

CA-Endevor[®]/DB

Batch Reference

15.0

OS/390



Computer Associates

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

THIS DOCUMENTATION MAY NOT BE COPIED, TRANSFERRED, REPRODUCED, DISCLOSED, OR DUPLICATED, IN WHOLE OR IN PART, WITHOUT THE PRIOR WRITTEN CONSENT OF CA. THIS DOCUMENTATION IS PROPRIETARY INFORMATION OF CA AND PROTECTED BY THE COPYRIGHT LAWS OF THE UNITED STATES AND INTERNATIONAL TREATIES.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

THE USE OF ANY PRODUCT REFERENCED IN THIS DOCUMENTATION AND THIS DOCUMENTATION IS GOVERNED BY THE END USER'S APPLICABLE LICENSE AGREEMENT.

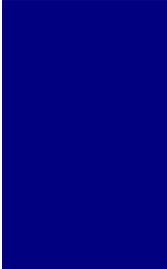
The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227.7013(c)(1)(ii) or applicable successor provisions.

First Edition, December 2000

© 2000 Computer Associates International, Inc.
One Computer Associates Plaza, Islandia, NY 11749
All rights reserved.

All trademarks, trade names, service marks, or logos referenced herein belong to their respective companies.



Contents

Preface

About This Guide	0-5
Related Documentation	0-5

Chapter 1: Overview

What Is CA-Endevor/DB?	1-1
What Is CA-Endevor/DB Batch?	1-2
Batch Features	1-2
Batch Coding Conventions	1-5

Chapter 2: Signon and Set Options Commands

SIGNON	2-2
SET OPTIONS	2-5

Chapter 3: Batch Syntax

Overview	3-1
<i>DISPLAY/PUNCH CCDB-entity-type</i>	3-2
CCID	3-5
DICTIONARY	3-9
ENTITY	3-15
MANAGEMENT GROUP	3-20
PREAUTHORIZATION	3-23
SECURITY CLASS	3-28
SIGNIN	3-40
SIGNOUT	3-43
STATUS	3-46
USER	3-49

Chapter 4: PUNCH Mode

What Is PUNCH Mode?	4-1
Why Use PUNCH Mode?	4-1
PUNCH Mode Syntax	4-2

Chapter 5: Batch Execution JCL

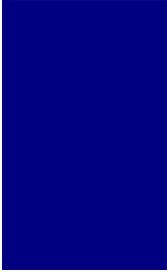
Introduction	5-1
OS/390 JCL	5-1

Appendix A: CA-Endevor/DB Batch Reserved Words

CA-Endevor/DB Batch Reserved Words	A-1
--	-----

Appendix B: CA-Endevor/DB Entity Types

Dictionary Entity Types	B-1
CCDB ENTITY TYPES	B-2



Preface

About This Guide

This document describes the CA-Endevor/DB Batch free-form language that allows users to modify their Change Control Databases (CCDBs). It contains the following chapters:

- Chapter 1 – provides an overview to CA-Endevor/DB Batch.
- Chapter 2 – discusses the SIGNON and SET OPTIONS commands.
- Chapter 3 – provides syntax and coding rules for all commands.
- Chapter 4 – discusses PUNCH Mode.
- Chapter 5 – discusses batch execution JCL.
- Appendix A – provides a list of CA-Endevor/DB Batch reserved words.
- Appendix B – provides a list of CA-Endevor/DB entity types.

