	Reference
Replication logs	Replication logs
Administering restart and recovery, logging concepts	Administering restart and recovery
Defining the logger environment for CICS, log stream types and sizings	Defining the logger environment for CICS
Replication logging, recovery and replication options	Replication logging
Administrative Data Utility (IXCMIAPU) for defining replication log streams	Administrative Data Utility

VSAM units of recovery

The Classic source engine sends only committed work to the target engine. If commit control is in effect at the target engine, the target engine applies all changes within a transaction automatically.

A unit of recovery (UOR) represents a group of operations completed by a CICS transaction to recoverable files that are either committed or backed out as a group. The source engine also groups multiple non-transactional changes into a UOR within the replication system to provide more efficient use of system resources while minimizing latency. Non-transactional changes include operations to non-recoverable data sets as well as changes captured by CICS VSAM Recovery (CICS VR).

Data type validation

Classic change data capture provides the option to enable additional validation of data types.

Some data types are not converted from source data to a different SQL data type when replicating data. The data server always validates the VARCHAR and VARGRAPHIC data types. To ensure that other data types are valid, you can configure the data server to validate data types and define the action for the data server to take if the validation fails. You can also limit the number of log records generated by data validation errors that the data server logs.

You can use the following configuration parameters to control data validation:
- CSDATAVALIDATEACT enables additional validation of data types.
- CSREPORTLOGCOUNT controls the number of log messages for badly-formed data or conversion errors.

For delimited format subscriptions, the following configuration parameters control data validation:
- PUBBDACTION: Set this parameter to TRUE to publish badly formed data or FALSE to fail when badly formed data is encountered.
- PUBBDCOMPAT: Set this parameter to define the options for publishing badly formed data in hexadecimal format or in a format compatible with Version 9.5.
  - 0: Publish in hexadecimal format
  - 1: Version 9.5: No repair, no flag
  - 2: Version 9.5: Repair, no flag

  - 3: Version 9.5: Repair and flag
- PUBBDDIAGLIMIT: Set this parameter if you want to limit the number of diagnostic messages issued for each table mapping.
- PUBBDWTOLIMIT: Set this parameter if you want to limit the number of WTO messages issued when badly formed data is encountered..

Developing a record selection exit for multiple record layouts

You can develop an optional record selection exit to define criteria for accepting or rejecting records for replication or to change the contents of records to fix invalid data.

After developing the exits, specify the modules names in the Capture service configuration parameters.

Use the record selection exit if your solution requires data manipulation or transformation. The exit provides both the before and after images with the change data, which enables you to perform sophisticated selection logic. You can also change the contents of the record I/O areas for any Classic data source.

Restriction: Do not change the size of the record by using the record selection exit.

In most situations, filter data sources with multiple record layouts by defining and altering a view on a table.

Develop your exit in assembler, then assemble and link the module into the Classic load library. A sample source for the exit is in the member User.SCACSAMP(CECRCSEL). This assembler module contains a description of the passed parameters and logic to accept or reject record data based on matching a mapped table name.

In order to have the Classic server call your exit, you must set the appropriate configuration parameter for the capture service. If your user exit is to be called for IMS changes, then set the value of the IMSRECSELECT parameter to the name of the load module you developed previously. For VSAM changes, set the value of the VSAMRECSELECT parameter.

For the support of prior versions of Classic CDC, if the value of the IMSRECSELECT or VSAMRECSELECT configuration parameter is blank, a load module with the name of CECRCSEL will be called if it exists in the STEPLIB library concatenation. This behavior is deprecated, however, and if you currently use an exit of the name CECRCSEL, you should set the value of the appropriate configuration parameter to that value.

The capture service passes the following parameters to the record selection exit:
- The address of a 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, decompress and decrypt it before calling the exit.
- A value that contains the binary length of the record data.

- The address of an 8-byte, left-aligned, padded with spaces database type name of the mapped table. The value `$IMS` indicates IMS database mapping, and the value `$VSAM` indicates VSAM.
- The address of an 8-byte, left-aligned, padded with spaces table owner ID as specified in the metadata grammar that mapped the table.
- The address of an 18-byte, left-aligned, padded with spaces table name as specified in the metadata grammar that mapped the table.
- A full-word binary field that contains the change type associated with the passed data. Here are the valid values for this field:
  - 1—The change is an update and the passed data is both the before image and after image of the update.
  - 2—The change is an insert and the passed data is the data inserted.
  - 3—The change is a delete and the passed data is the record deleted.
- The address of a DBMS-specific area (described in the following lists).
- The address of the after image record data for an update change.
- A full-word binary field that contains the length of the after image record data.
- The address of a null-terminated subscription name.

The DBMS-specific area for IMS is as follows:
- An 8-byte eye-catcher: “`$IMS` “
- A half-word binary length of this area.
- A half-word version number.
- A 4-byte reserved area.
- An 8-byte, left-aligned, padded with spaces DBD name.
- An 8-byte, left-aligned, padded with spaces segment name.
- An 8-byte, left-aligned, padded with spaces PSB name.
- An 8-byte, left-aligned, padded with spaces transaction name.
- An 8-byte, left-aligned, padded with spaces user id.
- An 8-byte binary system clock value.

The DBMS-specific area for VSAM is as follows:
- An 8-byte eye-catcher: “$VSAM “
- A half-word binary length of this area.
- A half-word version number.
- A 4-byte reserved area.
- An 8-byte, left-aligned, padded with spaces applid of the CICS TS system, or jobname of the CICS VR job, depending on the value of the source of the change indicator below.
- A 44-byte, left-aligned, padded with spaces dataset name.
- A full-word binary indicator of the source of the change. 1 = CICS TS, 2 = CICS VR.
- A 4-byte, left-aligned, padded with spaces name of the transaction for CICS TS.
- A 4-byte binary task number for CICS TS.
- A 4-byte, left-aligned, padded with spaces terminal id for CICS TS.
- An 8-byte, space-padded job name for CICS VR.
- A 4-byte, left-aligned, padded with spaces partial step name for CICS VR.
- An 8-byte binary system clock value of the log order time.

- An 8-byte binary system clock value of the original log time.

If the record selection exit module exists in the Classic load library, the Classic data server calls the exit each time the server receives a change from the log reader. The server makes this call once for each table name that matches the changed data source. The record selection exit determines whether the change is valid for the table name in the metadata catalog. A return code of zero (0) indicates that the exit accepted the record data for the passed table name. On acceptance, the Classic data server converts the record data to an SQL descriptor area (SQLDA) and forwards the change to the target engine.

With data sources that can include parent segments in the table mapping, such as IMS, the record data parameter contains information for all records in the defined path. When you calculate 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, add debugging logic in the sample record exit to dump passed records to the operator console.

You can filter certain table names based on the action. For example, an application might capture changes to a database only when inserts occur to certain records. In a case like this, you can use the record selection exit to reject all UPDATE and DELETE messages for a specified table name.

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

```
//jobname JOB (acctinfo),'CECRCSEL',CLASS=A,
//     MSGCLASS=X,NOTIFY=&SYSUID
//*******************************************************************
//*   ASSEMBLE CECRCSEL                                             *
//*******************************************************************
//ASSEMBLE EXEC PGM=ASMA90,PARM='LIST,NODECK,RENT'
//SYSLIB   DD  DSN=SYS1.MACLIB,DISP=SHR
//SYSLIN   DD  DISP=(NEW,PASS),DSN=&&OBJ,
//        UNIT=SYSDA,SPACE=(TRK,(1,1))
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=VIO,SPACE=(1700,(2000),,,ROUND)
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DISP=SHR,DSN=your.source.libray(CECRCSEL)
//*******************************************************************
//* LINK                                                            *
//*******************************************************************
//LINK    EXEC PGM=IEWL,PARM='LIST,RENT,REUS',COND=(4,LT)
//SYSLIN   DD  DISP=(OLD,DELETE),DSN=&&OBJ
//         DD  DDNAME=SYSIN
//SYSLMOD  DD  DISP=SHR,DSN=your.load.library
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1024,(120,120),,,ROUND),DCB=BUFNO=1
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
    ENTRY   CECRCSEL
    NAME    CECRCSEL(R)
/*
```

System exits for change capture

System exits are programs that the data server calls at predefined processing points to provide record filtering and other processing.

Table 27. System exits for change capture

Data source	Purpose	Exit
IMS	Identify whether a change should be captured and the subscription and name of the table to use	Table identification exit
IMS and VSAM	Determine whether a change should be captured for a specific table or view	Record selection exit

Table identification exit

You can develop a table identification exit to identify the subscription and the table or view to associate with a data change.

The capture process must identify the names of the subscriptions and of the table or view that are associated with a data change. Processing mechanisms establish this association by using information from the data change to identify the tables or views of interest. The tables or views of interest identify the subscriptions of interest.

Identifying the tables associated with a data change is a simple, efficient process. Similarly, identifying the subscriptions associated with a table or view is an efficient process. However, when views are involved in a data change, the capture process is more complicated. The process must first identify the set of all possible views that might be associated with the data change, and then process the selection criteria of each view against the data change to identify the set of views that are actually of interest.

The table identification exit offers a more efficient mechanism in the following situations:

- In an environment where many views are defined, or where views are defined with complex selection criteria, this exit can provide a more efficient mechanism for identifying the views that are associated with a particular data change.
- When many distinct database entities exist with identical layouts and you want to reduce the size of the metadata catalog.

An administrator can use the mechanisms that the table identification exit provides to perform the following actions:

- Identify the name of the subscription name and the table or view that is associated with a data change. In particular, this exit offers the administrator the ability to more quickly identify the views of interest directly from the data change record.
- Modify the identity of the database entity. For example, for IMS the exit can modify the DBD name or segment name in the DBMS-specific parameter thereby allowing the administrator to implement an aliasing scheme.

Table identification exit: Developing

A sample table identification exit is provided in SCACSAMP member CECRCID.

The sample code performs the following functions:

- Allocate a persistent work area at initialization time, placing the address of that work area in the User Word.
- Construct a table name based on the passed IMS DBD name and segment name.
- Deallocate the persistent work area at termination time.

The sample member CECRCID is written in non-LE assembly language. Sample JCL for assembling and linking the CECRCID is located in SCACSAMP member CECCJID.

Example

An IMS database contains four DBDs (DBD0, DBD1, DBD2, and DBD3). You can map each DBD through a common set of column definitions. Without the table identification exit, the administrator needs to define four tables:

- ABC.TAB_FOR_DBD0
- ABC.TAB_FOR_DBD1
- ABC.TAB_FOR_DBD2
- ABC.TAB_FOR_DBD3

The only differences are the table name and the DBD that each table references. Each tables is then associated with a distinct subscription:

- SUB_FOR_DBD0
- SUB_FOR_DBD1
- SUB_FOR_DBD2
- SUB_FOR_DBD3

By using this exit, the administrator can define one table, ABC.TAB_FOR_DBDS, associate this one table with the four subscriptions, and code the exit to establish the association between the input DBD name and the target subscription. In this case, the exit would produce the following results:

Table 28. Results

DBD name (input)	Table name (output)	Subscription name (output)
DBD0	ABC.TABLE_FOR_DBD0	SUB_FOR_DBD0
DBD1	ABC.TABLE_FOR_DBD1	SUB_FOR_DBD1
DBD2	ABC.TABLE_FOR_DBD2	SUB_FOR_DBD2
DBD3	ABC.TABLE_FOR_DBD3	SUB_FOR_DBD3

You can achieve similar processing with the exit modifying the IMS DBD name or segment name and signaling that normal processing should be performed for the data change. In this case, the administrator can define a single table for DBD0, ABC.TAB_FOR_DBD0, and associated this table with a single subscription, SUB_FOR_DBD0. Upon receiving a data change call associated with DBD1, DBD2, or DBD3, the exit will modify the DBD name to DBD0 and signal, through a return code, that the data change should be processed according to standard change processing mechanisms.

Table identification exit: Configuring

To enable use of the table identification exit for a subscription that references an IMS data source, you must update the IMSRECID configuration parameter for the capture service.

The IMSRECID parameter identifies the name of your implementation of the exit. You need to update this configuration setting while no subscriptions are replicating. The exit is loaded when you start replication for the first subscription that references an IMS data source. If replication is already active, you need to stop capture operations to enable use of the exit.

Table identification exit: Process

This topic describes how the capture service loads the table identification exit.

The exit is called in the following situations:

- After the capture service loads the exit with an initialization request
- Before the capture service loads the exit with a termination request
- With every data change processed by the capture service.

The capture service attempts to load the exit when the first subscription is started. If successful, the following message is issued:

CECC0247I The table identification exit *module-name* has been successfully loaded.

The exit is unloaded when capture operations end for the last active subscription. The following message is issued:

CECC0248I The table identification exit *module-name* has been unloaded from memory.

If the exit cannot be loaded, no console message is issued but an informational trace message is written to the trace log.

Due to this behavior you might see a series of load and unload messages during the life of the data server. This behavior enables you to replace the exit module without shutting down the entire data server. You can replace the exit in the load library for the data server. After capture operations end for all subscriptions, the new version is loaded when replication starts for the next subscription.

Table identification exit: Parameters

Programs that call the table identification exit pass address pointers in a parameter list.

Description

Register 1 points to a parameter list (PL) on entry to CECRCID. The parameter list consists of 12 entries as shown in the following table. Each entry is a full word (4 bytes) in length. Some of the entries contain the actual parameter value (for example, Action, Iteration Count) and other entries refer to an area that contains the actual value (for example, Address of User Word, Address of DBMS Type Name).

The following table describes each parameter list entry.

Table 29. Parameters passed to the table identification exit.

Parameter	Area length	Description
Address of user word	4 bytes	The user word is initialized to 0 before the initialization call. The exit can use the user word for any purpose. The value that is in the user word when the exit returns to the capture service will be presented to the exit when next called.
Action	4 bytes	Identifies the reason why the exit was called: **0: Initialize** The exit should perform initialization processing. **1: Update** The exit is called due to an update operation. **2: Insert** The exit is called due to an insert operation. **3: Delete** The exit is called due to a delete operation. **4: Terminate** The exit should perform termination processing.
Iteration count	4 bytes	Identifies the number of times that a particular data change is presented to the exit for processing. When a data change is first presented to the exit the iteration count is 1.
Address of database type name	8 bytes	Identifies the type of the data change that is being processed. This name identifies the structure that should be used to interpret the DBMS-specific parameter. Valid value: `"$IMS    "`
Address of subscription name	63 bytes	Area into which the exit must place the name of the subscription that will be associated with a data change of interest. The exit must initialize this area and the value returned must be padded with blanks to its full length.
Address of table owner	8 bytes	Area into which the exit must place the owner of the table or view that will be associated with a data change of interest. The exit must initialize this area and the value returned must be padded with blanks to its full length.
Address of table name	18 bytes	Area into which the exit must place the name of the table or view that will be associated with a data change of interest. The exit must initialize this area and the value returned must be padded with blanks to its full length.
Address of DBMS-specific parameter	Length depends on DBMS type	Contains information related to the data change. The structure of this area is database-type dependent. See “DBMS-specific parameters” on page 148 for more information.
Address of first buffer	Specified in length of first buffer	Contains a data change record. • For an insert, the first buffer will contain the after image associated with the data change. • For an update or delete, the first buffer will contain the before image associated with the data change.
Length of first buffer	4 bytes	The length of the second buffer. Provided in association with the insert, update, and delete action codes.

Table 29. Parameters passed to the table identification exit. (continued)

Parameter	Area length	Description
Address of second buffer	Specified in length of second buffer	Contains a data change record. This buffer is provided only for an update, in which case it will contain the after image associated with the data change.
Length of second buffer	4 bytes	The length of the second buffer. Provided in association with the update action code.

Table identification exit: Return codes

The table identification exit uses return codes to signal how the capture service should process a data change request.

Description

The exit provides the return codes listed in the following table.

Table 30. Return codes for the table identification exit.

Return codes	Description
0	The capture service should process the data change by using standard processing mechanisms.
1	No tables or subscriptions exist for this data change or no additional tables or subscriptions exist for this data change.
2	The capture service should process the data change in association with the identified subscription and table or view, and the exit has completed its processing for the data change.
3	The capture service should process the data change in association with the identified subscription and table or view, and the exit has additional subscriptions or tables or views to associate with the data change. In this situation, the Iteration Count is incremented by 1 and the exit is called again for the same data change. This processing continues until the exit provides a return code value that signals that it has completed processing for the data change (0, 1, or 2).

Important: The exit should provide these return codes only. Any other return codes can lead to unpredictable results and will be treated as a return code of 0.

For return codes 2 and 3, the exit must identify the subscription name, table owner, and table name (or view owner and view name) that will be associated with the data change.

The exit can control the number of times that it is called for a single data change by using return code 3. Each time the exit provides this return code value it will be called again for the same data change with the Iteration Count incriminated with each call. For a subsequent call, the exit can return any of the return codes 0, 1, 2, or 3. In this situation, these return codes are interpreted as shown in the following table:

Table 31. Return codes for multiple iterations of a data change.

Return codes	Description
0	The data change should be processed against the subscriptions and tables previously identified by the exit, and the data change should also be processed according to standard processing mechanisms.
1	The list of subscriptions and tables to associate with this data change was complete with the previous call.
2	The list of subscriptions and tables to associate with this data change is complete with this call.
3	The list of subscriptions and tables to associate with this data change is not complete and the exit should be called again.

Table identification exit: Special considerations

When working with the table identification exit, you should be aware of how processing occurs in specific change capture situations.

For IMS, the first buffer and second buffer contain information for all segments in the defined path. When you calculate offsets to specific fields in the data record, you must account for the concatenated length of all parent segments in the data area.

When variable length segments are captured for child segments with parents that are variable length, the starting positions of the child segments have fixed starting offsets. The starting offsets are based off the maximum length for each variable length segment. The first two bytes of each variable length segment contain the actual length of the data that was captured for that segment. Data that is not captured is initialized to binary zeros.

The table identification exit can change the contents of the record I/O areas in the first buffer or the second buffer. The exit might do this to correct invalid data.

Important: Use caution when modifying the contents of the record I/O area. The length of the buffers must not be changed by the exit.

Additional considerations apply to using the exit to deal with alias situations. When replication starts for a subscription, the capture service attempts to verify that the change capture environment is valid based on the active replication mappings associated with a subscription. The capture service attempts to validate that the change capture options for IMS that are set in the DBD definition are consistent based on the column definitions associated with a replication mapping. IMS uses the contents of the active ACB library to determine what gets written in a data capture record and the capture service uses the DBD load library form (which is used as input to create the ACB form). Although validation attempts to ensure proper setup, what gets written to the IMS log when a change is made might be inconsistent.

When aliasing is involved the administrator has had to augment the alias DBD definition to get IMS to generate the data capture log records in the first place, but because there might not be any replication mappings that reference that database, the capture service was potentially unable to perform any validation for these alias databases. the capture service is trusting that an alias databases structure and capture options are consistent with the database that was used for mapping purposes. If this turns out not to be the case data loss or corruption is possible.

Record selection exit

You can develop a record selection exit to control the processing of data change records for a given table or view.

During capture processing, a data change is processed against all tables or views that are associated with the data change. In some environments, you might want to process the data change record after it is associated with a table or view but before it is finally formatted and sent. Consider these situations:

- You might determine that the data change should not be sent for the table or view, for example, in a situation where all DELETE operations should be suppressed for a particular table.
- The view selection logic is complex and difficult to process by using an SQL engine.
- Invalid data values are likely to exist.

In an environment where views are defined, this exit might provide a more efficient mechanism for identifying the views that are associated with a particular data change. This exit also provides the ability to detect and correct invalid data values before those values are formatted and transmitted.

An administrator can use the mechanisms that the record selection exit provides to perform the following actions:

- Signal that a particular data change should not be processed for a given table or view. You might use the exit for this purpose because the view selection criteria are difficult to express through SQL or because the view selection criteria are more efficiently processed with the exit mechanism.
- Modify the contents of data change records. This capability can be useful for detecting invalid data values.

Record selection exit: Developing

A sample record selection exit is provided in SCACSAMP member CECRCSEL.

The sample code performs the following functions:

- Conditionally allocate and free a work area, based on whether the provided 256 byte area is large enough or not.
- Accept all records.
- Comments are provided for rejecting all Delete operations against the TEST.EMPLOYEE table.

The sample member CECRCSEL is written in non-LE assembly language. Sample JCL for assembling and linking the CECRCSEL is located in SCACSAMP member CECCJSEL.

The CECRCSEL exit should not modify the values in the DBMS-specific parameter. Any modification to these values will be ignored.

Record selection exit: Process

The load module for the record selection exit must be named CECRCSEL. The exit is invoked when the load module name is specified in the load library for the data server.

The exit is called once for every table or view associated with a data change processed by the capture service.

The capture service attempts to load the exit when the first subscription is started. If successful, the following message is issued:

```
CECC0245I The table identification exit CECRCSEL has been successfully loaded.
```

The exit is unloaded when capture operations end for the last active subscription. The following message is issued:

```
CECC0246I The table identification exit CECRCSEL has been unloaded from memory.
```

If the exit cannot be loaded, no console message will be issued but an informational trace message is written to the trace log.

Due to this behavior you might see a series of load and unload messages during the life of the data server. This behavior enables you to replace the exit module without shutting down the entire data server. You can replace the exit in the load library for the data server. After capture operations end for all subscriptions, the new version is loaded when replication starts for the next subscription.

Record selection exit: Parameters

Programs that call the record selection exit pass address pointers in a parameter list.

Register 1 points to a parameter list on entry to CECRCSEL. The parameter list consists of 11 entries as shown in the following table. Each entry is a full word (4 bytes) in length. Some of the entries contain the actual parameter value (for example, change type, length of first buffer) and other entries refer to an area that contains the actual value (for example, 256- byte work area, address of DBMS-type name).

The following table describes each parameter list entry.

Table 32. Parameters passed to the record selection exit.

Parameter	Area length	Description
Address of 256 byte work area	256 bytes	The work area is uninitialized. You can use this parameter as a save area so the module can call subroutines. Do not use this area for persistent storage across invocations of the module.
Address of first buffer	Specified in length of first buffer	The first buffer contains a data change record. For an insert, the first buffer will contain the after image associated with the data change. For an update or delete, the first buffer will contain the before image associated with the data change.
Length of first buffer	4 bytes	The length of the first buffer.
Address of database type name	8 bytes	Identifies the type of the data change that is being processed. The database type name identifies the structure that should be used to interpret the DBMS-specific parameter. Valid values: `"$IMS    "` or `"$VSAM    "`
Address of table owner	8 bytes	Area that contains the name of the owner (also referred to as creator or schema) of the table or view that is associated with the data change. The value is padded with blanks to its full length.
Address of table name	18 bytes	Area that contains the name of the table or view that is associated with the data change. The value is padded with blanks to its full length.

Table 32. Parameters passed to the record selection exit. (continued)

Parameter	Area length	Description
Change type		Identifies the reason why the exit was called: **1: Update** The exit is called due to an update operation. **2: Insert** The exit is called due to an insert operation. **3: Delete** The exit is called due to a delete operation.
Address of DBMS-specific parameter	Length depends on DBMS type	Contains information related to the data change. The structure of this area is database-type dependent. See "DBMS-specific parameters" on page 148 for more information.
Address of second buffer	Specified in length of second buffer	Contains a data change record. It is provided only for an update, in which case it will contain the after image associated with the data change.
Length of second buffer	4 bytes	The length of the second buffer. Provided in association with the update change type.
Address of subscription name	63 bytes	63-byte area that contains the name of the subscription that is associated with the data change. The value is blank padded to its full length.

Record selection exit: Return codes

The record selection exit uses return codes to signal how the capture service should process a data change request.

The exit provides the return codes listed in the following table:

Table 33. Return codes for the record selection exit.

Return codes	Description
0	The capture service should accept the data change.
!0	The capture service should ignore the data change.

When the exit signals that the data change should be accepted (return code 0), the capture service formats and prepares the data change record for transmission based on the value modified by the exit.

Record selection exit: Special considerations

When working with the record selection exit, you should be aware of how processing occurs in specific change capture situations.

For IMS, the first buffer and second buffer contain information for all segments in the defined path. When you calculate offsets to specific fields in the data record, you must account for the concatenated length of all parent segments in the data area.

When variable length segments are captured for child segments with parents that are variable length, the starting positions of the child segments have fixed starting offsets. The starting offsets are based off the maximum length for each variable length segment. The first two bytes of each variable length segment contain the actual length of the data that was captured for that segment. Data that is not captured is initialized to binary zeros.

The record selection exit can change the contents of the record I/O areas in the first buffer or the second buffer. The exit might do this to correct invalid data.

Important: Use caution when modifying the contents of the record I/O area. The length of the buffers must not be changed by the exit.

The exit should not modify the values in the DBMS-specific parameter. Any modification to these values will be ignored.

DBMS-specific parameters

The table identification exit (CECRCID) and the record selection exit (CECRCSEL) both receive the address of a DBMS-specific parameter.

Description

The DBMS-specific parameter contains information related to the data source for the data change that is being processed. This parameter contains a standard header followed by DBMS-specific information. The layout of the DBMS-specific parameter is shown in the following tables.

Table 34. Header

Parameter	Length	Description
Database type name	8 bytes	The database type name associated with the data change. The value is padded with blanks. This value matches the DBMS type name passed to the CECRCID or CECRCSEL exit from the Address of DBMS Type Name parameter. Possible values: IMS `"$IMS    "` or VSAM `"$VSAM   "`
Parameter length	2 bytes	The length of the DBMS-specific parameter, including this header.
Version number	2 bytes	The version of the DBMS-specific parameter.

Table 35. IMS-specific parameters

Parameter	Length	Description
DBD name	8 bytes	Name of the DBD associated with the data change.
Segment name	8 bytes	Name of the segment associated with the data change.
PSB name	8 bytes	Name of the PSB associated with the data change.
Transaction name	8 bytes	Application transaction name associated with the data change.

DSECTs for the DBMS-specific parameter layouts are located in SCACSAMP in CECMDBMS.

Chapter 12. Configuring replication

You can complete the replication configuration begun in Classic Data Architect by using Management Console.

Configuring replication in Management Console

You can use Management Console to complete the replication configuration and to administer and monitor replication.

In Management Console, you will need to complete the following tasks:

- Create datastores for the source and target data sources

  When publishing changes in a delimited format to WebSphere MQ, you specify the source server as the target server when you define a new subscription. You do not need to create a target data source for delimited format subscriptions.
- Set up and configure subscriptions
- Map tables and columns

Only the relevant information will be displayed in the interface and only the eligible features and options will be available for use.

After configuring replication, you are able to perform the following actions in Management Console:

- Start replication
- Stop replication
- Monitor replication

Be aware that the information, features, and options displayed in the Management Console interface is determined by several factors:

- Your role as a user
- The datastores to which you have access
- The type of the datastore
- The version of the datastore

Only the functionality that is applicable to InfoSphere Classic CDC for z/OS will be enabled. Features that are not visible or enabled on menus are not available to InfoSphere Classic CDC for z/OS datastores.

For more information, see the *Management Console Administration Guide*.

Chapter 13. After you install and configure InfoSphere CDC

After you instal and configure InfoSphere CDC, you can begin using it.

Refreshing your replication environment

Before replication begins, perform a refresh operation to load the contents of your target database and synchronize its contents with your data source.

Replication processing sends changes to a target database as they occur in near real time. Refresh processing, by contrast, synchronizes the entire contents of a source and target table so that replication processing can begin or resume with matching data.

In some cases, performing a refresh is preferable to resuming replication from a past restart position:

- You discover errors or inconsistencies in the target database
- Mass updates occurred at the data source that change most or all records
- Replication has been inactive long enough that refreshing the data will take less time than replicating the historical changes

You initiate refresh processing at the subscription level, and then the subscription refreshes any tables that require it. After the subscription finishes refreshing all the tables, it resumes replication automatically.

Administer refresh processing by using Management Console and ensure that you synchronize your tables and maintain data consistency:

- **Perform refresh to synchronize your tables—**Wherever possible, avoid using third-party utilities to load your target database. Management Console cannot ensure the accuracy of your data if you use an external load mechanism. If you must load the target data externally, be sure to perform the procedure in the correct sequence:
  1. Quiesce the data source.
  2. Set a log position to a point in time after the load.

     For VSAM, the source data set cannot be opened by any address space that will be logging updates when the log position is set.
  3. Reactivate the data source.
- **Protect data consistency when you refresh—**Quiesce the data source during refresh processing to maintain the consistency between your source and target tables, even though adaptive apply processing at the target database can handle most situations if you do not.

  Replication for the subscription is inactive during refresh processing, although log reading can continue. The source server stores the change messages until replication resumes. Replication processing begins from the point in time where refresh processing started.

  Unless all log positions for replication mappings are more current than the bookmark for the subscription, the bookmark determines where log reading begins.

You might see messages that refer to the Classic call level interface (CLI). InfoSphere Classic CDC for z/OS uses a CLI driver internally to fetch data and send it to the target engine.

Setting up your environment for refresh

The installation customization process handles the setup requirements for the Classic data server. You perform any setup steps in the data source environment, such as deploying a database interface or configuring PSBs for IMS.

