SQL/MX Connectivity Service Administrative Command Reference

Abstract

This manual describes the ODBC administrative commands running in the HP NonStop™ SQL/MX Connectivity Service (MXCS) environment and accessed through the SQL/MX conversational interface (MXCI). These commands support manipulation of the logical objects that define the user-visible attributes of the SQL/MX connectivity service (MXCS).

Product Version

MXCS ACL 1.1 APB

Supported Release Version Updates (RVUs)

This manual supports J06.03 and all subsequent J-series RVUs, H06.03 and all subsequent H-series RVUs, and G06.23 ABA and all subsequent G-series RVUs until otherwise indicated by its replacement publications.

Part Number	Published
526350-005	July 2013

Document History

Part Number	Product Version	Published
526350-001	MXCS ACL 1.0	April 2004
526350-002	MXCS ACL 1.1 ABA	August 2004
526350-003	MXCS ACL 1.1 ABO	March 2005
526350-004	MXCS ACL 1.1 APB	February 2013
526350-005	MXCS ACL 1.1 APB	July 2013

Legal Notices

Printed in the US

SQL/MX Connectivity Service Administrative Command Reference

Index Figures

Figures

What's New in This Manual

Manual Information

Abstract

This manual describes the ODBC administrative commands running in the HP NonStop™ SQL/MX Connectivity Service (MXCS) environment and accessed through the SQL/MX conversational interface (MXCI). These commands support manipulation of the logical objects that define the user-visible attributes of the SQL/MX connectivity service (MXCS).

Product Version

MXCS ACL 1.1 APB

Supported Release Version Updates (RVUs)

This manual supports J06.03 and all subsequent J-series RVUs, H06.03 and all subsequent H-series RVUs, and G06.23 ABA and all subsequent G-series RVUs until otherwise indicated by its replacement publications.

Part Number	Published
526350-005	July 2013

Document History

Part Number	Product Version	Published
526350-001	MXCS ACL 1.0	April 2004
526350-002	MXCS ACL 1.1 ABA	August 2004
526350-003	MXCS ACL 1.1 ABO	March 2005
526350-004	MXCS ACL 1.1 APB	February 2013
526350-005	MXCS ACL 1.1 APB	July 2013

New and Changed Information

- Updated the `initserver count` in the ADD DS and ALTER DS section.

About This Manual

This manual describes the administrative commands available to administer ODBC in an SQL/MX connectivity service (MXCS) running on a single HP NonStop system. These commands are implemented internally as the SQL/MX administrative command library (MACL), are linked into the SQL/MX command interface (MXCI), and support manipulation of logical objects defining the user-visible attributes of the ODBC facility of SQL/MX Connectivity Service (MXCS).

Audience

This manual is intended for database administrators and programmers who:

- Manage the ODBC facility of MXCS for connectivity to the NonStop SQL/MX.
- Use the MXCS administrative commands on the MXCI command-line rather than through the Web-based NSM/web product.
- Use batch files to change or add to the MXCS configuration.

Commands in this manual run at the MXCI prompt.

Related Documentation

This manual is part of the SQL/MX library of manuals, which includes:

Introductory Guides

SQL/MX Comparison Guide for SQL/MP Users	Describes SQL differences between SQL/MP and SQL/MX.
SQL/MX Quick Start	Describes basic techniques for using SQL in the SQL/MX conversational interface (MXCI). Includes information about installing the sample database.

Reference Manuals

SQL/MX Reference Manual	Describes the syntax of SQL/MX statements, MXCI commands, functions, and other SQL/MX language elements.
DataLoader/MX Reference Manual	Describes the features and functions of the DataLoader/MX product, a tool to load SQL/MX databases.
SQL/MX Messages Manual	Describes SQL/MX messages.
SQL/MX Glossary	Defines SQL/MX terminology.

Programming Manuals

SQL/MX Programming Manual for C and COBOL	Describes how to embed SQL/MX statements in ANSI C and COBOL programs.
SQL/MX Programming Manual for Java	Describes how to embed SQL/MX statements in Java programs according to the SQLJ standard.

Specialized Guides

SQL/MX Installation and Management Guide	Describes how to plan for, install, create, and manage an SQL/MX database. Explains how to use installation and management commands and utilities.
SQL/MX Query Guide	Describes how to understand query execution plans and write optimal queries for an SQL/MX database.
SQL/MX Data Mining Guide	Describes the SQL/MX data structures and operations to carry out the knowledge-discovery process.
SQL/MX Queuing and Publish/Subscribe Services	Describes how SQL/MX integrates transactional queuing and publish/subscribe services into its database infrastructure.
SQL/MX Report Writer Guide	Describes how to produce formatted reports using data from a NonStop SQL/MX database.
SQL/MX Connectivity Service Manual	Describes how to install and manage the SQL/MX Connectivity Service (MXCS), which enables applications developed for the Microsoft Open Database Connectivity (ODBC) application programming interface (API) and other connectivity APIs to use SQL/MX.
SQL/MX Guide to Stored Procedures in Java	Describes how to use stored procedures that are written in Java within SQL/MX.

Online Help

The SQL/MX Online Help consists of:

Reference Help	Overview and reference entries from the *SQL/MX Reference Manual*.
Messages Help	Individual messages grouped by source from the *SQL/MX Messages Manual.*
Glossary Help	Terms and definitions from the *SQL/MX Glossary*.
NSM/web Help	Context-sensitive help topics that describe how to use the NSM/web management tool.
Visual Query Planner Help	Context-sensitive help topics that describe how to use the Visual Query Planner graphical user interface.

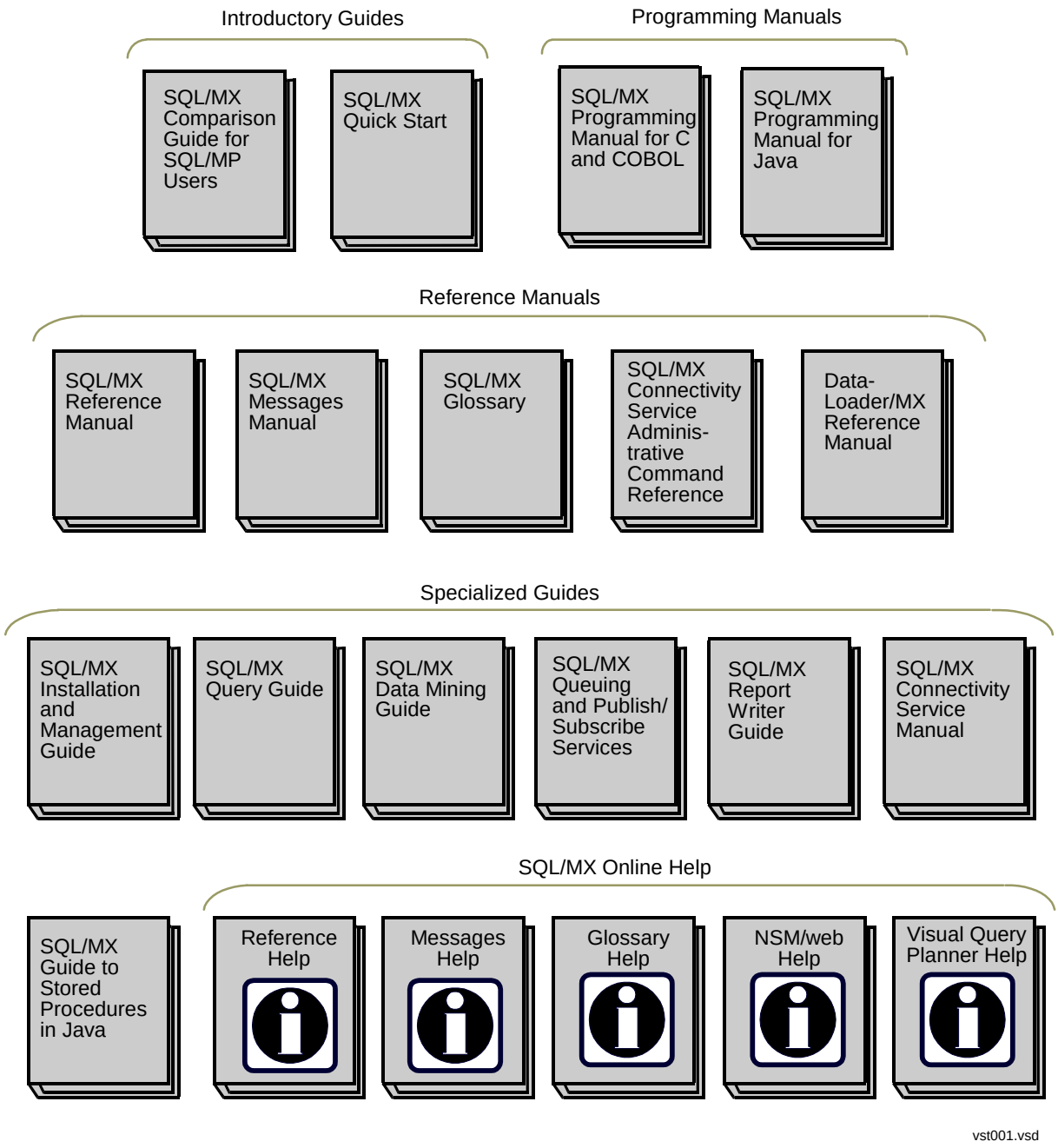
The NSM/web and Visual Query Planner help systems are accessible from their respective applications. You can download the Reference, Messages, and Glossary online help from the $SYSTEM.ZMXHELP subvolume or from the HP NonStop Technical Library (NTL). For more information about downloading online help, see the *SQL/MX Installation and Management Guide*.

These manuals are part of the SQL/MP library of manuals and are essential references for information about SQL/MP Data Definition Language (DDL) and SQL/MP installation and management:

Related SQL/MP Manuals

SQL/MP Reference Manual	Describes the SQL/MP language elements, expressions, predicates, functions, and statements.
SQL/MP Installation and Management Guide	Describes how to plan, install, create, manage an SQL/MP database. Describes installation and management commands and SQL/MP catalogs and files.

This figure shows the manuals in the SQL/MX library:



Notation Conventions

Hypertext Links

Blue underline is used to indicate a hypertext link within text. By clicking a passage of text with a blue underline, you are taken to the location described. For example:

This requirement is described under Backup DAM Volumes and Physical Disk Drives on page 3-2.

General Syntax Notation

This list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS. Uppercase letters indicate NSK keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

```
MAXATTACH
```

lowercase italic letters. Lowercase italic letters indicate variable items that you supply. Items not enclosed in brackets are required. For example:

```
file-name
```

[ ] Brackets. Brackets enclose optional syntax items. For example:

```
TERM [\system-name.]$terminal-name
```

```
INT[ERRUPTS]
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
FC [ num  ]
   [ -num ]
   [ text ]
```

```
K [ X | D ] address
```

{ } Braces. A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                  { $process-name  }
```

```
ALLOWSU { ON | OFF }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

… Ellipsis. An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address [ , new-value ]...

[ - ] {0|1|2|3|4|5|6|7|8|9}...
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char..."
```

Punctuation. Parentheses, commas, semicolons, and other symbols not previously described must be typed as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;

LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must type as shown. For example:

```
"[" repetition-constant-list "]"
```

Item Spacing. Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In this example, no spaces are permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing. If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] LINE

   [ , attribute-spec ]...
```

Notation for Messages

This list summarizes the notation conventions for the presentation of displayed messages in this manual.

Bold Text. Bold text in an example indicates user input typed at the terminal. For example:

```
ENTER RUN CODE

?123

CODE RECEIVED:      123.00
```

The user must press the Return key after typing the input.

Nonitalic text. Nonitalic letters, numbers, and punctuation indicate text that is displayed or returned exactly as shown. For example:

```
Backup Up.
```

lowercase italic letters. Lowercase italic letters indicate variable items whose values are displayed or returned. For example:

```
p-register
```

```
process-name
```

[ ] Brackets. Brackets enclose items that are sometimes, but not always, displayed. For example:

```
Event number = number [ Subject = first-subject-value ]
```

A group of items enclosed in brackets is a list of all possible items that can be displayed, of which one or none might actually be displayed. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
proc-name trapped [ in SQL | in SQL file system ]
```

{ } Braces. A group of items enclosed in braces is a list of all possible items that can be displayed, of which one is actually displayed. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
obj-type obj-name state changed to state, caused by
{ Object | Operator | Service }
```

```
process-name State changed from old-objstate to objstate
{ Operator Request. }
{ Unknown.          }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
Transfer status: { OK | Failed }
```

% Percent Sign. A percent sign precedes a number that is not in decimal notation. The% notation precedes an octal number. The%B notation precedes a binary number. The%H notation precedes a hexadecimal number. For example:

```
%005400
```

```
%B101111
```

```
%H2F
```

```
P=%p-register E=%e-register
```

Change Bar Notation

Change bars are used to indicate substantive differences between this manual and its preceding version. Change bars are vertical rules placed in the right margin of changed portions of text, figures, tables, examples, and so on. Change bars highlight new or revised information. For example:

> The message types specified in the REPORT clause are different in the COBOL85 environment and the Common Run-Time Environment (CRE).
>
> The CRE has many new message types and some new message type codes for old message types. In the CRE, the message type SYSTEM includes all messages except LOGICAL-CLOSE and LOGICAL-OPEN.

1 MXCS Architecture Overview

The NonStop SQL/MX connectivity service (MXCS) includes support for remote connections to the SQL/MX database, such as ODBC and JDBC. An associated MXCS configuration database allows creation of site-specific information to customize the ODBC facility to a specific use or application. This guide describes the commands for administrating the ODBC facility and the MXCS configuration database.

The MXCS administrative commands provide a command-line interface that can be scripted to administer the SQL/MX ODBC features. The commands are packaged in MXCI with SQL/MX commands and report writer commands. MXCI must be running to access these commands. An operating mode within MXCI directs commands to the proper command handlers.

Note. MXCS administrative commands do not administer the JDBC/MX product or the SQL/MX database itself.

If MXCS is installed on the system, the administrative commands are operational. Because MXCS often requires setup before it accesses applications, use these commands to configure MXCS initially (before it is started) and tune the configuration while it is operational. Some of the commands are persistent through subsystem shutdown, such as ADD DS, while others affect only currently running services or servers, such as START SERVICE or STOP SERVER.

MXCS administrative commands use command-line format. You enter MXCS administrative commands from a keyboard or from a command file after starting MXCI.

NSM/web, a Web-based GUI, administers the same objects that are administered through the command-line and MXCS administrative commands. However, NSM/web does not support command files, and it must first be installed on the system.

This section describes:

Features and Functionality

- Administers ODBC MXCS in NonStop SQL/MX Release 2.0 ABA and later releases
- Provides a scriptable command-line interface through MXCI on an HP NonStop operating system (OS)
- Reports if a command fails to supply requested information

- Configures a new installation of MXCS or modifies an existing installation
- Accesses objects only on the system where MXCI is running
- Provides database security so that all users can view objects, but only specified users can modify them
- Displays attributes and states of the objects within the ODBC MXCS configuration
- Displays and manages the ODBC MXCS configuration as five separate object types:
  - Service (the MXOAS command starts each service)
  - Server
  - Data source (DS)
  - Environment variable (EVAR) for each DS user (for permissions)
  - User (for permissions)
- Supports administrative commands:
  - INFO
  - ADD
  - DELETE
  - START
  - STOP
  - ALTER
  - VERSION
- Reports errors if the ODBC facility of MXCS is not running when needed

Architecture Overview

The MXCS administrative command library (MACL) is the software component inside MXCI that supports the MXCS administrative commands described in this guide. MACL is discussed only to show how the MXCS administrative commands are processed and to assist you in understanding error messages that might appear.