Before you refresh subscriptions, ensure you complete the following tasks on the data source environment:

- **Load query processor—**Ensure that your source server is running a query processor service for refresh operations (service class QPLD) so that InfoSphere Classic CDC for z/OS can refresh subscribed tables. Do not change the settings for this service.
- **IMS data sources—**Ensure that a DRA service is running in the source server. You must configure a DRA interface so that the source server can communicate with the IMS subsystem.

  Ensure that a DRA service is running in the source server. You must configure a DRA interface so that the source server can communicate with the IMS subsystem.

  You must also configure your PSBs to prevent uncommitted reads. If you do not configure your PSB correctly, refresh will fail and issue an error. using the IMS utilities, set `PROCOPT=GP` on the PCB level. You cannot use `PROCOPT=SENSEG` for refresh. For more information, see the *IMS Utilities Reference Manual*.
- **VSAM data sources—**Ensure that a VSAM access service is running in the source server. You must configure a VSAM access service so that the source server can access VSAM data sets.

Validation of the replication environment

InfoSphere Classic CDC for z/OS validates key components to ensure that your environment is ready to start replication.

Before you can start replication successfully, you must resolve any configuration problems or inconsistencies in a subscription that can prevent it from starting. The Classic Data Architect provides the **Validate replication mappings** function to help you troubleshoot and resolve errors when you are setting up your replication environment.

Validation processing enables you to fix subscription errors in the setup stage if you do not want to wait until you start replication. For example, you can try to address all the problems with a subscription that can be detected so that the subscription is more likely to start with the other subscriptions.

The source server can also detect schema changes while replication is active.

Subscription validation

Classic change data capture validates your subscription, replication mappings, and underlying data structures to help you to identify potential problems while you are setting up your replication environment.

The validation process compares the structure of the source and target tables in the subscription, and identifies configuration problems or inconsistencies in subscription metadata that can prevent replication from starting:
- A source database or file with incorrect augmentation
- A VSAM subscription that references two or more VSAM data sets that use different log stream IDs
- A mismatch between table and column attributes at the source and target

Validation processing continues to run to completion, regardless of whether it encounters one or more of these errors.

Validation at run time

When you start replication for a subscription, Classic change data capture validates the structure of your source and target tables.

Classic change data capture validates your replication environment regardless of which of the following methods you use to start the subscription:
- Start replication in the Management Console
- Issue a START,REPL command to the source server
- Mark a subscription as persistent and restart replication automatically

For IMS data, when you start replication for a subscription, your environment performs strict validations of the database descriptions (DBDs) and the program specification block (apply PSB) for any replication mappings that require it. Runtime validation ensures that, before replication begins, you resolve any errors that the describe process discovered when you created or modified the subscription. Possible problems include having source database descriptions (DBDs) without correct augmentation, and data mismatching between data sources and target databases. If your site uses global online change, your data server JCL must include the IMCPMDOL parameter for IMS DBD validation.

For VSAM data, the validation process uses the following information to validate the attributes of VSAM data sets:
- The name of the file
- The type of entry, such as VSAM cluster
- The type of organization
- The maximum record length
- The offset of the key
- The length of the key
- The source and target file definitions must either be recoverable (LOG=ALL or LOG=UNDO) or non-recoverable(LOG=NONE)

Understanding database transaction sequence and latency

Maintain low latency in your replication environment by optimizing the number of subscriptions that you deploy. In some cases, good subscription design involves a trade-off between low latency and applying transactions in strict order.

Several factors can affect transaction sequence and latency:
- **Transactional consistency—**Ordinary database processing maintains *transactional integrity* by applying transactions in the exact order in which they occur. For

example, a withdrawal from your bank account might succeed only if the bank first completes your transfer to that account.

- **Latency—**Replication processing differs from ordinary database processing in that low latency is also a high priority. In change data capture, you lower latency and maintain relative *transactional consistency* by deploying more subscriptions. For example, you can redistribute the replication mappings in an existing subscription into multiple subscriptions.

  The trade-off is that, as you increase the amount of parallel processing in your replication environment, you decrease the likelihood of applying transactions to the target database in perfect order.
- **Parallel processing—**When you deploy more subscriptions, the target engine increases the resources that it devotes to apply processing. Information management professionals sometimes refer to concurrent apply processing as *parallelism*, because the target engine applies changes to the target database concurrently.
- **Subscription design—**Design your subscriptions to address your business needs for low latency and precise order. For example, precise ordering is unlikely if replication stops for a subscription and a transaction cannot finish processing until you restart the subscription. Because subscriptions maintain independent restart positions, the target engine might apply later transactions before the original transaction can finish.

  Transactions that change databases in more than one subscription can result in splitting a UR into two or more apply transactions that arrive at the target database in unpredictable order. To keep the number of such transactions to a minimum, include all databases that are related to a single business application within one subscription wherever possible. For example, include your Customer data sources in one subscription and your Inventory data sources in another.

Chapter 14. Understanding data server services

The services required for the Classic data server are customized and running when you complete the installation customization process.

The following table summarizes these services.

Table 36. Summary of services

Service type	Task name	Service class
Administration service	CECPAA	PAA
Capture service	CECCAP	CAP
Client connection handler service	CACINIT	INIT
IMS database resource adapter (DRA) access service	CACDRA	DRA
IMS log reader service	CECLRS	LRSI
Logger service	CACLOG	LOG
Monitoring service	CECMAA	MAA
Query processor service	CACQP	QP
Refresh query processor service *	CACQP	QPLD
Region controller service	CACCNTL	CNTL
Remote operator command service	CACOPER	OPER

Notes:

- The refresh query processor service does not apply to delimited format subscriptions.

Administration service

The administration service is assigned to the PAA service class. The task name for the PAA service class is CECPAA.

The administration service manages the configuration and administration of replication. The management functions include setting up replication, error reporting during setup, and access to the system event log. The administration service also maintains the replication runtime environment.

The following table lists the configuration parameters that apply to the PAA service class.

Table 37. Configuration parameters for the PAA service class

Parameter	Default value	Description
IDLETIMEOUT	5M	The amount of time that a service remains idle before it polls the local message queue for messages that are to be processed
INITIALTHREADS	1	Number of instances of this service that the region controller starts during initialization of the Classic data server

Table 37. Configuration parameters for the PAA service class (continued)

Parameter	Default value	Description
MAXTHREADS	1	Maximum number of instances of this service that the region controller is allowed to start
MAXUSERS	100	Maximum number of user connections
RESPONSETIMEOUT	3S	Maximum amount of time to wait for response before terminating a connection
SAFEXIT	None	The System Authorization Facility (SAF) system exit that performs authorization checks for connections to the Classic data server.
SEQUENCE	0	Sequence number assigned to services
TRACELEVEL	4	Trace level

Capture service

The capture service is assigned to the CAP service class. The task name for the CAP service class is CECCAP.

The capture service manages change data capture. Those operations include management of the log reader service, change streams, publishing delimited format data to WebSphere MQ, and communication with the target server for non-delimited subscriptions.

The log reader service reports stream-related errors to the capture service that initiates end-of-replication for the affected subscriptions. The log reader service can also report other types of errors such as Classic data server or non-stream related errors that end replication for all active subscriptions and cause the capture service to stop.

Restriction: Only a single service of service class CAP can be configured in a single Classic data server. A single instance of this service runs in the address space of a Classic data server.

The following table lists the configuration parameters that apply to the CAP service class. The subscription type column identifies the configuration parameters that apply to a specific subscription type.

Table 38. Configuration parameters for the CAP service class

Parameter	Default value	Description	Subscription type
APPLYBUFFERSIZE*	252K	Size of the buffer that uses to send changes from the source server to the target server.	
BOOKMARKQUEUE	None	Name of the WebSphere® MQ local queue that stores restart information for use in delimited publishing.	Delimited format subscriptions
COLDELIMITER	,	Defines the character to use for delimiting columns.	Delimited format subscriptions
COMMITINTERVAL	500 MS	Defines how often delimited format messages should be committed to WebSphere MQ.	Delimited format subscriptions
CSDATAVALIDATEAC	0	Invalid data handling: 0 (no validation), 1 (repair), or 2 (repair and report)	Classic CDC subscriptions

Table 38. Configuration parameters for the CAP service class (continued)

Parameter	Default value	Description	Subscription type
CSREPORTLOGCOUNT	0	Limit for the number of messages logged when CSDATAVALIDATEAC = 2.	Classic CDC subscriptions
HEARTBEATTIMEOUT	15M	The interval at which the source and target servers exchange messages to detect communication problems.	Classic CDC subscriptions
IMSRECID		IMS table identification exit.	
IMSRECSELECT		Record selection exit to define criteria for accepting, rejecting or changing contents of records for IMS tables.	
PUBBDACTION	TRUE	Identifies the action for the capture service to take when badly formed data is encountered. The action controls whether replication for a subscription continues or stops.	Delimited format subscriptions
PUBBDCOMPAT	0	Identifies product compatiblity options for the format in which badly formed data is published.	Delimited format subscriptions
PUBBDDIAGLIMIT	100	Controls the number of diagnostic messages generated when badly formed data is encountered.	Delimited format subscriptions
PUBBDWTOLIMIT	1000	Controls the number of CACJ080W messages generated when badly formed data is encountered.	Delimited format subscriptions
PUBUTF8	FALSE	Identifies whether delimited messages are published in the host code page or converted to UTF-8 before writing to WebSphere MQ.	Delimited format subscriptions
QUEUEMGRNAME	CHAR(4)	Name of the WebSphere MQ queue manager that is used for delimited publishing	Delimited format subscriptions
ROWDELIMITER	/n	Defines the character to use for delimiting rows.	Delimited format subscriptions
SEQUENCE	0	Sequence number assigned to services	
STRINGDELIMITER	"	Defines the character to use for delimiting strings.	Delimited format subscriptions
TIMEZONE	Atlantic/ Reykjavik	The Olson database name for the time zone of the source server	Classic CDC subscriptions
TRACELEVEL	4	Trace level	
UORGROUPCOUNT	1	The number of messages to group into a common unit of recovery (UOR) during replication.	
VSAMRECSELECT		Record selection exit to define criteria for accepting, rejecting or changing contents of records for VSAM tables.	
* The APPLYBUFFERSIZE parameter does not apply to Classic CDC.			

Connection handler service

The connection handler service is assigned to the INIT service class. The task name for the INIT service is CACINIT.

A connection handler listens for connection requests from client applications and routes the requests to the appropriate administration, monitoring, operator, and query processor tasks.

Remote client applications use TCP/IP to communicate with a Classic data server.

The following table summarizes the configuration parameters that define connection handler services in the INIT service class.

Table 39. Configuration parameters for the INIT service class

Parameter	Default value	Description
COMMSTRING	TCP/0.0.0.0/ 9087	Client connection listen string
IDLETIMEOUT	5M	The amount of time that a service remains idle before it polls the local message queue for messages that are to be processed
INITIALTHREADS	1	Number of instances of this service that the region controller starts during initialization of the Classic data server
MAXTHREADS	1	Maximum number of instances of this service that the region controller is allowed to start
MAXUSERS	100	Maximum number of user connections
RESPONSETIMEOUT	5M	Maximum amount of time to wait for response before terminating a connection
SEQUENCE	0	Sequence number that is assigned to services.
TRACELEVEL	4	Trace level

Database resource adapter (DRA) service

The DRA service is assigned to the DRA service class. The task name for the DRA service class is CACDRA.

The DRA service interface initializes the DRA interface and connects to an IMS DBCTL region to access IMS data.

The following table lists the configuration parameters that define DRA services in the DRA service class.

Table 40. Configuration parameters for the DRA service class

Parameter	Default value	Description
CONNECTINTERVAL	15S	The interval time in seconds at which the data DRA service retries connecting to IMS when IMS is not available
DEFAULTPSBNAME	NOPSB	Default PSB name
DRATABLESUFFIX	None	DRA startup table suffix
DRAUSERID	None	Default DRA user ID that is used to connect to and register with a DBCTL subsystem. The DRAUSER is the name by which the Classic data server is known to DBCTL
IDLETIMEOUT	5M	The amount of time that a service remains idle before it polls the local message queue for messages that are to be processed

Table 40. Configuration parameters for the DRA service class (continued)

Parameter	Default value	Description
INITIALTHREADS	1	Number of instances of this service that the region controller starts during initialization of the Classic data server
MAXTHREADS	1	Maximum number of instances of this service that the region controller is allowed to start
MAXUSERS	10	Maximum number of user connections
RECONNECTWAIT	1M	The minimum amount of time in minutes that the DRA service waits after an IMS disconnect before attempting to reconnect to IMS
RESPONSETIMEOUT	5M	Maximum amount of time in minutes to wait for response before terminating a connection
SEQUENCE	0	Sequence number that is assigned to services
TRACELEVEL	4	Trace level

IMS log reader service

The IMS log reader service is assigned to the LRSI service class. The task name for the LRSI service class is CECLRS.

The IMS log reader service must communicate with the capture service. For communication between these services to occur successfully, the IMS log reader service and the capture service must be defined in the same Classic data server.

The IMS log reader service processes stream requests received from the capture service. Stream requests enable the IMS log reader service to return IMS change data from the necessary log streams.

Restriction: Only one instance of an IMS log reader service can be defined in a single Classic data server.

The following table lists the configuration parameters that apply to the LRSI service class.

Table 41. Configuration parameters for the LRSI service class.

Parameter	Default value	Description
INACTTHRESHOLD	30S	Interval time for issuing the inactivity message CECZ0400W for a given subsystem.
INITIALTHREADS	1	Number of instances of this service that the region controller starts during initialization of the Classic data server
NOTIFICATIONURL	None	TCP/IP address and port that the log reader interface uses for event notification. This parameter is defined as a 64-byte character string.
SEQUENCE	0	Sequence number assigned to services
SSIDEXCLUDELIST	None	Represents the IMS subsystem exclusion list for the log reader service.
TRACELEVEL	4	Trace level

Logger service

The logger service is assigned to the LOG service class. The task name for the LOG service class is CACLOG.

The logger service receives messages from all services in the data server and coordinates writing the messages to a common log. The logger also reports data server activities and is used for error diagnosis.

Restriction: A single logger task can run within a data server.

When the logger service is initialized, the configuration parameters in the LOG service class are defined with the parameter default values. You can modify the default values as needed.

The following table lists the configuration parameters that apply to the LOG service class.

Table 42. Configuration parameters for the LOG service class

Parameter	Default value	Description
CONSOLELEVEL	4	The amount of event messages that data server tasks record in the event log
DISPLAYLOG	FALSE	Display log
EIFEVENTSERVERS*	None	The service list for URL values that identifies the event server to which Event Integration Facility (EIF) events will be sent
EVENTLOG	None	The name of the event message log that is defined to the logger service
IDLETIMEOUT	5M	The amount of time that a service remains idle before it polls the local message queue for messages that are to be processed
INITIALTHREADS	1	Number of instances of this service that the region controller starts during data server initialization
LOGBUFSIZE	65536	Log buffer size
LOGURL	None	Log URL that provides a method of moving logging storage outside of the address space MESSAGEPOOLSIZE storage and into a data space.
MAXTHREADS	1	Maximum number of instances of this service that the region controller is allowed to start
MAXUSERS	100	Maximum number of user connections
MSGLIST	None	Represents a message list that is maintained as a service list.
RESPONSETIMEOUT	5M	Maximum amount of time to wait for response before terminating a connection
SEQUENCE	0	Sequence number that is assigned to services
STREAMNAME	None	Stream name used for the diagnostic log
TRACELEVEL	1	Trace level
* The EIFEVENTSERVERS parameter does not apply to Classic change data capture.		

Monitoring service

The monitoring service is assigned to the MAA service class. The task name for the MAA service class is CECMAA.

The monitoring service runs in address space of the source server. The purpose of this service is to manage all monitoring of the progress of replication for the source server.

The monitoring service reports metrics about replication data and latency information. Other management functions include reporting, receiving display and report commands, and producing the reports needed from runtime information.

The following table lists the configuration parameters that apply to the MAA service class.

Table 43. Configuration parameters for the MAA service class.

Parameter	Default value	Description
IDLETIMEOUT	5M	The amount of time that a service remains idle before it polls the local message queue for messages that are to be processed
INITIALTHREADS	1	Number of instances of this service that the region controller starts during initialization of the Classic data server
MAXTHREADS	1	Maximum number of instances of this service that the region controller is allowed to start
MAXUSERS	100	Maximum number of user connections
NMICOMMSTRING*	None	The communication path for the Network Management Interface (NMI) AF_UNIX domain socket.
RESPONSETIMEOUT	3S	Maximum amount of time to wait for response before terminating a connection
SAFEXIT	None	Name of the System Authorization Facility (SAF) system exit
SEQUENCE	0	Sequence number that is assigned to services
TRACELEVEL	4	Trace level

* The NMICOMMSTRING parameter does not apply to Classic change data capture.

Operator service

The command operator service is assigned to the OPER service class. The task name for the OPER service class is CACOPER.

The operator service supports a command operator interface for distributed client applications.

The operator service also handles communications between the Classic data server and the configuration support in the Classic Data Architect. To use the configuration support in the Classic Data Architect, the operator service must be running on the Classic data server.

The following table summarizes the configuration parameters that define command operator services in the OPER service class.

Table 44. Configuration parameters for the OPER service class

Parameter	Default value	Description
IDLETIMEOUT	5M	The amount of time that a service remains idle before it polls the local message queue for messages that are to be processed
INITIALTHREADS	1	Number of instances of this service that the region controller starts during initialization of the Classic data server
MAXTHREADS	1	Maximum number of instances of this service that the region controller is allowed to start
MAXUSERS	100	Maximum number of user connections
RESPONSETIMEOUT	5M	Maximum amount of time to wait for response before terminating a connection
SAFEXIT	None	Name of the System Authorization Facility (SAF) system exit
SEQUENCE	0	Sequence number that is assigned to services
SMFEXIT	None	Name of the System Management Facility (SMF) accounting exit that reports clock time and CPU time for a user session
SQLSECURITY	FALSE	Determines the level of access privilege verification that will be performed on operator commands
TRACELEVEL	4	Trace level

Query processor service

The query processor service is assigned to the QP service class. The task name for the QP service class is CACQP.

The query processor performs table and view maintenance and allows you to access your source data to validate your table mappings.

The query processor accesses and joins information from multiple data sources and performs updates to a single data source.

The following table summarizes the configuration parameters that define query processor services in the QP service class.

Table 45. Configuration parameters for the QP service class

Parameter	Default value	Description
BTREEBUFFS	4	Number of in-memory B-tree buffer caches used before data is written out
CPUGOVERNOR	None	CPU governor
IDLETIMEOUT	5M	The amount of time that a service remains idle before it polls the local message queue for messages that are to be processed

Table 45. Configuration parameters for the QP service class (continued)

Parameter	Default value	Description
INITIALTHREADS	5	Number of instances of this service that the region controller starts during initialization of the Classic data server
JOINTABLES*	4	Join optimization
MAXROWSEXAMINED	0	Maximum rows examined
MAXROWSEXCACTION	1 = RETURN	Maximum rows exceeded action
MAXROWSRETURNED	0	Maximum number of rows returned
MAXTHREADS	10	Maximum number of instances of this service that the region controller is allowed to start
MAXUSERS	20	Maximum number of user connections
RESPONSETIMEOUT	5M	Maximum amount of time to wait for response before terminating a connection
SAFEXIT	None	System Authorization Facility (SAF) system exit
SEQBUFNO*	0	Effective BUFNO value for a given file.
SEQBUFNOCALC*	0	Method to use for determining the BUFNO value when accessing sequential files.
SEQUENCE	0	Sequence number that is assigned to services
SMFEXIT	None	System Management Facility (SMF) accounting exit that reports clock time and CPU time for a user session
STMTRETENTION	0 = SYNCPOINT	Statement retention that defines the behavior of a prepared statement when a commit or rollback operation occurs
TEMPFILESPACE	HIPERSPACE, INIT=8M, MAX=2048M, EXTEND=8M	Temporary file space in megabytes
TRACELEVEL	4	Trace level
USERSUBPOOLMAX	8192	User pool maximum
VSAMAMPARMS	None	VSAM buffering information for VSAM files
WLMUOW	None	Workload Manager (WLM) unit of work activities

* The JOINTABLES, SEQBUFNO, and SEQBUFNOCALC parameters do not apply to Classic change data capture.

Refresh query processor service

A refresh query processor handles refresh processing for your subscriptions.

Note: This service does not apply to publishing changes for delimited format subscriptions to WebSphere MQ.

The following table summarizes the configuration parameters that define the refresh query processor services in the QPLD service class.

Table 46. Configuration parameters for the QPLD service class

Parameter	Default value	Description
BTREEBUFFS	4	Number of in-memory B-tree buffer caches used before data is written out
IDLETIMEOUT	5M	The amount of time that a service remains idle before it polls the local message queue for messages that are to be processed
MAXUSERS	100	Maximum number of user connections
RESPONSETIMEOUT	5M	Maximum amount of time to wait for response before terminating a connection
SEQUENCE	0	Sequence number that is assigned to services
TEMPFILESPACE	HIPERSPACE, INIT=8 M, MAX=2048 M, EXTEND=8 M	Temporary file space in megabytes
TRACELEVEL	4	Trace level
USERSUBPOOLMAX	8192	User pool maximum
WLMUOW	None	Workload Manager (WLM) unit of work activities

Region controller service

The region controller service is assigned to the CNTL service class. The task name for the CNTL service class is CACCNTL.

The region controller service monitors and controls the other services that run within the Classic data server.

The region controller directly or indirectly activates each service according to the configuration parameters that you define. The region controller starts, stops, and monitors the other tasks that run within the Classic data server.

The region controller also includes an IBM z/OS MTO (master terminal operator) interface that you can use to monitor and control an address space for a Classic data server.

The following table lists the configuration parameters for the CNTL service class.

Table 47. Configuration parameters for the CNTL service class

Parameter	Default value	Description
DBCSCODEPAGE	0	Double-byte CCSID that the z/OS operating system uses where the Classic data server is running.
HOSTCODEPAGE	37	Host code page of the Classic data server
IDLETIMEOUT	5M	The amount of time that a service remains idle before it polls the local message queue for messages that are to be processed
INITIALTHREADS	1	Number of instances of this service that the region controller starts during initialization of the Classic data server

Table 47. Configuration parameters for the CNTL service class (continued)

Parameter	Default value	Description
MAXTHREADS	1	Maximum number of instances of this service that the region controller is allowed to start
MAXUSERS	100	Maximum number of user connections
RESPONSETIMEOUT	5M	Maximum amount of time to wait for response before terminating a connection
SEQUENCE	0	Sequence number that is assigned to services
TRACELEVEL	4	Trace level

Chapter 15. Change messages and formats

In Classic CDC, change messages can be published in delimited format. Delimited messages are written to WebSphere MQ Series using MQMD version 2.

Specifying delimited messages

When you want Classic CDC to publish changes in a delimited message format, you define configuration parameters to the capture service.

The following configuration parameters allow you to specify the character to use, or its hexadecimal equivalent, for each delimiter type:

- COLDELIMITER for column delimiters
- ROWDELIMITER for row delimiters
- STRINGDELIMITER for string delimeters

The delimited parameters bind when the first delimited format subscription starts. Any changes to these parameters are ignored until all delimited format subscriptions stop. After all subscriptions stop and the cache is cleared, the next delimited format subscription causes a rebind.

The following rules apply to defining a delimiter:

- The length of the required delimiters (column delimiter, row delimiter, string delimiter) must be set to a valid value with a minimum length of one character
- The maximum length of a delimiter parameter is 8 characters
- A delimiter value must be unique.

  The same value cannot be defined for more than one type of delimiter type for the same service. For example, if you define the string delimiter for the capture service as a double quote ("), the column and row delimiters defined for the capture service cannot contain a ". If you define the string delimiter as multiple double quotes (""), the column and row delimiters cannot contain a "" or ".
- For hexadecimal values, you must specify an even number of hexadecimal constants

Restrictions

You cannot use the following characters or hexadecimal equivalents to create a delimiter:

- Upper and lower case alpha characters ('A'-'Z', 'a'-'z')
- Digits ('0'-'9')
- The space character (' ')
- The period character ('.')
- The negative and positive sign characters ('-', '+')
- Mixed mode shift-out and shift-in characters (x'0e' and x'0f')
- Double-byte characters

Any other printable or non-printable SBCS character is allowed. Printable characters are recommended.

Format of delimited messages

Delimited messages represent changed records as fields that are separated by a character, such as a comma.

You can delimit character data with characters such as double quotation marks, and delimit rows with characters such as newline characters. You specify which character to use for delimiting data by using the COLDELIMITER, ROWDELIMITER and STRINGDELIMITER parameters.

The following table describes the format of a delimited message.

Table 48.

Column position	Column type	Description
1	Integer	Used only by IBM. The value "9" identifies delimited messages.
2	Char	Indicates whether valid or invalid data was detected while processing the column. The literal value "IBM" indicates that the column contains valid data. Otherwise, the value is an error flag that identifies the first column that contains badly formed data. You specify the option to flag invalid data by defining the PUBBDCOMPAT configuration parameter.
3	Char	Date, in YYYYDDD format, that the capture service wrote the record to the WebSphere MQ message queue. This is the unit of recovery (UOR) commit date obtained from the source database system.
4	Char	For IMS, this value represents the IMS commit time. For other data sources, the value represents the date and time of when the commit was reported to the capture service. The date and time are obtained from the source database system.
5	Char	Schema name of the table or view.
6	Char	Name of the table or view.
7	Char	Indicates the operation that took place at the data source. **ISRT** Insert **REPL** Replace or update **DLET** Delete
8	Char	Indicates whether the record contains an after image or before image, if the operation was a replace or update. **A** After image **B** Before image

Table 48. (continued)

Column position	Column type	Description
9	Data column 1, data column 2, ... data column *n*	The values of the columns of the row inserted, deleted, or modified. Each column is separated by a single column delimiter (COLDELIMITER). • Each column with a data type of CHAR, VARCHAR, LONG VARCHAR, GRAPHIC, LONG VARGRAPHIC, VARGRAPHIC is enclosed in a user-specified string delimiter (STRINGDELIMITER). • Each column with a data type of BINARY or VARBINARY is represented as a hexadecimal encoded string enclosed in a user-specified string delimiter (STRINGDELIMITER). • Numeric data has no character delimiters.

When a delimited message contains badly formed data, you can define configuration parameters to control how the invalid data is processed. The PUBBDACTION parameter identifies the action to take when badly formed data is encountered. The following configuration parameters enable this processing:

- PUBBDACTION: Set this parameter to TRUE to publish badly formed data or FALSE to fail when badly formed data is encountered.
- PUBBDCOMPAT: Set this parameter to define the options for publishing badly formed data in hexadecimal format or in a format compatible with Version 9.5
  - 0: Publish in hexadecimal format
  - 1: Version 9.5: No repair, no flag
  - 2: Version 9.5: Repair, no flag
  - 3: Version 9.5: Repair and flag
- PUBBDDIAGLIMIT: Set this parameter if you want to limit the number of diagnostic messages issued for each table mapping.
- PUBBDWTOLIMIT: Set this parameter if you want to limit the number of WTO messages issued for all table mappings.

Example

Table TEST.EMPLOYEE has the following fields: First Name, Last Name, Position, Department, Salary, Commission. A message contains two change records: one insert and one update. In this example, the column delimiter is a comma, the row delimiter is a newline character, and the string delimiter is a double quotation mark.

```
9,"IBM  ","2014346","1523180005","TEST","EMPLOYEE","ISRT","A",100,"John","Doe","MGR","SALES",
120000,12000
9,"IBM  ","2014346","1523180305","TEST","EMPLOYEE","REPL","B",110,"Ed","Son","MGR","SALES",
109000,10000
9,"IBM  ","2014346","1523180305","TEST","EMPLOYEE","REPL","A",110,"Ed","Son","MGR","SALES",
129000,12000
```

Figure 7. Two example change records in the delimited format

The following examples show delimited output with the invalid data flag based on the value of the PUBDCOMPAT configuration parameter.

- PUBDCOMPAT: 0

  In this example, column 5 is flagged (IBM-INVALID-COLUMN-5-A) because the salary data contains spaces (X404040).

  ```
  9,"IBM-INVALID-COLUMN-5-A","2014346","1523180005","TEST","EMPLOYEE","ISRT","A",120,
  "Jane","Roe","MGR","SALES",X404040,12000
  ```
- PUBDCOMPAT: 1

  In this example, the salary data that contains spaces is treated as a valid numeric field (404.04).

  ```
  9,"IBM", "2014346","1523180005","TEST","EMPLOYEE","ISRT","A",120,
  "Jane","Roe","MGR","SALES",404.04,12000
  ```
- PUBDCOMPAT: 2

  In this example, the badly formed data in column 5 is replaced with -999999.

  ```
  9,"IBM","2014346","1523180005","TEST","EMPLOYEE","ISRT","A",120,
  "Jane","Roe","MGR","SALES",-999999,12000
  ```
- PUBDCOMPAT: 3

  In this example, the numeric error in column 5 is flagged (IBM-INVALID-NUMERIC-005A) and the badly formed data is replaced with -999999.

  ```
  9,"IBM-INVALID-NUMERIC-005A","2014346","1523180005","TEST","EMPLOYEE","ISRT","A",120,
  "Jane","Roe","MGR","SALES",-999999,12000
  ```

  In this example, a new column is added and flagged (IBM-INVALID-COLUMN-007A-HEX). Column 7 is a character column that contains a shift-out character without a corresponding shift-in character (X'0E4060').

  ```
  9,"IBM-INVALID-COLUMN-007A-HEX","2014346","1523180005","TEST","EMPLOYEE","ISRT","A",120,
  "Jane","Roe","MGR","SALES",4000.00,12000,X'0E4060'
  ```

  This example flags both the character column and the numeric column in error.

  ```
  9,"IBM-INVALID-COLUMN-007A-HEX+IBM-INVALID-NUMERIC-005A","2014346","1523180005","TEST","EMPLOYEE","ISRT","A",120,
  "Jane","Roe","MGR","SALES",-999999,12000,X'0E4060'
  ```

Figure 8. Delimited output with the invalid data flag

Segmentation of long messages

If change messages are too long, the data server divides the message into segments and sends the data in multiple physical messages.