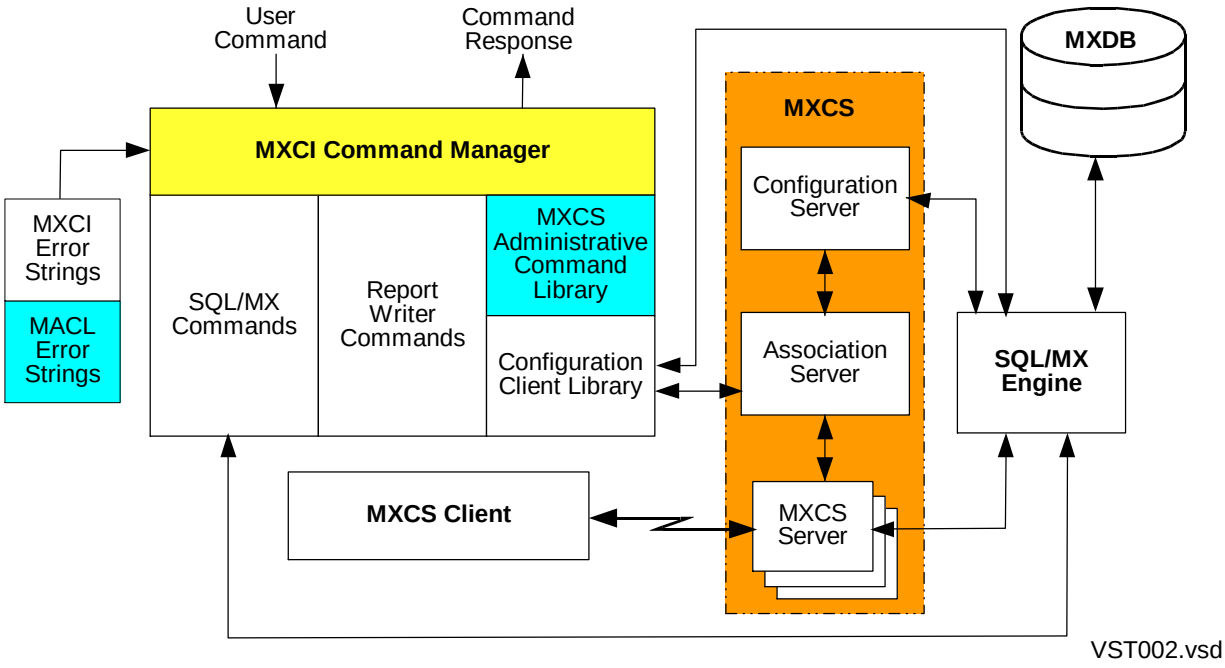
Figure 1-1 on page 1-3 shows how the MACL library fits in the system architecture with MXCS, MXCI, and the SQL/MX database. MACL links with MXCI and uses many of the existing generic features of the MXCI command-line interface. MXCI manages command entry and passes commands to MACL for processing. MXCI also displays report lines and error messages when a command is processed.

MXCI also supports execution of SQL statements through the SQL engine directly or through the report writer.

When you start MXCI, some (but not all) administrative commands are available when ODBC is not running on the system. Some of these commands require that the ODBC association server be operational. However, you can always configure a DS, EVAR, and user. MXCI is not required to be running while MXCS runs, so you can run MXCI for only as long as you need to maintain a session. Before entering any MXCS

administrative command, set MXCI to the MXCS mode as described in MXCI Mode Command on page 2-12.

Figure 1-1. MXCS Architecture



An operational MXCS association server is not required for data source and environmental variable configuration commands, but must exit (stopped state is OK) when you want to query the current status of the service, data source, or servers running in that service.

In Figure 1-1, assume that the MXCS facility is started. The normal flow for MXCS administrative command processing is:

1. MXCI starts and accepts a command from a terminal or disk file.
2. The MXCI command manager has been set to the MXCS mode so it passes the command to the MACL routines.
3. MACL parses the command and requests the client library perform the action.
4. The client library transmits requests to the appropriate MXCS component and returns responses.
5. MACL formats the responses into reports.
6. MACL returns status to the MXCI command manager.
7. The MXCI command manager displays all result and error lines and then waits for further input.

User Interface With MXCI

The MXCI interface with MXCS administrative commands is similar to its interface for the report writer or SQL command handlers. The basic features are:

- MXCS commands terminate with a semicolon (;), with command input occurring only after the semicolon is entered. Enclose embedded semicolons in single or double quotation marks.
- If the MXCS mode is set, the prompt changes from >> to CS> for terminal users, but not for log file or non-tty input streams. After you set the MXCS mode, all commands go to MACL until you either reset the mode or enter `exit`. In MXCS mode, basic SQL commands generate syntax errors.
- MXCI manages batch input files and output log files as usual, but now these files can contain MXCS commands (preceded by the mode command).
- MXCI passes a break event to MACL for handling. If a command is in progress, MACL stops generating report and error messages.
- No support exists for critical section or rollback events for MACL commands. These commands immediately complete. When a command initiates change in the MXCS environment, a break event arrival could interrupt the process. To determine if a break event is successful, check state information for the object in the current command.
- If an outstanding transaction exists, attempts to switch to the MXCS mode in MXCI fail with an error message. For example, executing queries with `autocommit` set to off (default is on) or having an open BEGIN WORK transaction causes failures.
- MACL supports no variable expansion in its input commands.

General Features

- Asterisk (*) is the only wild card supported in command names, but only for specific commands. No patterns are supported. Wild card `*xyz` or `a?b` causes an error.
- The asterisk (*) wild card applies only to objects on the specific system supporting the MXCI process. See Command Rules and General Syntax on page 2-3.
- Object name defaulting is supported to reduce typing. The most recent object named of each type persists until it is changed.
- Commonly used commands for the EVAR object have defaults to reduce typing. See Default Object Name on page 2-7.
- Each DS, EVAR, and user is checked for uniqueness. A duplicate causes failure of the ADD command. ALTER or DELETE commands act on the first match they detect and generate a warning about the successive duplicates.
- The INFO command displays duplicate objects, but does not issue a warning.

- Error messages describe the problem and, where possible, suggest a recovery action.

Software Installation

MXCS administrative commands are installed as part of MXCI. No special installation process is needed to install software for these commands.

Starting, Leaving, and Ending MXCI

The MXCI command has many options. One of them enables you to specify the name of a command file. For start-up information, see MXCI Mode Command on page 2-12. For examples, see Example Commands and Objects on page A-1

Complex System Configuration

This subsection describes a complex MXCS configuration on one system, showing the objects that MXCS administrative commands manage: service, server, DS, and EVAR. In Figure 1-2 on page 1-6, a system is shown with three of its CPUs represented and MXCS objects distributed across these CPUs. The service object is a logical entity consisting of two processes, one or more DS instances, a table of all servers it has started and owns, and memory-resident copies of EVARS owned by each DS instance.

Data sources exist in three places in three forms:

- A copy on the disk that has no state except existence or not
- A copy in the service, the DS instance, that has a state (stopped is default state)
- A partial copy kept in each server

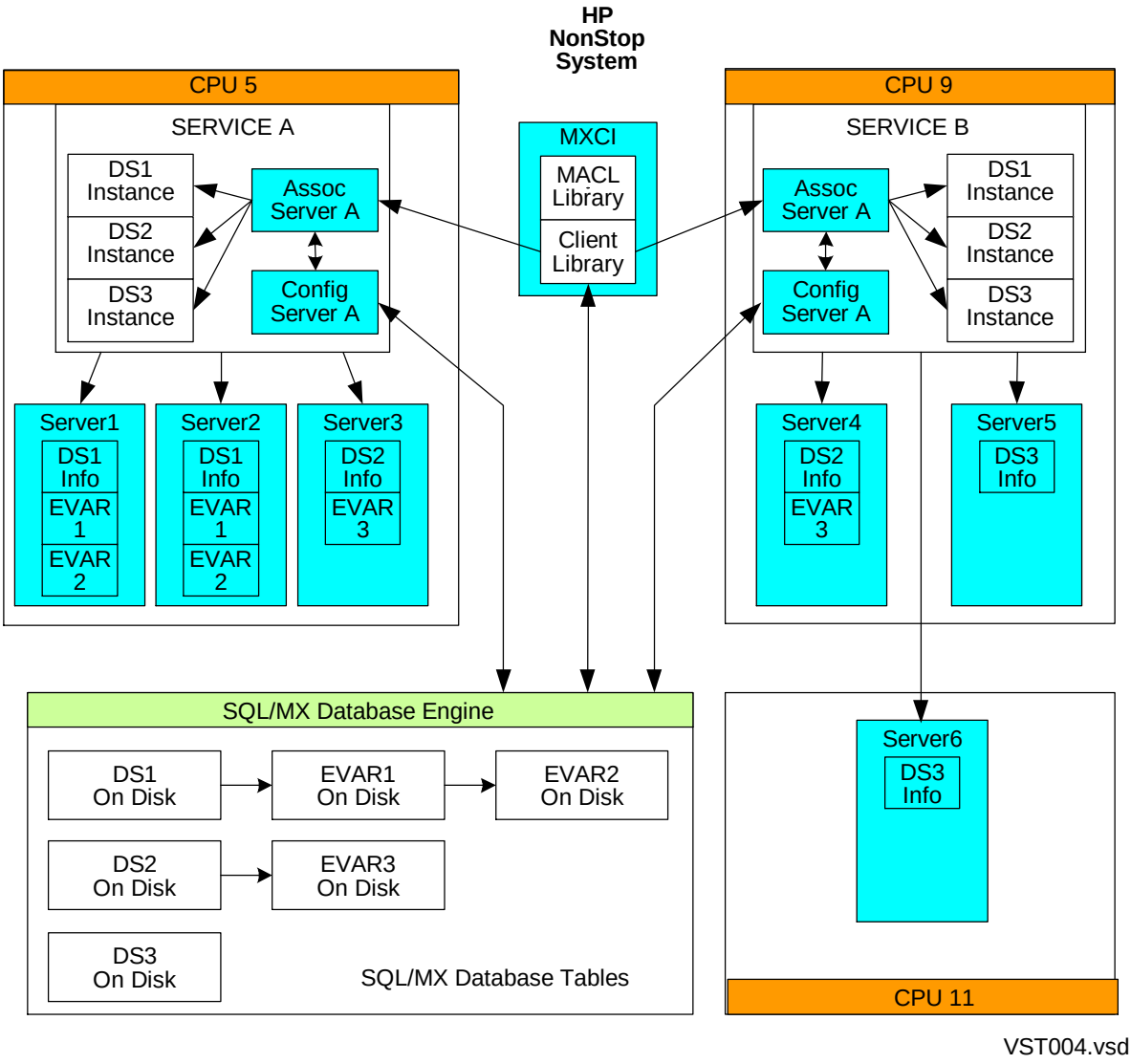
The configuration server in each service reads all DS instances at service start-up and keeps them current. However, states for DS instances can be different in different services. The association server starts each server and passes its DS attributes at server start-up and also when certain attributes change. However, when an EVAR for a DS instance is added, deleted, or changes value, it is not passed to connected servers because this action changes the environment during transactions. EVARs pass from a DS instance to a server only when that server starts or goes idle. Some DS attributes are never passed to the servers. For example, only the association server uses the maximum number of servers permitted.

In this configuration, two services are running: Service A on CPU 5 and Service B on CPU 9. The SQL/MX database engine contains the configuration database for the permanent ODBC configuration information (DS and EVAR definitions). Note that CPU 11 has no MXCS service running on it, but it does have a server object. Also, one instance of MXCI is running on one of the CPUs, which is not specified as it does not matter for this discussion.

On CPU 5, Service A is a logical entity consisting of an association server and a configuration server. All three DS definitions are read from the disk and kept current in

Service A. Service A monitors the three MXCS servers it has started in addition to the DS and EVARs they use. Each server uses only a few attributes of one DS (like time-out, trace, and statistics flags) and has all EVARs that belong to that DS defined in its environment.

Figure 1-2. Complex MXCS Configuration



The servers owned or started by Service A all run on CPU 5. Service B has two servers on CPU 9 and one server running on CPU 11. Using the `CpuList` parameter, each DS defines where the service can start servers using that DS. It is possible that DS 1 has only CPU 5 in its list, while DS 2 has both CPU 5 and 9 in its list, and DS 3 has CPU 9 and 11. To accurately determine which CPUs are permitted, display the detailed information about each DS by using one of the MXCS administrative commands.

All MXCS objects can be viewed or manipulated with MACL commands supported by MXCI. Through MACL and the configuration client library, MXCI establishes and terminates connections to the process that makes up the logical service and, in some cases, connections directly to the MXCS configuration database.

Although Figure 1-2 shows one system, an Expand network can connect to other systems. Each system must have its own copy of the configuration database, MXCI and its own MXCS services.

Server objects owned by one service can run on any CPU specified by the DS definition. Service objects can also run on any specified CPU when you manually start services from a TACL prompt by using the MXOAS command. No MXCS administrative command is available to launch a service.

The START SERVICE command requires that the MXOAS command has already started the processes and, as such, only moves the service from stopped to started and allows users to start connecting. Also, the STOP SERVICE command does not kill the service process. It merely prevents establishment of new connections and terminates existing connections and the servers supporting them.

Error Message Overview

MXCS administrative command error messages for any single event consist of one or more lines of error messages (limit of ten), displayed in this general format:

Note. A message with content as shown in the first line of this example is always displayed. Successive lines containing additional details are frequently displayed.

```
CS> command
*** Error[15nnn] Primary error message numbered 15200 to 15499 stating reason for
                 failure of the MXCS command.
*** Error[15nnn] (KEYWORD): Detailed error info from client library or usage
                 information. Keyword is "SUBSYSTEM" or "USAGE".
*** Error[15nnn] LOWLAYER 1: Report of problems from layers below client library.
*** Error[15nnn] LOWLAYER n: Additional report from layers below client library,
                 where n is 2, 3, 4, and so forth, indicating the sequential number
                    of the message up to 8.
```

To easily locate all errors for a particular event, you must be able to recognize all error report formats. Some error reports contain several error events, so it can be useful to group errors by event. Note that the initial message for an event contains no keyword after the error number.

A single error message for one error event looks like:

```
CS>info ds abc, option;
-- INFO DS \ODBC.$ASM.ABC Failed

*** ERROR[15242] Command has an invalid attribute "OPTION".   +++SUGG: Check spelling
    or use of extra commas.
```

A usage error message for one error event looks like:

```
CS>info xx abc;

*** ERROR[15453] Unknown object type, XX.

*** WARNING[15288] USAGE: "INFO" command supports object types: DS, EVAR, SERVER,
    SERVICE, USER.
```

A multiple error message for one error event looks like:

```
CS>start ds catalog_ds;
-- START DS \ODBC.$ASM.CATALOG_DS Failed

*** ERROR[15380] DS not found, \ODBC.$ASM.CATALOG_DS.

*** ERROR[15309] SUBSYSTEM: DS not found, library call CFGStartDS failed.

*** ERROR[15202] LOWLAYER 1: Data source does not exist.
```

These examples show actual error reports. See the *SQL/MX Messages Manual* for explanations of each message number, such as 15380, 15309, 15202, and so forth:

In some cases, error messages from lower layers appear redundant, as the preceding example shows. Note that the first line starts with a pair of hyphens (`--`) and is a result summary, not an error message. This first line of the message gives the basic action, the full object name, and summary status as success or failure.

Most commands using a wild card can produce multiple error events from one command. For example, if you request information on all DS objects and the system is busy, it is possible the command will fail for several, or even all, the DS objects it iterates over. Such iterative commands often stop after the fourth error event. A command with a wild card can quickly exceed the ten-message limit for error reports, with the tenth message stating that some messages were dropped.

Each MXCS administrative command error message starts with three asterisks (***), followed by `ERROR` or `WARNING` and the error message number. Warnings are informational and do not stop command processing. Errors usually stop command processing on the object where the error occurred. However, some commands can continue processing additional objects if the asterisk (*) wild card is specified.

When more than one error is reported for an event, the first message is the primary message describing the error and possible recovery. Subsequent messages result from internal calls to subsystems and often provide more details about the problem. Although primary error messages have no leading keyword, subsequent messages can contain one of these keywords:

- `USAGE:` Warning messages resulting from command entry syntax error
- `SUBSYSTEM:` Messages translated from client library codes
- `LOWLAYER 1:` Messages from layers below the client library
- `LOWLAYER 2:` Additional messages from layers below the client library

In the *SQL/MX Messages Manual*, notation for an MXCS administrative command error message often contains substitution variables, similar to this example:

```
15206  SUBSYSTEM: Bad return code int1, library call
functions-name failed.
```

Substitution variables appearing as *`int1`* are integers, and those appearing as *`string1`* are strings. Some cases use a more meaningful variable, such as *`functions-name`*, but substitution variables are always shown in italic.

Some error messages also suggest recovery action, as shown in this example of error 15215:

```
*** ERROR[15215] SUBSYSTEM: SERVICE name used cannot supply this info,
library call CFGGetStatusService failed.   +++SUGG: SERVICE name was probably
* and should be a specific SERVICE name.
```

When a recovery suggestion is given, it follows the error description and is introduced by the string:

```
+++SUGG:
```

Suggested recovery action is a “best effort” analysis of the problem indicated. Under certain circumstances, recovery might not always succeed.

2 MXCS Commands and Objects

This section describes:

MXCS Administrative Command Names

Commands that support administration of SQL/MX connectivity service (MXCS) are:

ADD	Creates and inserts one object into the configuration.
DELETE	Removes one or more objects from the configuration.
ALTER	Modifies one existing object in the configuration.
INFO	Shows attributes and state of one or more objects in the configuration.
START	Makes one or more objects available for use.
STOP	Removes one or more objects from service. Server processes are also deleted, but services and DSs continue to exist.
VERSION	Displays the product and release IDs, date, and build number.

These commands function in pairs with object types, but not all combinations of command and object pairs are supported. See Valid Command and Object Type Pairs on page 2-3.

MXCS Object Types

MXCS objects that support the command-line interface for SQL/MX connectivity service are:

DS (data source)	A named collection of connection attributes for the servers. A DS has a single definition on a particular system and can be seen by all services on the system. (A particular DS can be different between different systems.) The DS instance must be in the started state on any service that is using it (allowing connections). On any other service, a specific DS could be in the stopped state.
EVAR (environment variable)	A name and value pair defined in the server environment after the client connects to the MXCS server. Several different EVAR types exit, each with slightly different syntax. The definition or setting of an EVAR is per each DS. System-wide settings are not possible.
Server	A process that supports client connections to the database. Many MXCS server processes can run on different CPUs within the system, each owned by the service the started it. Each server also uses one DS. However, association and configuration servers are not in the MXCS server object class.
Service	A logical object started by the MXOAS command that always contains processes for the association server and usually those for the configuration server also. The service object monitors all DSs defined on the system and all user connection requests on the first of several HP NonStop TCP/IP ports assigned to it by the MXOAS command.
User	A logical object that defines access permissions for an administrator of the MXCS configuration database.

Object Ownership Hierarchy

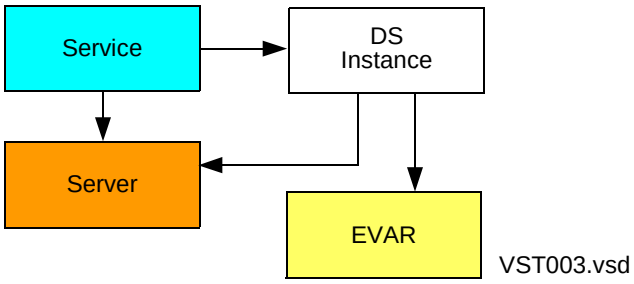
A DS is a named collection of attributes that resides on a disk. A DS instance is a logical entity in a service that has all attributes of the DS, plus a state. A DS instance also owns all the servers started by the service that uses it (Figure 2-1). A server uses one DS and its EVARS and is owned by one service. One DS owns each defined environment variable. Each EVAR is an attribute of one DS. Each service has a DS instance, and several services can use a DS at the same time.

Object Ownership Complexity

Complexity appears when the system has several services and DSs, as shown in Figure 2-1. The DS resides on a disk as a set of values and attributes, with the memory-resident DS instance built in each service from the DS definition on that disk. When the DS is added, altered, or deleted, all services are notified of the changes, and each DS instance is updated. EVAR changes are also propagated to all services, but only to idle servers. Servers with existing connections are not modified.

Two DSs can have an EVAR of the same name, but each EVAR is unique with its own values.

Figure 2-1. MXCS Object Relations



Valid Command and Object Type Pairs

All MXCS administrative commands do not support all objects types. The supported commands and object types are listed, with X indicating valid pairs, such as ADD DS:

			Object Type		
Command	**DS**	**EVAR**	**Server**	**Service**	**User**
ADD	X	X	-	-	X
ALTER	X	X	-	-	X
DELETE	X	X	-	-	X
INFO	X	X	X	X	X
START	X	-	-	X	-
STOP	X	-	X	X	-
VERSION	-	-	-	X	-

Command Rules and General Syntax

Command Termination

Each MXCS administrative command ends with a semicolon (;). Several commands can be on one input line, or one command can span several input lines. Use tabs, spaces, and new lines to separate parameters in a command.

Fields or Tokens in Commands

All fields or tokens in a command belong to one of these types:

Keyword	Single word without quotation marks, uppercase or lowercase (not case-sensitive), alphanumeric characters (*A* through *Z* or *a* through *z*, plus 0 through 9). Called key value when used as a value of an attribute.
Number	Single word, numeric characters (0 through 9) with sign (+ or -); no spaces and no commas.
System name	Single word beginning with a backslash (\), uppercase or lowercase (not case-sensitive), alphanumeric characters (*A* through *Z* or *a* through *z*, plus 0 through 9).
Process name	Single word beginning with dollar sign ($), uppercase or lowercase, alphanumeric characters (*A* through *Z* or *a* through *z*, plus 0 through 9).
SQL identifier	Single word, uppercase or lowercase, unless within optional double quotes (“), and used for object types DS and EVAR components. With double quotes, the character set expands to include spaces, tabs, punctuation, and lowercase alphabetic letters. See the Identifiers topic in the *SQL/MX Reference Manual* for an explanation of SQL identifiers. Unlike with a true SQL identifier, MXCS administrative commands accept regular identifiers containing special characters in unquoted strings, except token separators. Errors for leading or trailing space characters in a name are reported.
SQL string literal	Requires single quotes (‘), is single or multiword, is case-sensitive unless noted otherwise in the specific command, permits all printable characters, and is used only for attribute values. See the Literals topic in the *SQL/MX Reference Manual* for an explanation of SQL string literals. Unlike a true SQL string literal, MXCS administrative commands do not accept the optional leading character set identifier and the empty string literal. But these fields accept leading and trailing space in the string.

Generally, only these SQL types can be enclosed in quotes or split across multiple lines. Splitting requires quotes and the CONCAT operator.

General MXCS Administrative Command Syntax

```
COMMAND obj-type [obj-name][, attribute]...;
```

COMMAND

is listed in Valid Command and Object Type Pairs on page 2-3.

obj-type

is one of the objects listed in Valid Command and Object Type Pairs on page 2-3.

obj-name

is a string that defines the object to be acted upon (with its syntax described in Object Name Components on page 2-5). Object name (*obj-name*) is optional if the previous command acted on the same object. It can also be a subcomponent of the previously addressed object because the library retains the last object name and uses this name to complete the current object name, if needed. An error occurs if an object name is not provided and there is no default. A comma or semicolon delimits an object name.

attribute

is a keyword-value pair in the form of *keyword [value]* that further specifies the desired action. Use either a keyword alone or a keyword followed by a value with a comma or semicolon delimiting an attribute. Enclose in single or double quotes values containing a space, quote, comma, or semicolon. Keyword can be lowercase, uppercase, or mixed uppercase and lowercase. The value can be an integer, key value, SQL string literal (requires single quotes), or an SQL identifier (permits double quotes to protect lowercase or special characters). To specify several attributes, separate each one with a comma. Duplicating an attribute causes an error.

Command Syntax Case

Command syntax is case-sensitive only when enclosed in single or double quotes.

Object Name Components

An object name has one to four components, separated by periods. No space or carriage returns are permitted around period separators. The components of an object name depend on which object type is being addressed. A component can also be defaulted to the last specified value on an earlier command. Some commands permit the asterisk (*) wild card to be used as a component name.

General syntax is:

For service:
```
[[\system-name.]$service-name]
```
For DS:
```
[[[\system-name.]$service-name.]ds-name]
```
For EVAR:
```
[[[[\system-name.]$service-name.]ds-name.]evar-name]
```
For server:
```
[[[\system-name.]$service-name.]$server-name]
```
For user:
```
[[\system-name.]groupname.membername]
```

See individual commands for specific syntax formats.

\system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. The default is the system where MXCI is executing. This component is a system name data type (see Fields or Tokens in Commands on page 2-4).

$service-name

is the name of the service, the target MXCS association server process name. Using an asterisk (*) wild card is sometimes permitted and directs action on all services. The asterisk (*) service name often indicates contact with the MXCS configuration database directly, rather than with a specific service, in which case, the actions still affect all services, current or future. Use the asterisk (*) if no service is operational yet. In this case, use commands only on DS and EVAR objects. The initial default is one of the existing service or an asterisk (*) if no service is running. This component is a process name data type (see Fields or Tokens in Commands on page 2-4).

ds-name

is the target DS. The initial default is TDM_Default_DataSource. This component is an SQL identifier data type (see Fields or Tokens in Commands on page 2-4).

evar-name

is the target environment variable with no initial default. This component is an SQL identifier data type (see Fields or Tokens in Commands on page 2-4).

$server-name

is the name of an MXCS server with no initial default. This component is a process name data type (see Fields or Tokens in Commands on page 2-4).

groupname

is the group login name and must be given if *membername* is used. This component is an SQL identifier data type but is always set to uppercase after the command processes it (see Fields or Tokens in Commands on page 2-4).

membername

is the NonStop OS user login with no aliases permitted. This component is an SQL identifier data type but is always set to uppercase after the command processes it (see Fields or Tokens in Commands on page 2-4).

If you use the asterisk (*) or question mark (?) in a name, enclose it in single (‘) or double (“) quotation marks, or it is flagged as an invalid use of a pattern. The only valid pattern is an asterisk (*) alone, with no quotes, which generally indicates *all.*

DS Name and EVAR Name

The DS and EVAR name are not case-sensitive, unless they are enclosed in double quotes. To eliminate the need for double quotes, avoid spaces, punctuation, and lowercase characters. When necessary, include a double quote in such a name by doubling it. A backslash (\) is accepted as a character and not considered an escape character.

An example of a DS name with embedded special characters is:

```
$AS."ds',; ?)n""ame"      specifies DS:   ds',; ?)n"ame
```

Default Object Name

Use the default when object name components are long and difficult to type. To ensure all cases are well-defined, follow these guidelines:

- After you use an object name, it is retained until you enter another name component of the same type.
- When you use an object name, the `obj-type` field on the command defines the object type of the **last** specified component of the name. For example, the commands `info service $z0123;` and `info server $z0123;` are well-defined. (The first example will likely fail because `$z0123` is probably not an association server process ID.)
- When the final component of the name is defaulted while other components are specified, use trailing dot (period) separators to show missing components. This guideline does not apply to missing leading components.

  For example, this string addresses the named service on the same system as before (no leading period is needed):

  ```
  INFO SERVICE $AR
  ```

But the period is required in this command because the DS name is defaulted:

```
INFO DS $AR2.
```

This last example uses the DS named in the previous command but specifies a different service (it assumes a previous command acted on a DS). The example shows how to get state information on one specific DS from each service.

- When an intermediate component is specified by a default, you must specify its terminating period. For example, this command specifies the EVAR on $as2 owned by the same DS named in the previous example:

  ```
  INFO EVAR $as2..Evar2;
  ```

Command Size Limit

Commands have no size limit, so you can continue long statements or commands over multiple lines, breaking them at any space character. A semicolon (;) terminates a statement or command you break over several lines. To break a line within SQL identifier or string literal, use the CONCAT operator (||) and quote the two substrings with the same single or double quote character. Using space characters around the CONCAT operator is optional but is recommended for readability.

In this example, an EVAR name with a long DS name in it is broken. Make a break within a quoted string, not at the period after it.

```
add evar "This is a long DS name that will " ||
"be broken into two, not three, parts”.”ThisEvarName”,
Type set, value 34567;
```

Single Quotes for Attribute Values

Using single quotes around attribute values is optional except when an SQL string literal is required. That is, extra quotes around attribute values do not cause a syntax error. Extra quotes on keywords do cause an error. The MXCS administrative command parser removes unneeded single or double quotes before the strings are used. It also adds single quotes if they are required by the MXCS subsystem or database.

Double Quotes for Object Names

Using double quotes around object names (those that are SQL identifiers) is optional except when there are nonalphanumeric characters, spaces, or lowercase characters in the name. Extra double quotes around identifiers that do not require them do not cause a syntax error, but do inhibit automatically translating lowercase characters to uppercase. The command parser removes the quotes before using the name internally.

Empty Quoted Strings

Empty quoted strings are reported as syntax errors.

Command Caveats

Use care when specifying a backslash (\) and dollar sign ($) in commands when input is not from the command-line or an OBEY command file. These characters have special meaning to the OSS shell.

Output Report Syntax and Conventions

All output report lines contain a maximum of 79 characters, with longer lines automatically folded. Because a DS name can be up to 128 characters, a long DS name is the most likely cause of output report format problems. Field width for DS names is limited to 32 characters. Therefore, folding occurs for names longer than that limit. Obey the 32-character limit where possible.

All long lines fold to the next line at a space character nearest the end of the line, with the remainder of the line indented four spaces. Long DS names appear as in these examples:

```
                                        Start   Server Limits   Server Usage
Name                              State    Auto    Max Init Idle   Con  Avl  Tot
TDM_Default_DataSource            STARTED     ON     5    1    1     0    1    1
DS2                               STOPPED    OFF     5    3    3     0    0    0
ThisDataSourceNameIsLongerThan32CharactersButLessThan79Characters
                                  STOPPED    OFF    10    5    2     0    0    0
ThisDataSourceNameIsLongerThan32CharactersAndLongerThan79CharactersAndMakesTheRe
    portsLookReallyBAD_BAD_BAD_ReallyBAD_ReallyBAD
                                  STOPPED    OFF     5    1    2     0    0    0
```

Attributes display in either tables or two-column lists, with two-column lists period-filled between the attribute name and the value.

The first line of all reports contains the command and full object name. This line can be the only line that some commands display. In addition to the first line, some commands display a line for each object accessed as shown in the preceding example, while others display multiple lines per object.

If you use a wild card in a command and no objects are found, the word `None` appears. When a specific object is requested by name and not found, an error message appears.

Reports that display many objects (resulting from commands with wild cards) provide a count of objects output as the last line. The last line indicates how many errors, if any, were encountered during object iteration. In general, iterations do not stop on one error unless that error stops processing on other objects.

Security

MACL has two classes of commands:

- Informational commands, which return data. Any user can issue these commands:
  - INFO
  - VERSION
- Sensitive commands, which change states and values, and add or delete objects. Only a user with `operator` permissions can issue these commands:
  - ADD
  - ALTER
  - DELETE
  - START
  - STOP

Users issuing sensitive commands are validated against the list of user objects defined in the MXCS configuration database. Each user object has either `operator` or `user` permissions. Each system has its own set of user object definitions, just as it has its own MXCS configuration database. For information on managing these objects, see Section 7, Object User Commands.

Basic security rules are:

- Only the user who installed SQL/MX (super ID user, SUPER.SUPER) can modify the user list.
- Only users who have the `operator` permission in the user list can execute sensitive commands. The SUPER.SUPER user always has `operator` permissions, which cannot be changed.
- Everyone has the `user` permission in the user list by default, so anyone can execute informational commands. The generic user PUBLIC always has `user` permissions, which cannot be changed or removed. Adding individual users with the `user` permission is permitted, but is currently unnecessary.

Note. Use sensitive commands with caution to prevent disrupting users. Grant the `operator` permission only to users trained in ODBC concepts. If several users with the `operator` permission are changing things at the same time, maintaining consistency of object definitions can be at risk.