Overview

If a message is too long for the message size limit that you configured for the send queue, the data server segments it into a logical message consisting of multiple physical messages.

The data server splits the messages at operation boundaries to keep each insert, update, or delete operation within a single segment.

You specify a value for maximum message size when you create a publishing map. For more information, see the topic "Publishing map properties".

PutApplType values that describe segmentation

When the data server formats a message, it builds an MQ message descriptor that contains metadata fields. For example, the value in the **`MsgSeqNumber`** field maintains the correct message order.

The **PutApplType** field identifies the type of application that put the message on the queue. Classic data event publishing populates the following user-defined values in the **PutApplType** field to demarcate segmented messages and describe the status of segmentation to target applications:

- 0x00100008 : Signifies that the message is complete, and is not part of a chain of segmented messages.
- 0x00100001 : Signifies that the message is the first in a chain of segmented messages.
- 0x00100002 : Signifies that the message is one of any number of messages between the first and the last message in the chain.
- 0x00100004 : Signifies that the message is the last in a chain of segmented messages.

Chapter 16. Configuration parameters for Classic data servers and services

Configuration parameters define settings for Classic data servers and for the services required for source data servers.

The global configuration parameters define server-wide settings. Standard configuration parameters define settings that are common to most services. All other configuration parameters are service-specific.

Global parameters for Classic data servers

Global parameters define configuration values that affect the entire Classic data server. Unlike other configuration parameters, global parameters are not related to a single service.

The following table summarizes the global configuration parameters and lists parameter default values.

Table 49. Global configuration parameters

Parameter	Default value	Description
DATACONVERRACT	0	Data conversion action
DATAVALIDATEACT*	0	Data validation action
DECODEBUFSIZE*	8192	Decode buffer size
FETCHBUFSIZE*	32000	Size of the result set buffer returned to a client application
MESSAGEPOOLSIZE	16777216	Message pool size. This value is set during the installation and customization process.
REPORTLOGCOUNT*	0	Log record limit for badly-formed data or conversion errors
STATICCATALOGS	0	Activates static catalog processing
TASKPARM	None	Specifies runtime options that are passed to subtasks through the IBM z/OS ATTACH macro

* The DATAVALIDATEACT, DECODEBUFSIZE, FETCHBUFSIZE and REPORTLOGCOUNT parameters do not apply to Classic change data capture.

DATACONVERRACT

The DATACONVERRACT parameter identifies the action for the Classic data server to take if a conversion error occurs when it converts numeric data between zoned and packed decimal formats and between binary and packed decimal formats.

Specification

Use: Global configuration parameter for the Classic data server.

Data type: INT

Default: 0 = FAIL

Valid values: 0 - 2

0 = FAIL
Specifies that the query ends with a -4908 return code that indicates non-valid mapped data.

1 = REPAIR
The Classic data server changes non-valid data to -99...99s, and the SQL statement ends successfully with a SQL_SUCCESS_WITH_INFO return code. One or more 002f0002 warning messages are also returned to the client application. A 002f0002 warning message is returned for each column each row that is changed to a value of -99..999s. The Classic data server does not write any log messages to indicate that conversion errors occurred.

Example: The Classic data server changes nonvalid data to the highest possible negative value for the column precision. A column with a precision of DECIMAL(3,0) changes to a value of -999, and a column with a precision of DEC(3,2) changes to a value of -9.99. If you sort the result set, rows with nonvalid data appear last.

2 = REPAIR REPORT
The Classic data server processes the conversion error as the REPAIR option describes. The Classic data server also writes the data conversion error message x002f0001 to the server log for each row that contains one or more conversion errors.

DATAVALIDATEACT

The DATAVALIDATEACT parameter enables additional validation of data types that are not converted from source data to a different SQL data type. This parameter controls whether additional data type validation occurs and identifies the action for the Classic data server to take if a validation error occurs due to badly-formed data.

Specification

Use: Global configuration parameter for the Classic data server.

Data type: INT

Default: 0 = NO VALIDATION

Valid values: 0 - 3

0 = NO VALIDATION
The data is not checked. This is the default

1 = REPAIR
The Classic data server changes non-valid data as described in the table below, and the SQL statement ends successfully with a SQL_SUCCESS_WITH_INFO return code. One or more 002f0002 warning messages are also returned to the client application. A 002f0002 warning message is returned for each column and each row that is changed. The Classic data server does not write any log messages to indicate that conversion errors occurred.

2 = REPAIR REPORT
The Classic data server processes the validation error as the REPAIR option

describes. The Classic data server also writes the data conversion error message x002f0001 to the server log for each row that contains one or more conversion errors.

3 = FAIL
The query ends with a -4908 return code that indicates non-valid mapped data.

Table 50. Data type validation

Data type value	SQL data type	Data validated	Repair value
P	DECIMAL	Packed decimal data	-9
V	VARCHAR	Field length	0: For a negative length value Maximum length: If the length is greater than the maximum length of the field
VB	VARBINARY	Field length	0: For a negative length value Maximum length: If the length is greater than the maximum length of the field
UP	DECIMAL	Packed positive number	-9
UF	INTEGER	Value between 0 and X'7FFFFFFF'	-9
UH	SMALLINT	Value between 0 and X'7FFF'	-9

DECODEBUFSIZE

DECODEBUFSIZE defines the size of the DECODE buffer. This buffer is a staging area that decodes data from the network format into the host local data format.

Description

Data is taken from the FETCH buffer in pieces that are the size that is specified for the DECODE buffer. The data is converted until a single row of data is completely processed and returned to the application. For optimum use, set the DECODE buffer to a size that is at least equivalent to a single row of data.

The DECODEBUFSIZE and FETCHBUFSIZE parameters work together. If the DECODEBUFSIZE is omitted, its value is set to the value of FETCHBUFSIZE. If a value higher than the FETCHBUFSIZE is used, the value of DECODEBUFSIZE is set to the FETCHBUFSIZE. Thus, coordinate the settings of the DECODEBUFSIZE and FETCHBUFSIZE parameters.

Specifications

Use: Global configuration parameter for the Classic data server.

Data type: INT

Default: 8192

Valid values: 4096 - 64000

FETCHBUFSIZE

The FETCHBUFSIZE parameter specifies the size of the result set buffer that is returned to a client application. You specify this parameter in the configuration file for the client application.

Description

When you set the fetch buffer size to 1, single rows of data are returned to the client application.

An appropriate FETCHBUFSIZE depends upon the average size of the result set rows that are sent to the client application and the optimum communication packet size. To improve performance, pack as many rows as possible into a fetch buffer. The default fetch buffer size is generally adequate for most queries.

If FETCHBUFSIZE is set smaller than a single result set row, the size of the actual fetch buffer that is transmitted is based on the result set row size. The size of a single result set row in the fetch buffer depends on the number of columns in the result set and the size of the data that is returned for each column.

The FETCHBUFSIZE and DECODEBUFSIZE parameters work together. If the DECODEBUFSIZE is omitted, its value is set to the value of FETCHBUFSIZE. If a value higher than the FETCHBUFSIZE is used, the value of DECODEBUFSIZE is set to the FETCHBUFSIZE.

You can use the following calculations to determine the size of a result set row in the buffer:

```
fetch buffer row size = (number of data bytes returned) x
(number of columns * 6)
```

Each fetch buffer has a fixed overhead. You can compute the overhead as follows:

```
fetch buffer overhead = 100 + (number of columns * 8)
```

If your applications routinely retrieve large result sets, contact your network administrator to determine the optimum communication packet size. Then, set the FETCHBUFSIZE to a size that accommodates large result sets.

Specifications

Use: Global configuration parameter for the Classic data server.

Data type: INT

Default: 32000

Valid values: 1 - 524288

MESSAGEPOOLSIZE

The **MESSAGEPOOLSIZE** parameter specifies the size of the memory region in bytes for most memory allocation.

Description

Specify a region size that is at least 8 MB lower than the site limit, and use the greater of these values:
- 8 MB higher than the message pool
- 20% higher than the message pool

If the 8 MB gap between the region and the message pool is still not sufficient, increase this difference in increments of 8 MB.

Set the **MESSAGEPOOLSIZE** parameter to the greater of these values:
- 20% less than the region size
- 8 MB below the REGION value or 8 MB below any site limit imposed by exits.

If you increase the value of the **MESSAGEPOOLSIZE** parameter, set the region size higher to maintain the 8 MB gap.

Specification

Use: Global configuration parameter for the Classic data server.

Data type: INT

Default: 16777216

Valid values: 1048576 - upper limit not applicable

REPORTLOGCOUNT

The REPORTLOGCOUNT parameter sets the maximum number of messages written to the log for badly-formed data or conversion errors.

Description

This parameter value prevents excessive logging to the log file for the Classic data server when a large amount of badly-formed data records are processed. The count controls the number of rows in the result set that generate the log messages for a given access to a table or view.

Specification

Use: Global configuration parameter for the Classic data server.

Data type: INT

Default: 0 = No logging limit

Valid values: 0 - 100000

STATICCATALOGS

The STATICCATALOGS parameter activates static catalog processing for the system catalog data sets that are referenced by the Classic data server.

Description

With static catalog processing, the system catalog files are opened once for a query processor task. The system catalog files remain open until that Classic data server is shut down. In normal operating mode, the system catalogs are closed after the required table and column information is retrieved in order to process a query, for each query that is processed by the query processor.

Activate static catalog processing to substantially improve query performance in outer cursor and inner cursor situations when a large number of queries are issued serially.

Close the static catalog when the system is not updating catalogs information. Use this parameter when the Classic data server operates in production mode and the system catalogs are static.

Specification

Use: Global configuration parameter for the Classic data server.

Data type: INT

Default: 0

Valid values: 0 - 1

`0`	Close system catalog files and establish read locks for each query.
`1`	Close system catalog files when the Classic data server is shut down.

TASKPARM

The TASKPARM parameter specifies IBM C runtime options that are passed to system child tasks through the z/OS ATTACH macro.

Description

One common use of this parameter is to pass TCP/IP information to the Communications Interface task. IBM Software Support can provide a current value.

Specifications

Use: Global configuration parameter for the Classic data server.

Data type: CHAR(64)

Default: None

Standard parameters for services

Standard service parameters are available in more than one service, but like service-specific parameters, you can define values separately for each service.

IDLETIMEOUT

IDLETIMEOUT indicates idle time out.

Description

The IDLETIMEOUT value specifies the amount of time that a service remains idle before it polls the local message queue for messages that are to be processed.

Specification

Use: Configuration parameter for the Classic data server that applies to services that close connections after a specific time period.

Data type: TIME

Default values:
- 5M

Valid formats:

*n***MS**	*n* milliseconds
*n***S**	*n* seconds
*n***M**	*n* minutes
*n***H**	*n* hours

Setting the value to zero (0) indicates no time out. However, the setting the value to 0 is not recommended.

INITIALTHREADS

INITIALTHREADS identifies minimum tasks.

Description

The value of INITIALTHREADS specifies the number of instances of this service that the region controller starts during initialization of the Classic data server. You can use the MTO START command to start an occurrence when needed, unless the service already has MAXTHREADS instances.

The default settings for this parameter are adequate for a default data server configuration. It is important to be cautious when you set the INITIALTHREADS parameter to a value that is greater than the default value. The region controller might not be able to start as many threads as the number specified due to system resource restrictions.

Specification

Use: Configuration parameter for the Classic data server that applies to all services.

Data type: INT

Default values:
- 1: For all services (exception: query processor service)
- 5: For the query processor service only

Valid values: 0 and above.

1 If a service must be limited to a single occurrence.

`0`	Indicates to the region controller that occurrences of this service must not be started during initialization of the Classic data server.

MAXTHREADS

MAXTHREADS identifies the maximum number of instances for a service.

Description

The value of MAXTHREADS specifies the maximum number of instances of this service that the region controller can start.

The default settings for this parameter are adequate for a default data server configuration.

Specification

Use: Configuration parameter for the Classic data server.

Data type: INT

Default values:
- 1: For all services (exception: query processor service)
- 10: For the query processor service only

Valid values: 0 and above.

`1`	If a service must be limited to a single instance.
`0`	The currently deployed number of threads remains the same. A new instance is not started.

MAXUSERS

MAXUSERS identifies the maximum number of connections per task.

Description

The MAXUSERS value is the maximum number of connections that are allowed per instance of this service. Set this field to 1 to disable multi-tasking for all instances of this service.

Specification

Use: Configuration parameter for the Classic data server that is common to all services.

Data type: INT

Default value: Varies by service.

10	DRA
20	QP
	QPRR
50	VSMS
100	CNTL

INIT
LOG
MAA
OPER
PAA

Valid values: 1 and above

RESPONSETIMEOUT

RESPONSETIMEOUT specifies the maximum amount of wait time for an expected response.

Specification

Use: Configuration parameter for the Classic data server.

Data type: TIME

Default value: 5M

Valid formats:

*n***MS**	Number of milliseconds
*n***S**	Number of seconds
*n***M**	Number of minutes

If you set the RESPONSETIMEOUT value to zero (0), the response timeout function is disabled.

SEQUENCE

Each service in the Classic data server is assigned a SEQUENCE number. The SEQUENCE parameter controls the order in which the region controller starts services in the Classic data server.

Description

The controller service starts first and is assigned a SEQUENCE value of 1. The logger service starts next and is assigned a SEQUENCE value of 2. You cannot modify the values of these core services.

When you add a service, that service is assigned the next available SEQUENCE value of 3 or higher. You can change the order in which services start by using SEQUENCE to reassign sequence numbers to services. You can modify these SEQUENCE values.

The non-core services can be assigned the same SEQUENCE number value. If this occurs, the services with the same SEQUENCE value will be started in alphabetical order based on service name.

SEQUENCE also affects termination processing for the Classic data server. Services stop in the reverse order that they start.

Specification

Use: Configuration parameter for the Classic data server that is common to all services.

Data type: INT

Default value: 0

Valid values:
- 1: Assigned to controller service
- 2: Assigned to logger service
- 3 - 999: Services are assigned the next available SEQUENCE value in the range of 3 - 999 when the service is added. You can modify these SEQUENCE values.

TRACELEVEL

The TRACELEVEL parameter regulates the amount of information that tasks in the Classic data server record in the trace log.

Specification

Use: Configuration parameter that is common to all services.

Data type: INT

Default values:
- 4: For all services (exception: logger service)
- 1: For the logger service only. This value controls what other services send to the logger service and what the logger service writes to the log.

Valid values: 0 - 20:

20	No trace information generated
12	Identify non-recoverable error conditions
8	Identify all recoverable error conditions
4	Generate warning messages
3	Generate debugging information
2	Generate a detailed trace, but do not include binary buffers.
1	Generate function call information
0	Trace all

Important: Change this parameter only at the request of IBM Software Support. Settings lower than 4 cause response time degradation and higher CPU costs.

Service-specific parameters

Service-specific parameters pertain to a single service, so the values that you define affect only that service.

This section provides an explanation of each configuration parameter. Chapter 14, “Understanding data server services,” on page 155 summarizes the configuration parameters associated with each service.

Note: The configuration parameters for the DRA and refresh query processor services do not apply to delimited format subscriptions.

BOOKMARKQUEUE

The BOOKMARKQUEUE configuration parameter identifies the name of the WebSphere MQ local queue that stores restart information for use in delimited publishing.

Description

You must specify BOOKMARKQUEUE when you want InfoSphere Classic CDC for z/OS to publish changes in a delimited format to WebSphere MQ.

Specify the name of the local queue where bookmark information is stored. This queue must have the following attributes:

DEFPSIST(YES)
: Specifies that messages are logged and survive a restart of the queue manager.

GET(ENABLED)
: Allows the capture service to get messages from the queue.

INDXTYPE(MSGID)
: Enables the queue manager to maintain an index based on message identifiers.

PUT(ENABLED)
: Allows the capture service to put messages on the queue.

DEFSOPT(SHARED)
: Specifies that, by default, multiple reader tasks can read the queue concurrently.

SHARE
: Identifies to the WebSphere MQ queue manager that multiple reader tasks can read this queue concurrently.

The user ID associated with the source server must have the following WebSphere MQ privileges to access the bookmark queue:

- Connect to the queue manager (MQCONN or MQCONNX) on the system where the source server runs.
- Open (MQOPEN)
- Inquire about attributes (MQINQ)
- Put messages (MQPUT)
- Get messages (MQGET)

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: CHAR(48)

Default: None

BTREEBUFFS

The BTREEBUFFS parameter determines the number of B-tree buffer caches in memory that are used before spooling the staged result set to hiperspaces or to physical files.

Description

You can set BTREEBUFFS to override the default value of four. If sufficient memory is available in the MESSAGEPOOLSIZE memory pool, this parameter can be increased, and performance might improve, depending on the size of the result set and whether the result set is ordered or grouped.

Specification

Use: Configuration parameter for the QP and QPLD services.

Data type: INT

Default value: 4

Valid values: 4 - 214730

COLDELIMITER

COLDELIMITER is a required configuration parameter that specifies which character to use for delimiting columns.

Description

You specify COLDELIMITER when you want InfoSphere Classic CDC for z/OS to publish changes in a delimited format.

Specify the value as a hexadecimal number or enclose the value in single quotation marks.

Specifications

In general, delimiter values should be enclosed in single quotes.
- If you use character delimiters and would like to preserve the case you need to enclose the value with single quotes or it will be converted into uppercase. Use of delimiters that contain letters or numbers is not recommended.
- If the delimiter value contains a single quote, you can enclose the value in double quotes.
- If the delimiter value contains a double quote, you can enclose the value in single quotes.
- You cannot define a delimiter string with a mixture of single and double quotes.
- The following escape characters can be specified in single quotes:

\r – carriage return, equivalent to 0X0D
\n – new line character, equivalent to 0X0A
\f – form feed, equivalent to 0X0C
\t – tab, equivalent to 0X09
\v – vertical tab, equivalent to 0X08

- To specify an unsupported escape character, you can enter the hexadecimal representation of the value. You must define hexadecimal values with the prefix "0X". Inclusion of a null character (0x00) in the delimiter value will lead to unpredictable results.

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: CHAR(8)

Default:,

COMMITINTERVAL

The COMMITINTERVAL configuration parameter identifies how often changes are committed to WebSphere MQ for a subscription. Each subscription commits changes independently.

Description

At each interval, the capture 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 user applications.

Changes for a subscription are committed at COMMITINTERVAL frequencies when the interval expires and all changes for the UOR currently being processed are written to WebSphere MQ. The WebSphere MQ transactions for a subscription are also committed if there are no more changes in a subscriptions capture cache regardless of the COMMITINTERVAL.

Finding the best commit interval is a compromise between latency (the delay between the time that transactions are originally committed at the source and finally committed to WebSphere MQ) and the 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.
- To reduce CPU overhead, lengthen the commit interval.

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

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: TIME

Default: 500 MS

Valid values:

The COMMITINTERVAL may be set to a value between 1 millisecond and 1 hour. You can specify the value in the following form:

*n***MS**	*n* milliseconds
nS	*n* seconds
nM	*n* minutes
nH	*n* hours

COMMSTRING

The COMMSTRING parameter specifies the protocol identifier and address for communication.

Description

The connection handler supports TCP/IP. The COMMSTRING value defines the protocol followed by the protocol specific information

- For developing remote client applications with a TCP/IP connection handler, you need the protocol identifier TCP, followed by the IP address of the machine that the Classic data server is running on, and the port number that is assigned to this server as a listen port. For example: TCP/*host-name*/*port-number*.

  The client connection string supports Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6). For example:

  ```
  SET,CONFIG,SERVICE=INIT,COMMSTRING='TCP/0.0.0.0/9087'; (IPv4)
  SET,CONFIG,SERVICE=INIT,COMMSTRING='TCP/::/9087';        (IPv6)
  ```

  If you try to connect to the data server by using IPv6, you might need to provide a scope if you are using a link-local address. The scope is typically the network interface name that you specify following the IPv6 address. The format is `ipv6 address%scope`. For example:

  ```
  SET,CONFIG,SERVICE=INIT,COMMSTRING='TCP/fe80::xxxx:xxxx:xxxx:xxxx%INTF0001/9087';
  ```

Specification

Use: Configuration parameter for the connection handler service.

Service class: INIT

Service task: CACINIT

Data type: CHAR(64)

Default: TCP/0.0.0.0/9087

CONNECTINTERVAL

The CONNECTINTERVAL parameter defines the frequency at which the DRA service retries connecting to IMS when IMS is not available.

Description

The default value for this parameter is 15S (seconds). This value indicates that the DRA service will retry failed connections to IMS every 15 seconds.

Specifications

Use: Configuration parameter for the DRA service.

Service class: DRA

Service task: CACDRA

Data type: TIME

Default: 15S

Valid values:

*n***MS**	*n* milliseconds
*n***S**	*n* seconds
*n***M**	*n* minutes
*n***H**	*n* hours

CONSOLELEVEL

CONSOLELEVEL is a required parameter that controls when event messages are sent to the z/OS console.

Description

All event messages are written to the event log and the system trace. The value of the CONSOLELEVEL parameter controls when event messages are also routed to the z/OS console. When the trace level of an event message equals or exceeds the value specified for the CONSOLELEVEL parameter, that message is sent to the console.

Specifications

Use: Configuration parameter for the logger service.

Service class: LOG

Task name: CACLOG

Data type: INT

Default: 4

Valid values:

20 Generates no event messages to the console. Messages are written to the event log.

4 Generates the following event messages:
- Subscription group operation messages issued from the capture and administration services
- Stream activation and destruction messages issued from the capture and log reader services

Specifying a value less than 4 generates more console messages and increases the number of messages in the console buffers.

3 Generates the following event messages:
- Replication messages issued from the the capture and administration services
- Table, view, and DBMS object cache operation messages issued from the administration service
- SSID start and stop messages issued from the IMS log reader service

2 Generates the following event messages:
- Open and close messages issued from the IMS log reader service
- Roll-off cache maintenance messages issued from the capture service

0 Writes all event messages to the console.

CPUGOVERNOR

The CPUGOVERNOR parameter specifies the name and the time limit for the exit to implement a CPU resource governor. If this parameter is omitted, there is no limit to the amount of CPU time for query processing.

Specifications

Use: Configuration parameter for the query processor services.

Service class: QP

Service task: CACQP

Data type: CHAR(64)

Default: None

CSDATAVALIDATEAC

The CSDATAVALIDATEAC parameter enables additional validation of data types that are not converted from source data to a different data type when replicating data.

Description

This parameter controls whether additional data type validation occurs and identifies the action for the source server to take if a validation error occurs due to badly-formed data.

Note: The CSDATAVALIDATEAC parameter does not apply to delimited format subscriptions.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Task name: CECCAP

Data type: INT

Default: 0 = NO VALIDATION

Valid values: 0 - 2

0 = NO VALIDATION
The data is not checked. This is the default.

Validation of variable (character and graphic) and zoned decimal data is an exception. The Classic data server always validates variable and zoned decimal data regardless of the CSDATAVALIDATEAC value.

1 = REPAIR
The Classic data server changes invalid data as described in the table below, and the data is replicated with the repaired values. The Classic data server does not write any log messages to indicate that conversion errors occurred.

2 = REPAIR REPORT
The Classic data server processes the validation error as the REPAIR option describes. The Classic data server also writes the data conversion error message x002f0001 to the server log for each row that contains one or more validation errors.

Table 51. Data type validation

Data type value	SQL data type	Data validated	Repair value
P	DECIMAL	Packed decimal data	-9999
V	VARCHAR	Field length	0: For a negative length value Maximum length: If the length is greater than the maximum length of the field
UP	DECIMAL	Packed positive number	-9999
UF	INTEGER	Value between 0 and X'7FFFFFFF'	-999999999
UH	SMALLINT	Value between 0 and X'7FFF'	-9999
C	DECIMAL	Ensures that source data is valid zoned decimal data.	-9999

CSREPORTLOGCOUNT

The CSREPORTLOGCOUNT parameter sets the maximum number of messages written to the log for badly-formed data or conversion errors.

Description

This parameter value prevents excessive logging to the Classic data server log file when a large amount of badly-formed data records are processed. The count controls the number of log records produced when the Classic data server is configured to report badly-formed data.

Note: The CSREPORTLOGCOUNT parameter does not apply to delimited format subscriptions.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Task name: CECCAP

Data type: INT

Default: 0 = No logging limit

Valid values: 0 - 100000

DBCSCODEPAGE

The DBCSCODEPAGE parameter identifies the double-byte CCSID that the z/OS operating system uses where the Classic data server is running.

Description

This parameter defines the code page for double-byte data when you define the HOSTCODEPAGE parameter as a multi-byte code page. Classic change data capture communicates this value to the target server as the code page for double-byte data that is mapped in graphic columns.

Specifications

Use: Configuration parameter for the region controller service.

Service class: CNTL

Task name: CACCNTL

Data type: INT

Default: 0

Valid values: 0 - 65535

DEFAULTPSBNAME

The DEFAULTPSBNAME parameter identifies a default PSB name.

Description

The PSB name is used when a CREATE TABLE statement references an IMS table that contains no PSB name.

Specification

Use: Configuration parameter for the DRA access service.

Service class: DRA

Task name: CACDRA

Data type: CHAR(8)

Default value: NOPSB

DISPLAYLOG

The DISPLAYLOG parameter allows you to view log messages for the logger service.

Description

This parameter controls whether log records are mirrored to the data set that is specified in the SYSOUT DD statement. The default data set is the system output data set (SYSOUT).

Specifications

Use: Configuration parameter for the logger service.

Service class: LOG

Task name: CACLOG

Data type: Boolean

Default: FALSE

DRATABLESUFFIX

The DRATABLESUFFIX parameter identifies the suffix of the DRA startup table.

Description

Use DRATABLESUFFIX to specify the suffix of the load module name that you created for IMS DRA initialization.

Specification

Use: Configuration parameter for the IMS DRA access service.

Service class: DRA

Task name: CACDRA

Data type: CHAR(3)

Default value: None

DRAUSERID

The DRAUSERID parameter identifies the DRA user ID.

Description

Use DRAUSERID to specify the default DRA user ID to use for connecting to and registering with DBCTL. The DRA user ID is the name by which the Classic data server is known to the IMS database manager subsystem, DBCTL.

Specification

Use: Configuration parameter for the IMS DRA access service.

Service class: DRA

Task name: CACDRA

Data type: CHAR(9)

Default value: None

EVENTLOG

EVENTLOG is an optional parameter identifies the name of the event message log that is defined to the logger service.

Description

The logger service writes event messages to the log specified in the EVENTLOG parameter. This parameter identifies a z/OS system log stream. The event log file cannot be shared among multiple Classic data servers. You can use one event log file with one Classic data server.

If you do not specify the EVENTLOG parameter, event messages are not captured. Event messages will not be available for retrieval by the Management Console. In this case, the Classic data server formats event messages to SYSPRINT and incurs processing overhead at runtime.

Specifications

Use: Configuration parameter for the logger service.

Service class: LOG

Task name: CACLOG

Data type: CHAR (26)

Default: None

HEARTBEATTIMEOUT

HEARTBEATTIMEOUT is an optional parameter that controls whether or not heartbeat messages are enabled.

Description

When enabled, the HEARTBEATTIMEOUT parameter defines the heartbeat interval at which messages are sent to any connected target engines.

Heartbeats allow IBM InfoSphere Classic Change Data Capture for z/OS to detect the failure of a target engine. Classic change data capture performs a graceful shutdown of the subscriptions replicating to the failed target server.

The source and target servers exchange heartbeat messages and expect a response from the partner within the negotiated heartbeat timeout interval that both environments define.

Any positive integer assigned to HEARTBEATTIMEOUT that is within the acceptable range enables heartbeats and specifies the heartbeat timeout period in minutes. A value of zero (0) disables heartbeats.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Task name: CECCAP

Data type: TIME

Default: 15M

Valid values:

3M - 999M
Positive integer between 3 minutes and 999 minutes. 15 minutes is the default.

0 Disables heartbeats.

HOSTCODEPAGE

The HOSTCODEPAGE parameter identifies the CCSID that the z/OS operating system uses where the data server is running.

Description

Classic change data capture communicates this value to the target engine to control how code page conversion occurs for replicated data.