Example MXCS Configuration

The next list of objects and attributes defines the static portion of the MXCS configuration that appears in all examples in this manual. You start MXCS from a TACL prompt with a command-line that specifies the service name ($AS in this example), the

NonStop TCP/IP address (`-TCP = driver process name`), the initial port number (`-PN 15000`), the trace detail level (`-E 0`), and the port range (`-PR 250`):

```
MXOAS /name $AS, nowait, term $zhome/ -TCP $ZTC0 -PN 15000 -E 0 -PR 250
```

When SQL/MX is initialized, the TDM_Default_DataSource is created with these defaults:

```
MaxServer:                  5
IdleServer:                 1
InitServer:                 1
IdleTimeout:                SYSTEM_DEFAULT
ConnTimeout:                SYSTEM_DEFAULT
StartAutomatic:             OFF
<No Resource Statistics>
<No Resource Management Policy EVARs>
<No Control EVARs>
<No Define EVARs>
<No Set EVARs>
```

A sample configuration of a new DS normally entered for testing is:

```
Data source name:           TDM_Default_DataSource
MaxServer:                  20
IdleServer:                 5
InitServer:                 10
IdleTimeout:                SYSTEM_DEFAULT
ConnTimeout:                60 minutes
StartAutomatic:             Manual
Trace:                      Off
SQLStmtStat:                On
SQLPrepareStat:             On
SQLExecuteStat:             On
SQLExecDirectStat:          On
SQLFetchStat:               On
ConnInfoStat:               On
SessionInfoStat             On
RM: Estimated Cost:         STOP_LOG_WITH_ERROR
Define: =EMPLOYEE           CLASS MAP, FILE $DATA01.PERSNL.EMPLOYEE
Define: =DEPT               CLASS MAP, FILE $DATA02.PERSNL.DEPT
Control: QUERY DEFAULT      HIST_NO_STATS_ROWCOUNT '123456789'
Control: QUERY DEFAULT      HIST_NO_STATS_UEC '99'
Control: TABLE              * TIMEOUT '3000'
Control: TABLE              PERSNL.JOB MDAM 'OFF'
Sets:    CATALOG            TANDEM_SYSTEM_NSK
Sets:    SCHEMA             PUBLIC_ACCESS_SCHEMA
```

Related Commands

You can use several related commands, outside of the MXCS administrative command library, in conjunction with commands supported by the library:

- MXCI Mode Command on page 2-12
- Session Variables on page 2-12
- MXCI Comment on page 2-12

MXCI Mode Command

The MXCI MODE command directs all subsequent commands to the MXCS command library. To enter the MXCS mode, enter:

```
>>mode mxcs;
CS>
```

To indicate the session is in MXCS mode, the prompt changes to `CS>`. The prompt for continuation lines is the normal +> for all modes. The mode prompt of CS persists until the mode is reset. Use CS as the prompt for connection services commands.

To end the MXCS mode, enter:

`cs>mode sql;`	Return to SQL/MX mode; the default at MXCI start.
`CS>mode report;`	Go to report writer mode.
`CS>exit;`	Terminate MXCI.

To determine the active mode, enter:

```
CS>mode display;
```

For an example session, see Appendix A, Example Commands and Objects.

If there is an SQL transaction in progress and not yet committed, the mode switch to `mode mxcs` is not permitted. To enter MXCS administrative commands in this situation, start a second instance of MXCI.

Session Variables

Do not use the TACL DEFINE command or OSS shell environment variables to define session variables. These are not expanded in the input command by MXCI or MXCS.

MXCI Comment

An MXCI comment:

- Begins with two hyphens (`--`) and ends at the end of the line
- Can be on a line by itself or it can follow command text
- Can be embedded in multiline input provided it is outside of keywords and literals

3 Object Service Commands

This section describes the MXCS administrative commands you use for the service object (see Valid Command and Object Type Pairs on page 2-3):

The service object represents one running MXCS ODBC instance on a single CPU. One system can have several service objects running simultaneously on the same or different CPUs. However, each service on the system requires a unique set of NonStop TCP/IP ports assigned to it at start-up.

The service object consists of one association server process plus one configuration server process, and it can own several DS instances and server objects.

The stop and start service commands do not initiate or terminate the association server process. Use the TACL commands RUN MXOAS and STOP to initiate and terminate the server process and to also name the service object. For examples of these commands, see Appendix A, Example Commands and Objects.

INFO SERVICE

This command displays the name and attributes of the service object, including state information, DS summary, and user connection information. The name of the service is always the name of the ODBC association server process (defined by the RUN MXOAS command).

Command syntax is:

```
INFO SERVICE [obj-name][, DETAIL];
```

obj-name

```
[[\system-name.]$service-name | *]
```

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server process of the service to be displayed and must be an association server. When the asterisk (*) wild card is used, each association server on the system is contacted.

DETAIL

displays all attributes. If you omit this option, the report provides only the most commonly used attributes.

Considerations—INFO SERVICE

- INFO SERVICE is a nonsensitive command.
- The asterisk (*) wild card is supported but supplies information about services only on the local system.
- Processes that represent the service object are the association server and the configuration server for each service supported.
- You can change only the state of the service object dynamically. Other attribute changes require terminating the service and starting it again with new options by using the TACL command STOP and RUN MXOAS.
- The association server process must be running, but service can be in any state.

Output Report States—INFO SERVICE

The INFO SERVICE report contains state values with these definitions:

STARTING	Service is initializing (starting).
AVL	Service is available.
STOPPING	Service is going to STOP state, but some DS not yet stopped.
STOPPED	Service is running but not accepting users.

Examples—INFO SERVICE

These examples display the attributes of the service $AS:

- Without DETAIL option:

```
CS>info service $AS;
-- INFO SERVICE \ODBC.$AS Report
PName    State        SPid        TCP/IP PN     Port      Last Updated    Type
$AS      AVL          1,387       $ZTC0         21000     Jan 22 12:15     AS
$Z1094   AVL          1,377       $ZTC0         21001     Jan 22 12:15     CFG
```

- With DETAIL option:

```
CS> info service $AS, detail
-- INFO SERVICE \ODBC.$AS Report with details

PName    State        SPid        TCP/IP PN     Port      Last Updated    Type
$AS      AVL          1,387       $ZTC0         21000     Jan 22 12:15     AS
$Z1094   AVL          1,377       $ZTC0         21001     Jan 22 12:15     CFG

DS Name (active only)                      DS State   Svr: Con     Avl     Tot
Test2                                      STARTED           0       1       1
TDM_Default_DataSource                     STARTED           9       2      11
--------------------                                       -----   -----   -----
Totals For Service                                             9       3      12
```

The last line of the report, Totals for Service, displays only if more than one active DS exists.

Terms used in info service reports are defined as listed:

PName	Process name
State	Service state
SPid	Service program ID (CPU process number)
TCP/IP PN	NonStop TCP/IP process name (use SCF to find IP address)
Port	NonStop TCP/IP port number
Last Updated	Last state change time
Type	Server type: AS = Association server CFG = Configuration server

START SERVICE

This command starts:

- The target service (after the TACL command RUN MXOAS has initiated the process)
- All DSs that auto-start
- The initial servers associated with a started DS

After service starts, clients can begin connecting to the servers.

Command syntax is:

```
START SERVICE [obj-name];
```

obj-name

[[*system-name.*]*$service-name*]

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server process of the service to be started. This name must be an association server process that is in the stopped state.

Considerations—START SERVICE

- START SERVICE is a sensitive command.
- The asterisk (*) wild card is not supported.
- The association server can be in any state but it must exist (after invoking the TACL command RUN MXOAS).
- The service can be in the stopped state, because it is configured to enter it at startup or because an operator previously issued STOP. If this service is already in the available state, START SERVICE is successful but has no effect.

Example—START SERVICE

This example starts the service $AS:

```
CS>start service $AS;
-- START SERVICE \ODBC.$AS Successful
```

STOP SERVICE

This command, which executes the controlled shutdown of the service:

- Puts the service object in the stopping state but does not terminate the association server process
- Stops all DSs and therefore all servers on the target service

Actual termination of the server occurs at different times, depending on the *stop-mode* attribute you specify.

Command syntax is:

```
STOP SERVICE [obj-name] [,AFTER stop-mode], REASON text;
```

obj-name

[[*system-name.*]*$service-name*]

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server process of the service to be stopped.

AFTER *stop-mode*

specifies how quickly the command stops the service, DSs, and servers, by using these key values:

NOW

terminates all servers immediately. A client connected to the server receives the error `communication link failure` on the next action that accesses the server. The client can disconnect and connect to another server or quit. Incomplete transactions are rolled back.

DISCONNECT

closes the servers only after the remote client terminates the connection. A client connected to a server does not know a termination is requested and can continue working until it disconnects normally. This is the default mode.

REASON *text*

is a nonempty SQL string literal of up to 70 characters, which indicates why the service is being shut down. It is written to the EMS log by the service if the command is accepted. Strings of more than 70 characters are truncated with a warning. Single quotes are required.

Considerations—STOP SERVICE

- STOP SERVICE is a sensitive command.
- The asterisk (*) wild card is not supported.
- Service goes to the stopped state only when all active DSs and servers have also transitioned to the stopped state.
- New clients cannot connect to a server associated with a stopping or stopped service.
- The administrator can check progress by using the INFO command.
- The association server and the configuration server are not terminated, but they recognize only administrative commands.
- If the service is already in the stopped or stopping state, STOP SERVICE is successful but has no effect.

Example—STOP SERVICE

This example stops the service $AS:

```
CS>stop service $AS, after now, reason 'text';
-- STOP SERVICE \ODBC.$AS Successful
```

VERSION SERVICE

This command returns version information for these components:

- The MXCS association server
- The Client library
- The MACL

This information helps when reporting problems.

Command syntax is:

```
VERSION SERVICE [obj-name];
```

obj-name

```
[[\system-name.]$service-name | *]
```

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server process for the target service. When the asterisk (*) wild card is used, the report will not contain a version for the MXCS process.

Considerations—VERSION SERVICE

- VERSION SERVICE is a nonsensitive command.
- When an asterisk (*) is used, no service is contacted so version information for MXCS is not reported.
- When an asterisk (*) is not used, the named service must be running, but it can be in any state.

Example—VERSION SERVICE

This example uses an active service and displays NonStop VPROC information for the service $AS and the two libraries, showing the subsystem name and build number/date:

```
CS>version service $AS;
-- VERSION SERVICE \ODBC.$AS Successful
    MXCS        T7969G10_01JAN2004_AS_1202
    CFGCL       T7969G10_01JAN2004_CFL_1215
    MACL        T1054G10_01JAN2004_MACL_0129
```

This example does not connect to a service or display an MXCS VPROC:

```
CS>version service *;
-- VERSION SERVICE \ODBC.* Successful
    MXCS
    CFGCL       T7969G10_01JAN2004_CFL_1215
    MACL        T1054G10_01JAN2004_MACL_0129
```

4 Object Server Commands

This object is the server to which the remote client connects to access a database and is usually called the MXCS server. This object type does not include association servers or configuration servers, so the INFO SERVER and STOP SERVER commands cannot administer these servers.

The association server starts new servers as needed, so only the INFO SERVER and STOP SERVER commands are available to manage a server object.

This section describes the MXCS administrative commands to use for the server object (see Valid Command and Object Type Pairs on page 2-3):

INFO SERVER

This command displays attributes (state, connection, and configuration) of MXCS servers. You cannot change these attributes directly.

Command syntax is:

```
INFO SERVER [obj-name][, DETAIL][, DS ds-name];
```

obj-name

```
[[[\system-name.]$service-name.]$server-name | *]
```

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server process that owns this server.

server-name

is the name of the MXCS server process to be displayed. It must not be an association or configuration server process. It can be the asterisk (*) wild card.

DETAIL

displays all attributes. If you omit this option, the report provides only the most commonly used attributes. When more details are needed, especially for the client side, this option provides additional status. The server must be connected to retrieve useful information for a detailed display.

DS *ds-name*

is the name of the DS to be used to select a subset of all MXCS servers on this service. This option is supported only when the *server-name* specified is the asterisk (*) wild card. It then displays only the servers using this DS that were started by the service process specified in *obj-name*.

Note. *ds-name* is an SQL identifier and is therefore case-sensitive. When using lowercase characters, enclose *ds-name* in quotes to preserve the lowercase characters. Failure to do so causes ***ds-name*** to be automatically uppercased during command execution, which can cause errors. Enter the default *ds-name* as "TMD_Default_DataSource".

Considerations—INFO SERVER

- INFO SERVER is a nonsensitive command.
- The asterisk (*) wild card is supported
- All displays are grouped by DS.
- Display attributes cannot be changed with administrative commands.

Output Report—INFO SERVER

The output report contains state values with these definitions:

STARTING	Server initializing (starting)
AVL	Server available (idle)
CONING	Client connecting
CON	Client connected
DCONING	Client disconnecting
STOPPING	Server Stopping

Example—INFO SERVER

These command examples display the attributes of servers owned by $AS. The first example does not use the detail option. The other examples use the detail option showing an idle server and one connected to a Microsoft Windows client.

Report without detail option:

```
CS>info server $AS.*;
-- INFO SERVER \ODBC.$AS.* Report

DS Name: Test2
PName    State    SPid   TCP/IP PN  Port   Last State Chg    Remote System Name
$Z0481   AVL      1,394   $ZTC0     21002  Jan 22 12:15

DS Name: TDM_Default_DataSource
PName    State    SPid   TCP/IP PN  Port   Last State Chg    Remote System Name
$Z0483   CON      0,404   $ZTC0     21009  Jan 22 12:55              TUSER1-P1
$Z1114   AVL      0,387   $ZTC0     21021  Jan 22 12:16
$Z1115   AVL      1,361   $ZTC0     21022  Jan 22 12:16

-- Cmd Processed 4 objects, 0 failed.
```

Detailed report for a server that is not connected:

```
CS>info server $AS.$Z0481, detail;
-- INFO SERVER \ODBC.$AS.$Z0481 Report, DETAIL

DS Name: TDM_Default_DataSource
PName    State    SPid  TCP/IP PN   Port   Last State Chg    Remote System Name
$Z0481   AVL      0,387   $ZTC0     21021  Jan 22 12:16
  Cl User ID.................Not Avail     Cl App Name.................Not Avail
  Cl Proc ID.................Not Avail
```

Detailed report for a server connected to a PC application (Microsoft Access):

```
CS>info server $AS.$Z0483, detail;
-- INFO SERVER \ODBC.$AS.$Z0483 Report, DETAIL

DS Name: TDM_Default_DataSource
PName    State    SPid  TCP/IP PN   Port   Last State Chg    Remote System Name
$Z0483   CON      0,404   $ZTC0     21009  Jan 22 12:55            TXNUSER1-P1
  Cl User ID.............swdev1.user1     Cl App Name..............MS Access
  Cl Proc ID.....................836
```

Terms used in `info server` reports are defined as listed:

`Cl`	Client
`Cl User ID`	NonStop OS user ID used by client application to connect to server, if connected
`Cl Proc ID`	Client process ID, if connected
`Cl App ID`	Client application ID, if connected
`Last State Change`	Time connection opened or idle state began
`Port`	NonStop TCP/IP port number
`PName`	Process name
`State`	Service state
`SPid`	Service program ID (CPU process number)
`TCP/IP PN`	NonStop TCP/IP process name (use SCF to find IP address)

STOP SERVER

This command puts the server in the stopping state. The actual termination occurs at different times, depending on the *stop-mode* attribute you specify.

Check on progress by using the INFO SERVER command.

Command syntax is:

```
STOP SERVER [obj-name] [,AFTER stop-mode];
```

obj-name

[[[*system-name.*]*$service-name.*]*$server-name*]

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server process that owns this server.

server-name

is the name of the MXCS server process to be stopped. It must not be an association or configuration server. It cannot be the asterisk (*) wild card.

AFTER *stop_mode*

specifies the termination time of the MXCS server with a key value:

NOW

terminates the server abruptly (abort as fast as possible). A client connected to the server receives the `communication link failure` error on the next action that accesses the server. The client is required to either disconnect and connect to another server or to quit. Incomplete transactions are rolled back. This is the default mode.

Considerations—STOP SERVER

- STOP SERVER is a sensitive command.
- STOP SERVER deletes the server from the system. The server does not transition to a stopped state and wait for a start command after STOP SERVER.
- The asterisk (*) wild card is not supported.
- The service that starts the MXCS server owns it and stops it.

- Stopping a DS stops all servers it owns on the specified SERVICE.
- Stopping a service stops all servers it owns.
- Client connections can change rapidly. Issue INFO SERVER DETAIL before issuing STOP server to verify that the server is connected to the appropriate client.
- If the target server does not exist, STOP SERVER returns an error.
- If no client is connected to the target server, it terminates immediately.
- If the server is already in the stopping state, STOP SERVER returns only a success message.
- To start and stop DS trace and statistics attributes for all servers owned by a DS, use the ALTER DS command.

Example—STOP SERVER

This example stops the server $Z0123 owned by $AS:

```
CS>STOP SERVER $AS.$z0123;
-- STOP SERVER \ODBC.$AS.$Z0123 Successful
```

5 Object DS Commands

This object is a server-side data source (DS) that represents a named collection of use, connection, and database attributes. Clients specify the desired DS at connection time to establish the run-time environment. The DS is a permanent object that is recorded in the MXCS administrative database.

The DS is owned by each service that is using it. The same DS has the same configured values in each service using it, but it can have different states, trace states, and server counts.

All services see all DSs on a system, and changes to configured values of the DS are propagated to all services. You can start a DS instance in some services and stop it in others. Before you can view DSs on a service, you must start the service once. You can view DSs in the MXCS administrative database without starting a service.

This section describes the MXCS administrative commands that are available for the DS object (see Valid Command and Object Type Pairs on page 2-3):

ADD DS

This command adds one DS to the MXCS configuration on the system.

Command syntax is:

```
ADD DS [obj-name][, COPY old-ds-name]
     [, MaxServer count][, IdleServer count][, InitServer count]
     [, IdleTimeout timeout-val][, ConnTimeout timeout-val]
     [, StartAutomatic [OFF]][, InitPri priority-val]
     [, Trace [OFF]][, CpuList cpu-val][, AllStat [OFF]]
     [, SQLExecuteStat [OFF]][, SQLExecDirectStat [OFF]]
     [, SQLStmtStat [OFF]][, SQLFetchStat [OFF]]
     [, SQLPrepareStat [OFF]][, SessionInfoStat [OFF]]
     [, ConnInfoStat [OFF]];
```

obj-name

[[[*system-name.*][*$service-name* | *].]*ds-name*]

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server that handles or processes the command or the asterisk (*) wild card. The new DS is added to all services in either case. An asterisk (*) causes connection to the database rather than to any specific service, and it is the only valid value when no services are active.

ds-name

is the name of the DS to be added (created) and cannot be the asterisk (*) wild card. It must be a new name or the command will fail.

Note. *ds-name* is an SQL identifier and is therefore case-sensitive. When using lowercase characters, enclose *ds-name* in quotes to preserve the lowercase characters. Failure to do so causes ***ds-name*** to be automatically uppercased during command execution, which can cause errors. Enter the default *ds-name* as "TMD_Default_DataSource".

COPY *old-ds-name*

is the name of any existing DS used as a model for the new one on the same system. It is an SQL identifier. The name must be a DS component name only and not a system or service name. All attributes of this existing DS are defaults for the new one. Parameters specified on this ADD command override values copied from the existing DS. The CpuList values must all come from the command-line or from the old DS. If you specify one of these numbers on the input, specify them all. All attributes from the existing DS copy into the new DS, creating a clone of the

existing DS. The existing DS and the new one must have different names. Two DSs with the same name generate an error.

Using this parameter can be the first step in renaming a DS, followed by stopping and deleting the old DS.

`MaxServer` *`count`*

is the upper limit on the servers that can be operational for this service on this DS. The default value is 1. A value of 0 gives the system default, which is a value determined by the system at run time. Valid values are 0 through the total number of ports allowed, minus 3. The MXOAS command determines the total number of ports allowed. Different services can have different limits, which prevents you from validating them when you create them.

`IdleServer` *`count`*

is the lower limit on the available idle servers that are operational for this service on this DS. Valid values are 0 through `MaxServer` count, with 0 as the default. This count represents idle servers that are waiting for user connections. At DS startup, once the `InitServer` *`count`* is satisfied, additional servers are started until the `IdleServer` *`count`* is reached. Initially, idle servers are started in blocks of five, with a check for incoming connection requests occurring between each block. After peak user loads, idle servers are terminated to the `IdleServer` *`count`* value (see `IdleTimeout` on this page).

`InitServer` *`count`*

is the suggested number of servers that each service starts when the DS first starts. Legal values are from 0 to the smaller of either 1000 or the value specified by the `MaxServer` *`count`*, with the default value as 0. If the initial server count exceeds 1000, an error is displayed. When the value for `InitServer` *`count`* is greater than 0, the servers are started in blocks of five with a check for incoming connection requests occurring between each block. To permit servicing incoming users, the DS goes to the STARTED state as soon as the first server starts. These servers count as idle if they are not allocated to new users immediately.

Note. The value of 1000 is applicable from H06.26/J06.15 RVU and SQL/MX version R3.2.1.

`IdleTimeout` *`timeout-val`*

is a keyword or the integer number of minutes a server waits in the available state for a connection before it stops itself when the count of servers exceeds the `IdleServer` *`count`* value for the DS. The default value is SYSTEM_DEFAULT, which is defined by MXCS (ten minutes, currently). Other valid values are NO_TIMEOUT and from the integer one to a large number.

`timeout-val`

```
SYSTEM_DEFAULT | NO_TIMEOUT | minutes
```

ConnTimeout *timeout-val*

is a keyword or the number of minutes a client-server connection can remain idle before the server terminates the connection and becomes available. The default value is SYSTEM_DEFAULT, which is defined by MXCS (ten minutes currently). Other valid values are NO_TIMEOUT and from the integer one to a large number.

timeout-val

SYSTEM_DEFAULT | NO_TIMEOUT | *minutes*

StartAutomatic [OFF]

is a flag to request the automatic starting of this DS when its association server process starts. Use this flag without the OFF option to turn it ON. The default value is OFF when the attribute is not used. The DS is created in the stopped state and must be explicitly started on one or more services.

InitPri *priority-val*

is the initial system priority used to start a new server on this DS. Valid values are 1 to 199 and SYSTEM_DEFAULT, which is the association server's process priority.

priority-val

SYSTEM_DEFAULT | *number*

Trace [OFF]

is a flag for activating the server trace facility in all servers using this DS and started by this service. The Trace flag is not a permanent setting, is not stored in the configuration database, and is not retained through a service shutdown. It is set as specified on this DS for the current service or all services when the asterisk (*) wild card is used for the *service-name*. Trace data is written to the collector specified when the service is started. Use this flag without the OFF option to turn it ON. The default value is OFF when the attribute is not used.

CpuList *cpu-val*

is an SQL string literal (with multiple values enclosed in single quotes), containing the list of CPU numbers where the service can start servers for this DS. When you use this parameter, specify all requested CPUs. Valid values are 0 to 15 and ALL. The value ALL specifies all available CPUs on the system where the service is running.

When to using CPUList, note:

- Duplicate CPU numbers are ignored.
- The order of CPU numbers from CPUList appears as the next higher sequential CPU number in the list and not in the order entered on the command-line.

- Nonexistent CPU numbers are not reported. Because a CPU can be down now and up later, nonexistent CPU numbers return an EMS warning only when you attempt to use them.
- Only new servers are affected, not currently running servers.

The `CpuList` values come from either the command-line or from the source DS, but not a combination of the two. If you specify one of these CPUs on the command-line, you must specify them all.

`cpu-val`

`ALL | ‘`*`nbr0`* `[`*`nbr1`*`] ... [`*`nbr15`*`]’`

`AllStat [OFF]`

is a flag to set for this DS, all seven individual statistics gathering flags. Use this option without the OFF option to turn these flags ON. Use this flag along with the `copy` option to reset all flags, regardless of the flags an existing DS has set for statistics.This flag must precede other individual statistic flags, which reset just those individual flags. The default value is OFF when the attribute or copy is not used.

`SQLExecuteStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of SQLExecute statistics at the time an EXECUTE statement is received. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset just this one flag. Statistics are written to the collector specified when the service is started. The default value is OFF when the attribute is not used.

`SQLExecDirectStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of SQLExecDirect statistics at the time an EXECUTE statement is received. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset only this flag. Statistics are written to the collector specified when the service is started. The default value is OFF when the attribute is not used.

`SQLStmtStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of statement statistics at the time a PREPARE statement is received. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset only this flag. Statistics are written to the collector specified when the service is started. The default value is OFF when the attribute is not used.

`SQLFetchStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of SQLFetch statistics at the time a statement is closed. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset only this flag.

Statistics are written to the collector specified when the service is started. The default value is OFF when the attribute is not used.

`SQLPrepareStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of SQLPrepare statistics at the time a PREPARE statement is received. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset only this flag. Statistics are written to the collector specified when the service is started. The default value is OFF when the attribute is not used.

`SessionInfoStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of session statistics at the time a session is terminated. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset only this flag. Statistics are written to the collector specified when the service is started. The default value is OFF when the attribute is not used.

`ConnInfoStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of session statistics at the time a session is established. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset only this flag. Statistics are written to the collector specified when the service is started. The default value is OFF when the attribute is not used.

Considerations—ADD DS

- ADD DS is a sensitive command.
- The asterisk (*) wild card is not supported for DS name.
- Parameters are optional and have default values.
- Default value for all flags is ON. Specifying the OFF option with the flag is permitted but is unnecessary. The OFF option is most useful with ALTER DS and is better omitted with ADD DS.
- Statistics parameters are processed as they occur on the command-line. For example, the `AllStat` flag sets all statistics flags to ON, with successive commands on the same command-line setting individual flags to OFF.
- Services see the DS as soon as it is created. After creation, the DS has no EVARs defined unless the COPY parameter creates them.
- The DS is created in the stopped state and must be explicitly started on one or more services. Specifying the `StartAutomatic` attribute starts the target DS when its associated server process starts.
- Except for errors when setting the trace flag during an ADD command, an error causes the DS to be removed completely.

Examples—ADD DS

This example creates the DS `Mixed Case DSN` in all services, even though the service $AS is specifically addressed:

```
CS>add ds $AS."Mixed Case DSN", MaxServer 30, IdleServer 5, InitServer 10,
            IdleTimeout SYSTEM_DEFAULT, ConnTimeout 60, AllStat;
-- ADD DS \ODBC.$AS.Mixed Case DSN Successful
```

Use this command to clone a DS and change one parameter:

```
CS>add ds $AS."Son of Mixed Case DSN", COPY "Mixed Case DSN", MaxServer 50;
-- ADD DS \ODBC.$AS.Son of Mixed Case DSN Successful
```

ALTER DS

This command is similar to the ADD DS command, except ALTER DS modifies the attributes of an existing DS in the MXCS configuration on one system.

Command syntax is:

```
ALTER DS [obj-name]
     [, MaxServer count][, IdleServer count][, InitServer count]
     [, IdleTimeout timeout-val][, ConnTimeout timeout-val]
     [, StartAutomatic [OFF]][, InitPri priority-val]
     [, Trace [OFF]][, CpuList cpu-val][, AllStat [OFF]]
     [, SQLExecuteStat [OFF]][, SQLExecDirectStat [OFF]]
     [, SQLStmtStat [OFF]][, SQLFetchStat [OFF]]
     [, SQLPrepareStat [OFF]][, SessionInfoStat [OFF]]
     [, ConnInfoStat [OFF]];
```

obj-name

[[[*system-name.*][*$service-name* | *].]*ds-name*]

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server that handles or processes the command or the asterisk (*) wild card. Most changes effect the DS on all services in either case. An asterisk (*) causes connection to the database rather than to any specific service, and it is the only valid value when no services are active.

ds-name

is the name of the DS to be modified and cannot be the asterisk (*) wild card.

Note. *ds-name* is an SQL identifier and is therefore case-sensitive. When using lowercase characters, enclose *ds-name* in quotes to preserve the lowercase characters. Failure to do so causes ***ds-name*** to be automatically uppercased during command execution, which can cause errors. Enter the default *ds-name* as "TMD_Default_DataSource".

MaxServer *count*

is the upper limit on the servers that are operational for this service on this DS. Valid values are 0 through the total number of ports allowed minus 3. See documentation on the MXOAS command for the total number of ports allowed. The practical limit can be different for each service and is often in the range of 20 to 500. A value of 0 gives the system default, which is a value determined by the system at run time.

`IdleServer` *`count`*

is the lower limit on the available idle servers that are operational for this service on this DS. Valid values are 0 through `MaxServer` count, with 0 as the default. This count represents idle servers that are waiting for user connections. At DS startup, once the `InitServer` *`count`* is satisfied, additional servers are started until the `IdleServer` *`count`* is reached. Initially, idle servers are started in blocks of five, with a check for incoming connection requests occurring between each block. After peak user loads, idle servers are terminated to the `IdleServer` *`count`* value (see `IdleTimeout` on this page).

`InitServer` *`count`*

is the suggested number of servers that each service starts when the DS first starts. Legal values are from 0 to the smaller of either 1000 or the value specified by `MaxServer` *`count`*, with the default value as 0. If the initial server count exceeds 1000, an error is displayed. When the value for `InitServer` *`count`* is greater than 0, the servers are started in blocks of five with a check for incoming connection requests occurring between each block. To permit servicing incoming users, the DS goes to the STARTED state as soon as the first server starts. These servers count as idle servers if they are not allocated to new users immediately.

Note. The value of 1000 is applicable from H06.26/J06.15 RVU and SQL/MX version R3.2.1.

`IdleTimeout` *`timeout-val`*

is a keyword or integer number of minutes a server waist in the available state for a connection before it stops itself when the count of servers exceeds the DS server count for the DS. Valid values are SYSTEM_DEFAULT, which is defined by MXCS (ten minutes, currently); NO_TIMEOUT; and the integer one to a large number.

`ConnTimeout` *`timeout-val`*

is a keyword or the integer number of minutes a client-server connection remains idle before the server terminates the connection and becomes available. Valid values are SYSTEM_DEFAULT, which is defined by MXCS (ten minutes, currently); NO_TIMEOUT; and the integer one to a large number.

`StartAutomatic [OFF]`

is a flag to request the automatic starting of this DS when its associated server process starts. Use this flag without the OFF option to turn it ON. The default value is OFF when the attribute is not used. The DS is created in the stopped state and must be explicitly started on one or more services.

`InitPri [`*`priority-val`*`]`

is the initial system priority used to start a new server on this DS. The valid values are `1` to `199` and SYSTEM_DEFAULT | number, which is the association server process priority.

`priority-val`

SYSTEM_DEFAULT | *`number`*

`Trace [OFF]`

is a flag for activating the server trace facility in all servers using this DS and started by this service. The Trace flag is not a permanent setting, is not stored in the configuration database, and is not retained through a service shutdown. It is set as specified on this DS for the current service or all services when *`service-name`* is the asterisk (*) wild card. Trace data is written to the collector specified when the service is started. Use this flag without the OFF option to turn it ON. The default value is not changed when the attribute is not used. Changing this flag affects currently running servers started by this service.

`CpuList` *`cpu_val`*

is an SQL string literal (in single quotes if specifying more than one value), containing the list of CPU numbers where the service can start servers for this DS (the next CPU is selected round-robin). When you use this parameter, all requested values must be specified again. The valid values are `0` to `15` and `ALL`, which specifies all available CPUs on the system where the service is running. Duplicate numbers are accepted. Because a CPU can be down now and up later, nonexistent CPU numbers are not reported when set but return an EMS warning when you attempt to use them. This option affects only new servers; currently running servers are not changed.

The `CpuList` values come from either the command-line or from the source DS, and not a combination of the two. If you specify one of these CPUs on the command-line, you must specify them all:

`cpu-val`

ALL | *`‘nbr0`* [*`nbr1`*] ... [*`nbr15`*]*`’`*

`AllStat [OFF]`

is a flag to set, for this DS, all seven individual statistics gathering flags for servers started by this service. Use this flag without the OFF option to turn all seven individual flags ON. This flag must precede other individual statistics flags, which reset just those individual flags. Currently running servers started by any service are affected by changing this flag.

`SQLExecuteStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of SQLExecute statistics at the time an EXECUTE statement is received for servers started by this service. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset just this one flag. Statistics are written to the collector specified when the service is started. Currently running servers started by any service are affected by changing this flag.

`SQLExecDirectStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of SQLExecDirect statistics at the time an EXECUTE statement is received for servers started by this service. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset only this flag. Statistics are written to the collector specified when the service is started. Currently running servers started by any service are affected by changing this flag.

`SQLStmtStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of statement statistics at the time a PREPARE statement is received for servers started by this service. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset only this flag. Statistics are written to the collector specified when the service is started. Currently running servers started by this service are affected by changing this flag.

`SQLFetchStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of SQLFetch statistics at the time a statement is closed for servers started by this service. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset only this flag. Statistics are written to the collector specified when the service is started. Currently running servers using this DS and started by any service are affected by changing any flag.

`SQLPrepareStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of SQLPrepare statistics at the time a PREPARE statement is received. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset only this flag. Statistics are written to the collector specified when the service is started. Currently running servers using this DS and started by any service are affected by changing any flag.

`SessionInfoStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of session statistics at the time a session is terminated. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset only this flag. Statistics are written to the collector specified when the service is started. Currently

running servers using this DS and started by any service are affected by changing any flag.

`ConnInfoStat [OFF]`

is a flag to turn ON, for this DS, the statistics gathering of session statistics at the time a session is established. Use this flag without the OFF option to turn the individual flag ON. This flag can follow the `AllStat` flag to reset only this flag. Statistics are written to the collector specified when the service is started. Currently running servers using this DS and started by any service are affected by changing any flag.

Considerations—ALTER DS

- ALTER DS is a sensitive command.
- The asterisk (*) wild card is not supported fro the DS name.
- Changes are immediately visible to all services.
- The DS can be in any state.
- Changing the `StartAutomatic` attribute does not change the current state of the DS.
- Omitting a parameter indicates no change to that attribute and does not reset that attribute to its default value. Omitting all parameters results in an error.
- The default value for all flags is the current value. To set it to on, specify the attribute without the OFF option. To turn it off, specify the OFF option.
- An error while changing one parameter does not stop or roll back changes to other parameters.

Examples—ALTER DS

- Modify two attributes of the DS `TDM_Default_DataSource` in all services, even though the service $AS is specifically addressed:

```
CS>alter ds $as."TDM_Default_DataSource", MaxServer 10, ConnTimeout no_timeout;
-- ALTER DS \ODBC.$AS.TDM_Default_DataSource Begun
-- ALTER DS \ODBC.$AS.TDM_Default_DataSource Successful for basic settings
```

- Turn on trace for all active MXCS servers started by the service $AS that are using the DS `TDM_Default_DataSource`:

```
CS>alter ds $as."TDM_Default_DataSource", Trace;
-- ALTER DS \ODBC.$AS.TDM_Default_DataSource Begun
-- ALTER DS \ODBC.$AS.TDM_Default_DataSource Successful for trace
```

- Turn off trace for all active MXCS servers started by any service that is using DS `TDM_Default_DataSource` and also modify other parameters. Because failures do not roll back or stop the actions, this output shows the progress of changes to the DS attributes in the database and the trace results on each active service. In this example, the DS name, `TDM_Default_DataSource`, is enclosed in quotes to

preserve its lowercase characters. If quotes are not used, the DS name automatically changes to uppercase `TDM_Default_DataSource` when the command executes.

```
CS>alter ds *."TDM_Default_DataSource", Trace off, maxserver 6, initpri 52,
cpulist '1', allstat;
-- ALTER DS \ODBC.*.TDM_Default_DataSource Begun
-- ALTER DS \ODBC.*.TDM_Default_DataSource Successful for basic settings
-- ALTER DS \ODBC.*.TDM_Default_DataSource Successful for CpuList
-- ALTER DS \ODBC.*.TDM_Default_DataSource Successful for InitPri
-- ALTER DS \ODBC.*.TDM_Default_DataSource Successful for statistics
-- ALTER DS \ODBC.$AS1.TDM_Default_DataSource Failed for trace
-- ALTER DS \ODBC.$AS2.TDM_Default_DataSource Successful for trace
-- ALTER DS \ODBC.$AS3.TDM_Default_DataSource Failed for trace
-- ALTER DS \ODBC.$AS4.TDM_Default_DataSource Failed for trace
-- Cmd processed 4 objects, 3 failed. Iteration stopped after 3 failures
```

DELETE DS

This command removes one DS from the MXCS configuration on the target system. The DS must be in the stopped state in all services on the system, or an error is generated and the DS is not deleted.

Command syntax is:

```
DELETE DS [obj-name];
```

obj-name

```
[[[\system-name.][$service-name | *].]ds-name]
```

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server that handles or processes the command or the asterisk (*) wild card. In either case, the DS must be in the stopped state on all services and is deleted from all services. The asterisk indicates to connect to the database rather than to a specific service, and it is the only valid value when no services are active.

ds-name

is the name of the DS to be deleted. It cannot be the asterisk (*) wild card or the default DS, TDM_Default_DataSource.

Note. *ds-name* is an SQL identifier and is therefore case-sensitive. When using lowercase characters, enclose *ds-name* in quotes to preserve the lowercase characters. Failure to do so causes ***ds-name*** to be automatically uppercased during command execution, which can cause errors. Enter the default *ds-name* as "TMD_Default_DataSource".

Considerations—DELETE DS

- DELETE DS is a sensitive command.
- The asterisk (*) wild card is not supported for the DS name.
- The REASON argument is not required for DELETE DS because it was captured for STOP DS.

Example—DELETE DS

This example removes the DS `Test2` from the system:

```
CS>delete ds $AS."Test2";
-- DELETE DS \ODBC.$AS.Test2 Successful
```

INFO DS

This command displays attributes of one or all defined DSs on this system.

Command syntax is:.

```
INFO DS [obj-name][, DETAIL][, ACTIVE];
```

obj-name

[[[*system-name.*][*$service-name* | *].]*ds-name* | *]

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server that handles or processes the command or the asterisk (*) wild card. An asterisk causes a connection to the database rather than a specific service, and it is the only valid value when no services are active. The state information for one DS instance can be different in each service, so a specific service must be selected to get state information. When the asterisk is used, state information is displayed as n/a, or not available. It cannot be an asterisk when the ACTIVE attribute is specified.

ds-name

is the name of the DS to be displayed and can be the asterisk (*) wild card.

Note. *ds-name* is an SQL identifier and is therefore case sensitive. When using lower case characters, enclose *ds-name* in quotes to preserve the lower case characters. Failure to do so causes ***ds-name*** to be automatically upper cased during command execution, which can cause errors. Enter the default *ds-name* as "TMD_Default_DataSource".

DETAIL

displays all attributes. When you omit this option, the report provides only the most commonly used attributes. When more details are needed, this option provides all attributes.

ACTIVE

displays information on only the DSs that are in the starting, started, or stopping states on one service. This flag is valid only when an asterisk (*) is used for the DS name, and a specific single service name is specified. If no DSs are active on the specified service, the command report displays a line stating None.

The output report contains a column of state values with these definitions:

STARTING	Data source initializing (starting first server)
STARTED	Data source started
STOPPING	Data source stopping (all servers not down)
STOPPED	Data source stopped

Considerations—INFO DS

- INFO DS is a nonsensitive command.
- The asterisk (*) wild card is valid for *$service-name* and *ds-name*, but no state information is displayed when *$service-name* is an asterisk (*).
- Most attributes are configured attributes, but state and server counts are also shown.
- The operator can change any attributes.
- None of the servers or the EVARs are shown in this display. Use INFO on those objects directly if that information is desired.

Examples—INFO DS

These commands display the attributes of all DSs on the system. This first example has four DSs and is invoked without the `detail` option:

```
CS>info ds $AS.*;
-- INFO DS \ODBC.$AS.* Report

                                        Start   Server Limits   Server Usage
Name                              State    Auto    Max Init Idle   Con  Avl  Tot
TDM_Default_DataSource            STARTED    ON     5    1    1     0    1    1
DS2                               STOPPED   OFF     5    3    3     0    0    0
ThisDataSourceNameIsLongerThan32CharactersButLessThan79Characters
                                  STOPPED   OFF     1    0    0     0    0    0
Mixed Case DSN                    STARTING  OFF    20    7    1     1    1    2
Unused_DS_with_32_Character_Name STOPPED    OFF   247  150    7     0    0    0
-- Cmd Processed 5 objects, 0 failed.
```

This second example appears with the `detail` option but was done at a time different than the preceding one. This example has two DSs.

```
CS>INFO DS $AS.*, detail, active;
-- INFO DS \ODBC.$AS.* Report, detailed

Name: \ODBC.$AS.TDM_Default_DataSource
  CpuList: 0,1,2,3,4
  InitPri.....................140     CurrentState.................STARTED
  ConnectedServers.............1     AvailableServers.................1
  LastStateChg.........Feb 14 10:08  LastUpdate..................Not Set
  IdleServer...................1     IdleTimeout............SYSTEM_DEFAULT
  MaxServer....................5     ConnTimeout............SYSTEM_DEFAULT
  InitServer...................1     StartAutomatic.....................ON
  Trace.......................OFF     SQLPrepareStat.....................OFF
  ConnInfoStat................OFF     SQLExecuteStat.....................OFF
  SessionInfoStat.............OFF     SQLExecDirectStat..................OFF
  SQLStmtStat.................OFF     SQLFetchStat.......................OFF
```

```
Name: \ODBC.$AS.Mixed Case DSN
  CpuList: 3,4,5,12,13,14,15
  InitPri.....................150     CurrentState.............STARTING
  ConnectedServers.............1     AvailableServers.................1
  LastStateChg.........Oct 18 11:43  LastUpdate................Not Set
  IdleServer...................5     IdleTimeout...........SYSTEM_DEFAULT
  MaxServer...................20     ConnTimeout.....................60
  InitServer..................10     StartAutomatic..................OFF
  Trace.....................OFF     SQLPrepareStat..................ON
  ConnInfoStat................ON     SQLExecuteStat..................ON
  SessionInfoStat.............ON     SQLExecDirectStat...............ON
  SQLStmtStat.................ON     SQLFetchStat....................ON
-- Cmd Processed 2 objects, 0 failed.
```

Terms used in `info DS` reports are:

`State`	DS state
`StartAuto`	Automatic DS start when its association server process starts
`Max`	Upper limit of operational servers for this service on this DS
`Init`	Number of idle servers to start when DS starts
`Idle`	Lower limit of idle servers for this service on this DS
`Con`	Number of connected servers
`Avl`	Available servers for this service on this DS (registered minus connected servers)
`Tot`	Number of registered servers

Terms used in `info DS` reports with the `detail` option are:

`CpuList`	List of CPUs that the service can start the servers on (round-robin)
`InitPri`	Initial process priority to start a new server on this DS
`CurrentState`	Current state of the data source
`ConnectedServers`	Number of connected servers
`AvailableServers`	Servers available for this service on this DS
`LastStateChg`	Time the connection opened or the idle state began
`LastUpdate`	Last time a DS attribute was updated
`IdleServer`	Lower limit of the idle servers for this service on this DS
`IdleTimeout`	Number of minutes a server waits in the available state before stopping itself
`MaxServer`	Upper limit of servers for this service on this DS
`ConnTimeout`	Number of minutes a client server connection remains idle before the server terminates
`InitServer`	Number of idle servers to start when the DS starts

`StartAutomatic`	Automatic DS start when its associated server process starts
`Trace`	Activates server trace facility in all servers started by and using this DS
`SQLPrepareStat`	SQL prepare statistics gathered when a PREPARE statement is received
`ConnInfoStat`	Connection information statistics gathered when a connection is established
`SQLExecuteSta`	SQL execute statistics gathered when an EXECUTE statement is received
`SessionInfoStat`	Session information statistics gathered when a session is terminated
`SQLExecDirectStat`	SQL execute direct statistics gathered when an EXECUTEDIRECT statement is received
`SQLStmtStat`	SQL statement statistics gathered when a PREPARE statement is received
`SQLFetchStat`	SQL fetch statistics when closing a statement

START DS

This command starts one DS on one service or on all services if the asterisk (*) wild card is specified. New clients can then use this DS. If the DS is already started, START DS returns a success message.

Command syntax is:.

```
START DS [obj-name];
```

obj-name

```
[[[\system-name.][$service-name | *].]ds-name]
```

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server that handles or processes the command where this DS should be started. Using the asterisk (*) wild card indicates to start all services, which enables the DS to be started in all services for one system on one command.

ds-name

is the name of the DS to be started and cannot be the asterisk (*) wild card.

Note. *ds-name* is an SQL identifier and is therefore case-sensitive. When using lowercase characters, enclose *ds-name* in quotes to preserve the lowercase characters. Failure to do so causes ***ds-name*** to be automatically uppercased during command execution, which can cause errors. Enter the default *ds-name* as `"TMD_Default_DataSource"`

Considerations—START DS

- START DS is a sensitive command.
- The asterisk (*) wild card is not supported for DS name.

Example—START DS

This example starts the DS `TDM_Default_DataSource` for the service $AS:

```
CS>start ds $AS.”TDM_Default_DataSource”;
-- START DS \ODBC.$AS.TDM_Default_DataSource Successful
```

STOP DS

This command stops one DS on all services or all DSs on one service. The actual termination occurs at different times, depending on the *stop-mode* attribute you specify. The DS goes to the stopped state only after all servers are stopped.

Note. Use INFO DS to determine when a DS is in the STOPPED state.

Command syntax is:

```
STOP DS [obj-name], REASON text, [AFTER stop-mode];
```

obj-name

[[[*system-name.*][*$service-name* | *].]*ds-name* | *]

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server that handles or processes the command where this DS should be stopped. Using the asterisk (*) wild card indicates to stop all services and is valid only when a specific DS is selected. This value allows the DS to be stopped on all services on one system in one command, so a subsequent delete is possible.

ds-name

is the name of the DS to be stopped. It can be the asterisk (*) wild card, provided a specific service is selected.

Note. *ds-name* is an SQL identifier and is therefore case-sensitive. When using lowercase characters, enclose *ds-name* in quotes to preserve the lowercase characters. Failure to do so causes ***ds-name*** to be automatically uppercased during command execution, which can cause errors. Enter the default *ds-name* as "TMD_Default_DataSource".

AFTER *stop_mode*

specifies the termination time of the server with one of these key values:

NOW

terminates the server abruptly (abort as fast as possible). A client connected to the server receives the communication link failure error on the next action that accesses the server. The client is expected to either disconnect and connect to another server or to quit. Incomplete transactions are rolled back.

DISCONNECT

terminates the server but not until the remote client terminates the connection. The client connected to a server is not notified that a termination is requested and continues working until it disconnects normally. It could take hours or days for the client to disconnect normally. This is the default mode.

REASON *text*

is a nonempty SQL string literal of up to 70 characters, which indicates why the DS is being shut down. It is written to the EMS log by the service if the command is accepted. Strings of 70 or more characters are truncated with a warning. Single quoting is required.

Considerations—STOP DS

- STOP DS is a sensitive command.
- The asterisk (*) wild card is valid for *$service-name* or *ds-name* but not both in the same command.
- Check on the progress of the STOP DS command by using the INFO DS command.
- Issuing STOP DS inhibits new client access to the target DS.
- If target DS is already stopped or stopping, STOP DS returns a success message.
- Target DS must be stopped in all services before deleting it using DELETE DS.

Examples—STOP DS

- This example stops the DS TDM_Default_DataSource being used by the service $AS and also stops all servers started by the service $AS that are using this DS:

```
CS>stop ds $AS."TDM_Default_DataSource", REASON 'text, text';
-- STOP DS \ODBC.$AS.TDM_Default_DataSource Successful
```

- this example stops all DSs used by the service $AS along with all MXCS servers:

```
CS>stop ds $AS.*, after now, reason 'Shutdown for the day';
-- STOP DS \ODBC.$AS.* Begun
-- STOP DS \ODBC.$AS.TDM_Default_DataSource Successful
-- STOP DS \ODBC.$AS.Test2 Successful
-- Cmd Processed 2 objects, 0 failed.
```

6 Object EVAR Commands

This section describes the MXCS administrative commands to use for the object EVAR (see Valid Command and Object Type Pairs on page 2-3):

EVAR Types

The EVAR object defines NonStop OS or SQL environment variables and provides commands to manage them. The server environment uses these variables to define various aspects of queries, usually substitution variables or SQL operational parameters. These variables affect operations after the client has connected to the MXCS server.

An EVAR has three main components: type, name, and value. Each DS has its own collection of EVARs. EVARs are not shared among DSs.

In these examples of different types of EVARs, the uppercase text example is displayed by the INFO command and is stored in the database after the input command is parsed and defaults added:

Note. Single quotes delimiting ‘OFF’ are required.

```
CONTROL QUERY DEFAULT QUERY_CACHE_STATISTICS 'OFF'    (control type example)
ADD DEFINE =KEVIN_TABLE, FILE $DATA02.KEVIN2.KTABLE   (define type example)
SET NAMETYPE ANSI                                     (set type example
```

This list indicates the supported types and the default prefix:

`all`	No default prefix
`control`	CONTROL QUERY DEFAULT
`define`	ADD DEFINE
`rmp`	Not applicable
`set`	SET

EVAR Order

Use the INFO EVAR command to determine the order in which the environment variables are applied when used. EVARS appear in the order of their creation or addition, with the latest EVAR added for a specific type at the end of the list. If the order of the EVARs is important, individually delete one or more of the existing target EVARS by using the DELETE EVAR command. Add them back individually by using the ADD EVAR command in the order that you want them to appear.