This value is also used for delimited format subscriptions when the PUBUTF8 parameter is set to TRUE. When UCS conversion is enabled, the delimited messages are converted from the HOSTCODEPAGE to code page 1208.

Specification

Use: Configuration parameter for the region controller service.

Service class: CNTL

Service tasks: CACCNTL

Data type: INT

Default: 37

Valid values: 0 - 99999

IMSRECID

The IMSRECID configuration parameter identifies the name of the table identification exit load module that controls the capture of changes made to IMS data.

Description

If you develop a table identification exit to control publishing of your IMS data, you need to use the IMSRECID configuration parameter to activate loading and calling the exit. See "Table identification exit" on page 139 for more information about using and creating the exit.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: CHAR(8)

Default: None

IMSRECSELECT

The IMSRECSELECT configuration parameter identifies the name of the record selection exit load module that controls the capture of changes made to IMS data.

Description

If you develop a record selection exit to control publishing of your IMS data, you need to use the IMSRECSELECT configuration parameter to activate loading and calling the exit. See the topic "Developing a record selection exit for multiple record layouts" on page 136 for more information about using and creating the exit.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: CHAR(8)

Default: None

INACTTHRESHOLD

INACTTHRESHOLD is an optional parameter that defines the frequency for issuing the inactivity message CECZ0400W for a given subsystem.

Description

In an IMS data sharing environment, changes are captured from multiple DB/DC or DBCTL subsystems. The CECZ0400W message is issued in situations where the activity of one or more subsystems is not detected for an extended period of time.

The value that you specify for the INACTTHRESHOLD parameter controls how frequently the CECZ0400W message is issued. You can specify any value, in seconds, between the minimum value of 1 and the maximum value of 86400 (24 hours). The new value takes effect after the previous INACTTHRESHOLD value is reached.

Specifications

Use: Configuration parameter for the IMS log reader service.

Service class: LRSI

Task name: CECLRS

Data type: INT

Default: 30s

Valid values: 1s - 86400s

LOGBUFSIZE

The LOGBUFSIZE parameter defines the size of the log buffer.

Specifications

Use: Configuration parameter for the logger service.

Service class: LOG

Task name: CACLOG

Data type: INT

Default: 65536

Valid values: 4096 - 1024000

LOGURL

The LOGURL parameter identifies the communication protocol for the logger service.

Description

You can use LOGURL to override the protocol defined for local queues. This parameter is typically used for XM queues.

Specifications

Use: Configuration parameter for the logger service.

Service class: LOG

Task name: CACLOG

Data type: CHAR(32)

Default: None

Example

The following sample command sets the value of the **LOGURL** parameter for a logger service with the name LOG.

```
F <Data-Server-Name>,SET,CONFIG,SERVICE=LOG,LOGURL=XM1/DSLG1/LOGQ1/256
```

MAXROWSEXAMINED

The MAXROWSEXAMINED parameter implements the governor.

Description

MAXROWSEXAMINED provides protection from excessive resource use that inefficient or erroneous queries cause. The governor limits the number of examined rows. In the examination phase, a restriction on the number of examined rows is put into effect after native retrieval is performed and before additional filtering takes place to satisfy any WHERE clause specifications.

In general, you should set the value of MAXROWSEXAMINED to the default value or to meet the requirements of the table size. You can use a larger value of roughly 10000 in a test environment.

Specification

Use: Configuration parameter for the query processor services.

Service class: QP

Service task: CACQP

Data type: INT

Default: 0

Valid values: 0 - 2147483647

MAXROWSEXCACTION

The MAXROWSEXCACTION parameter determines the behavior of the governor when the MAXROWSEXAMINED or the MAXROWSRETURNED governor limits are reached.

Description

The MAXROWSEXCACTION parameter tests a query to ensure that the query returns the correct result set and does not expend large amounts of resources. This test is performed before the full query is run.

Specification

Use: Configuration parameter for the query processor services.

Service class: QP

Service task: CACQP

Data type: INT

Default: 1

Valid values: 0 - 1

Valid values and results:

0 = ABORT
Stops the query when a governor limit for MAXROWSEXAMINED or MAXROWSRETURNED is reached. The -9999 return code is issued.

1 = RETURN
Returns a normal result set when a governor limit is reached. A truncated result set is returned to the client. The result might not be complete, and there is no indication that the governor limit is reached.

MAXROWSRETURNED

The MAXROWSRETURNED parameter specifies the maximum number of rows a query can return to the client. The governor imposes the restriction you specify after any WHERE clause is fully processed, but before the result table is returned to the application.

Description

MAXROWSRETURNED provides protection from excessive resource use that inefficient or erroneous queries cause.

Specifications

Use: Configuration parameter for the query processor services.

Service class: QP

Service task: CACQP

Data type: INT

Default: 0

Valid values: 0 - 2147483647

MAXUSERS

MAXUSERS identifies the maximum number of connections per task.

Description

The MAXUSERS value is the maximum number of connections that are allowed per instance of this service. Set this field to 1 to disable multi-tasking for all instances of this service.

Specification

Use: Configuration parameter for the Classic data server that is common to all services.

Data type: INT

Default value: Varies by service.

10	DRA
20	QP QPRR
50	VSMS
100	CNTL INIT LOG MAA OPER PAA

Valid values: 1 and above

MSGLIST

The MSGLIST parameter is maintained as a service list. You maintain service list parameters by using service list commands.

Description

You specify a message list as *message-number/destination*:

Message-number
The message number must begin with prefix CEC and contain nine characters.

Destination
You can specify one of the following destinations:

- CONSOLE: z/OS console
- DIAGLOG: Diagnostic trace log
- EVENT: Event log
- SUPPRESS: No destination

The service list configuration commands create or update the destination (or suppression) for a particular message based on message IDs. You can change a message destination to another destination. The destination hierarchy is as follows:

- CONSOLE: The message is routed to the console, event log, and diagnostic log.
- EVENT: The message is routed to the event log and the diagnostic log.
- DIAGLOG: The message is routed to the diagnostic log only.
- SUPPRESS: The message does not appear in any destination.

Changes to this service list are effective immediately.

Specification

Use: Configuration parameter for the logger service.

Service class: LOG

Task name: CACLOG

Data type: CHAR

Default: None

NOTIFICATIONURL

NOTIFICATIONURL is a required parameter that identifies the TCP/IP address and port used for event notification.

Description

The TCP/IP address and port specified for the NOTIFICATIONURL parameter and for the configuration table module (CECE1OPT) for the notification exit must be the same.

Specifications

Use: Configuration parameter for the IMS log reader service.

Service class: LRSI

Task name: CECLRS

Data type: CHAR (64)

Default: None

PUBBDACTION

The PUBBDACTION configuration parameter identifies the action for the capture service to take when badly formed data is encountered.

Description

Note: The PUBBDACTION parameter applies to delimited format subscriptions only.

You can set the value of PUBBDACTION to TRUE to publish badly formed data or set the value to FALSE to fail when badly formed data is encountered.
- When you specify TRUE, subscription publishing continues when badly formed data is encountered. TRUE is the default value.
- When you specify FALSE, the table row that contains badly formed data is not published and the capture service stops replication for the subscription.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: Boolean

Default: TRUE

Valid values: TRUE or FALSE

PUBBDCOMPAT

The PUBBDCOMPAT configuration parameter identifies whether to publish badly formed data in hexadecimal format or in a format that is compatible with version 9.5 of InfoSphere Classic Data Event Publisher for z/OS.

Description

Note: The PUBBDCOMPAT parameter applies to delimited format subscriptions only.

Beginning in version 11.3, any badly formed data that is encountered in a table row is published in hexadecimal format by default.

The PUBBDCOMPAT configuration parameter provides a compatibility option for existing customers to continue handling badly formed data in the same way that it was handled in version 9.5.

Recommendations:
- New customers should keep the default value (0) to publish badly formed data in hexadecimal format.
- Existing customers should keep the default value if their consuming applications are not designed to handle the way badly formed data is flagged and published in version 9.5.
- Existing customers who have consuming applications that are designed to handle badly formed data in the way it is flagged and published in version 9.5 should specify option 1, 2, or 3 to identify which format their applications are designed to work with.

Important: Regardless of the PUBBDCOMPAT setting, badly formed data in non-numeric format is always published as a hexadecimal value preceded by an "X" identifier.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: INT

Default: 0

Valid values:

0: Default

Badly formed data is published in hexadecimal format preceded by an "X".

Flag Insert the invalid data flag `IBM-INVALID-COLUMN-`*n*`-`*i* into published output.

n The first column in error. This identifier applies to badly formed data in numeric and non-numeric format.

i Indicates whether the record contains a before image (B) or an after image (A).

1: Version 9.5, No repair, No flag

The publishing format of badly formed data is compatible with version 9.5 of InfoSphere Classic Data Event Publisher for z/OS. Invalid data is not repaired or flagged.

No repair

Do not replace invalid numeric data with -9's.

Note: If the original numeric value cannot be displayed, an exception occurs and -9's are displayed. However, diagnostic messages in the server log will contain the original value in hexadecimal format.

No flag

Do not insert the invalid data flag into published output.

2: Version 9.5, Repair, No flag

The publishing format of badly formed data is compatible with version 9.5 of InfoSphere Classic Data Event Publisher for z/OS. Invalid data is repaired but not flagged.

Repair

Replace invalid numeric data with -9's.

No flag

Do not insert the invalid data flag into published output.

3: Version 9.5, Repair, Flag

The publishing format of badly formed data is compatible with version 9.5 of InfoSphere Classic Data Event Publisher for z/OS. Invalid data is repaired and flagged.

Repair

Replace invalid numeric data with -9's.

Flag Insert the invalid data flag `IBM-INVALID-COLUMN-`*nnni*`-HEX` or

`IBM-INVALID-NUMERIC-`*`nnni`* into published output. If a table row contains badly formed numeric data and non-numeric data, both invalid data flags are shown: `IBM-INVALID-COLUMN-`*`nnni`*`-HEX+IBM-INVALID-NUMERIC-`*`nnni`*.

nnn	Identifies the 3-digit number of the column that contains badly formed data.
i	Indicates whether the record contains a before image (B) or an after image (A).

When the length of the column number is less than 3 digits, the number is prefixed with zeroes. For example, `IBM-INVALID-NUMERIC-005A`.

Table 52. Data type validation

Data type value	SQL data type	Data validated	Repair value
P	DECIMAL	Packed decimal data	-9 or hexadecimal
V	VARCHAR	Field length	0: For a negative length value Maximum length: If the length is greater than the maximum length of the field
UP	DECIMAL	Packed positive number	-9 or hexadecimal
UF	INTEGER	Value between 0 and X'7FFFFFFF'	-9 or hexadecimal
UH	SMALLINT	Value between 0 and X'7FFF'	-9 or hexadecimal

PUBBDDIAGLIMIT

The PUBBDDIAGLIMIT configuration parameter controls the number of diagnostic messages generated for each table mapping when badly formed data is encountered.

Description

Note: The PUBBDDIAGLIMIT parameter applies to delimited format subscriptions only.

The capture service generates a pair of messages in the diagnostic log for each table row that contains badly formed data. The first message displays the content of the table columns that contains badly formed data. The second message displays the entire delimited table row in the form in which it was mapped.

You can limit the number of diagnostic messages by setting the value of PUBBDDIAGLIMIT to a maximum limit. The default value is 100 message pairs.

To disable limit checking, you can set the PUBBDDIAGLIMIT value to 0.

Example

The following example shows a pair of diagnostic messages generated for badly formed data.

The trace header contains a specific return code (SpcRC) and a data field. The data field contains two values:

- The first field identifies the operation: 1: Delete, 2: Insert, 3: Update.
- The second field contains the image identifier: B4: Before image, AF: After image.

Each data element within a message is displayed on a new 16-byte line. If the length of a data element is greater than 16 bytes, the content of that element continues on the next line.

First message

```
2014-03-21-13.56.48.036458 RC(04), SpcRC(002f0004), Data(00000003,000000B4)
   Node(54), Task(9076112)
   QP, func fnGenBadDataDiagMsg, line 2572 in SYS14080.T114038.RA000.RRINKECI.SRCC.H01(AABCCDCP)
   0000 * e3c5e2e3 c2c5c44b e3c2f7f7 f2f9f06d *  * TESTBED.TB77290_ *
   0010 * f0f14040 40404040 40404040 40404040 *  * 01               *
   0020 * c4c5c3c3 d6d3f140 40404040 40404040 *  * DECCOL1          *
   0030 * a7f4f0f4 f0f4f0                     *  * X404040          *
```

The first diagnostic messages includes the following characteristics:

- The message is identified by a specific return code: SpcRC 002f0004.
- The Data field identifies an update operation: 00000003, and before image: 000000B4.
- The first data element is the associated *schema.table* name that contains the badly formed data: TESTBED.TB772901_01.
- The remaining elements consist of name/value pairs for columns that contain badly formed data. The column names are displayed first. In this example, the name of the column in error is DECCOL1.
- Column values are displayed in hexadecimal format prefixed with the identifer "X". Each value is displayed on a separate line. In this example, the column value of the badly formed data is X404040.

Second message

```
2014-03-21-13.56.48.036468 RC(04), SpcRC(002f0005), Data(00000003,000000B4)
   Node(54), Task(9076112)
   QP, func fnGenBadDataDiagMsg, line 2603 in SYS14080.T114038.RA000.RRINKECI.SRCC.H01(AABCCDCP)
   0000 * e3c5e2e3 c2c5c44b e3c2f7f7 f2f9f06d *  * TESTBED.TB77290_ *
   0010 * f0f14040 40404040 40404040 40404040 *  * 01               *
   0020 * f16b7f83 888199a5 8193f140 40404040 *  * 1,"charval1      *
   0030 * 4040407f 6b7ff283 888199a5 8193f140 *  *    ","2charval1  *
   0040 * 40404040 40407f6b a7f4f0f4 f0f4f0   *  *       ",X404040  *
```

The second diagnostic message displays the associated delimited row that contains badly formed data. This message includes the following characteristics:

- The message is identified by a specific return code: SpcRc 002f0005.
- The Data field identifies an update operation: 00000003, and before image: 000000B4.
- The first data element is the associated *schema.table* name that contains the badly formed data: TESTBED.TB772901_01.
- The remaining data represents the associated delimited row. The delimited row is displayed on a new 16-byte line. Each line of the delimited row is displayed in hexadecimal format followed by the equivalent character representation of the content. The character representation is shown in the format of the host code page.
- In this example, the value of the badly formed data is ",X404040.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: INT

Default: 100

PUBBDWTOLIMIT

The PUBBDWTOLIMIT configuration parameter controls the number of CACJ080W and CACJ078W console messages generated when badly formed data is encountered.

Description

Note: The PUBBDWTOLIMIT parameter applies to delimited format subscriptions only.

When the capture service encounters badly formed data, the following sets of messages are issued.

- For numeric data:

  CACJ080W INVALID NUMERIC DATA DETECTED FOR SUBSCRIPTION <*Subscription*> AND TABLE <*Table*> PUBLISHING TO SEND QUEUE <*queue*>

  CACJ081I LIMIT OF CACJ080W MESSAGES HAS BEEN REACHED.
- For UCS conversion errors:

  CACJ078W CONVERSION FAILURE DETECTED FOR SUBSCRIPTION <*Subscription*> AND TABLE <*Table*> PUBLISHING TO SEND QUEUE <*queue*> MESSAGE CONVERTED TO HEXADECIMAL

  CACJ079I LIMIT OF CACJ078W MESSAGES HAS BEEN REACHED.

The PUBBDWTOLIMIT parameter controls the issuance of both sets of messages. You can limit the number of CACJ080W and CACJ078W warning messages by setting the value of PUBBDWTOLIMIT to a maximum limit. This is a server-wide limit across all subscriptions. The default value is 1000.

The PUBBDWTOLIMIT value applies to each message type individually. When the PUBBDWTOLIMIT is reached, the informational console message CACJ080I is issued to notify users that no additional CACJ080W messages will be generated and message CACJ079I is issued to notify users that no additional CACJ078W messages will be generated.

For example, if you set PUBBDWTOLIMIT to 10, you can receive up to 10 CACJ080W messages and up to 10 CACJ078W messages before the CACJ081I and CACJ079I messages are generated.

When badly formed data is encountered, it is likely that multiple rows within a table mapping will be impacted. To prevent redundant console messages, only one CACJ080W or CACJ078W message is issued per table mapping.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: INT

Default: 1000

PUBUTF8

Setting the PUBUTF8 parameter to TRUE changes the code page that is used for publishing messages to 1208 (UTF-8).

Description

Note: The PUBUTF8 parameter applies to delimited format subscriptions only.

By default, the capture service publishes messages that use the code page specified by the HOSTCODEPAGE configuration parameter. If you want the capture service to publish messages that use the 1208 (UTF-8) code page, set the value of PUBUTF8 to TRUE.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: Boolean

Default: False

QUEUEMGRNAME

The QUEUEMGRNAME configuration parameter identifies the name of the WebSphere MQ queue manager that is used for delimited publishing.

Description

You specify QUEUEMGRNAME when you want InfoSphere Classic CDC for z/OS to publish changes in a delimited format to WebSphere MQ.

Specify the name of the queue manager or the name of the queue-sharing group.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: CHAR(4)

Default: None

RECONNECTWAIT

The RECONNECTWAIT parameter defines the minimum amount of time that the DRA service waits after an IMS disconnect before attempting to reconnect to IMS.

Description

This wait time ensures that the DRA service does not try to reconnect to IMS while IMS is still in the process of stopping and prevents abends in the DBCTL interface.

The RECONNECTWAIT parameter is enforced when the console message CAC00137W is issued to ensure that IMS has enough time to stop before the DRA service attempts to reestablish a connection.

The default value for this parameter is 1M (minute). This value indicates that the DRA service will start trying to connect to IMS one minute after a disconnect is received from IMS and message CAC00137W is issued.

Specifications

Use: Configuration parameter for the DRA service.

Service class: DRA

Service task: CACDRA

Data type: TIME

Default: 1M

Valid values:

*n***MS**	*n* milliseconds
*n***S**	*n* seconds
*n***M**	*n* minutes
*n***H**	*n* hours

ROWDELIMITER

ROWDELIMITER is a required configuration parameter that specifies which character to use for delimiting rows.

Description

You specify ROWDELIMITER when you want InfoSphere Classic CDC for z/OS to publish changes in a delimited format.

Specify the value as a hexadecimal number or enclose the value in single quotation marks.

Specifications

"Specifying delimited messages" on page 167 describes the rules that apply to defining a delimiter.

In general, delimiter values should be enclosed in single quotes.

- If you use character delimiters and would like to preserve the case you need to enclose the value with single quotes or it will be converted into uppercase. Use of delimiters that contain letters or numbers is not recommended.
- If the delimiter value contains a single quote, you can enclose the value in double quotes.
- If the delimiter value contains a double quote, you can enclose the value in single quotes.
- You cannot define a delimiter string with a mixture of single and double quotes.
-
- The following escape characters can be specified in single quotes:

  ```
  \r – carriage return, equivalent to 0X0D
  \n – new line character, equivalent to 0X0A
  \f – form feed, equivalent to 0X0C
  \t – tab, equivalent to 0X09
  \v – vertical tab, equivalent to 0X08
  ```
- To specify an unsupported escape character, you can enter the hexadecimal representation of the value. You must define hexadecimal values with the prefix "0X". Inclusion of a null character (0x00) in the delimiter value will lead to unpredictable results.

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: CHAR(8)

Default: \n (new line character)

SAFEXIT

The SAFEXIT parameter specifies the System Authorization Facility (SAF) system exit that performs authorization checks for the administration, monitoring, and console connections to the Classic data server.

Description

The values that you specify for the SAFEXIT parameter control the actions that a user can perform when connected to a Classic data server for the following types of connections:

- Administration connections from IBM InfoSphere Change Data Capture Management Console. You use administration connections to view and edit subscription definitions, start and stop replication, and view system events.
- Monitoring connections established by the management console separately from administration connections. These connections are authenticated at the z/OS host. Monitoring connections also include Network Management Interface (NMI)

connections to the z/OS data server. You use NMI connections to retrieve metrics data and for access to subscription states or statuses.
- Console connections from the Classic Data Architect that allow remote operators to issue console commands to the Classic data server.
- Query processor connections that use the SAF exit to authenticate users who map tables and run test queries.

You specify the optional parameters for the SAF exit in the following format:

```
CACSX04{,optional-parameters...}
```

If you do not specify any optional parameters, the SAF exit load module CACSX04 activates user ID and password authentication when a user connects to a Classic data server. The optional parameters provide additional security checking that the administration service, monitoring service, and operator service perform.

All connections

The following optional parameter for validation of IP addresses applies to all connections:

NETACCESS=Y/N
Indicates whether the exit should validate the IP address of the connected client to authenticate access to the Classic data server.

Set the value to Y when the IP address of the connected client is known and the SERVAUTH parameter of the RACROUTE REQUEST=VERIFY invocation is supplied. The RACROUTE operation is successful when the associated user ID has at least READ-level access rights to the network security zone resource. If the security system indicates that it cannot make a decision in response to the request because a corresponding network security zone resource profile does not exist, the SAF exit regards the response as Access Denied.

A value of N indicates that the SERVAUTH parameter is omitted from the RACROUTE REQUEST=VERIFY invocation. This is the default.

Administration connections

The following optional parameters for the administration service activate security checking for client connections:

VALIDATE=*Y/N*
Indicates whether the SAF exit should perform resource class checking for each connected user. The default value for VALIDATE is Y. If Y is specified, resource access checking occurs when users make requests to either query data from the Classic data server or update information about replication.
- For query requests, READ level access is checked.
- For operations that change subscription information or the state of subscriptions (for example, starting or stopping subscriptions), CONTROL access is checked.

If N is specified, resource class checking is not performed. This is the default.

ADMCLASS=*administrator-class-name*
Indicates the name of the security class that contains a profile that requires access authentication.

This parameter is valid if VALIDATE=Y on the administration service for the SAF exit. If this parameter is not specified, SERVAUTH is the default security class.

ADMPROF=*administrator-profile-name*
: Indicates the name of the resource profile that requires access authentication.

  This parameter is valid if VALIDATE=Y on the administration service for the SAF exit. If this parameter is not specified, CEC.ADMIN is the default profile name.

Monitoring connections

The following optional parameters for the monitoring service activate security checking for client connections from the Classic Data Architect:

VALIDATE=*Y/N*
: Indicates whether the SAF exit should perform resource class checking for each connected user. If Y is specified, resource access checking occurs when users make requests to retrieve metrics information. READ level access is checked. If N is specified, resource class checking is not performed. This is the default.

MONCLASS=*monitor-class-name*
: Indicates the name of the security class that contains a profile that requires access authentication.

  This parameter is valid if VALIDATE=Y on the monitoring service for the SAF exit. If this parameter is not specified, SERVAUTH is the default class name.

MONPROF=*monitor-profile-name*
: Indicates the name of the resource profile that requires access authentication.

  This parameter is valid if VALIDATE=Y on the monitor service for the SAF exit. If this parameter is not specified, CEC.MONITOR is the default profile name.

Console connections

The following optional parameters for the operator service activate security checking for client connections from the Classic Data Architect for issuing console commands:

VALIDATE=*Y/N*
: Indicates whether the SAF exit should perform resource class checking for each connected user. If Y is specified, CONTROL level access is checked when users issue console commands through the remote operator. If N is specified, resource class checking is not performed. This is the default.

OPERCLASS=*operator-class-name*
: Indicates the name of the security class that contains a profile that requires access authentication.

  This parameter is valid if VALIDATE=Y on the operator service for the SAF exit. If this parameter is not specified, SERVAUTH is the default class name.

OPERPROF=*operator-profile-name*

Indicates the name of the resource profile that requires access authentication.

This parameter is valid if VALIDATE=Y on the operator service for the SAF exit. If this parameter is not specified, CEC.OPER is the default profile name.

Query processor connections

The following optional parameters for the query processor service activate authorization checks for data server connections.

VALIDATE=*Y/N*

Indicates whether the exit should validate that the user ID has authority to access a specified database, file, or PSB name. Use both SQL security and the SAF exit in conjunction with your site security package to restrict access to your data.

Set a value of **`Y`** to use RACF security and issue RACROUTE validation calls for specified resource names. Set a value of **`N`** to suppress validation processing for resources. The default value for VALIDATE is **`Y`**.

This parameter helps you to control access with greater precision:

- Ensure that only valid users can access resources by setting VALIDATE=Y against the QP service.

  VALIDATE=Y authenticates each individual resource in the QP.
- Eliminate the overhead of verifying that the user has authority to access a resource by setting VALIDATE=N against the QP service.

  Do this if you have elected to use SQL security to control access to tables and stored procedures. This setting is also useful in a test or development environment if you trust any user with a valid z/OS user ID to access the data. For example, you might not want to use DB2® privileges or use RACF to verify each resource.

IMS CLASS=*Class Name*

Specifies the name of the RACF resource class that is checked to determine whether the user has authority to schedule or access the PSBs associated with the tables referenced in a query. The Class Name can be up to eight characters long. This sub-parameter is required when accessing IMS data.

PSB PREFIX=*Prefix*

Specifies a value to prefix to the PSB names before a RACF authorization call is issued. If specified, the PSB name is appended to the Prefix value, for example, IMSPPSB1 where the Prefix is IMSP and the PSB name is PSB1.

If you are planning to access IMS data, you might need to modify the IMS CLASS subparameter to define the RACF class where IMS PSBs are defined at your site.

To use a PSB, a user ID must have at least CONTROL access to that PSB's corresponding RACF profile within the class.

The combination of the length of the PSB name and the length of the prefix must be eight characters or less. This is a RACF restriction. If a larger PSB name or prefix combination is encountered, an error message is issued.

Specification

Use: Configuration parameter for the administration, monitoring, operator, and query processor services.

Service classes: PAA, MAA, OPER, QP

Service tasks: CECPAA. CECMAA, CACOPER, CACQP

Data type: CHAR

Default: None

SMFEXIT

SMFEXIT reports clock time and CPU time for an individual user session with a query processor task.

Description

You can supply the following values for the SMFEXIT parameter:

RECTYPE=*nnn*
: This is a required parameter that defines the SMF user record type. This parameter contains a numeric value between 128 and 255.

SYSID=*xxxx*
: This is a required parameter that contains the primary JES subsystem ID. SYSID can be a maximum of four characters.

Specification

Use: Configuration parameter for the operator and query processor services.

Service classes: OPER, QP

Service tasks: OPER, CACQP

Data type: CHAR(64)

Default value: None

SQLSECURITY

The SQLSECURITY parameter determines the level of access privilege verification that will be performed on operator commands.

Description

When SQLSECURITY is set to TRUE, user access privileges are checked in the system catalog for DISPLAY and SYSOPER privileges. If no privileges are found, operator requests are not allowed in the Classic data server to which the user is connected.

Specification

Use: Configuration parameter for the command operator services.

Service class: OPER

Service task: CACOPER

Data type: Boolean

Default: FALSE

Valid values:

FALSE
: Authenticated users can enter all valid commands. Valid commands include the STOP command that you can issue to stop the Classic data server. This setting also allows the Classic data server to run without defining a set of metadata catalogs.

TRUE
: Allows different levels of access level privileges for each user. For example, you can allow all users to issue the DISPLAY command but allow only a subset of users to issue the STOP command. If TRUE is specified, the server JCL must contain DD statements for the metadata catalogs.

SSIDEXCLUDELIST

The SSIDEXCLUDELIST parameter is an optional parameter that represents the IMS subsystem exclusion list for the log reader service.

Description

SSIDEXCLUDELIST is maintained as a service list. You can specify the name of an IMS subsystem ID (SSID) to exclude from ordering decisions.

By default, all subsystems that exist in the RECON that the Classic data server references are eligible to be captured. The actual state of each subsystem and the starting position of a change stream determine whether a subsystem change needs to be captured.

Specification

Use: Configuration parameter for the IMS log reader service.

Service class: LRSI

Task name: CECLRS

Data type: CHAR

Default: None

STMTRETENTION

The STMTRETENTION parameter defines the behavior of a prepared statement when a commit or rollback operation occurs.

Specification

Use: Configuration parameter for the query processor services.

Service class: QP

Service task: CACQP

Data type: INT

Default: 0

Valid values: 0 - 2

`0 = SYNCPOINT`
: Release the statement whenever a COMMIT or ROLLBACK is issued.

`1 = ROLLBACK`
: Release the statement only when a ROLLBACK syncpoint is issued.

`2 = DISCONNECT`
: Release the statement only when the user disconnects from the Classic data server. All prepared statements are retained across both COMMIT and ROLLBACK calls.

STREAMNAME

The STREAMNAME parameter identifies the name of the log stream that is defined in the z/OS system logger.

Description

The logger service writes log records to the z/OS log stream specified in the STREAMNAME parameter. The log stream is used for the diagnostic log that runs in the Classic data server.

Specifications

Use: Configuration parameter for the logger service.

Service class: LOG

Task name: CACLOG

Data type: CHAR(26)

Default: None

STRINGDELIMITER

STRINGDELIMITER is a required configuration parameter that specifies which character to use for delimiting strings.

Description

"Specifying delimited messages" on page 167 describes the rules that apply to defining a delimiter.

You specify STRINGDELIMITER when you want InfoSphere Classic CDC for z/OS to publish changes in a delimited format.

Specify the value as a hexadecimal number or enclose the value in single quotation marks.

Specifications

In general, delimiter values should be enclosed in single quotes.

- If you use character delimiters and would like to preserve the case you need to enclose the value with single quotes or it will be converted into uppercase. Use of delimiters that contain letters or numbers is not recommended.
- If the delimiter value contains a single quote, you can enclose the value in double quotes.
- If the delimiter value contains a double quote, you can enclose the value in single quotes.
- You cannot define a delimiter string with a mixture of single and double quotes.
-
- The following escape characters can be specified in single quotes:

  ```
  \r – carriage return, equivalent to 0X0D
  \n – new line character, equivalent to 0X0A
  \f – form feed, equivalent to 0X0C
  \t – tab, equivalent to 0X09
  \v – vertical tab, equivalent to 0X08
  ```
- To specify an unsupported escape character, you can enter the hexadecimal representation of the value. You must define hexadecimal values with the prefix "0X". Inclusion of a null character (0x00) in the delimiter value will lead to unpredictable results.

Use: Configuration parameter for the capture service.

Service class: CAP

Service task name: CECCAP

Data type: CHAR(8)

Default: "

TEMPFILESPACE

TEMPFILESPACE defines a temporary data set that is dynamically allocated by a Classic data server to store the intermediate result set. How you define the TEMPFILESPACE value varies by service.

Description: QP

Temporary data set information is a set of parameters separated by commas. Parameters not specified are set to the defaults. Set this parameter so that the resulting file is large enough to hold any intermediate result sets that are generated from a typical query that runs on a particular Classic data server. If your site has a storage unit name for VIO storage, specify `VIO`.

Hiperspace places temporary data files, such as spill files, in expanded storage. Hiperspace improves performance and mainly affects complex queries, for example, queries that contain the ORDER BY clause. Hiperspace requires Authorized Program Facility (APF) authorization.

Specification

Service class: QP

Service task: CACQP

Data type: CHAR(64)

Default value: HIPERSPACE,INIT=8M,MAX=2048M,EXTEND=8M

Valid values:

ALCUNIT = BLOCK|TRK|CYL
Specifies a unit of space allocation in block, track, or cylinder units. The default value is TRK.

SPACE = *bytes*
Specifies the primary amount of space to allocate. The default value is 15.

EXTEND = *bytes*
secondary amount of space to allocate. The default value is 5.

VOL = VOLSER
Specifies the volume serial number. The default is the z/OS default for your site.

UNIT = unit name
Specifies 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
Specifies the record format to allocate that corresponds to a z/OS Record Format (RECFM) of FB, VB, or U. The default is V.

RECLEN = *nnn*
Specifies the record length. For variable format records, z/OS LRECL (maximum record length) is set to the fixed record length RECLEN +4. Default is 255.

BLKSIZE = *nnn*
Specifies the block size. The default is 6144.

Example: DASD

Here are example entries for DASD:

```
TEMPFILESPACE = ALCUNIT=TRK,SPACE=15,VOL=CACVOL
TEMPFILESPACE = ALCUNIT=CYL,SPACE=2
TEMPFILESPACE = ALCUNIT=CYL,SPACE=2,EXTEND=1,UNIT=VIO
```

Specification for hiperspace

Valid values:

INIT
Initial region size for the hiperspace

MAX
: Maximum region size for the hiperspace.

EXTEND
: Unit of growth when INIT is exceeded

In general, you should specify these sizes in megabytes (for example, 8M). You can use other units for the query processor.

The estimate for determining these values is related to system installation limits and expected query types. Roughly, make the maximum size equivalent to that of the regular temporary space file as described for the non-hiperspace TEMPFILESPACE setting.

Example: hiperspace

To specify hiperspace, specify the TEMPFILESPACE parameter as follows:

```
TEMPFILESPACE = HIPERSPACE,INIT=16M,MAX=24M,EXTEND=8M
```

TIMEZONE

TIMEZONE is a required parameter that represents the Olson database name for the time zone of the source server.

Description

The TIMEZONE configuration parameter is typically set once during the installation and customization process. The installation and customization process provides the keyword CDCTIMEZ for setting the name of the Olson time zone. The TIMEZONE configuration parameter is set to the value that you specify for the CDCTIMEZ keyword. If a value is not specified for the CDCTIMEZ keyword, the environment is configured with the default value for the TIMEZONE parameter.

You set the TIMEZONE configuration parameter to the string that represents the time zone name. For example, if the source server is running in California you can set the configuration value to `'America/Los_Angeles'`. The table below lists the valid time zone names

Note: The TIMEZONE parameter does not apply to delimited format subscriptions.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Task name: CECCAP

Data type: CHAR

Default: Atlantic/Reykjavik

Valid values: The following table lists the valid names for the TIMEZONE configuration parameter. The value that you specify must match at least one string in the first column in the table of valid string values. You must enclose the string in single quotes.

Table 53. Valid values for the TIMEZONE parameter.

String value	Country code	Coordinates
Africa/Abidjan	CI	+0519-00402
Africa/Accra	GH	+0533-00013
Africa/Addis_Ababa	ET	+0902+03842
Africa/Algiers	DZ	+3647+00303
Africa/Asmara	ER	+1520+03853
Africa/Bamako	ML	+1239-00800
Africa/Bangui	CF	+0422+01835
Africa/Banjul	GM	+1328-01639
Africa/Bissau	GW	+1151-01535
Africa/Blantyre	MW	-1547+03500
Africa/Brazzaville	CG	-0416+01517
Africa/Bujumbura	BI	-0323+02922
Africa/Cairo	EG	+3003+03115
Africa/Casablanca	MA	+3339-00735
Africa/Ceuta	ES	+3553-00519
Africa/Conakry	GN	+0931-01343
Africa/Dakar	SN	+1440-01726
Africa/Dar_es_Salaam	TZ	-0648+03917
Africa/Djibouti	DJ	+1136+04309
Africa/Douala	CM	+0403+00942
Africa/El_Aaiun	EH	+2709-01312
Africa/Freetown	SL	+0830-01315
Africa/Gaborone	BW	-2439+02555
Africa/Harare	ZW	-1750+03103
Africa/Johannesburg	ZA	-2615+02800
Africa/Kampala	UG	+0019+03225
Africa/Khartoum	SD	+1536+03232
Africa/Kigali	RW	-0157+03004
Africa/Kinshasa	CD	-0418+01518
Africa/Lagos	NG	+0627+00324
Africa/Libreville	GA	+0023+00927
Africa/Lome	TG	+0608+00113
Africa/Luanda	AO	-0848+01314
Africa/Lubumbashi	CD	-1140+02728
Africa/Lusaka	ZM	-1525+02817
Africa/Malabo	GQ	+0345+00847

Table 53. Valid values for the TIMEZONE parameter. (continued)

String value	Country code	Coordinates
Africa/Maputo	MZ	-2558+03235
Africa/Maseru	LS	-2928+02730
Africa/Mbabane	SZ	-2618+03106
Africa/Mogadishu	SO	+0204+04522
Africa/Monrovia	LR	+0618-01047
Africa/Nairobi	KE	-0117+03649
Africa/Ndjamena	TD	+1207+01503
Africa/Niamey	NE	+1331+00207
Africa/Nouakchott	MR	+1806-01557
Africa/Ouagadougou	BF	+1222-00131
Africa/Porto-Novo	BJ	+0629+00237
Africa/Sao_Tome	ST	+0020+00644
Africa/Tripoli	LY	+3254+01311
Africa/Tunis	TN	+3648+01011
Africa/Windhoek	NA	-2234+01706
America/Adak	US	+515248-1763929
America/Anchorage	US	+611305-1495401
America/Anguilla	AI	+1812-06304
America/Antigua	AG	+1703-06148
America/Araguaina	BR	-0712-04812
America/Argentina/ Buenos_Aires	AR	-3436-05827
America/Argentina/Catamarca	AR	-2828-06547
America/Argentina/Cordoba	AR	-3124-06411
America/Argentina/Jujuy	AR	-2411-06518
America/Argentina/La_Rioja	AR	-2926-06651
America/Argentina/Mendoza	AR	-3253-06849
America/Argentina/Rio_Gallegos	AR	-5138-06913
America/Argentina/Salta	AR	-2447-06525
America/Argentina/San_Juan	AR	-3132-06831
America/Argentina/San_Luis	AR	-3319-06621
America/Argentina/Tucuman	AR	-2649-06513
America/Argentina/Ushuaia	AR	-5448-06818
America/Aruba	AW	+1230-06958
America/Asuncion	PY	-2516-05740
America/Atikokan	CA	+484531-0913718
America/Bahia	BR	-1259-03831
America/Bahia_Banderas	MX	+2048-10515
America/Barbados	BB	+1306-05937
America/Belem	BR	-0127-04829

Table 53. Valid values for the TIMEZONE parameter. (continued)

String value	Country code	Coordinates
America/Belize	BZ	+1730-08812
America/Blanc-Sablon	CA	+5125-05707
America/Boa_Vista	BR	+0249-06040
America/Bogota	CO	+0436-07405
America/Boise	US	+433649-1161209
America/Cambridge_Bay	CA	+690650-1050310
America/Campo_Grande	BR	-2027-05437
America/Cancun	MX	+2105-08646
America/Caracas	VE	+1030-06656
America/Cayenne	GF	+0456-05220
America/Cayman	KY	+1918-08123
America/Chicago	US	+415100-0873900
America/Chihuahua	MX	+2838-10605
America/Costa_Rica	CR	+0956-08405
America/Cuiaba	BR	-1535-05605
America/Curacao	AN	+1211-06900
America/Danmarkshavn	GL	+7646-01840
America/Dawson	CA	+6404-13925
America/Dawson_Creek	CA	+5946-12014
America/Denver	US	+394421-1045903
America/Detroit	US	+421953-0830245
America/Dominica	DM	+1518-06124
America/Edmonton	CA	+5333-11328
America/Eirunepe	BR	-0640-06952
America/El_Salvador	SV	+1342-08912
America/Fortaleza	BR	-0343-03830
America/Glace_Bay	CA	+4612-05957
America/Godthab	GL	+6411-05144
America/Goose_Bay	CA	+5320-06025
America/Grand_Turk	TC	+2128-07108
America/Grenada	GD	+1203-06145
America/Guadeloupe	GP	+1614-06132
America/Guatemala	GT	+1438-09031
America/Guayaquil	EC	-0210-07950
America/Guyana	GY	+0648-05810
America/Halifax	CA	+4439-06336
America/Havana	CU	+2308-08222
America/Hermosillo	MX	+2904-11058
America/Indiana/Indianapolis	US	+394606-0860929
America/Indiana/Knox	US	+411745-0863730

Table 53. Valid values for the TIMEZONE parameter. (continued)

String value	Country code	Coordinates
America/Indiana/Marengo	US	+382232-0862041
America/Indiana/Petersburg	US	+382931-0871643
America/Indiana/Tell_City	US	+375711-0864541
America/Indiana/Vevay	US	+384452-0850402
America/Indiana/Vincennes	US	+384038-0873143
America/Indiana/Winamac	US	+410305-0863611
America/Inuvik	CA	+682059-1334300
America/Iqaluit	CA	+6344-06828
America/Jamaica	JM	+1800-07648
America/Juneau	US	+581807-1342511
America/Kentucky/Louisville	US	+381515-0854534
America/Kentucky/Monticello	US	+364947-0845057
America/La_Paz	BO	-1630-06809
America/Lima	PE	-1203-07703
America/Los_Angeles	US	+340308-1181434
America/Maceio	BR	-0940-03543
America/Managua	NI	+1209-08617
America/Manaus	BR	-0308-06001
America/Marigot	MF	+1804-06305
America/Martinique	MQ	+1436-06105
America/Matamoros	MX	+2550-09730
America/Mazatlan	MX	+2313-10625
America/Menominee	US	+450628-0873651
America/Merida	MX	+2058-08937
America/Mexico_City	MX	+1924-09909
America/Miquelon	PM	+4703-05620
America/Moncton	CA	+4606-06447
America/Monterrey	MX	+2540-10019
America/Montevideo	UY	-3453-05611
America/Montreal	CA	+4531-07334
America/Montserrat	MS	+1643-06213
America/Nassau	BS	+2505-07721
America/New_York	US	+404251-0740023
America/Nipigon	CA	+4901-08816
America/Nome	US	+643004-1652423
America/Noronha	BR	-0351-03225
America/North_Dakota/Center	US	+470659-1011757
America/North_Dakota/New_Salem	US	+465042-1012439
America/Ojinaga	MX	+2934-10425

Table 53. Valid values for the TIMEZONE parameter. (continued)

String value	Country code	Coordinates
America/Panama	PA	+0858-07932
America/Pangnirtung	CA	+6608-06544
America/Paramaribo	SR	+0550-05510
America/Phoenix	US	+332654-1120424
America/Port_of_Spain	TT	+1039-06131
America/Port-au-Prince	HT	+1832-07220
America/Porto_Velho	BR	-0846-06354
America/Puerto_Rico	PR	+182806-0660622
America/Rainy_River	CA	+4843-09434
America/Rankin_Inlet	CA	+624900-0920459
America/Recife	BR	-0803-03454
America/Regina	CA	+5024-10439
America/Resolute	CA	+744144-0944945
America/Rio_Branco	BR	-0958-06748
America/Santa_Isabel	MX	+3018-11452
America/Santarem	BR	-0226-05452
America/Santiago	CL	-3327-07040
America/Santo_Domingo	DO	+1828-06954
America/Sao_Paulo	BR	-2332-04637
America/Scoresbysund	GL	+7029-02158
America/Shiprock	US	+364708-1084111
America/St_Barthelemy	BL	+1753-06251
America/St_Johns	CA	+4734-05243
America/St_Kitts	KN	+1718-06243
America/St_Lucia	LC	+1401-06100
America/St_Thomas	VI	+1821-06456
America/St_Vincent	VC	+1309-06114
America/Swift_Current	CA	+5017-10750
America/Tegucigalpa	HN	+1406-08713
America/Thule	GL	+7634-06847
America/Thunder_Bay	CA	+4823-08915
America/Tijuana	MX	+3232-11701
America/Toronto	CA	+4339-07923
America/Tortola	VG	+1827-06437
America/Vancouver	CA	+4916-12307
America/Whitehorse	CA	+6043-13503
America/Winnipeg	CA	+4953-09709
America/Yakutat	US	+593249-1394338
America/Yellowknife	CA	+6227-11421
Antarctica/Casey	AQ	-6617+11031

Table 53. Valid values for the TIMEZONE parameter. (continued)

String value	Country code	Coordinates
Antarctica/Davis	AQ	-6835+07758
Antarctica/DumontDUrville	AQ	-6640+14001
Antarctica/Macquarie	AQ	-5430+15857
Antarctica/Mawson	AQ	-6736+06253
Antarctica/McMurdo	AQ	-7750+16636
Antarctica/Palmer	AQ	-6448-06406
Antarctica/Rothera	AQ	-6734-06808
Antarctica/South_Pole	AQ	-9000+00000
Antarctica/Syowa	AQ	-690022+0393524
Antarctica/Vostok	AQ	-7824+10654
Arctic/Longyearbyen	SJ	+7800+01600
Asia/Aden	YE	+1245+04512
Asia/Almaty	KZ	+4315+07657
Asia/Amman	JO	+3157+03556
Asia/Anadyr	RU	+6445+17729
Asia/Aqtau	KZ	+4431+05016
Asia/Aqtobe	KZ	+5017+05710
Asia/Ashgabat	TM	+3757+05823
Asia/Baghdad	IQ	+3321+04425
Asia/Bahrain	BH	+2623+05035
Asia/Baku	AZ	+4023+04951
Asia/Bangkok	TH	+1345+10031
Asia/Beirut	LB	+3353+03530
Asia/Bishkek	KG	+4254+07436
Asia/Brunei	BN	+0456+11455
Asia/Choibalsan	MN	+4804+11430
Asia/Chongqing	CN	+2934+10635
Asia/Colombo	LK	+0656+07951
Asia/Damascus	SY	+3330+03618
Asia/Dhaka	BD	+2343+09025
Asia/Dili	TL	-0833+12535
Asia/Dubai	AE	+2518+05518
Asia/Dushanbe	TJ	+3835+06848
Asia/Gaza	PS	+3130+03428
Asia/Harbin	CN	+4545+12641
Asia/Ho_Chi_Minh	VN	+1045+10640
Asia/Hong_Kong	HK	+2217+11409
Asia/Hovd	MN	+4801+09139
Asia/Irkutsk	RU	+5216+10420
Asia/Jakarta	ID	-0610+10648

Table 53. Valid values for the TIMEZONE parameter. (continued)

String value	Country code	Coordinates
Asia/Jayapura	ID	-0232+14042
Asia/Jerusalem	IL	+3146+03514
Asia/Kabul	AF	+3431+06912
Asia/Kamchatka	RU	+5301+15839
Asia/Karachi	PK	+2452+06703
Asia/Kashgar	CN	+3929+07559
Asia/Kathmandu	NP	+2743+08519
Asia/Kolkata	IN	+2232+08822
Asia/Krasnoyarsk	RU	+5601+09250
Asia/Kuala_Lumpur	MY	+0310+10142
Asia/Kuching	MY	+0133+11020
Asia/Kuwait	KW	+2920+04759
Asia/Macau	MO	+2214+11335
Asia/Magadan	RU	+5934+15048
Asia/Makassar	ID	-0507+11924
Asia/Manila	PH	+1435+12100
Asia/Muscat	OM	+2336+05835
Asia/Nicosia	CY	+3510+03322
Asia/Novokuznetsk	RU	+5345+08707
Asia/Novosibirsk	RU	+5502+08255
Asia/Omsk	RU	+5500+07324
Asia/Oral	KZ	+5113+05121
Asia/Phnom_Penh	KH	+1133+10455
Asia/Pontianak	ID	-0002+10920
Asia/Pyongyang	KP	+3901+12545
Asia/Qatar	QA	+2517+05132
Asia/Qyzylorda	KZ	+4448+06528
Asia/Rangoon	MM	+1647+09610
Asia/Riyadh	SA	+2438+04643
Asia/Sakhalin	RU	+4658+14242
Asia/Samarkand	UZ	+3940+06648
Asia/Seoul	KR	+3733+12658
Asia/Shanghai	CN	+3114+12128
Asia/Singapore	SG	+0117+10351
Asia/Taipei	TW	+2503+12130
Asia/Tashkent	UZ	+4120+06918
Asia/Tbilisi	GE	+4143+04449
Asia/Tehran	IR	+3540+05126
Asia/Thimphu	BT	+2728+08939
Asia/Tokyo	JP	+353916+1394441

Table 53. Valid values for the TIMEZONE parameter. (continued)

String value	Country code	Coordinates
Asia/Ulaanbaatar	MN	+4755+10653
Asia/Urumqi	CN	+4348+08735
Asia/Vientiane	LA	+1758+10236
Asia/Vladivostok	RU	+4310+13156
Asia/Yakutsk	RU	+6200+12940
Asia/Yekaterinburg	RU	+5651+06036
Asia/Yerevan	AM	+4011+04430
Atlantic/Azores	PT	+3744-02540
Atlantic/Bermuda	BM	+3217-06446
Atlantic/Canary	ES	+2806-01524
Atlantic/Cape_Verde	CV	+1455-02331
Atlantic/Faroe	FO	+6201-00646
Atlantic/Madeira	PT	+3238-01654
Atlantic/Reykjavik	IS	+6409-02151
Atlantic/South_Georgia	GS	-5416-03632
Atlantic/St_Helena	SH	-1555-00542
Atlantic/Stanley	FK	-5142-05751
Australia/Adelaide	AU	-3455+13835
Australia/Brisbane	AU	-2728+15302
Australia/Broken_Hill	AU	-3157+14127
Australia/Currie	AU	-3956+14352
Australia/Darwin	AU	-1228+13050
Australia/Eucla	AU	-3143+12852
Australia/Hobart	AU	-4253+14719
Australia/Lindeman	AU	-2016+14900
Australia/Lord_Howe	AU	-3133+15905
Australia/Melbourne	AU	-3749+14458
Australia/Perth	AU	-3157+11551
Australia/Sydney	AU	-3352+15113
Europe/Amsterdam	NL	+5222+00454
Europe/Andorra	AD	+4230+00131
Europe/Athens	GR	+3758+02343
Europe/Belgrade	RS	+4450+02030
Europe/Berlin	DE	+5230+01322
Europe/Bratislava	SK	+4809+01707
Europe/Brussels	BE	+5050+00420
Europe/Bucharest	RO	+4426+02606
Europe/Budapest	HU	+4730+01905
Europe/Chisinau	MD	+4700+02850
Europe/Copenhagen	DK	+5540+01235

Table 53. Valid values for the TIMEZONE parameter. (continued)

String value	Country code	Coordinates
Europe/Dublin	IE	+5320-00615
Europe/Gibraltar	GI	+3608-00521
Europe/Guernsey	GG	+4927-00232
Europe/Helsinki	FI	+6010+02458
Europe/Isle_of_Man	IM	+5409-00428
Europe/Istanbul	TR	+4101+02858
Europe/Jersey	JE	+4912-00207
Europe/Kaliningrad	RU	+5443+02030
Europe/Kiev	UA	+5026+03031
Europe/Lisbon	PT	+3843-00908
Europe/Ljubljana	SI	+4603+01431
Europe/London	GB	+513030-0000731
Europe/Luxembourg	LU	+4936+00609
Europe/Madrid	ES	+4024-00341
Europe/Malta	MT	+3554+01431
Europe/Mariehamn	AX	+6006+01957
Europe/Minsk	BY	+5354+02734
Europe/Monaco	MC	+4342+00723
Europe/Moscow	RU	+5545+03735
Europe/Oslo	NO	+5955+01045
Europe/Paris	FR	+4852+00220
Europe/Podgorica	ME	+4226+01916
Europe/Prague	CZ	+5005+01426
Europe/Riga	LV	+5657+02406
Europe/Rome	IT	+4154+01229
Europe/Samara	RU	+5312+05009
Europe/San_Marino	SM	+4355+01228
Europe/Sarajevo	BA	+4352+01825
Europe/Simferopol	UA	+4457+03406
Europe/Skopje	MK	+4159+02126
Europe/Sofia	BG	+4241+02319
Europe/Stockholm	SE	+5920+01803
Europe/Tallinn	EE	+5925+02445
Europe/Tirane	AL	+4120+01950
Europe/Uzhgorod	UA	+4837+02218
Europe/Vaduz	LI	+4709+00931
Europe/Vatican	VA	+415408+0122711
Europe/Vienna	AT	+4813+01620
Europe/Vilnius	LT	+5441+02519
Europe/Volgograd	RU	+4844+04425

Table 53. Valid values for the TIMEZONE parameter. (continued)

String value	Country code	Coordinates
Europe/Warsaw	PL	+5215+02100
Europe/Zagreb	HR	+4548+01558
Europe/Zaporozhye	UA	+4750+03510
Europe/Zurich	CH	+4723+00832
Indian/Antananarivo	MG	-1855+04731
Indian/Chagos	IO	-0720+07225
Indian/Christmas	CX	-1025+10543
Indian/Cocos	CC	-1210+09655
Indian/Comoro	KM	-1141+04316
Indian/Kerguelen	TF	-492110+0701303
Indian/Mahe	SC	-0440+05528
Indian/Maldives	MV	+0410+07330
Indian/Mauritius	MU	-2010+05730
Indian/Mayotte	YT	-1247+04514
Indian/Reunion	RE	-2052+05528
Pacific/Apia	WS	-1350-17144
Pacific/Auckland	NZ	-3652+17446
Pacific/Chatham	NZ	-4357-17633
Pacific/Easter	CL	-2709-10926
Pacific/Efate	VU	-1740+16825
Pacific/Enderbury	KI	-0308-17105
Pacific/Fakaofo	TK	-0922-17114
Pacific/Fiji	FJ	-1808+17825
Pacific/Funafuti	TV	-0831+17913
Pacific/Galapagos	EC	-0054-08936
Pacific/Gambier	PF	-2308-13457
Pacific/Guadalcanal	SB	-0932+16012
Pacific/Guam	GU	+1328+14445
Pacific/Honolulu	US	+211825-1575130
Pacific/Johnston	UM	+1645-16931
Pacific/Kiritimati	KI	+0152-15720
Pacific/Kosrae	FM	+0519+16259
Pacific/Kwajalein	MH	+0905+16720
Pacific/Majuro	MH	+0709+17112
Pacific/Marquesas	PF	-0900-13930
Pacific/Midway	UM	+2813-17722
Pacific/Nauru	NR	-0031+16655
Pacific/Niue	NU	-1901-16955
Pacific/Norfolk	NF	-2903+16758
Pacific/Noumea	NC	-2216+16627

Table 53. Valid values for the TIMEZONE parameter. (continued)

String value	Country code	Coordinates
Pacific/Pago_Pago	AS	-1416-17042
Pacific/Palau	PW	+0720+13429
Pacific/Pitcairn	PN	-2504-13005
Pacific/Ponape	FM	+0658+15813
Pacific/Port_Moresby	PG	-0930+14710
Pacific/Rarotonga	CK	-2114-15946
Pacific/Saipan	MP	+1512+14545
Pacific/Tahiti	PF	-1732-14934
Pacific/Tarawa	KI	+0125+17300
Pacific/Tongatapu	TO	-2110-17510
Pacific/Truk	FM	+0725+15147
Pacific/Wake	UM	+1917+16637
Pacific/Wallis	WF	-1318-17610

UORGROUPCOUNT

The UORGROUPCOUNT parameter identifies the number of messages that the capture service should group into a common unit of recovery (UOR) during replication.

Description

The source server, by default, respects the transaction boundaries of the source database managements system (DBMS) and replicates transactions within those boundaries. This means that each transaction is individually managed at the target server.

This parameter also applies to delimited publishing. If UOR grouping is active, source UORs can be combined and written to WebSphere MQ as part of the same WebSphere MQ transaction up to the UORGROUPCOUNT limit (or until there is no more committed data in the subscriptions capture cache). UOR grouping does not take the value of the COMMITINTERVAL parameter into account. However if grouping is active, additional source UORs are added to the same WebSphere MQ message even if multiple source UORs are combined into the same MQ transaction and the COMMITINTERVAL timer has expired. The cost of UOR grouping is not expected to delay the final WebSphere MQ commit for long.

For small transactions, you might achieve higher throughput and lower CPU by grouping multiple small transactions into a single larger transaction. You can use the UORGROUPCOUNT parameter to instruct the source server to combine smaller transactions together before they are sent to the target server.

The UORGROUPCOUNT parameter represents the maximum number of updates to group together. The source server combines transactions until the number of updates will exceed the UORGROUPCOUNT value if the next transaction is grouped. At this point, the group is sent to the target server and a new group is started. A group is also completed and a new group is started when the following situations occur:

- The source server runs out of committed work in the capture cache

- The group is serial and the next UOR is parallel, or the group is parallel and the next UOR is serial

Grouping transactions with the UORGROUPCOUNT parameter does not cause an original transaction from the source DBMS to be divided into multiple groups. An original transaction is always contained in a single group when transaction grouping is active.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Task name: CECCAP

Data type: INT

Default: 1

The default value indicates that transaction grouping is inactive because each transaction meets or exceeds the UORGROUPCOUNT value and causes the UOR to be sent to the target.

VSAMRECSELECT

The VSAMRECSELECT configuration parameter identifies the name of the record selection exit load module that controls the capture of changes made to VSAM data.

Description

If you develop a record selection exit to control publishing of your VSAM data, you need to use the VSAMRECSELECT configuration parameter to activate loading and calling the exit. See the topic "“Developing a record selection exit for multiple record layouts” on page 136 for more information about using and creating the exit.

Specifications

Use: Configuration parameter for the capture service.

Service class: CAP

Task name: CECCAP

Data type: CHAR(8)

Default: None

USERSUBPOOLMAX

The USERSUBPOOLMAX parameter determines the maximum size of a user sub pool.

Description

Each user that connects to a Classic data source is assigned a user sub pool. A query processor task is assigned to a user connection. The USERSUBPOOLMAX parameter limits the size of the memory pool that a query processor task can use to optimize the use of system resources.

A user sub pool can grow to 256 times the USERSUBPOOLMAX value, resulting in the maximum user sub pool size in bytes. For example, if you set USERSUBPOOLMAX to the default value of 8192, the memory requirements for all of your queries for the current connection cannot exceed 2MB.

Recommendation: For complex queries, or in environments that process many queries simultaneously per user connection, set USERSUBPOOLMAX to a value greater than 8192. If you have many users simultaneously connecting to the data source, you might need to increase the value of the MESSAGEPOOLSIZE parameter because each user sub pool is allocated out of the main message pool.

Configure this parameter carefully to avoid using too much data server storage.

Specification

Use: Configuration parameter for the QP and QPLD service classes.

Data type: INT

Default value: 8192

Valid values: 4 - 214730

WLMUOW

The WLMUOW parameter specifies the Workload Manager (WLM) unit-of-work activities and manages queries in WLM goal mode.