ADD EVAR

This command adds one EVAR to the DS definition specified.

Command syntax is:

```
ADD EVAR [obj-name][, TYPE type-name][, PREFIX string], VALUE string;
```

or

```
ADD EVAR [obj-name], TYPE RMP, LIMIT number[, ACTION key-val];
```

obj-name

[[[[*system-name*.][*$service-name* | *].]*ds-name*.]*evar-name*]

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server that handles or processes the command or the asterisk (*) wild card. In either case, the EVAR is added to this DS on all servers.

ds-name

is the name of the DS owning the EVAR and cannot be the asterisk (*) wild card.

Note. *ds-name* is an SQL identifier and is therefore case-sensitive. When using lowercase characters, enclose *ds-name* in quotes to preserve the lowercase characters. Failure to do so causes ***ds-name*** to be automatically uppercased during command execution, which can cause errors. Enter the default *ds-name* as "TMD_Default_DataSource".

---

evar-name

is the name of the EVAR to be added and cannot be the asterisk (*) wild card. It is an SQL identifier, and so requires double quotes in some cases. When the type is a define, the name can optionally contain a leading equal sign, with or without quotes. The name is always set to uppercase, even when enclosed in quotes.

TYPE *type-name*

is one of the valid types in the table under EVAR Types on page 6-1. The type all is not valid for this command. The default type is control.

`PREFIX` *`string`*

is the portion of the object definition that appears before the name. It is an SQL string literal and so requires single quotes. The default value is determined by the type. See EVAR Types on page 6-1. For consistency, this value is displayed in uppercase letters, even though it is a quoted literal.

`VALUE` *`string`*

is the portion of the object definition that appears after the name. It is an SQL string literal, and so requires single quotes. There is no default. See the *SQL/MX Reference Manual* for name and value syntax details.

`LIMIT` *`number`*

is the threshold value for the monitored attribute for this resource management policy. There is no default. This value must be positive and not exceed the maximum value of the LARGEINT SQL data type of 9,223,372,036,854,775,807. Note that the number entered should have no commas and should appear as 9223372036854775807. Commas are inserted only for clarity.

`ACTION` *`key-val`*

is the desired action when the limit is exceeded for this resource management policy. The valid values are LOG, LOG_WITH_INFO, and STOP (with LOG also). The LOG option is the default.

Considerations—ADD EVAR

- ADD EVAR is a sensitive command.
- The asterisk (*) wild card is not supported for *`evar-name`*.
- ADD EVAR fails if you use the same name in an existing EVAR of the same type.
- The new EVAR is added after existing ones of this type. If order is important, see EVAR Order on page 6-2.
- All SERVICEs can detect the added EVAR on the DS as soon as it is created, but connected servers do not detect it until the current connection is closed.
- You need not stop the DS to execute ADD EVAR.
- Names for some EVARs must be multiword to be unique, so blank spaces are permitted in this name with double quotes. The resulting definition consists of the prefix, name, and value fields, concatenated together in that order.

Example—ADD EVAR

This example adds an EVAR of type `define` with the name GROUP to the DS `TDM_Default_DataSource`. You can use the equal (=) sign in the name, but it is not

required. Use of the specific service $AS is not significant here; the asterisk (*) wild card is less ambiguous.

```
CS>add evar $AS."TDM_Default_DataSource".=GROUP, type define, value 'CLASS MAP,
     FILE $DATA3.PERSNL.GROUP';
-- ADD EVAR \ODBC.*.TDM_Default_DataSource.=GROUP Successful
```

This example adds an EVAR type `rmp` and displays it. The EVAR is named "ESTIMATED COST" in uppercase and enclosed in double quotes to protect the space character. Entering the name in lowercase is permitted, but all characters are automatically converted to uppercase.

In this example, the service name and data source name are defaulted to those used in the preceding example:

```
CS>add evar "ESTIMATED COST", type rmp, limit 5000, action log_with_info;
-- ADD EVAR \ODBC.$AS.TDM_DEFAULT_DATASOURCE.ESTIMATED COST Successful

CS>info evar *, type rmp;
-- INFO EVAR \ODBC.$AS.TDM_DEFAULT_DATASOURCE.* Report
Resource Management Policies:
ESTIMATED COST       Limit: 5000     Action: LOG_WITH_INFO
```

This example is a set of `adds` showing at least one `add` for each EVAR type. The full object name in the first command is the default for subsequent commands. Each command produces a confirmation (not shown).

Note. Single and double quotes are required where shown.

```
>>mode mxcs;
CS>ADD EVAR $AS."TDM_Default_DataSource".EMPLOYEE, type define, value
'CLASS MAP, FILE $DATA01.PERSNL.EMPLOYEE';
CS>ADD EVAR DEPT, type define, value 'CLASS MAP, FILE $DATA02.PERSNL.DEPT';
CS>ADD EVAR HIST_NO_STATS_ROWCOUNT, value '123456789';
CS>ADD EVAR HIST_NO_STATS_UEC, value '99';
CS>ADD EVAR QUERY_CACHE, value '10000';
CS>ADD EVAR QUERY_CACHE_MAX_VICTIMS, value '100';
CS>ADD EVAR QUERY_CACHE_STATISTICS, value 'ON';
CS>ADD EVAR QUERY_CACHE_STATISTICS_FILE, value '/home/testcachefiles/myStats';
CS>ADD EVAR TIMEOUT, value 'RESET';
CS>ADD EVAR SCRATCH_DISKS, value '$data01, $data02, \testsys.$data03';
CS>ADD EVAR "* TIMEOUT", prefix 'control table', value '3000';
CS>ADD EVAR "PERSNL.JOB MDAM", prefix 'control table', value 'OFF';
CS>ADD EVAR "ESTIMATED COST", type rmp, limit 123456, action STOP;
CS>ADD EVAR CATALOG, type set, value 'TANDEM_SYSTEM_NSK';
CS>ADD EVAR SCHEMA, type set, value 'PUBLIC_ACCESS_SCHEMA';
CS>ADD EVAR MPLOC, type set, value '$DATA03.PERSNL';
CS>mode sql;
>>
```

ALTER EVAR

This command modifies one existing EVAR in the DS definition specified.

Command syntax is:.

```
ALTER EVAR [obj-name][, TYPE type-name][, PREFIX string], VALUE string;
```

or

```
ALTER EVAR [obj-name], TYPE RMP, LIMIT number[, ACTION key-val];
```

obj-name

[[[[*system-name.*][*$service-name* | *].]*ds-name.*]*evar-name*]

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server that handles or processes the command or the asterisk (*) wild card. In either case, the EVAR is altered on this DS for all servers.

ds-name

is the name of the DS owning the EVAR and cannot be the asterisk (*) wild card.

Note. *ds-name* is an SQL identifier and is therefore case-sensitive. When using lowercase characters, enclose *ds-name* in quotes to preserve the lowercase characters. Failure to do so causes ***ds-name*** to be automatically uppercased during command execution, which can cause errors. Enter the default *ds-name* as "TMD_Default_DataSource".

---

evar-name

is the name of the EVAR to be altered and cannot be the asterisk (*) wild card. It is an SQL identifier, and so requires double quotes in some cases. When the type is a define, the name can optionally contain a leading equal sign, with or without quote. The name is always set to uppercase, even when enclosed in quotes.

TYPE *type-name*

is one of the valid types from the table in EVAR Types on page 6-1. The type all is not valid for this command. The default type is control.

`PREFIX` *`string`*

is the portion of the object definition that appears before the name. It is an SQL string literal, and so requires single quotes. The default value is determined by the type. See EVAR Types on page 6-1. For consistency, this value is stored in uppercase letters, even though it is a quoted literal.

`VALUE` *`string`*

is the portion of the object definition that appears after the name. It is an SQL string literal, and so requires single quotes. There is no default. See the *SQL/MX Reference Manual* for name and value syntax details.

`LIMIT` *`number`*

is the threshold value for the monitored attribute for this resource management policy. There is no default. This value must be positive and not exceed the maximum value of the LARGEINT SQL data type of 9,223,372,036,854,775,807. The number entered should have no commas and should appear as 9223372036854775807. Commas are inserted only for clarity

`ACTION` *`key-val`*

is the desired action when the limit is exceeded for this resource management policy. The valid values are LOG, LOG_WITH_INFO, and STOP (with LOG also). The LOG option is the default.

Considerations—ALTER EVAR

- ALTER EVAR is a sensitive command.
- The asterisk (*) wild card is not supported for *`evar-name`*.
- The order of the EVARs is not changed. Use the INFO EVAR to determine the order in which the environment variables can be applied when used.
- ALTER EVAR changes the first EVAR it finds with a prefix and name match. It then scans for additional matches and displays warnings because subsequent names can be used at run time instead of the first name.
- ALTER EVAR rebuilds the object from the data provided in the command.
- Syntax and use of this command are similar to the ADD EVAR command. But unlike ADD EVAR, which places the new EVAR at the end of the list, ALTER EVAR replaces the object where it originally appeared in the list.
- Names for some EVARs must be multiword to be unique, so blank spaces are permitted in this name with double quotes. The resulting definition consists of the prefix, name, and value fields, concatenated together in that order.
- The existing values for PREFIX, LIMIT, and ACTION become the defaults, but VALUE does not default.

- Omitting all parameters causes an error.
- All SERVICEs can detect this changed EVAR as soon as it is updated on the disk, but connected servers do not detect it until the current connection is closed.
- You need not stop the DS to execute ALTER EVAR.

Example—ALTER EVAR

This example modifies an EVAR of type `define,` with the name GROUP in the DS `TDM_Default_DataSource`. You can use the equal (=) sign in the name, but it is not required.

```
CS>alter evar *."TDM_Default_DataSource".GROUP, type define,
   value 'CLASS MAP, FILE $DATA55.PERSNL.GROUP';
   -- ALTER EVAR \ODBC.*.TDM_Default_DataSource.=GROUP Successful
```

DELETE EVAR

This command removes the named EVAR from the specified DS definition.

Command syntax is:

```
DELETE EVAR [obj-name][, TYPE type-name];
```

obj-name

```
[[[[\system-name.][$service-name | *].]ds-name.]evar-name | *]
```

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server that handles or processes the command or the asterisk (*) wild card. In either case, the EVAR is deleted on this DS for all servers.

ds-name

is the name of the DS owning the EVAR and cannot be the asterisk (*) wild card.

Note. *ds-name* is an SQL identifier and is therefore case-sensitive. When using lowercase characters, enclose *ds-name* in quotes to preserve the lowercase characters. Failure to do so causes ***ds-name*** to be automatically uppercased during command execution, which can cause errors. Enter the default *ds-name* as `"TMD_Default_DataSource"`.

---

evar-name

is the name of the EVAR to be deleted and can be the asterisk (*) wild card. If the EVAR name is an asterisk (*) and the type is `all`, all EVARs under that DS are deleted. If the EVAR name is an asterisk (*) and a type other than `all` is specified, EVARs of that type are deleted from the DS. If the EVAR name is not an asterisk (*), the type cannot be `all`, and only that EVAR is deleted. The EVAR name is an SQL identifier and requires double quotes in some cases. When the type is a `define`, the name can optionally contain a leading equal sign, with or without quotes. The name is always set to uppercase, even when enclosed in quotes.

TYPE *type-name*

is one of the valid types in the table under EVAR Types on page 6-1.

Considerations—DELETE EVAR

- DELETE EVAR is a sensitive command.
- The asterisk (*) wild card is supported for *evar-name*.
- DELETE EVAR deletes the first EVAR it finds with a name match and then scans for additional matches and displays warnings if found.
- All SERVICEs remove this deleted EVAR as soon as it is removed from the database, but connected servers do not remove it until the current connection is closed.
- You need not stop the DS to execute DELETE EVAR.

Example—DELETE EVAR

This example deletes an EVAR of type `define`, with the name GROUP from the DS `TDM_Default_DataSource`. You can use the equal (=) sign in the name, but it is not required.

```
CS>delete evar *."TDM_Default_DataSource".GROUP, type define;
-- DELETE EVAR \ODBC.*.TDM_Default_DataSource.=GROUP Successful
```

INFO EVAR

This command displays the named EVAR from the DS definition specified.

Command syntax is:

```
INFO EVAR [obj-name][, TYPE type-name];
```

obj-name

```
[[[[\system-name.][$service-name | *].]ds-name.]evar-name | *]
```

system-name

is the name of the current system of up to 16 CPUs with the same Expand address and where the target MXCS process is running. *system-name* and *service-name* identify one service instance.

service-name

is the name of the MXCS association server that handles or processes the command or the asterisk (*) wild card.

ds-name

is the name of the DS owning the EVAR and cannot be the asterisk (*) wild card.

Note. *ds-name* is an SQL identifier and is therefore case-sensitive. When using lowercase characters, enclose *ds-name* in quotes to preserve the lowercase characters. Failure to do so causes ***ds-name*** to be automatically uppercased during command execution, which can cause errors. Enter the default *ds-name* as `"TMD_Default_DataSource"`.

---

evar-name

Is the name of the EVAR to be displayed. It can be the asterisk (*) wild card. If the EVAR name is an asterisk (*) and the type is `all`, the EVARs for this DS are displayed. If the EVAR name is an asterisk (*) and a type other than *all* is specified, the EVARs of that type for this DS are displayed. If the EVAR name is specified and a type other than `all` is specified, that EVAR for that DS is displayed. The EVAR name is an SQL identifier and requires double quotes in some cases. When the type is a `define`, the name can optionally contain a leading equal sign, with or without quotes. The name is always set to uppercase, even when enclosed in quotes.

TYPE *type-name*

is one of the valid types from the table in EVAR Types on page 6-1. The default type is `control`.

Considerations—INFO EVAR

- INFO EVAR is not a sensitive command.
- The asterisk (*) wild card is supported for *evar-name*.
- INFO EVAR displays every EVAR with a name match. If more than one match is found, no warning is generated.
- INFO EVAR displays the EVARs in the order that they are applied.
- When an EVAR does not fit on a 79-character line, it is folded at the most reasonable blank space (if possible), with subsequent lines indented four spaces.

Example—INFO EVAR

This example displays an EVAR of type `define,` with the name GROUP in the DS `TDM_Default_DataSource`. You can use the equal (=) sign in the name, but it is not required.

```
CS>info evar *."TDM_Default_DataSource".GROUP, type define;
-- INFO EVAR \ODBC.*.TDM_Default_DataSource.=GROUP Report
ADD DEFINE =GROUP, CLASS MAP, FILE $DATA03.PERSNL.GROUP
```

This example uses wild cards and is type `all`. This is the default DS, and it has no EVARs defined.

```
CS>info evar *."TDM_Default_DataSource".*, type all;
-- INFO EVAR \ODBC.*.TDM_Default_DataSource.* Report
Resource Management Policies:
None

Defines:
None

Controls:
None

Sets:
None
```

This example uses wild cards and the type is `all`. This DS has some of each EVAR defined. Note the control table EVARS:

```
CS>info evar *."TDM_Default_DataSource".*, type all;
-- INFO EVAR \ODBC.*.TDM_Default_DataSource.* Report
Resource Management Policies:
ESTIMATED COST   Limit: 123456   Action: STOP

Defines:
ADD DEFINE =EMPLOYEE, CLASS MAP, FILE $DATA01.PERSNL.EMPLOYEE
ADD DEFINE =DEPT, CLASS MAP, FILE $DATA02.PERSNL.DEPT

Controls:
CONTROL QUERY DEFAULT HIST_NO_STATS_ROWCOUNT '123456789'
```

```
CONTROL QUERY DEFAULT HIST_NO_STATS_UEC '99'
CONTROL TABLE * TIMEOUT '3000'
CONTROL TABLE PERSNL.JOB MDAM 'OFF'

Sets:
SET CATALOG TANDEM_SYSTEM_NSK
SET SCHEMA PUBLIC_ACCESS_SCHEMA
```

This example uses a compound name. It assumes the same DS as the last example and uses the default type:

```
CS>info evar "PERSNL.JOB MDAM";
-- INFO EVAR \ODBC.*.TDM_Default_DataSource.PERSNL.JOB MDAM Report
CONTROL TABLE PERSNL.JOB MDAM 'OFF'
```

7 Object User Commands

This section describes the MACL commands that are available to an MXCS administrator for the user object (see Valid Command and Object Type Pairs on page 2-3):

Only the NonStop OS super ID user SUPER.SUPER, or its alias, is registered in the MXCS configuration database with the `grant` permission in addition to the `operator` permission. See Security on page 2-10

The user object assigns access permission for an administrator of the MXCS configuration database: `User` or `Operator`. The permission of `User` maps internally to TABLE READ. The permission of `Operator` maps to TABLE READ and TABLE WRITE. This object does not support `grant` permission.