Description

Define a subsystem type and classification rules for the workload that is processed by the query processor. For an existing subsystem type, select which of these parameters fits that subtype. For example, the z/OS Started Task Control (STC) type supports user ID and TRANNAME. Job Entry Subsystem (JES) supports user ID, TRANNAME, and TRANCLASS.

For information about how to define service classes and classification rules, see the description of z/OS Workload Manager in the CICS product information for WLM goal mode. The priority for units of work need to be less than VTAM® and IMS. The discretionary goal might result in very slow response times. Performance periods allow you to define a high number of service units for short transactions and a smaller number for long running transactions.

Specification

Use: Configuration parameter for the query processor services.

Service class: QP

Service task: CACQP

Data type: CHAR(64)

Default: None

Chapter 17. Command reference

Look up syntax and explanations for commands that help you to manage subscriptions, Classic data servers, and configurations.

Subscription management commands

You can use the commands for replication management to start and stop replication, modify subscriptions, and gather metric data about subscriptions running on source and target servers.

The administration service supports the operational and administration commands for replication management. The monitoring service supports the commands that report metric data for subscriptions and replication mappings.

You can issue the subscription management commands by using the master terminal operator (MTO) interface. You can also use the Management Console to manage subscriptions and obtain subscription metrics, or use the Classic Data Architect to obtain diagnostic metrics.

DISPLAY,REPL command

You can use the DISPLAY,REPL command to query a source server for metrics for subscriptions and replication mappings. The commands generate reports that are issued to the console as WTO messages.

Displaying subscription summary

This command displays subscription metrics for the specified subscription or for all subscriptions.

```
►►──DISPLAY──,──REPL──,──┬─SUBSCR──=──subscription-name─┬──────────────────────►◄
                         └─SUBSCR──=──*─────────────────┘
```

A warning message displays on the report if no subscriptions are found.

For delimited format subscriptions, the command displays summary information for the source server for all applicable subscriptions followed by target information that contains latency information. If there are multiple subscriptions, the source information displays first followed by target information for any delimited format subscriptions.

Displaying subscription detail

This command displays replication mapping and replication object metrics for the specified subscription or for all subscriptions.

```
►►──DISPLAY──,──REPL──,──┬─SUBSCR──=──subscription-name─┬──,──┬────────┬──────►◄
                         └─SUBSCR──=──*─────────────────┘     └─DETAIL─┘
```

The report output provides the same subscription metrics as the subscription summary report with the addition of replication mapping information.

For a delimited format subscription that writes to WebSphere MQ, the metric information that is written to the source might be inaccurate.

- In cases when transactions were rolled back, metrics for aborted transactions are included in the report; they are not corrected.
- If an end immediate stop of replication occurs, the mapping counts might be inaccurate because they reflect data that was rolled back for the active WebSphere MQ transaction when the command was issued.
- If a subscription is in error, the mapping counts might be inaccurate.

A warning message displays on the report if no subscriptions are found.

Displaying replication mappings

This command displays metrics for all subscriptions associated with a replication object.

```
►►──DISPLAY──,──REPL──,────MAPPING──=──replication-object──────────────────────►◄
```

The report provides information about the subscriptions that contain replication mappings for the specified replication object. It includes the status of subscriptions and replication mappings.

Parameters

subscription name
: The name of the subscription to display. You can specify the subscription name as an identifier with or without quotation marks.

asterisk (*)
: An asterisk (*) displays information for all subscriptions. You can also specify an * as a wildcard character at the end of a partial subscription name. For example, specifying SUBSCR=ABC* will display all the subscription names that begin with ABC.

DETAIL
: Displays detailed information about subscriptions and replication mappings.

MAPPING
: Displays information about subscriptions that contain replication mappings.

replication object
: The name of the replication object to display. You can specify the replication object name as an identifier with or without quotation marks.

Examples

The first example shows the output of the DISPLAY,REPL,SUBSCR command for all subscriptions whose name begins with the string 'UDB'. Note that the quantity and size of data that is received and sent can vary widely, depending on factors such as these:

- Log data size
- Table mappings (number and data size of mapped columns)
- Change messages that are the result of formatting the source data
- Data that is stored in caches after you stop and restart replication for a subscription

```
CAC00200I DISPLAY,REPL,SUBSCR=UDB*
CECM0052I SUBSCRIPTION METRICS REPORT
SrcSysID Subscription Name
======== ============================================================
UDBSUB01 UDBSUB01
              Received        Sent        State: REPLICATE CONTINUOUS
          =============== ===============   Cache:   3%
 Bytes           5430644         7550973
 Rows              16456           16456
 Refresh               0               0
 Commits            1153            1152
 Inserts           16456           16456
 Updates               0               0
 Deletes               0               0

UDBSUB02 UDBSUB02
              Received        Sent        State: REPLICATE CONTINUOUS
          =============== ===============   Cache:   3%
 Bytes         684234436       111590430
 Rows            1002000         1002000
 Refresh               0               0
 Commits            1004            1000
 Inserts               0               0
 Updates         1002000          102000
 Deletes               0               0

UDBSUB03 UDBSUB03
              Received        Sent        State: REFRESH BEFORE REPLI
          =============== ===============   Cache:   0%
 Bytes             13464           85916
 Rows                 42              42
 Refresh              42              42
 Commits               0               0
 Inserts               0               0
 Updates               0               0
 Deletes               0               0

UDBSUB04 UDBSUB04
              Received        Sent        State: STARTING
          =============== ===============   Cache:   0%
 Bytes                 0            1723
 Rows                  0               0
 Refresh               0               0
 Commits               0               0
 Inserts               0               0
 Updates               0               0
 Deletes               0               0

UDBSUB05 UDBSUB05
              Received        Sent        State: INACTIVE
          =============== ===============   Cache:   0%
 Bytes                 0               0
 Rows                  0               0
 Refresh               0               0
 Commits               0               0
 Inserts               0               0
 Updates               0               0
 Deletes               0               0

Number of source subscriptions reported: 5
END OF REPORT
```

In the next example, the subscription UDBSUB6 is in the state REFRESH PENDING. Table refresh operations can be expensive, so a single subscription refreshes its replication mappings serially to save system resources. The data server automatically limits the number of concurrent refreshes across multiple subscriptions, placing refresh requests that exceed the limit on a pending queue.

```
UDBSUB6 UDBSUB6
              Received         Sent         State: REFRESH PENDING
          =============== ===============   Cache:   0%
 Bytes                 0            1514
 Rows                  0               0
 Refresh               0               0
 Commits               0               0
 Inserts               0               0
 Updates               0               0
 Deletes               0               0

Total number of subscriptions in refresh pending: 1
```

START,REPL command

You can use the START,REPL command to start replication.

Starting replication

This command starts replication activity for the specified subscription or for all subscriptions.

```
►►──START──,──REPL──,──┬─SUBSCR──=──subscription-name──┬──,──┬──────────────┬──────────►◄
                       └─ALL───────────────────────────┘     ├─CONTINUOUS─┤
                                                             ├─DESCRIBE───┤
                                                             └─PERIODIC───┘
```

Parameters

subscription name
: The name of the subscription that contains the data that you want to replicate. You can specify the subscription name as an identifier with or without quotation marks.

ALL
: Starts all subscriptions that are available for processing. Any subscriptions that are locked for other processing (for example, for replication processing or administrative updates) are not started.

CONTINUOUS
: Starts continuous replication for the subscription. Replication continues until a command to stop replication is received.

DESCRIBE
: Activates the Describe process. This process validates metadata at the source server and transfers metadata about active subscriptions and replication mappings to the target server.

PERIODIC
: Starts net change replication for a subscription. This process stops replication when the source server receives a unit of recovery (UOR) that occurred after replication was started. This UOR is not replicated and initiates the controlled end of replication.

SET,REPL command

You can use the SET,REPL command to modify existing subscriptions.

Modifying subscription information

You can use the following SET command to modify the attributes of an existing subscription or of all subscriptions.

```
►►—SET—,—REPL—,——┬SUBSCR—=—subscription-name─┬—,——————————————————————►
                 └ALL────────────────────────┘

►─┬──────────────────────────────────────────────────┬————————————————————►◄
  ├SETLOGPOS—,——┬GMTTIME—=—time────┬──────────────────┤
  │             └SERVERTIME—=—time─┘                  │
  ├'CAPTURECACHE—=—n'─────────────────────────────────┤
  ├'PERSISTENT—=——┬YES'─┬─────────────────────────────┤
  │               └NO'──┘                             │
  └'STATUS—=——┬ACTIVE——┬—,—OBJECT—=—object'───────────┘
              ├PARK────┤
              └REFRESH─┘
```

Parameters

subscription name
: The name of the subscription to modify. You can specify the subscription name as an identifier with or without quotation marks.

ALL
: Modifies the specified attributes for all subscriptions.

CAPTURECACHE
: The size of the capture cache. A value of 64 to 2048 MB is allowed.

PERSISTENT
: Automatically restarts replication for the specified subscription. Specify this keyword only for subscriptions that use continuous replication. Valid values are YES or NO.

SETLOGPOS
: Updates the bookmark information stored at the target server.

Specify the time in the format YYYY-MM-DD-hh.mm.ss.thmiju:

- YYYY-MM-DD is a calendar date
- hh.mm.ss.thmiju is a time of day

Valid values: You can specify a value to get the current time or specify GMTTIME or SERVERTIME.

GMTTIME
: The Greenwich Mean Time (GMT) time. You must specify this value up to the ss (seconds). For example:

```
SET,REPL,SUBSCR=subscription name,SETLOGPOS,
GMTTIME=2011-04-14-18.01.00.123456
```

SERVERTIME
: The server time. Specify the local time of the server time zone. You must specify this value up to the ss (seconds) value. For example:

```
SET,REPL,SUBSCR=subscription name,SETLOGPOS,
SERVERTIME=2011-04-14-18.01.00.123456
```

Important: If you modify a bookmark, you might need to refresh the entire subscription and restart replication for the subscription to recreate a valid bookmark. You should only change a bookmark under the direction of IBM Software Support.

Note: Refresh processing is not supported for delimited format subscriptions.

"STATUS= <*status*>,OBJECT=<*object*>"
The status of the subscription. Valid values:

ACTIVE
For any subscription, you can set which mappings you want to be set as active for replication. If a mapping is not set as active, it will not replicate when replication is started for the subscription.

PARK For any subscription, you can park replication mappings. If a mapping is set as parked, it will not replicate when replication is started for the subscription.

REFRESH
Starts refresh processing for the subscription.

Note: Refresh processing is not supported for delimited format subscriptions.

The STATUS and OBJECT keywords must be enclosed in quotation marks.

STOP,REPL command

You can use the STOP,REPL command to stop replication activity for a single subscription or for all subscriptions.

Stopping replication

This command ends replication activity for the specified subscription.

```
►►──STOP──,──REPL──,──┬─SUBSCR──=──subscription-name─┬──,──┬──────────────────┬────►◄
                      └─ALL──────────────────────────┘     │  ┌─CONTROLLED─┐  │
                                                           ├──────────────────┤
                                                           ├─IMMEDIATE────────┤
                                                           └─CAPTURE──────────┘
```

Parameters

subscription name
The name of the subscription to stop. You can specify the subscription name as a quoted or an unquoted identifier.

ALL
Stops all subscriptions. The capture service examines the state of each subscription and processes the request if all subscriptions are in an appropriate state.

CONTROLLED
Controls how replication stops by allowing all transactions that are in process to finish before replication stops. Following a CONTROLLED stop, the subscription is placed in an inactive state. The capture service continues to read log data for the subscription in anticipation of a subsequent START command for the subscription. This is the default.

IMMEDIATE
Stops all transactions immediately, rolling back any that are not finished. Following an IMMEDIATE stop, the subscription is placed in the inactive state. The capture service continues to read log data for the subscription in anticipation of a subsequent START command for the subscription.

CAPTURE
Stops data capture and discards the capture cache for one or more subscriptions.

You can only stop data capture for a subscription that is in an inactive state. When you issue the STOP,REPL command with the CAPTURE keyword, the subscription remains in an inactive state and the capture service stops reading data for the subscription.

Examples

The CAPTURE keyword is useful when you change the definition of a subscription and the cached data will no longer match the new definition. The CAPTURE keyword is also useful in situations when you want to reduce the amount of storage that the capture cache uses.

- To stop data capture and discard any currently cached data for all inactive subscriptions:
  ```
  STOP,REPL,ALL,CAPTURE
  ```
- To put subscription *sub1* into an inactive state and then stop data capture and discard any currently cached associated with the *sub1* subscription, issue the following sequence of commands:
  ```
  STOP,REPL,SUBSCR=sub1,IMMEDIATE
  STOP,REPL,SUBSCR=sub1,CAPTURE
  ```
- To put all eligible subscriptions into an inactive state and then stop data capture and discard any currently cached associated with the subscriptions, issue the following sequence of commands:
  ```
  STOP,REPL,ALL,IMMEDIATE
  STOP,REPL,ALL,CAPTURE
  ```

Classic data server administration commands

Use these commands to start and stop Classic data servers and services, list connected users, cancel user sessions, and view or print log messages.

Starting a data server

When you start a Classic data server, you start all of the services defined in the configuration file for the Classic data server.

About this task

You can perform either of the steps described in the following procedure to start a Classic data server. All services start if the value of the INITIALTHREADS configuration parameter is greater than 0.

Procedure

- Issue a console command to start the JCL procedure for the Classic data server:
  ```
  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.

STOP command

Stopping a Classic data server stops all of the services that are running within it.

About this task

The purpose of the STOP,ALL command is to shutdown the data server. The data server stops after the services running in the data server complete their required processing.

If the shutdown process does not complete after issuing a STOP,ALL command, you can issue the STOP,ALL,IMMEDIATE command. For example, the shutdown process might not complete if a service encounters a problem and cannot complete its quiesce processing. In this case, you can issue the STOP,ALL,IMMEDIATE command to bypass service quiesce processing and stop the data server.

Procedure

- To stop a Classic data server, issue the following command in an MTO interface:

  ```
  F name,STOP,ALL
  ```

  name The name of the started task or batch job for the Classic data server.

- To stop a Classic data server immediately, issue the following command in an MTO interface:

  ```
  F name,STOP,ALL,IMMEDIATE
  ```

  name The name of the started task or batch job for the Classic data server.

START,SERVICE command

You can start an instance of a service that is defined in the configuration for the Classic data server.

About this task

You can use this command when you want to start a service without stopping and restarting the Classic data server.

To start a service in a data server:

Procedure

Issue the following command in an MTO interface, where *name_of_job* is the name of the started task for the Classic data server:

```
F name_of_job,START,SERVICE=name_of_service
```

STOP,SERVICE command

You can stop an instance of a non-critical service that is defined in the configuration for the Classic data server.

About this task

Important: You should not issue the STOP,SERVICE command regularly. When a data server is configured, you typically do not need to start and stop the services that run in the data server individually.

The STOP command cancels any user activity in a service and disconnects all active users from that service.

Restriction: You cannot stop a critical service. The following services are critical to the operations of a Classic data server:
- Administration service
- Capture service
- IMS log reader service
- Logger service

If you attempt to stop a critical service by issuing a STOP,SERVICE command or a STOP,TASKID command, a warning message is issued.

Procedure

- To stop a service by means of its task ID, issue this command:

  ```
  F server_name,STOP,TASKID=task_ID
  ```

  server_name
  : The name of the task or batch job started by the Classic data server. This name is CACCS for change capture.
- To stop a service by means of its name, issue this command:

  ```
  F server_name,STOP,SERVICE=name_of_service
  ```

  server_name
  : The name of the task or batch job started by the Classic data server. This name is CACCS for change capture.

DISPLAY,ALL command

The **DISPLAY,ALL** command outputs a formatted list of the current usage information about a data server.

Procedure

To display current usage information about services, users, configurations, and the memory pool, issue the DISPLAY,ALL command:

```
F name,DISPLAY,ALL
```

name The name of the task or batch job started by the data server.

DISPLAY,MEMORY command

The **DISPLAY,MEMORY** command outputs a formatted list of usage information about data server memory.

Procedure

To display the current use of the memory pool in the data server, issue the DISPLAY,MEMORY 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.

DISPLAY,SERVICES command

The **DISPLAY,SERVICES** command outputs a formatted list of information about the services running in a data server.

Procedure

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
Service name.

TASKID
TCB address (in decimal notation) of the service instance that is displayed.

TASKNAME
Same as TYPE.

STATUS
One of the values that is displayed in the table below.

USER The user ID that is currently being serviced. Generally, this value is blank.

The following table lists the most common statuses:

Table 54. States and descriptions

Status	Description
QUIESCE	Unused.
READY	Idle and waiting for requests.
RECEIVING	Receiving a request.
RESPONDING	Sending a response.
STOP	Processing a STOP,ALL request.

DISPLAY,USERS command

You can list all of the users that are connected to a Classic data server.

Procedure

To list users, issue this command in an MTO interface:

```
F name,DISPLAY,USERS
```

name The name of the started task or batch job for the Classic data server.

Results

When information is requested about active users, a WTO display message is generated for each user that is connected to the Classic data server. For each user, the following information is displayed:

USER The user ID that was supplied when the client application connected to the server. The example below provides information about different types of users.

TASKID
The TCB address of the query processor service that the client connected to.

SERVICE
The first 8 characters of the data source name that the client application connected to. This value corresponds to one of the names of a query processor service definition.

SESSIONID
The address (in decimal format) of a control block that is used to track the user.

HOSTNAME
The name of the host computer where the connection originated.

PROCESSID
The process ID on the host computer that the client application is executing in.

THREADID
The thread identifier on the host computer that the client application is executing in.

STMTS
This is a two-part value that is separated by a forward slash (/). The first value identifies the number of statements that the user currently has active. The second number identifies the maximum number of statements that were active.

MEMORY
This is a two-part value that is separated by a forward slash (/). The values are in kilobytes. The first value identifies the amount of memory that is currently being used to process the active statement. The second value identifies the maximum amount of memory that was used to process all statements that were issued by the user.

Example

The following example show output from the DISPLAY,USERS command that identifies different types of users:

```
06.34.42 JOB03544  CAC00200I DISPLAY,USERS
06.34.42 JOB03544  USER     TASKID     SERVICE
06.34.42 JOB03544  PROCESSID  THREADID   STMTS MEMORY
```

```
06.34.42 JOB03544  PUBLIC   9104784    PAA
06.34.42 JOB03544  0        0          00/00 633K/633K
06.34.42 JOB03544  CDCMCUSR 9104128    MAA
06.34.42 JOB03544  127      9104128    00/00 0K/0K
06.34.42 JOB03544  JOHNS    9104784    PAA
06.34.42 JOB03544  0        0          00/00 0K/0K
06.34.42 JOB03544  Total Number of USERS (current/maximum) = 3/3
```

In this example:

- PUBLIC is an internal user that the data server establishes.
- CDCMCUSR is a connection from InfoSphere CDC Access Server from IBM InfoSphere Change Data Capture Management Console. The data server receives one connection for an access server. Because the data server does not receive one connection for each user, a generic user ID, CDCMCUSR, is shown.
- JOHNS is the management console connection to the administration service. Because there is connection one for each management console user, the actual user ID, JOHNS, displays.

CANCEL command

You can cancel particular user sessions or all sessions for particular users.

About this task

You can find out which users are currently connected by issuing the DISPLAY,USERS command. The output of this command gives you the IDs of users and the IDs of the user sessions.

The cancel command takes effect when the query processor is not actively processing a query for the user that is being canceled. Users are notified upon completion of any action of theirs that follows the issuing of the command.

To disconnect users and user sessions, issue either of the following commands by using the MTO interface.

Procedure

- To cancel a particular user session, issue this command:
  `F name,CANCEL,SESSIONID=session ID`
- To cancel all sessions for a particular user, issue this command:
  `F name,CANCEL,USER=user ID`

DISPLAY,QUERIES command

You can list all of the queries that the query processors in a Classic data server are currently processing.

About this task

With the information provided in the output, you can issue the CANCEL,QUERY command to cancel a particular query.

Procedure

To list queries, issue this command in an MTO interface:

`F name,DISPLAY,QUERIES`

name The name of the started task or batch job for the Classic data server.

Results

When information is requested about the queries that are being tracked by the Classic data server, a WTO display message is generated for each query. For each query, the following information is displayed:

QUERY
The SQL statement name that is assigned by the client application or JDBC/ODBC Driver to track the query.

USER The user ID that issued the query.

SESSONID
The user control block address (in decimal format) of the user that issued the query.

SERVICE
The first 8-characters of the service name that is processing the query.

TASKID
The TCB address (in decimal format) of the query processor that is handling the query.

TYPE One of the following values that indicates the type of query that is being processed:

XQRY
SELECT statement

XJOI SELECT statement that contains a join condition

XUNI SELECT statement that contains a union

XINS INSERT statement

XUPD
UPDATE statement

XDEL DELETE statement

CALL CALL statement

STATE
One of the following values that indicates the state that the query is in:

INITIAL
Statement is created and not yet prepared.

PREPARED
Statement is prepared.

OPENED
Statement is opened.

EXECUTING
Statement is being executed.

FETCHED
Result sets are being fetched.

WAITING
Statement is waiting to be executed.

SUSPENDED
Statement is suspended.

CLOSED
Statement is closed.

MEMORY
This is a two-part value that is separated by a forward slash (/). The values that are displayed are in kilobytes. The first value identifies how much memory the query is currently using. The second value identifies the maximum amount of memory that has been allocated to process the statement.

CANCEL,QUERY command

You can cancel queries that are running on a Classic data server or enterprise server.

Before you begin

Before issuing this command, issue the `DISPLAY,QUERIES` command to get the name and session ID for the query that you want to cancel.

About this task

The cancel command takes effect when the query processor is not actively processing the query to be canceled. Users are notified upon completion of any action of theirs that follows the issuing of the command.

To cancel a query, issue the following command using the MTO interface:

```
F name,CANCEL,QUERY=name,SESSIONID=session ID
```

Classic data server configuration commands

After a Classic data server is created and running as a result of the installation customization process, you can modify the configuration for the Classic data server as needed by using a set of operator commands.

You use configuration commands for the following actions:

- Updating and displaying configuration data for a Classic data server.
  The commands include ADD, SET, DELETE, and DISPLAY.
- Importing and exporting configuration data for a Classic data server.
  The commands include EXPORT and IMPORT.

You can issue the configuration-related commands by using the master terminal operator (MTO) interface. You can also use the Classic Data Architect to update your configuration for a Classic data server. With both the Classic Data Architect and the MTO interface, you make configuration updates against a running Classic data server.

The configuration migration and maintenance utility, CACCFGUT, also supports the EXPORT and IMPORT commands and provides a REPORT command. You can use the CACCFGUT utility to issue the commands offline, when the Classic data server is not running. For example, the utility supports the EXPORT and IMPORT commands that enable you to restore a configuration environment to a previous point in time.

Commands for updating and displaying configurations for Classic data servers

You use the ADD, SET, DELETE, and DISPLAY configuration commands to update configuration information for services, service lists, global configuration parameters, and user-specific configurations.

When you add a service with the ADD command, you add the service to the configuration for the Classic data server. Adding the service does not automatically start the service. A service starts automatically during the next startup of the Classic data server if the value of the INITIALTHREADS configuration parameter is set to a value of 1 or greater. Otherwise, if you do not restart the Classic data server, you must issue a START,SERVICE command to start the service.

After a service is added, you cannot update the name of the service. To change a service name, you must delete the existing service and then add a new service with the new service name.

Basic validation occurs when you modify configuration parameters with the SET command. These validations are limited to general parameter data type and numeric range checks. The individual services validate specific configuration parameters to verify parameter content and relationships during service startup.

In addition to services configurations, you can add and modify service lists. A service list is a type of parameter that represents list values.

You use the ADD, SET, and DELETE commands to maintain these parameters.

To update a service list entry, you must delete the existing entry and then add a new entry.

ADD, DELETE, and DISPLAY service list commands

You can use the ADD and DELETE commands to add and remove service list entries. You can use the DISPLAY command to display service lists.

Description

A service list is a type of parameter that represents list values. A single service list can contain an unlimited number of entries.

A service list is always associated with a specific service.

- The MSGLIST parameter associated with the logger service defines the destination for a message or suppresses a message. Changes made to this service list are effective immediately.
- The SSIDEXCLUDELIST parameter associated with the log reader service represents an exclusion list. Changes made to this service list are effective when the Classic data server or the log reader service is restarted.

You use the ADD and DELETE commands to maintain these parameters. To update a service list entry, you must delete the existing entry, and then add a new entry. You cannot use the SET command to update a service list entry.

Adding service list entries

You can add a service list entry with the following ADD command:

ADD,CONFIG,SERVICELIST=*listname*,SERVICE=*servicename*,VALUE=*value*

Example

```
ADD,CONFIG,SERVICELIST=SSIDEXCLUDELIST,SERVICE=LRSI,VALUE=ID1
```

See the information about the specific service list parameter for any operational issues involved with adding service list entries.

Adding service list entries for message destinations

You can add a message filter and a message destination to a message list with the following ADD command:

```
ADD,CONFIG,SERVICELIST=MSGLIST,SERVICE=servicename,VALUE=<msg>/<dest>
```

Examples

To suppress a message:

```
ADD,CONFIG,SERVICELIST=MSGLIST,SERVICE=LOG,VALUE=CECN0002I/SUPPRESS
```

To change the destination of a message:

```
ADD,CONFIG,SERVICELIST=MSGLIST,SERVICE=LOG,VALUE=CECN0002I/DIAGLOG
```

Deleting service list entries

You can delete entries from a service list with the following DELETE command:

```
DELETE,CONFIG,SERVICELIST=listname,SERVICE=servicename,VALUE=value
```

Example

```
DELETE,CONFIG,SERVICELIST=SSIDEXCLUDELIST,SERVICE=LRSI,VALUE=ID1
```

When you delete an entry from a service list, you remove the entry from the configuration. See the information about the specific service list parameter for any operational issues involved with deleting service list entries.

Deleting service list entries for message destinations

You can remove a message filter with the following DELETE command:

```
DELETE,CONFIG,SERVICELIST=MSGLIST,SERVICE=servicename,VALUE=<msg>/<dest>
```

Deleting a message from the MSGLIST service list resets the destination of the message to its default destination.

Displaying service list entries

Service list information is displayed when you display configuration information for the related service. For example, to display exclusion list information related to the log reader service, issue the following command:

```
DISPLAY,CONFIG,SERVICE=LRSI
```

Parameters

listname
The service list name. You can specify one of the following service list names:
- MSGLIST
- SSIDEXCLUDELIST

`servicename`
: The service name associated with the service list name.

`value`
: List value.

  For MSGLIST, value is a string that defines the message ID and the destination. The string is in the format `<msg>/<dest>`, where:
  - *msg* is the message ID. The message number must begin with prefix CEC and contain nine characters.
  - *dest* is one of the following destinations:
    - CONSOLE: z/OS console
    - DIAGLOG: Diagnostic trace log
    - EVENT: Event log
    - SUPPRESS: No destination

  For SSIDEXCLUDELIST, value is the name of the IMS subsystem ID.

ADD configuration command

You can use the ADD command to add new service definitions, user-specific definitions, and service lists to a configuration.

Adding services

Adding a new service typically occurs during the installation customization process. Otherwise, you can add a service by using the ADD configuration command.

When you add a service, you specify the service name and select the service class for the new service. The service class is predefined.

The following table lists the service classes and the service type that each service class defines.

Table 55. Service classes.

Service class	Service type
CAP	Capture service
CNTL	Region controller service
DRA	IBM IMS database resource adapter (DRA) access service
GLOB	Global service
INIT	Client connection handler service
LOG	Logger service
LRSI	IMS log reader service
MAA	Monitoring service
OPER	Remote operator command service
PAA	Administration service
QP	Single-phase commit query processor service
QPLD	Refresh query processor service

You can add services with the following ADD command:

```
ADD,CONFIG,SERVICE=servicename,SERVICECLASS=serviceclass
```

Example

To add a new capture service named CS to the configuration, issue the following command:

```
ADD,CONFIG,SERVICE=CS,SERVICECLASS=CAP
```

In this example, the configuration parameters for the TESTV10 service are created with the default values for the LOG service class.

Adding user-specific configurations

When you add a user-specific configuration, you must specify a user ID. The user ID is associated with a defined service. You can override the default values for the configuration parameters for the service that you specify. In version 9.5, you can define user-specific configurations for query processor services only.

Command format:

```
ADD,CONFIG,USER=userid,SERVICE=servicename
```

Example

To add a new user record for TESTUSER to the configuration for service TESTV10, issue the following command:

```
ADD,CONFIG,CONFIG,USER=TESTUSER,SERVICE=TESTV10
```

Parameters

serviceclass
The name of the service class that the service is associated with.

servicename
When adding a service, specify the name of the service to create. When adding a user-specific configuration, specify the name of the service to override. The maximum length allowed for the name is 64 bytes.

userid
For a user-specific configuration, the user logon ID used to connect to the Classic data server. The maximum length allowed for the user ID is 8 bytes.

DELETE configuration command

You can use the DELETE command to remove service definitions, service list entries, and user-specific definitions from a configuration.

Deleting services

To delete a service, you must specify the name of the service to delete. If a service is running, you must stop the service before you delete the service. When a service configuration is deleted, any associated user-specific configuration is also deleted.

You cannot delete the core default services — the controller service and the logger service. You also cannot delete the GLOBAL service name which represents global parameters.

You can remove a non-core service with the following DELETE command:

```
DELETE,CONFIG,SERVICE=servicename
```

Example

To delete the service TESTV10 from the configuration, issue the following command:

```
DELETE,CONFIG,SERVICE=TESTV10
```

Deleting user-specific configurations

To delete a user-specific configuration, you must stop the associated service. You specify the user logon ID that is used to connect to the Classic data server.

DELETE command format:

```
DELETE,CONFIG,USER=userid,SERVICE=servicename
```

Example

To delete the user configuration for user ID TESTUSER related to service CACLOG from the configuration, issue the following command:

```
DELETE,CONFIG,USER=TESTUSER,SERVICE=CACLOG
```

Parameters

servicename
: When deleting a service, specify the name of the service to delete. When deleting a user-specific configuration, specify the name of the related service.

userid
: The user ID of the user to delete.

DISPLAY configuration command

You can use the DISPLAY command to display configuration information about services, service lists, user-specific configurations, and global configuration parameters.

Displaying all configuration information

You can display all configuration information for a Classic data server with the following DISPLAY command:

```
DISPLAY,CONFIG,ALL
```

During startup of the Classic data server, the contents of the configuration file are written to the first entry in the diagnostic log for the Classic data server.

Displaying service information

You can display configuration information about a specific service with the following DISPLAY command:

```
DISPLAY,CONFIG,SERVICE=servicename
```

Example

To display all configuration parameters in the service named CAPTURE, issue the following command:

```
DISPLAY,CONFIG,SERVICE=CAPTURE
```

Displaying user-specific configuration information

You can display user-specific configuration information about a specific user configuration, a specific user, or all users with the following DISPLAY commands.

- Display a user-specific configuration for the specified service:

  ```
  DISPLAY,CONFIG,USER=userid,SERVICE=servicename
  ```

  Example

  To display all parameters in user configuration for USER1 and service CACSAMP, issue the following command:

  ```
  DISPLAY,CONFIG,USER=USER1,SERVICE=CACSAMP
  ```

- Display all user-specific configurations across all services that contain the specified user ID:

  ```
  DISPLAY,CONFIG,USER=userid,SERVICE=ALL
  ```

  Example

  To display all parameters in all user configurations for USER1, issue the following command:

  ```
  DISPLAY,CONFIG,USER=USER1,SERVICE=ALL
  ```

- Display all user-specific configurations across all services:

  ```
  DISPLAY,CONFIG,USER=ALL
  ```

Parameters

servicename
: The name of the service to display. To display global parameters, specify GLOBAL as the service name.

userid
: The user ID of the user-specific configuration.

SET configuration command

You can use the SET command to modify parameter values defined for an existing service, a user-specific configuration, and global parameters.

Modifying service information

You can use the following SET commands to modify parameters in a single service, modify all services within a given service class, and modify all services across all service classes.

You can also use the SET command to reset the values of a configuration parameter value to the parameter default value.

- For a specific service, use the following SET command:

  ```
  SET,CONFIG,SERVICE=servicename,parm=value
  ```

  Example

  To set the TRACELEVEL parameter in service CAPTURE to 8, issue the following command:

  ```
  SET,CONFIG,SERVICE=CAPTURE,TRACELEVEL=8
  ```

- To reset a parameter value to the default for specific service, use the following SET command:

  ```
  SET,CONFIG,SERVICE=servicename,parm=DEFAULT
  ```

  Example

To set the TRACELEVEL parameter in service CAPTURE to the default value (4), issue the following command:

```
SET,CONFIG,SERVICE=CAPTURE,TRACELEVEL=DEFAULT
```

- For services within a given service class, use the following SET command:

  ```
  SET,CONFIG,SERVICECLASS=serviceclass,parm=value
  ```

  Example

  To set the TRACELEVEL parameter in all monitoring services to 3, issue the following command:

  ```
  SET,CONFIG,SERVICECLASS=MAA,TRACELEVEL=3
  ```

- For all services, use the following SET command:

  ```
  SET,CONFIG,SERVICE=ALL,parm=value
  ```

  Example

  To set the value of the TRACELEVEL parameter in all services to 2, issue the following command:

  ```
  SET,CONFIG,SERVICE=ALL,TRACELEVEL=2
  ```

Modifying user-specific configuration information

You can set the value of a specific parameter in a user-specific configuration with the following SET command.

```
SET,CONFIG,USER=userid,SERVICE=servicename,parm=value
```

Example: Set the value of the TRACELEVEL parameter in the user-specific configuration for USER1 for TEST95 to 2.

```
SET,CONFIG,USER=USER1,SERVICE=TEST95,TRACELEVEL=2
```

You can also use the DEFAULT keyword on the SET command to return a parameter to its default value.

Parameters

parm
: The name of the configuration parameter to modify.

serviceclass
: The service class of the service to modify.

servicename
: The name of the service that contains the configuration parameter to modify. For global parameters, specify GLOBAL as the service name. The maximum length allowed for the name is 64 bytes.

userid
: The name of the user logon ID. The maximum length allowed for the user ID is 8 bytes.

value
: The new parameter value.

Usage notes

The following rules apply to specifying lowercase and uppercase character values for the SET command:

- Character strings that contain embedded spaces or special characters must be enclosed in either single or double quotation marks.

– For double quotation marks (“), the character string is set to uppercase.
– For single quotation marks ('), values specified in lowercase are saved as lowercase.

Commands for importing and exporting configurations for a Classic data server

An export does not affect an existing configuration for a Classic data server. The EXPORT command creates a snapshot of the configuration in a command file that contains ADD and SET commands. The command file can be used as input to the IMPORT command.

You can use the IMPORT and EXPORT commands to perform the following functions:

- Back up and restore a current configuration environment. The EXPORT command creates a command file that consists of ADD and SET commands based on the current environment that can later be imported into a Classic data server with newly initialized configuration files to complete the restore process.

  Important: Frequently back up configuration files by copying them or by using the EXPORT command.
- Apply updates to the current configuration environment. You can use the IMPORT command to apply updates to the current configuration environment in multiple updates or single updates. You can use the EXPORT command to create a command file or create a command file manually.
- Save different versions of a configuration environment. You can use exported configuration output to create a clone of the existing configuration for a Classic data server. For example, the EXPORT command is useful in a test environment where you can rebuild the configuration required for a specific test scenario by importing the saved configuration. The EXPORT and IMPORT commands can also be an effective mechanism for cloning Classic data servers.

Example

You can use the EXPORT and IMPORT process to restore a specific configuration environment to a previous point in time. By using the EXPORT command, you can create a command file that is based on the configuration environment of a running Classic data server. You can then use this command file to update a different configuration file by using the IMPORT command.

For example, you can use the EXPORT command to generate a command file, and then IMPORT that command file on a server that is running with a newly initialized configuration file.

To build a new configuration environment that is identical to an existing configuration environment, export the source configuration to the desired target file. Then run the configuration migration and maintenance utility CACCFGUT to import the original source configuration to the new target environment.

You can also issue the IMPORT command while a Classic data server is running to update parameter settings in the current configuration environment.

EXPORT configuration command

The EXPORT command is useful for multiple purposes, such as backing up configuration information and cloning a configuration for a Classic data server.

Description

You can use the EXPORT and IMPORT process to restore a specific configuration environment to a previous point in time. By using the EXPORT command, you can create a command file that is based on the configuration environment of a running Classic data server. You can then use this command file to update a different configuration file by using the IMPORT command.

The target of the EXPORT command is a PDS member or a sequential file. If the file or member that you specify in the EXPORT command already exists, it is rewritten. If the file or member does not exist, it is created.

If you predefine the EXPORT target:
- The minimum record length is 80 bytes.
- The format can be either fixed or variable length records.

The EXPORT command does not support GDGs.

The owner ID associated with the Classic data server job must have authorization with an external security manager (ESM), such as the Resource Access Control Facility (RACF), to create or access the EXPORT target data set (sequential file or PDS member).

When a PDS member is specified as the target of the EXPORT command, an attempt to run the EXPORT command fails if another user or job accesses the PDS member at the same time. To avoid this situation, you create a PDS member to use for the IMPORT and EXPORT process only.

Exporting configuration information

EXPORT command format:

```
EXPORT,CONFIG,FILENAME=DSN:dsname | DSN:dsname(member) | DDN:ddname
 | DDN:ddname(member)
```

Example

The following example shows sample file contents of an EXPORT file:

```
--SET,CONFIG,SERVICE=GLOBAL,MESSAGEPOOLSIZE=67108864;
--SET,CONFIG,SERVICE=GLOBAL,DATACONVERRACT=1;
--SET,CONFIG,SERVICE=GLOBAL,FETCHBUFSIZE=32000;
--SET,CONFIG,SERVICE=GLOBAL,DECODEBUFSIZE=8192;
--SET,CONFIG,SERVICE=GLOBAL,STATICCATALOGS=0;
--SET,CONFIG,SERVICE=GLOBAL,TASKPARM='';
--SET,CONFIG,SERVICE=GLOBAL,DATAVALIDATEACT=0;
--SET,CONFIG,SERVICE=GLOBAL,REPORTLOGCOUNT=0;
ADD,CONFIG,SERVICE=CNTL,SERVICECLASS=CNTL;
--SET,CONFIG,SERVICE=CNTL,INITIALTHREADS=1;
--SET,CONFIG,SERVICE=CNTL,MAXTHREADS=1;
--SET,CONFIG,SERVICE=CNTL,MAXUSERS=100;
--SET,CONFIG,SERVICE=CNTL,TRACELEVEL=4;
--SET,CONFIG,SERVICE=CNTL,RESPONSETIMEOUT=5M;
--SET,CONFIG,SERVICE=CNTL,IDLETIMEOUT=5M;
SET,CONFIG,SERVICE=CNTL,SEQUENCE=1;
```

In the example, the export operation produces a command file of all overridden parameters as active SET commands. SET commands are also generated for all parameters that use default values. These particular SET commands are generated as comments. The prefix "– –" in the first two columns of the command identifies comments.

Parameters

ddname
The DD statement defined in the JCL that starts the Classic data server. The DD name points to the target of the EXPORT command.

dsname
The name of the EXPORT data set.

Usage notes

The following rules apply to the format of command files:

- Commands must end with a semicolon.
- A comment begins with two dashes ("– –") in columns 1 and 2.
- A single command can span multiple lines. You can break a line after a comma or a space. You cannot break a keyword or value across multiple lines. For example:

```
SET,CONFIG,SERVICE=CAPTURE_SERVICE_NAME/CRAR/QRAR/2048/8,TEMPFILESPACE=
'HIPERSPACE,INIT=2048M,MAX=2048M,EXTEND=0M';
```

Example

This EXPORT command creates the file USER.TEST.FILE if the file does not already exist.

```
EXPORT,CONFIG,FILENAME=DSN:USER.TEST.FILE
```

IMPORT configuration command

You can use the IMPORT command to apply multiple updates or single updates to the configuration file for a running Classic data server. You can also use the IMPORT command to perform recovery operations to restore configuration information.

Description

The updates that the IMPORT command processes must reside in an existing IBM z/OS PDS member or sequential file. You can use either of the following methods to build the input file:

- Manually create the file and populate it with a defined set of commands.

  When you manually create a command file, you can specify any valid format of the ADD, SET, or DELETE configuration commands.
- Issue the EXPORT command to generate an IMPORT command file.

All commands in the file are processed whether or not a single command fails. Any errors encountered during the IMPORT process are displayed on the operator console. For example, unknown commands or incorrect command syntax can cause errors. Attempting to add a service that already exists can also cause an error. A status message is displayed when the process is complete.

You can use the IMPORT command at any time to change configuration parameter settings for existing services on a running Classic data server.

Importing configuration command files

IMPORT command format:

```
IMPORT,CONFIG,FILENAME=DSN:dsname | DSN:dsname(member) | DDN:ddname
 | DDN:ddname(member)
```

The data that you import must be in the command file format as described for the EXPORT command.

Example:

The following sample shows the contents of a sample IMPORT command file:

```
ADD,CONFIG,SERVICE=IMSLRS,SERVICECLASS=LRSI;
SET,CONFIG,SERVICE=IMSLRS,TRACELEVEL=3;
ADD,CONFIG,SERVICELIST=SSIDEXCLUDELIST,SERVICE=LRSI,VALUE="IMS1";
ADD,CONFIG,SERVICELIST=SSIDEXCLUDELIST,SERVICE=LRSI,VALUE="IMS2";
```

When this command file is imported, these changes occur:
- A new service named IMSLRS is added
- The value of TRACELEVEL is set to 3 for the IMSLRS service
- The value IMS1 is added to the exclusion list SSIDEXCLUDELIST for service IMSLRS
- The value IMS2 is added to the exclusion list SSIDEXCLUDELIST for service IMSLRS

Parameters

ddname
The DD statement defined in the JCL that starts the Classic data server. The DD name points to the source IMPORT command file.

dsname
The name of the IMPORT data set.

Example

This IMPORT command imports the file USER.PDS.FILE.

```
IMPORT,CONFIG,FILENAME=DSN:USER.PDS.FILE(USER1)
```

IMS-specific commands

IMS-specific commands apply to IMS replication.

RELEASE,RECON command

You can use the RELEASE,RECON command to release the existing holds (enqueues) on the current RECON data set.

Description

The RELEASE,RECON command is useful for releasing enqueues when your IMS administrator reorganizes RECON data sets.

This command is associated with the IMS log reader service. When the log reader interface (LRI) completes command processing, it passes a return code to the IMS log reader service. The IMS log reader service then logs the result of the command.

Syntax

This command releases existing holds:

```
'RELEASE,RECON'
```

Syntax

This command releases existing holds:

```
data-server-name,RELEASE,RECON
```

Parameters

data-server name
: The name of the Classic data server that hosts the IMS log reader service.

Chapter 18. 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 a metadata catalog when the data server is stopped.

The catalog-related JCL is customized during the installation customization process. You can configure the standalone CACCATUT JCL to perform one of the following operations when it runs:

- Create and initialize a version 11.3 zFS metadata catalog
- Create and initialize a version 11.3 sequential metadata catalog
- Create and initialize a version 11.3 linear metadata catalog
- Upgrade an existing pre-version 11.3 sequential metadata catalog to a version 11.3 zFS or sequential metadata catalog.
- Upgrade an existing pre-version 11.3 linear metadata catalog to a version 11.3 zFS or sequential metadata catalog.
- Report on space utilization, the contents of a zFS or sequential metadata catalog, and any corruptions that might exist within the metadata catalog.
- Reorganize the contents of a zFS or sequential metadata catalog to reclaim unused space and, where possible, correct any corruptions that might exist.
- Create a version 11.3 zFS or sequential copy of a version 11.3 zFS, sequential, or linear metadata catalog.
- Load system objects into the metadata catalog.

You can set up your metadata catalogs manually. See the guidelines for estimating the size of the metadata catalog for detailed information.

The catalog initialization and maintenance utility is distributed in library SCACLOAD as member name CACCATUT.

Estimating the size of the catalog initialization and maintenance utility

You can use the guidelines provided with the formula in this topic to estimate the size of the metadata catalog.

During the installation customization process you provide input to estimate the size of the metadata catalog. This estimate is used to allocate the file space that will store the metadata catalogs. The estimate is based on the formula below.

You can also use the formula if you set up your metadata catalogs manually. This is a generous estimate because an accurate size determination is not feasible and there is no guarantee to fully store a catalog. In the unlikely event that you run out of file space, you will receive an explicit error message providing further instructions.

```
ESTIMATED CATALOG OBJECT SIZE in Bytes = 314572800

   + <number of expected tables> * 3710
   + <number of expected tables> * <maximum or average number of expected columns per table>  * 776

   + <number of expected views> * 3086
   + <number of expected views> * <maximum or average number of expected columns per view>  * 776
   + <number of expected indexes> * 1278
   + <number of expected indexes> * <maximum or average number of expected columns per index>  * 264

   + <number of expected stored procedures> * 1543
   + <number of expected stored procedures> * <maximum or average number of parameters per stored procedure>  * 264

TOTAL ESTIMATED CATALOG SIZE in Bytes = ESTIMATED CATALOG OBJECT SIZE

   + <user-defined reserved space in bytes>

TOTAL ESTIMATED CATALOG SIZE in Megabytes =  TOTAL ESTIMATED CATALOG SIZE in Bytes / 1048576
```

Notes:

- You supply the number in brackets based on your planned usage of the catalog.
- A conservative user would use the maximum number instead of the average number in the formula.
- Minimally, you should reserve an additional 1/3 of the calculated catalog object size.
- Indexes are typically used in Classic Federation for IMS, VSAM, DB2, and Adabas data sources only.
- Because stored procedures are rarely used, it is likely that you can ignore providing input for them.
- A reserve for GRANT authorities is included in the formula which should be sufficient, even for the largest catalog.

Creating and initializing zFS metadata catalogs

The zSeries File System (zFS) metadata catalog is initialized as part of the installation customization process.

Before you begin

The zFS format of the metadata catalog is the default. This format is the recommended configuration for the metadata catalog that provides significant performance and capacity capabilities in comparison to the sequential or linear data set formats used in releases prior to version 11.3. The sequential and linear formats continue to be supported for compatability.

Using the zFS format of the metadata catalog requires the definition of a z/FS file system. The data set that contains the zFS file systems is call a zFS *aggregate*. This data set will contain the files and directories for the file system and is mounted into the z/OS UNIX hierarchy.

You must create and mount the file system before trying to create the metadata catalog. To mount the file system, the user issuing the MOUNT command must have the following Unix privileges:

- SAF READ-access level authorization is required for the SUPERUSER.FILESYS.PFSCTL resource in the UNIXPRIV class to run the zFS administration command, IOEZADM.
- SAF READ-access level authorization is required for the SUPERUSER.FILESYS.MOUNT resource in the UNIXPRIV class to perform MOUNT and UNMOUNT operations against USS file systems

For more information about zFS files, see *Distributed File Service, SMB, and zFS* and *z/OS UNIX System Services* in the z/OS product documentation.

The installation customization provides steps to setup the files system and an example for mounting the file system in addition to the steps included below.

About this task

zFS metadata catalogs provide improved performance and capacity compared to sequential and linear metadata catalogs. You can update a zFS metadata catalog and eliminate the need to maintain both an updatable sequential catalog and a non-updatable linear catalog. With zFS support, catalogs can be greater than 4GB in size in contrast to the 2GB limitation for sequential and linear catalogs.

Recommendation: Use the sizing formula during the installation customization process to estimate the required catalog size. See the topic 'Estimating the size of the catalog initialization and maintenance utility.

Procedure

1. Define and format a zFS aggregate by using *USERHLQ*.USERSAMP(CECCRZCT).
2. Mount the file system using the sample MOUNT command provided in *USERHLQ*.USERSAMP(CECCRZCT). Ensure that the user issuing the MOUNT command has the necessary privileges.
3. Follow the installation customization process. During the customization process, the CACCATLG member in the SCACSAMP data set is customized. The zFS metadata catalog is initialized as part of the installation customization process.
4. Optional: To initialize additional catalogs as needed, customize and run JCL member CACCATLG. Member CACCATLG in the SCACSAMP data set contains JCL to run CACCATUT with the INIT option. The INIT operation of the catalog initialization and maintenance utility (CACCATUT) initializes USS file system files for a version 11.3 zFS metadata catalog and creates the SYSIBM and SYSCAC system tables for the metadata catalog.
   a. Provide a job card that is valid for your site.
   b. Change the value of the CAC parameter to your high-level qualifier.
   c. Change the value of the DISKU parameter to a valid DASD unit type for your site.
   d. Change the value of the DISKVOL parameter to identify the volume where you want to locate the system catalog.

Results

When the catalog initializes, it grants SYSADM privileges to the user ID that creates the catalog.

Creating and initializing sequential metadata catalogs

The sequential metadata catalog is initialized as part of the installation customization process.

Before you begin

Important: The zFS format of the metadata catalog is the default. This format is the recommended configuration for the metadata catalog that provides significant performance and capacity capabilities in comparison to the sequential data set format.

To grant ownership privileges for the catalog, ensure that each TSO user ID is assigned a unique OMVS UID. Set an automatically generated UID for the user IDs that you are using in RACF:

```
ALTUSER USER01 OMVS(NOUID)
ALTUSER USER01 OMVS(AUTOUID)
```

Procedure

- Follow the installation customization process. The CDCCATNM member in the SCACSAMP data set is customized. The sequential metadata catalog is initialized as part of the installation customization process.
- Optional: To initialize additional catalogs as needed, customize and run JCL member CDCCATNM. Member CDCCATNM in the SCACSAMP data set contains JCL to run CACCATUT with the INIT option. The INIT operation of the catalog initialization and maintenance utility (CACCATUT) initializes data sets for a version 11.3 sequential metadata catalog and creates the SYSIBM and SYSCAC system tables that make up the 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.

Results

When the catalog initializes, it grants SYSADM privileges to the user ID that creates the catalog automatically.

Creating and initializing linear metadata catalogs

To use a 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 linear metadata catalog.

Before you begin

A sequential metadata catalog must exist on the z/OS LPAR where the Classic data server is located. To create a sequential metadata catalog, see "Creating and initializing sequential metadata catalogs" on page 259.

The sequential metadata catalog must contain the final versions of all of the mapped tables and other objects that you plan to use. 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.

About this task

Prior to version 11.1, you could use linear metadata catalogs to improve performance. However, they are dependent on a sequential metadata catalog because linear metadata catalogs cannot be updated.

Important: The zFS format of the metadata catalog is the default. A zFS metadata catalog is updatable and its performance matches, if not exceeds, the performance of a linear metadata catalog. zFS also accommodates a catalog size greater than 4GB, making it the most flexible metadata catalog option.

Procedure

1. Stop the Classic data server.
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 sequential metadata catalog into the linear metadata catalog. See "Copying metadata catalogs."
4. In the configuration file for the Classic data server, set the value of the configuration parameter STATICCATALOGS to 1.
5. Update the JCL for the Classic data server to point to the linear metadata catalog.
6. Start the Classic data server.

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 metadata catalog for the same release as the Classic Federation data server.

The INCAT and ININDX DD statements are required and identify the metadata catalog that is to be copied into the target metadata catalog identified by the CACCAT and CACINDX DD statements. The INCAT and ININDX DD statements cannot refer to the same data sets names that CACCAT and CACINDX 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.

Note: If the target metadata catalog is not large enough, CACCATUT issues informational message 0x00710401.

Procedure

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. For the PARM keyword of the EXEC statement, specify COPY.

2. Update the JCL for the Classic data server to point to the target metadata catalog.

Upgrading metadata catalogs

You can upgrade a sequential or zFS metadata catalog to a new version of the metadata catalog.

About this task

You can upgrade metadata catalogs from:

- A previous version of a sequential metadata catalog to a new version of a sequential or zFS metadata catalog.
- A current version of a sequential metadata catalog to a new version of a zFS metadata catalog.
- A previous version of a zFS metadata catalog to a new version of a zFS metadata catalog.

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 USS file system path of the zFS metadata catalog or to sequential metadata catalog data sets. These statements identify 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

1. Follow the installation customization and migration process. The CACCATUP member in the SCACSAMP data set is customized. The sequential metadata catalog is upgraded as part of the installation customization and migration process. See Customizing installation environments.
2. To update the catalog as needed after the installation customization and migration process, customize and run member CACCATUP.
   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 DISKU parameter to a valid DASD unit type for your site.

   e. Change the VOLSER parameter to identify the volume where you want the metadata catalog located.
3. Update the JCL for the Classic data server to point to the new metadata catalog.

Reorganizing metadata catalogs

You can reorganize a zFS or sequential metadata catalog to reclaim wasted disk space.

About this task

You can reorganize zFS and sequential metadata catalogs. You cannot reorganize linear metadata catalogs.

The REORG operation compresses the contents of an existing metadata catalog by moving the metadata catalog. The new version of the 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.

Note: If the target metadata catalog is not large enough, CACCATUT issues informational message 0x00710401.

Procedure

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 Classic data server to point to the new metadata catalog.

Loading system objects into metadata catalogs

You can load system objects into the metadata catalog.

About this task

The METALOAD operation populates new system objects in the catalog such as stored procedures and table definitions.

The catalog that the CACCAT and CACINDX DD statements point to is updated with the most current object definitions.

Procedure

1. Run member CACCATMD. The customized CACCATMD member provides an example of how to run the METALOAD command. CACCATMD is available in the SCACSAMP data set.
2. For the PARM keyword of the EXEC statement, specify METALOAD.

Results

Message 0x00760205 is written to SYSPRINT for each system object that is successfully loaded.

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 265.
- For more information on detail reports, see “Object detail reports” on page 268.

Procedure

Run member CACCATRP. The customized CACCATRP member is available 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 V11.3 metadata catalog, the version identifier is reported as 11.03.00.

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 V11.3 metadata catalog, the version identifier is reported as 11.03.00.

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:

- V11.3 metadata catalog – IBM InfoSphere Classic V11.3 *build-date*
- V11.1 metadata catalog – IBM InfoSphere Classic V11.1 *build-date*
- V10.1 metadata catalog – IBM InfoSphere Classic V10.1 *build-date*
- V9.5 metadata catalog – IBM InfoSphere Classic V9.5 *build-date*
- V9.1 metadata catalog – IBM WebSphere Classic V9.1 *build-date*

The build-date takes the form *mmddyyyy* and is updated for major releases and roll-up PTFs.

Date created
The date and time that the data 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 and later releases of the 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 number *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.

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 IMS tables the segments 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

                                       Authorization Column  Table  Frag.
                          Next   Prev  Start    End / Parm  Index  Other
                        ------ ------ ------ ------ ------ ------ ------
 Record    RID Size Type Name
      1      0  128  IRD IBM InfoSphere Classic V11.3
                           N/A    N/A    N/A    N/A    N/A    N/A    N/A
      2      1 1664  TAB SYSIBM.SYSTABLES
                           200  41135      0      0     14   2596      0
      3     14  768  COL SYSIBM.SYSTABLES.NAME
                            20      1    N/A    N/A      1    N/A    N/A
      4     20  768  COL SYSIBM.SYSTABLES.CREATOR
                            26     14    N/A    N/A      1    N/A    N/A
      5     26  768  COL SYSIBM.SYSTABLES.TYPE
                            32     20    N/A    N/A      1    N/A    N/A
      6     32  768  COL SYSIBM.SYSTABLES.DBNAME
                            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.

Object type	Name
TAB	Qualified table name: *owner-name.table-name*
COL	Qualified column name: *owner-name.table-name.column-name*

Object type	Name
FRG	Depends on the type of fragment, as follows: • Record Array Fragment Definitions – Fragment ID and level number • IMS tables – segment name • Table-level fragments for tables other than IMS – Fragment ID and level number • System fragments – ‘SYSTEM'
IDX	Qualified index name: *owner-name.index-name*
KEY	Qualified key name: *owner-name.index-name.column-name*
VEW	Qualified view name: *owner-name.view-name*
VDP	Qualified view name and view dependent number: *owner-name.view-name(number)*
RTN	Qualified stored procedure name: *owner-name.routine-name*
PRM	Qualified parameter name: *owner-name.routine-name.parameter-name*
ADB	User and database class: *user-name.database-class*
ATB	Qualified table name and user: *owner-name.table-name.authorization-ID*
ART	Qualified stored procedure name and user: *owner-name.routine-name.authorization-ID*
AUS	Qualified object name and user: *owner-name.table-name.authorization-ID*
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.

Chapter 19. The metadata utility

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

The metadata utility accepts as input the DDL that is generated by Classic Data Architect.

Running the metadata utility

The metadata utility executes as a z/OS batch job. Sample JCL to execute the metadata utility is distributed as member name CACMETAU in the SCACSAMP library. The name of the load module for the metadata utility is CACMETA.

Before you begin

You must know the name of the query processor that you want the metadata utility to connect to. You must also know the user ID and password required for a connection.

The query processor that you want to connect to must be running.

About this task

The names of the sample JCL and load modules distributed in V11.3 are the same as those used in prior releases of the metadata utility. You cannot use an earlier release of the metadata utility JCL to run V11.3 of the utility without modification. Use the *USERHLQ*.USERSAMP(CACMETAU) sample for V11.3.

MSGCAT DD
: The metadata utility JCL must contain an MSGCAT DD statement that references the message catalog. The message catalog is accessed by the CLI component and the metadata utility to retrieve the text for error messages reported by the Classic data server and error conditions detected by CLI or by the metadata utility.

SYSOUT DD
: The SYSOUT DD statement is used to record a summary of the processing performed by the metadata utility.

SYSPRINT DD
: As the metadata utility reads input records and executes each statement, the metadata utility echoes each statement and the execution status of each statement out to the SYSPRINT DD.

SYSIN DD
: The sample JCL uses data set concatenation on the SYSIN DD statement to provide the CONNECT TO SERVER statement. The metadata utility uses the statement to configure the run-time environment so that the CLI interface can connect to the correct Classic data server to process the DDL statements. The DDL statements are referenced in the second data set referenced by the SYSIN DD statement and referenced by the DDLIN substitution variable. A sample CONNECT TO SERVER statement is provided in SCACCONF sample member CACMUCON.

The file referenced by the SYSIN DD statement is treated as a text input stream, and can be in fixed length or variable length format. There is no restriction on the record length.

DDLOUT DD
The DDLOUT DD statement can be used to support GENERATE DDL statements. If you specify this statement, the metadata utility will write DDL to the specified DDLOUT file for any successful GENERATE DDL statements that are received from SYSIN. If the statement is not specified, any generated DDL is written to SYSPRINT.

Procedure

1. Open for editing member CACMETAU in the SCACSAMP data set and make these changes:
   a. Provide a job card that is valid for your site.
   b. Change the `CAC` parameter to the installed high-level qualifier.
   c. Customize the connection statement in member CACMUCON to point to the Classic data server with the metadata catalog that you want to update. See "CONNECT TO SERVER statement for the metadata utility" on page 278.
   d. Change the `DDLIN` parameter to the member that contains the DDL statements to process.
   e. If you intend to run GENERATE DDL statements and you want to write the output to a file, add a `DDLOUT DD` statement that points to the file that will contain the generated DDL.
   f. If you need to process large DDL statements, change the RGN parameter to change the region size. Increase the region size in increments of two megabytes.
2. Submit the job.

Results

As the metadata utility reads input records, it echoes them out to the SYSPRINT DD statement.

A header is printed for each statement in the SYSIN input file. Each input record read from SYSIN is echoed in the SYSPRINT file. The line number appears in front of the statement text. The line numbers increase monotonically. When the SYSIN data set has a record length that is greater than 80-bytes, the SYSIN input is wrapped in 80-character increments. Wrapped lines do not have line numbers.

After the statements, the output specifies whether the statements ran successfully or failed. Next, any error, warning, or information messages associated with the execution of the statements appear.

Example

The following example shows sample SYSPRINT output:

```
LINE NO.    STATEMENT
      74    CONNECT TO SERVER "CACSAMP" "TCP/0.0.0.0/9087"
      75    USERID USER01;
CACM001I SQLCODE = 0, INFO: Statement execution was successful.
LINE NO.    STATEMENT
      77    DROP TABLE "CAC"."EMPLOYEE";
CACM001I SQLCODE = 0, INFO: Statement execution was successful.
LINE NO.    STATEMENT
      78
      79    CREATE TABLE "CAC"."EMPLOYEE" DBTYPE VSAM
      80      DS "USER01.EMPLOYEE"
      81    (
      82
      83      "ENAME" SOURCE DEFINITION
      84              DATAMAP OFFSET 0 LENGTH 20
      85              DATATYPE C
      86              USE AS CHAR(20),
      87      "PHONE" SOURCE DEFINITION
      88              DATAMAP OFFSET 20 LENGTH 4
      89              DATATYPE UF
      90              USE AS INTEGER,
      91      "MAILID" SOURCE DEFINITION
      92              DATAMAP OFFSET 25 LENGTH 6
      93              DATATYPE C
      94              USE AS CHAR(6),
      95      "SALARY" SOURCE DEFINITION
      96              DATAMAP OFFSET 31 LENGTH 4
      97              DATATYPE P
      98              USE AS DECIMAL(7 , 2),
      99      "JOBID" SOURCE DEFINITION
     100              DATAMAP OFFSET 35 LENGTH 4
     101              DATATYPE D
     102              USE AS REAL,
     103      "EMPID" SOURCE DEFINITION
     104              DATAMAP OFFSET 39 LENGTH 4
     105              DATATYPE UF
     106              USE AS INTEGER,
     107      "DEPTID" SOURCE DEFINITION
     108              DATAMAP OFFSET 43 LENGTH 2
     109              DATATYPE UH
     110              USE AS SMALLINT,
     111      "DEPARTMENT" SOURCE DEFINITION
     112              DATAMAP OFFSET 47 LENGTH 15
     113              DATATYPE C
     114              USE AS CHAR(15) );
CACM001I SQLCODE = 0, INFO: Statement execution was successful.
LINE NO.    STATEMENT
     115    GRANT DELETE, INSERT, UPDATE, SELECT ON TABLE "CAC"."EMPLOYEE" TO
             PUBLIC;
CACM001I SQLCODE = 0, INFO: Statement execution was successful.
```

Figure 9. Sample SYSPRINT output

The sample output shows that the metadata utility ran four statements. These statements existed in the SYSIN input stream.

Summary reports that are generated by the metadata utility

The metadata utility writes error and summary information to the SYSOUT data set.

During normal processing, the output written to SYSOUT consists of a two line “header” that identifies the format of the information displayed. For each statement processed by the metadata utility, the starting and ending line numbers from the SYSIN file are reported. The starting line number identifies the actual line on which the statement started and does not include any comments that might have been encountered before the statement. The last line of the statement is the line where the ; is detected and does not take into account any comment lines that might exist after the statement. Therefore, it is possible to see "jumps" between ending and starting line numbers.

The third column in the summary report output identifies the statements processing status. For example, in the SYSPRINT output, the processing status is 0 if no errors were reported, a negative SQL error code value or a hexadecimal form of a system error message. Additionally, N/A is displayed for generated statements that were not part of the SYSIN input stream, such as DISCONNECT statements.

After the processing status is the name of the statement that was encountered followed by the object name that the statement is referencing. Table 56 identifies the statement types that are supported by the metadata utility. For each entry in the table, the statement name is followed by the type of object name that is extracted from the statement. For each entry the statement class is also identified, which determines how the statement is processed by the metadata utility.

Table 56. Statement types that are supported by the metadata utility

Types of statement	Names of objects	Class
CREATE TABLE	Table name	DDL
CREATE INDEX	Index name	DDL
CREATE PROCEDURE	Procedure name	DDL
CREATE VIEW	View name	DDL
COMMENT ON	Table name, index name, column name, or stored procedure name	DDL
ALTER TABLE	Table name	DDL
DROP INDEX	Index name	DDL
DROP TABLE	Table name	DDL
DROP PROCEDURE	Procedure name	DDL
DROP VIEW	View name	DDL
GRANT	N/A	DDL
REVOKE	N/A	DDL
CONNECT TO SERVER	Data source name	Connect
DISCONNECT FROM SERVER	Data source name	Connect

If a statement is encountered that is not in Table 56, then the metadata utility issues error 0x00760002 (Unsupported statement encountered by the metadata utility: start-of-statement). The first 25 characters of the statement are displayed in the summary report, and the object name is set to UNKNOWN. The line numbers that display for this unknown/unsupported statement start with the first line of the statement. The ending line number is where a ; is encountered or end-of-file is reached. The contents of this statement are also echoed in SYSPRINT followed by the 0x00760008 error message text and the first 70 characters of the statement.

For supported statements, a single line is generated in the SYSOUT summary report for each statement that was identified in the SYSIN input stream. A single line is also generated in the SYSOUT summary report for any DISCONNECT or

COMMENT ON statements generated by the metadata utility. For generated statements, the starting and ending line numbers are identified as N/A. Following each statement summary line, for statements that either reported errors when they were executed or if the statement reported one or more warning messages, information messages, or both, the SYSOUT output echoes the same error, warning, and information messages that display in the SYSPRINT output stream.

```
  Start    End  Processing
 Line # Line #      Status Statement              Object
      1      1           0 CONNECT TO SERVER      CACSAMP
CACM0001I SQLCODE = 0, INFO: Statement execution was successful.

      2      2        -204 DROP TABLE             CAC.EMPLOYEE
CACM0002I SQLCODE = -204, ERROR: CAC.EMPLOYEE is an undefined name.

      3     22           0 CREATE TABLE           CAC.EMPLOYEE
CACM0001I SQLCODE = 0, INFO: Statement execution was successful.

    N/A    N/A           0 DISCONNECT FROM SERVER CACSAMP
CACM0001I SQLCODE = 0, INFO: Statement execution was successful.
```

Figure 10. Sample SYSOUT summary report

Return codes for the metadata utility

The metadata utility uses the z/OS return code to identify the overall success or failure of its execution.

The following table identifies the return codes that the metadata utility can issue.

Table 57. List of return codes for the metadata utility

Return code	Meaning
20	Environmental setup error detected. Processing is not attempted.
16	Errors detected in the configuration member or while initializing the run-time environment. Processing is not attempted.
12	Error reported connecting to the Classic data server. Processing is not attempted.
8	Error reported by the server, CLI, or metadata utility while processing a statement. Processing continues with the next statement in the SYSIN input stream.
4	Warning messages issued by the server or metadata utility. Statements were processed successfully.
0	No errors were reported.

CONNECT statements for the metadata utility

The metadata utility supports two different types of CONNECT statements.

The metadata utility does not check that CICS connection statements are valid. When it encountered one of these statements, the metadata utility echoes the statement in the SYSOUT output and displays message CACM010I with the following information:

```
0x00760200 - CONNECT TO CICS statement no longer used.
```

CONNECT TO SERVER statement for the metadata utility

The metadata utility uses the information in this statement to connect to a query processor and update the content of a metadata catalog.

The CONNECT TO SERVER statement uses the following syntax:

```
►►──CONNECT TO SERVER──datasource-name──connection-URL──┬───────────────────┬──────────►
                                                         └─USERID──user-ID──┘

►──┬───────────────────────┬──;──────────────────────────────────────────────────────►◄
   └─PASSWORD──password───┘
```

Parameters

datasource-name
: A 1- to 16-character native identifier that names the query processor (data source) to connect to in the Classic data server identified by *connection-URL*. *datasource-name* is used until it is explicitly changed by another CONNECT TO SERVER statement or until the metadata utility terminates.

connection-URL
: A native identifier that provides information used to connect to a Classic data server. The URL must be delimited and match the connection information of an active TCP/IP connection handler in the Classic data server.

USERID *user-ID*
: This optional clause specifies the user ID for connecting to the Classic data server. The default value is the TSO user ID associated with the metadata utility job. The user ID is a 1- to 7-character short-native identifier.

PASSWORD *password*
: This clause is required when the Classic data server has the SAF exit active. *password* provides authentication information when the connection with the Classic data server is established. By default, the connection is established without a password.

If the PASSWORD clause is specified, the password is a 16-character DES encrypted character string specified in hexadecimal format. This password needs to be generated using the password generator utility on Windows or on z/OS.

GENERATE DDL statement for the metadata utility

To generate DDL from the catalog, you can use Classic Data Architect. You can also use the metadata utility directly on the z/OS system. The metadata utility can generate the same kinds of objects that Classic Data Architect can generate.

Table 58. Generated Statements supported for each object type

Generated DDL	Object
DROP statements	TABLE, VIEW, INDEX, PROCEDURE
CREATE statements	TABLE, VIEW, INDEX, PROCEDURE
COMMENT ON statements	TABLE, VIEW, INDEX, PROCEDURE
GRANT statements	TABLE, VIEW, INDEX, PROCEDURE

Table 58. Generated Statements supported for each object type (continued)

Generated DDL	Object
ALTER statements	TABLE, VIEW
Indexes	TABLE

One situation where you might choose to create DDL from the metadata utility is in a migration. For example, you could generate DDL for objects on a test system and import those objects onto a production system. The DDL is output to the media that you specify for subsequent processing. The generated DDL is formatted for direct input into the metadata utility for creating or modifying catalog entries.

To generate DDL, the metadata utility first connects to an appropriate Classic data server specified in a CONNECT TO SERVER statement. Once connected, the catalog is accessible for queries necessary to gather data for the GENERATE DDL statements. The generated DDL is printed to SYSPRINT or recorded in the file associated with the DDLOUT DD in the CACMETAU JCL.

Syntax

```
►►──GENERATE DDL FOR──┬─TABLE─────┬──┬───────────────┬────────────────────►
                      ├─INDEX─────┤  └─┤ Like-Clause ├─┘
                      ├─VIEW──────┤
                      └─PROCEDURE─┘

►─┤ Object-Name ├──┬────────────────┬─────────────────────────────────►◄
                   └─┤ With-Options ├─┘
```

Like-Clause:

```
├──┬─────┬─LIKE──┬─────────────┬──────────────────────────────────────┤
   └─NOT─┘       └─ESCAPE─char─┘
```

Object-Name:

```
├──┬─object-name─────────────┬────────────────────────────────────────┤
   └─schema-name.object-name─┘
```

With-Options:

```
             ┌─,─────────────┐
             ▼               │
├──WITH────────┬─ALL────────┬┴────────────────────────────────────────┤
               ├─INDEX──────┤
               ├─DROP───────┤
               ├─ALTER──────┤
               ├─GRANT──────┤
               └─COMMENT ON─┘
```

Parameters

Like-Clause:

LIKE
LIKE ESCAPE *char***NOT LIKE ESCAPE** *char*

Specify this clause when the object name contains wildcard characters. When the object name contains an underscore name character, an escape character must be defined to preserve the context. Once defined, place the escape character immediately preceding the underscore character to indicate that it is not a wildcard character.

```
GENERATE DDL FOR TABLE LIKE ESCAPE '!' "USER1"."IMS!_TAB%"
  WITH DROP,INDEX;
```

Object-Name:

object-name
schema-name.object-name

Specify the object with or without a schema name. If schema name is not specified, the schema defaults to the user ID assigned to the metadata utility job. When you specify objects after a like clause, you can use wildcard characters in the schema or object name. The supported wildcard characters are the same as those characters that are supported by the Query Processor (underscore and percent sign).

System catalog objects with a schema name of SYSIBM or SYSCAC cannot be generated using the GENERATE DDL statement. When the like clause is used, all system objects that qualify for the like processing are ignored. If you attempt to generate a specific object in the SYSIBM or SYSCAC schema, the following error is returned:

```
CACM004I Non-SQLCODE = 0x0071001E, ERROR: You cannot create or
generate a catalog object with an owner of SYSIBM or SYSCAC.
```

With-Options:

WITH *option*

The with options clause is used to describe additional DDL statement types to be included with the CREATE *object* statement that is generated. You can specify additional statement types individually or you can specify ALL, which generates all additional statement types that are applicable for the CREATE *object* statement. The Table 58 on page 278 table shows the valid additional statement types that can be generated.

A redundant or duplicate specification in the with options clause returns an error during parsing (for example, if other options are specified with ALL). A specification of an invalid option returns an error during parsing (for example, when an index is specified on a view).

Considerations for the DDL output

- By default, the generated DDL is written to SYSPRINT in the standard output format for the metadata utility report.

  If you want to generate the DDL to a file, allocate the file to the DDLOUT DD in the CACMETAU JCL. When the metadata utility is run, the DDL is written to the data set referenced by the DDLOUT DD instead of SYSPRINT. And, SYSPRINT contains the following informational message for each statement:

```
CACM001I SQLCODE = 0,  INFO: The generated DDL was successfully written to
        DDLOUT.
```

- As with other metadata utility statements, you can choose to set up the SYSIN files for CACMETAU as two concatenated LRECL 80 files. The first file contains the CONNECT TO SERVER statement and the second contains the statements to run. For example:

  ```
  //SYSIN  DD  DISP=SHR,DSN=&USRHLQ..SCACCONF(&CONNECT)
  //       DD  DISP=SHR,DSN=&USRHLQ..SCACSAMP(&DDLIN)
  ```

  Note: When concatenating with SYSIN, ensure that the files are consistent in their LRECL definitions.

  The metadata utility can handle a SYSIN of 80 bytes or greater. Cases where you might require more than 80 bytes:
  - If you specify DB2 tables and indexes using DB2 long names. (DB2 long names could require as much as 132 bytes.)
  - If you specify COMMENT ON statements with long text strings.

Example 1

Generate DDL for table names belonging to USER1 and beginning with the characters VSAM_TAB. Include indexes defined on the resultant tables. Generate DROP statements for the resultant tables and indexes. Record the generated DDL in the specified z/OS data set associated with the DDLOUT DD.

Provide the following input to SYSIN, with a DDLOUT DD specified in the CACMETAU JCL:

```
GENERATE DDL FOR TABLE LIKE ESCAPE '!' "USER1"."VSAM!_TAB%"
  WITH DROP,INDEX;
```

SYSTERM contains the following metadata utility output:

```
********************************* TOP OF DATA *************************************
  Start     End Processing
 Line #  Line #     Status Statement                 Object
      1       1         0  CONNECT TO SERVER         CACDAS
CACM001I SQLCODE = 0, INFO: Statement execution was successful.

      2       3         0  GENERATE DDL FOR          LIKE
CACM001I SQLCODE = 0, INFO: The generated DDL was successfully written to DDLOUT.

    N/A     N/A         0  DISCONNECT FROM SERVER    CACDAS
CACM001I SQLCODE = 0, INFO: Statement execution was successful.
******************************* BOTTOM OF DATA ***********************************
```

SYSPRINT contains the following metadata utility output:

```
********************************* TOP OF DATA *************************************
LINE NO.    STATEMENT

      1     CONNECT TO SERVER CACDAS "TCP/9.30.136.90/5002";
CACM001I SQLCODE = 0, INFO: Statement execution was successful.

LINE NO.    STATEMENT

2 GENERATE DDL FOR TABLE LIKE ESCAPE '!' "USER1"."VSAM!_TAB%"
3    WITH DROP,INDEX TO DSN:USER1.GENERATE.OUTPUT;
CACM001I SQLCODE = 0, INFO: The generated DDL was successfully written to DDLOUT
******************************* BOTTOM OF DATA ***********************************
```

Only one table created by USER1 fulfilled the search criteria for a name beginning with the characters VSAM_TAB. The z/OS data set associated with the DDLOUT DD contains the following generated DDL:

```
********************************* TOP OF DATA **************************************
DROP INDEX "USER1"."VSAM_TABLE_IDX1";

DROP TABLE "USER1"."VSAM_TABLE";

CREATE TABLE "USER1"."VSAM_TABLE" DBTYPE VSAM
   DS "USER1.VSAM.VSAMFILE"
(
   "NAME" SOURCE DEFINITION
 DATAMAP OFFSET 0 LENGTH 20
 DATATYPE UC
 USE AS CHAR(20),
   "ADDRESS1" SOURCE DEFINITION
 DATAMAP OFFSET 20 LENGTH 20
 DATATYPE UC
 USE AS CHAR(20),
   "ADDRESS2" SOURCE DEFINITION
 DATAMAP OFFSET 40 LENGTH 20
 DATATYPE UC
 USE AS CHAR(20),
   "CITY" SOURCE DEFINITION
 DATAMAP OFFSET 60 LENGTH 12
 DATATYPE UC
USE AS CHAR(12),
   "STATE" SOURCE DEFINITION
 DATAMAP OFFSET 72 LENGTH 2
 DATATYPE UC
 USE AS CHAR(2),
   "ZIP" SOURCE DEFINITION
 DATAMAP OFFSET 74 LENGTH 5
 DATATYPE UC
 USE AS CHAR(5),
   "PLUS4" SOURCE DEFINITION
 DATAMAP OFFSET 79 LENGTH 4
 DATATYPE UC
 USE AS CHAR(4) );

CREATE INDEX "USER1"."VSAM_TABLE_IDX1" ON "USER1"."VSAM_TABLE" ("STATE"
   ASC, "ZIP" ASC, "NAME" ASC);

******************************** BOTTOM OF DATA ************************************
```

Example 2

Generate DDL for table names belonging to USER1 and beginning with the characters IMS_TAB. Include indexes defined on the resultant tables. Generate DROP statements for the resultant tables and indexes. Record the generated DDL in the specified z/OS data set associated with the DDLOUT DD.

Provide the following input to SYSIN, with a DDLOUT DD specified in the CACMETAU JCL:

```
GENERATE DDL FOR TABLE LIKE ESCAPE '!' "USER1"."IMS!_TAB%"
  WITH DROP,INDEX;
```

SYSOUT contains the following metadata utility output:

```
********************************* TOP OF DATA **************************************
  Start     End Processing
 Line #  Line #     Status Statement                 Object
      1       1         0  CONNECT TO SERVER         CACDAS
CACM001I SQLCODE = 0, INFO: Statement execution was successful.

      2       3         0  GENERATE DDL FOR          LIKE
CACM001I SQLCODE = 0, INFO: The generated DDL was successfully written to DDLOUT.

    N/A     N/A         0  DISCONNECT FROM SERVER    CACDAS
CACM001I SQLCODE = 0, INFO: Statement execution was successful.
******************************** BOTTOM OF DATA ************************************
```

SYSPRINT contains the following metadata utility output:

```
********************************* TOP OF DATA **************************************
LINE NO.    STATEMENT

      1     CONNECT TO SERVER CACDAS "TCP/9.30.136.90/5002";
CACM001I SQLCODE = 0, INFO: Statement execution was successful.

LINE NO.    STATEMENT

2 GENERATE DDL FOR TABLE LIKE ESCAPE '!' "USER1"."IMS!_TAB%"
3    WITH DROP,INDEX TO DSN:USER1.GENERATE.OUTPUT;
CACM001I SQLCODE = 0, INFO: The generated DDL was successfully written to DDLOUT
******************************** BOTTOM OF DATA ************************************
```

Only one table created by USER1 fulfilled the search criteria for a name beginning with the characters IMS_TAB. The z/OS data set associated with the DDLOUT DD contains the following generated DDL:

```
********************************* TOP OF DATA **************************************
DROP INDEX "USER1"."IMS_TABLE_IDX1";

DROP TABLE "USER1"."IMS_TABLE";

CREATE TABLE "USER1"."IMS_TABLE" DBTYPE IMS
   DS "USER1.IMS.IMSFILE"
(
   "NAME" SOURCE DEFINITION
 DATAMAP OFFSET 0 LENGTH 20
 DATATYPE UC
 USE AS CHAR(20),
   "ADDRESS1" SOURCE DEFINITION
 DATAMAP OFFSET 20 LENGTH 20
 DATATYPE UC
 USE AS CHAR(20),
   "ADDRESS2" SOURCE DEFINITION
 DATAMAP OFFSET 40 LENGTH 20
 DATATYPE UC
 USE AS CHAR(20),
   "CITY" SOURCE DEFINITION
 DATAMAP OFFSET 60 LENGTH 12
 DATATYPE UC
USE AS CHAR(12),
   "STATE" SOURCE DEFINITION
 DATAMAP OFFSET 72 LENGTH 2
 DATATYPE UC
 USE AS CHAR(2),
   "ZIP" SOURCE DEFINITION
 DATAMAP OFFSET 74 LENGTH 5
 DATATYPE UC
 USE AS CHAR(5),
   "PLUS4" SOURCE DEFINITION
 DATAMAP OFFSET 79 LENGTH 4
 DATATYPE UC
 USE AS CHAR(4) );

CREATE INDEX "USER1"."IMS_TABLE_IDX1" ON "USER1"."IMS_TABLE" ("STATE"
   ASC, "ZIP" ASC, "NAME" ASC);

******************************** BOTTOM OF DATA ************************************
```

Encrypting passwords for connecting to Classic data servers when the SAF exit is active

If the Classic data server that the metadata utility communicates with has an active SAF exit, the CONNECT TO SERVER statement used by the metadata utility must include both a USERID and PASSWORD clause.

About this task

The password must be encrypted according to the Data Encryption Standard (DES). Use the password encryption utility for Windows or the password encryption utility for z/OS to encrypt the password according to this standard.

- The password generator utility for Windows, cacencr.exe, is a Windows-based command-line utility.
- The password generator utility for z/OS runs as a batch job.

The SAF exit does not use encryption. If your site uses SAF exits you can connect to a server by using the Classic Data Architect. This GUI tool masks your password, but does not use encryption. Supply your z/OS user ID and password.

User-written JDBC and ODBC applications do not use encrypted passwords. If your site uses these applications, you can connect by using a z/OS user ID and password.

Running the password generator utility for Windows

The password generator utility for Windows verifies the password that you supply, encrypts the password, and writes the encrypted password to the password.txt file in the current directory.

About this task

The utility is installed in the ODBC\bin directory where you installed the Classic ODBC client.

Procedure

1. Update the PATH variable to include the following references to the ODBC installation directories: `PATH=ODBC\lib;ODBC\bin;....`
2. Edit the cac.ini file.
   a. In a Windows command window, navigate to the ODBC\lib directory.
   b. Edit the cac.ini file to provide the following required settings based on the location of your Classic ODBC client installation and your locale information:

      ```
      NL CAT = c:\\Program Files\\IBM\\ISClassic113\\ODBC\\lib
      NL = US English
      ```
3. Set the CAC_CONFIG environment variable to point to the full path of the cac.ini file that you updated in the previous step.

   ```
   SET CAC_CONFIG= c:\Program Files\IBM\ISClassic113\ODBC\lib\cac.ini
   ```
4. In a Windows command window, navigate to the ODBC\bin directory.
5. Type **cacencr** and press **Enter**.
6. At the prompt, type the password that you want to encrypt. The password encryption utility creates a file named **password.txt** that contains the password in a 16-byte hexadecimal string format.

   ```
   ENCRYPTED=x'<16-byte hexadecimal value>'
   ```
7. Open the password.txt file in a file editor and copy the hexadecimal string (x'<*16-byte hexadecimal value*>') to the Windows clipboard.
8. In a mainframe session, edit the data set that contains the CONNECT TO SERVER statement used for the metadata utility. Paste the encrypted password after the PASSWORD keyword.

Chapter 20. The configuration migration and maintenance utility

The configuration migration and maintenance utility, CACCFGUT, runs as an IBM z/OS batch job that manages configurations for your Classic data servers. You can use the utility for backup and recovery, monitoring, and maintenance.

Features of the configuration migration and maintenance utility

You can use the configuration migration and maintenance utility to manage the configurations of your Classic data servers. The utility can monitor, back up, and recover your configurations.

The utility supports the following configuration-related Master Terminal Operator (MTO) commands:

- EXPORT,CONFIG
- IMPORT,CONFIG

The utility also provides a REPORT command.

To run the configuration migration and maintenance utility, enter one or more of the configuration commands on the SYSIN DD statement.

Backups

You can use the utility to run the EXPORT command at any time to create backups of configuration files.

Recovery

You can use the EXPORT and IMPORT commands to restore a configuration environment to a previous point in time. By using the EXPORT command, you can create a command file that is based on the configuration environment of a running Classic data server. You can then use this command file to update a different configuration file by using the IMPORT command.

To restore the full configuration, run the utility offline against empty configuration data sets when the Classic data server is not running. To avoid conflicts with default services, run the utility against empty configuration data sets.

Monitoring

You can use the utility to run the REPORT command at any time to monitor the configuration of the Classic data server.

Chapter 21. 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 summarize the log messages or filter them. You can also format and print event messages.

About this task

Perform the steps in the following procedure to view log messages:

Procedure

1. Configure CACPRTLG. See "Parameters for configuring the log print utility (CACPRTLG)."
2. Create filters for the output. See "Filters for modifying output from the log print utility (CACPRTLG)" on page 290.
3. Run CACPRTLG. There are two ways to run this utility:
   - Run CACPRTLG as a step in the same job used to run the Classic data server.
   - Run CACPRTLG as a separate job from the Classic data server job or started task.

Parameters for configuring the log print utility (CACPRTLG)

You supply values to the **PARM** parameter of the CACPRTLG EXEC statement to determine which information CACPRTLG displays and where CACPRTLG extracts the information from.

Specify the **PARM** parameter in the JCL for the Classic data server. See the sample JCL for CACPRTLG in the sample members for the Classic data servers found in *USERHLQ*.SCACSAMP, such as CECCDSRC.

See also the sample JCL for CACPRTLG in the sample member *USERHLQ*.SCACSAMP(CACPRTLS), which shows how to run CACPRTLG against a system logger stream and print a log stream separately.

Recommendation: Use log streams for the diagnostic log or the event log (CACLOG) so that you can print the log while the data server is running. By using log streams, you do not need to set the logger service parameter DISPLAYLOG=TRUE to see logged information and can avoid the processing overhead costs related to formatting and displaying the logs.

The following list shows the possible values for the PARM parameter.

SUMMARY=N|Y

N Displays all of the messages that are in the log if you configured the logger service to write to the CACLOG DD statement or system log streams.

Y Displays a report about the contents of the log if you configured the logger service to write to the CACLOG DD statement or system log streams.

`STREAM=log_stream`
: The *log_stream* value 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.

`PURGE`
: 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.

`PURGEALL`
: Marks for deletion all of the log messages that are in the log stream.

`EVENTS=eventlog_stream_name`
: The name of the event log stream that was specified for the EVENTLOG configuration parameter for the logger service.

`LOCALE=locale`
: The message locale to use when translating the event messages. If you do not specify the LOCALE parameter, the default value EN_US is used to translate event messages using the US English message catalog. Valid values:

EN_US	US English message catalog.
JA_JP	Japanese message catalog.
KO_KR	Korean message catalog.
ZH_CH	Traditional Chinese message catalog.
ZH_TW	Simplified Chinese message catalog.

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 Classic data server, you might find it helpful to review the JES output for the Classic 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 Classic 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.

Chapter 22. Troubleshooting and contacting IBM Support

The following support page contains the latest troubleshooting information and details on how to open a service request with IBM Support:

- http://www.ibm.com/software/data/infosphere/support/change-data-capture/

For contact information in your region:

- http://www.ibm.com/planetwide/

Notices

Trademarks

Printed in USA