Format of the user's name within the commands for the user object is the same as the NonStop OS group and name format: *`groupname.membername`*, with these considerations:

- You can default both name components.
- If you default only *`groupname`*, provide the leading dot.
- You do not have to add quote marks.
- Names are not case-sensitive in input. However, command execution sets the name to uppercase even if it is enclosed in quotes.

The two default user names and permissions are:

- PUBLIC with `User` permissions
- SUPER.SUPER with `Operator` permissions

You cannot remove or change these default user names. The PUBLIC entry gives everyone `User` permissions. Use the ADD USER command to assign `Operator` permissions to a few users.

ADD USER

This command adds one user name and permissions to the MXCS configuration database access list. Command syntax is:

```
ADD USER [obj-name][, PERM perm-val];
```

obj-name

[[*system-name.*]*groupname.membername*]

system-name

is the name of a group of up to 16 CPUs with the same Expand address. *system name* and *service-name* identify one instance.

groupname

is the NonStop OS group logon name.

membername

is the NonStop OS user logon name with no aliases permitted.

PERM *perm-val*

is the permission value of `Oper`, `Operator`, or `User` assigned to the named user. The permission keyword `Operator` is equivalent to `Oper` and is the default.

Considerations—ADD USER

- ADD USER is a sensitive command.
- The asterisk (*) wild card is not supported.
- Only the user SUPER.SUPER or an alias can use this command.
- To change the permission of users already assigned a permission, use the ALTER USER command.
- ADD USER command terminates with an error if the user is already defined.

Example—ADD USER

This example adds the named user and gives database permissions to execute any command:

```
CS>add user grouptst.george, perm oper;
-- ADD USER \ODBC.GROUPTST.GEORGE Successful
```

ALTER USER

This command changes one user's permission in the MXCS configuration database access list. Command syntax is:

```
ALTER USER [obj-name][, PERM perm-val];
```

obj-name

[[*system-name.*]*groupname.membername*]

system-name

is the name of a group of up to 16 CPUs with the same Expand address. *system name* and *service-name* identify one instance.

groupname

is the NonStop OS group logon name.

membername

is the NonStop OS user logon name with no aliases permitted.

PERM *perm-value*

is the permission value to be assigned this user. The default is either `User` or `Operator` depending on what is already assigned. No error is reported if the explicate permission is the same as the current permission, and no change occurs. The permission keyword `Operator` is equivalent to `Oper`.

Considerations—ALTER USER

- ALTER USER is a sensitive command.
- The asterisk (*) wild card is not supported.
- Only the SUPER.SUPER user can execute this command.
- SUPER.SUPER and DEFAULT users cannot be altered.

Example—ALTER USER

This example changes the database permission for the named user to the other value:

```
CS>alter user grouptst.george;
-- ALTER USER \ODBC.GROUPTST.GEORGE Successful
```

DELETE USER

This command deletes one user name and permissions from the MXCS configuration database access list. Command syntax is:

```
DELETE USER [obj-name];
```

obj-name

[[*system-name.*]*groupname.membername*]

system-name

is the name of a group of up to 16 CPUs with the same Expand address. *system name* and *service-name* identify one instance.

groupname

is the NonStop OS group logon name.

membername

is the NonStop OS user logon name with no aliases permitted.

Considerations—DELETE USER

- DELETE USER is a sensitive command.
- The asterisk (*) wild card is not supported.
- Only the user SUPER.SUPER can use this command.
- SUPER.SUPER and DEFAULT users cannot be deleted.

Example—DELETE USER

This example removes the named user from database permissions:

```
CS>delete user grouptst.george;
-- DELETE USER \ODBC.GROUPTST.GEORGE Successful
```

INFO USER

This command displays user names and permissions to the MXCS configuration database access list. It allows display of permissions for one user, all users in one group, or all users with permissions defined. Command syntax is:

```
INFO USER [obj-name];
```

obj-name

[[*system-name.*][*groupname* | *].[*membername* | *]]

system-name

is the name of a group of up to 16 CPUs with the same Expand address. *system name* and *service-name* identify one instance.

groupname

is the NonStop OS group logon name.

membername

is the NonStop OS user logon name with no aliases permitted.

Considerations—INFO USER

- INFO USER is not a sensitive command.
- Can be used by anyone to display which users have `Oper` permissions to the MXCS configuration database.
- *groupname* can be the asterisk (*) wild card only when *membername* is also the wild card.

Example—INFO USER

This example first displays the named user's database permissions and then all users with permissions:

```
CS>info user grouptst.george;
-- INFO USER \ODBC.GROUPTST.GEORGE Report

GROUPTST.GEORGE     Oper

>CS>info user *.*;
-- INFO USER \ODBC.*.* Report

1234567891....^....2....^....3....^....4....^....5....^....6....^....7....^....8
PUBLIC............User     SUPER.SUPER........Oper     GROUPTST.SUSAN.....Oper
GROUPTST.TOM.......Oper     GROUPTST.GEORGE....User     GROUPTST.MARY......Oper
```

A

Example Commands and Objects

The appendix provides examples to:

- Start an MXCS ODBC service named $AS
- Create and start a DS named TEST_DSN
- Review the resulting MXCS configuration
- Stop and delete the DS
- Stop the service
- Delete the service

Because example object names are not case-sensitive, they appear in lowercase. However, they appear in uppercase in reports. To protect the case of a DS name, enclose it in double quotes (“). Service names are automatically translated to uppercase.

Commands in these examples use the defaulting feature for component names, where component names are retained from previous commands.

These MXCI commands appear in the order that they appear in this example.

- `mode mxcs;`
- `info service $as;`
- `add ds Test_DSN, ... ;`
- `add evar "ESTIMATED COST", ... ;`
- `add evar timeout, ... ;`
- `add evar sratch_disks, ... ;` Misspelled EVAR name to demonstrate an error
- `add evar scratch_disks, ... ;`
- `add evar EMP, type define, ... ;`
- `info ds, detail;`
- `info evar *, ... ;`
- `info ds *;`
- `start service;`
- `start ds Test_DSN;`
- `info service, detail;`
- `info ds, detail;`
- `info server *;`
- `info server *, ... ;`
- `stop ds, ... ;`
- `delete ds;`
- `stop service $as, ... ;`

This is an example of running the commands and messages that result. Operator permissions are required to execute `add`, `start`, `stop`, and `delete`, so it is assumed the user has this permission. No security permissions are necessary to run `info`.

This example starts the service object $AS using the MXOAS command at the TACL prompt, where `-i` indicates that the service enters the stopped state after initialization (no servers start until the DS and service are started):

```
volume $system.zmxodbc
run mxoas /name $as, nowait, term $zhome/ -tcp $ztc0 -i -pn 21000
```

This example starts MXCI and verifies that the service started properly:

```
mxci
Hewlett-Packard NonStop(TM) SQL/MX Conversational Interface 2.0
(c) Copyright 2003 Hewlett-Packard Development Company, LP.
>>mode mxcs;
CS>info service $as;
-- INFO SERVICE \ODBC.$AS Report

PName    State        SPid        TCP/IP PN     Port      Last Updated    Type
$AS      STOP         1,386       $ZTC0         21000     Mar 26 13:37     AS
$Z3445   AVL          1,394       $ZTC0         21001     Mar 26 13:37     CFG
```

This example creates the DS and uses options that are not optimal for normal use. For example, its `CpuList` option limits the servers to running on CPU 0 and 1. It also starts statistic recording that can reduce performance, fill the logs, and disable idle server and idle connection timeouts. After starting the servers, they are persistent and do not end. You can configure the DS without starting a service by using an asterisk (*) for the service name, which connects it directly to the database. Regardless of how the DS is created, all existing services immediately see the DS. This DS is created in the stopped state:

```
CS>add ds Test_DSN, MaxServer 5, IdleServer 3, InitServer 2, IdleTimeout
+>NO_TIMEOUT, ConnTimeout NO_TIMEOUT, InitPri 140, CpuList '0 1',
+>SQLExecDirectStat, SQLFetchStat, SQLPrepareStat;
-- ADD DS \ODBC.$AS.TEST_DSN Successful
```

This example adds EVARs to the DS definition:

```
CS>add evar "ESTIMATED COST", type rmp, limit 500, action stop;
-- ADD EVAR \ODBC.$AS.TEST_DSN.ESTIMATED COST Successful
CS>add evar timeout, value 'reset';
-- ADD EVAR \ODBC.$AS.TEST_DSN.TIMEOUT Successful
```

This command produces an error because of a misspelled EVAR name:

```
CS>add evar sratch_disks, value '$data01, $data02, \ODBC.$data03';
-- ADD EVAR \ODBC.$AS.TEST_DSN.SRATCH_DISKS Failed

*** ERROR[15408] Lower layers failed to perform action requested.

*** ERROR[15310] SUBSYSTEM: SQL error from LowLayer, library call
    CFGSetEnvironmentValues failed.

*** ERROR[15202] LOWLAYER 1: *** ERROR[2050] SRATCH_DISKS is not the name of any
    DEFAULTS table attribute.*** ERROR[8822] Unable to
    prepare the statement.
```

These examples complete adding EVARs and displaying the results:

```
CS>add evar scratch_disks, value '$data01, $data02, \ODBC.$data03';
-- ADD EVAR \ODBC.$AS.TEST_DSN.SCRATCH_DISKS Successful
CS>add evar EMP, type define, value 'CLASS MAP, FILE $DATA01.PERSNL.EMPLOYEE';
-- ADD EVAR \ODBC.$AS.TEST_DSN.=EMP Successful
CS>info ds, detail;
-- INFO DS \ODBC.$AS.TEST_DSN Report, detailed
Name: \ODBC.$AS.TEST_DSN
CpuList: 0,1
InitPri.........................140     CurrentState.................STOPPED
ConnectedServers.....................0     AvailableServers.....................0
LastStateChg.............Dec 31 18:00     LastUpdate..............Mar 26 13:39
IdleServer.........................3     IdleTimeout................NO_TIMEOUT
MaxServer..........................5     ConnTimeout................NO_TIMEOUT
InitServer.........................2     StartAutomatic.....................OFF
Trace.............................OFF     SQLPrepareStat.....................ON
ConnInfoStat......................OFF     SQLExecuteStat.....................OFF
```

```
SessionInfoStat..................OFF     SQLExecDirectStat.................ON
SQLStmtStat.....................OFF     SQLFetchStat......................ON

CS>info evar *, type all;
-- INFO EVAR \ODBC.$AS.TEST_DSN.* Report
Resource Management Policies:
ESTIMATED COST       Limit: 500     Action: STOP

Defines:
ADD DEFINE =EMP, CLASS MAP, FILE $DATA01.PERSNL.EMPLOYEE

Controls:
CONTROL QUERY DEFAULT TIMEOUT 'RESET'
CONTROL QUERY DEFAULT SCRATCH_DISKS '$data01, $data02, \ODBC.$data03'

Sets:
None
```

This example displays all DSs on the system:

```
CS>info ds *;
-- INFO DS \ODBC.$AS.* Begun
                                    Start   Server Limits   Server Usage
Name                          State Auto    Max Init Idle   Con  Avl  Tot
TDM_Default_DataSource        STARTED   ON     5    1    1     0    1    1
TEST_DSN                      STOPPED   OFF    5    2    3     0    0    0
-- Cmd processed 2 objects, 0 failed.
```

This example starts the service and the DS in the service $AS:

```
CS>start service;
-- START SERVICE \ODBC.$AS Successful
CS>start ds Test_DSN;
-- START DS \ODBC.$AS.TEST_DSN Successful
```

This example verifies the results of starting the service and DS. Because servers are not connected to clients yet, all servers are in shown the AVL state, and all connected client information is listed as `Not Avail` in the displays.

```
CS>info service, detail;
-- INFO SERVICE \ODBC.$AS Report

PName    State        SPid        TCP/IP PN     Port      Last Updated    Type
$AS      AVL          1,386       $ZTC0         21000     Mar 26 13:49     AS
$Z3445   AVL          1,394       $ZTC0         21001     Mar 26 13:37     CFG

DS Name (active only)                      DS State   Svr: Con     Avl     Tot
TDM_Default_DataSource                     STARTED           0       1       1
TEST_DSN                                   STARTED           0       3       3
--------------------                                      -----   -----   -----
Totals For Service                                            0       4       4
CS>info ds, detail;
-- INFO DS \ODBC.$AS.TEST_DSN Report, detailed
Name: \ODBC.$AS.TEST_DSN
CpuList: 0,1
InitPri.........................140     CurrentState..................STARTED
ConnectedServers.....................0     AvailableServers...................3
LastStateChg.............Mar 26 14:07     LastUpdate...............Mar 26 13:39
IdleServer.........................3     IdleTimeout................NO_TIMEOUT
MaxServer..........................5     ConnTimeout................NO_TIMEOUT
InitServer.........................2     StartAutomatic.....................OFF
Trace............................OFF     SQLPrepareStat.....................ON
ConnInfoStat.....................OFF     SQLExecuteStat.....................OFF
SessionInfoStat..................OFF     SQLExecDirectStat..................ON
SQLStmtStat......................OFF     SQLFetchStat.......................ON

CS>info server *;
-- INFO SERVER \ODBC.$AS.* Report

DS Name: TDM_Default_DataSource
PName    State     SPid  TCP/IP PN  Port   Last State Chg    Remote System Name
$Z3471   AVL       1,375   $ZTC0    21002  Mar 26 13:49

DS Name: TEST_DSN
PName    State     SPid  TCP/IP PN  Port   Last State Chg    Remote System Name
```

```
$Z3503   AVL       0,388   $ZTC0    21007  Mar 26 14:07
$Z3504   AVL       1,381   $ZTC0    21008  Mar 26 14:07
$Z3505   AVL       0,414   $ZTC0    21009  Mar 26 14:07
-- Cmd processed 4 objects, 0 failed.

CS>info server *, detail, ds Test_DSN;
-- INFO SERVER \ODBC.$AS.* Report, DETAIL

DS Name: TEST_DSN
PName    State     SPid  TCP/IP PN  Port   Last State Chg    Remote System Name
$Z3503   AVL       0,388   $ZTC0    21007  Mar 26 14:07
  Cl User ID.................Not Avail     Cl App Name................Not Avail
  Cl Proc ID.................Not Avail

DS Name: TEST_DSN
PName    State     SPid  TCP/IP PN  Port   Last State Chg    Remote System Name
$Z3504   AVL       1,381   $ZTC0    21008  Mar 26 14:07
  Cl User ID.................Not Avail     Cl App Name................Not Avail
  Cl Proc ID.................Not Avail

DS Name: TEST_DSN
PName    State     SPid  TCP/IP PN  Port   Last State Chg    Remote System Name
$Z3505   AVL       0,414   $ZTC0    21009  Mar 26 14:07
  Cl User ID.................Not Avail     Cl App Name................Not Avail
  Cl Proc ID.................Not Avail
-- Cmd processed 3 objects, 0 failed.
```

This example stops the DS:

```
CS>stop ds, after now, reason 'test complete';
-- STOP DS \ODBC.$AS.TEST_DSN Successful
```

This example deletes the DS, stops the service, and exits MXCI:

```
CS>delete ds;
-- DELETE DS \ODBC.$AS.TEST_DSN Successful
CS>stop service $as, after now, reason 'test complete';
-- STOP SERVICE \ODBC.$AS Successful

CS>exit;

End of MXCI Session
```

You cannot delete the service through the MXCI interface. Delete the service at the TACL prompt by entering:

```
stop $as
```

Index

L

M

O

P

R

S

T

U

V

W

Special Characters