Related Documentation

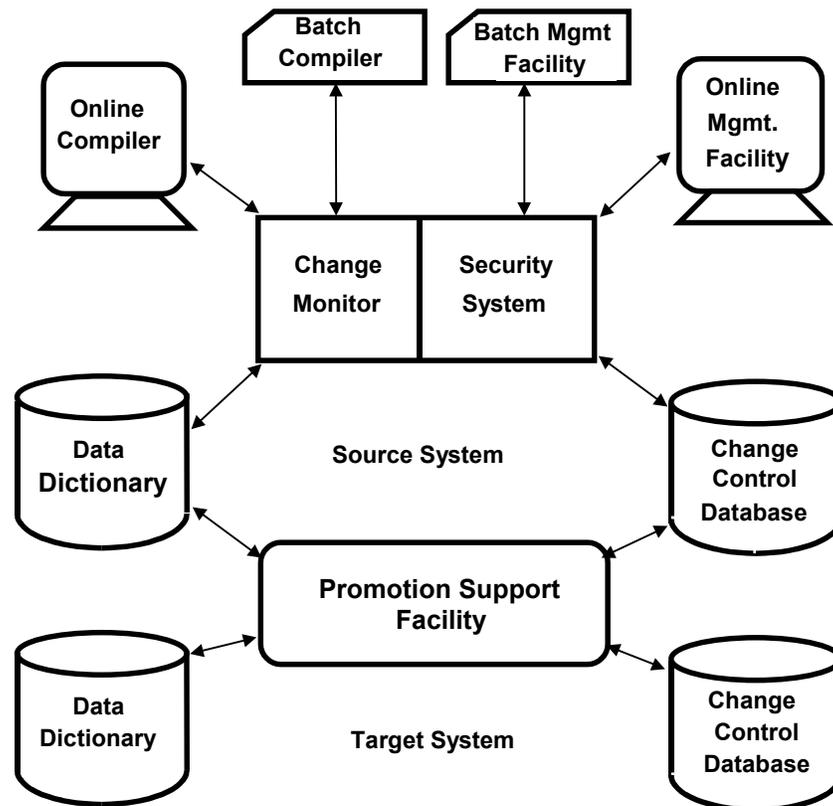
Refer to the following manuals for more information about CA-Endevor/DB:

- *CA-Endevor/DB Concepts and Facilities Guide*
- *CA-Endevor/DB Administrator Guide*
- *CA-Endevor/DB User Guide*
- *CA-Endevor/DB Messages and Codes*
- *CA-IDMS Installation and Maintenance Guide-OS/390*

What Is CA-Endevor/DB?

CA-Endevor/DB controls and monitors change processing within the CA-IDMS environment. The use of CA-Endevor/DB streamlines the administration of the CA-IDMS environment and helps ensure a smooth migration from one stage of development (such as system testing) to the next.

CA-Endevor/DB is fully integrated with CA-IDMS and provides an interface between user requests against the CA-IDMS Integrated Data Dictionary (IDD) and the IDD itself. CA-Endevor/DB operates under OS/390.



The CA-Endevor/DB change monitor/security system:

- Captures each update request made to the IDD (except for requests against entities for which update information is not being captured, as defined by the CA-Endevor/DB Administrator).
- Guarantees that the user making the request is authorized to do so (if access-level security is in effect).
- Documents the change in the CA-Endevor/DB Change Control Database (CCDB), automatically associating it with the user, the entity being modified, and any CA-Endevor/DB management information applicable to the change.
- Passes the update request through to the IDD for processing.

Because CA-Endevor/DB is fully integrated with CA-IDMS, it monitors itself, as well as the IDD. Any changes to the CA-Endevor/DB CCDB are monitored just as carefully as those made to the IDD.

What Is CA-Endevor/DB Batch?

CA-Endevor/DB Batch is a free-form language that enables users to modify their Change Control Data Bases (CCDBs) through the use of English-like commands.

CA-Endevor/DB Batch is the language used for the non-interactive (batch) execution of CA-Endevor/DB. Batch processing is desirable when you have to process large numbers of elements. This eliminates much of the time spent in interactive screen navigation.

This chapter of the *CA-Endevor/DB Batch Reference* explains the benefits of using Batch, and provides an introduction to Batch structure as well as a discussion of the attributes and characteristics contained in Batch syntax.

The features and benefits of using Batch are described below.

Batch Features

The features listed below emphasize those aspects of CA-Endevor/DB Batch that both facilitate and enhance CA-Endevor/DB processing. Using Batch, you can:

- Build command syntax for multiple commands with minimum input.
- Make mass updates.
- Easily clone CCDB information.
- Perform CCDB updates in background.

Processing Flow

When you submit your Batch requests, CA-Endevor/DB follows a specific processing flow to execute the commands.

1. CA-Endevor/DB processes the SIGNON command.
2. For each command, CA-Endevor/DB performs the following command processing:
 - Parses and validates the Batch syntax.
A Syntax Report is produced, echoing the Batch command entered and flagging any syntax errors.
 - When the request has been validated, CA-Endevor/DB checks for errors. If errors exist within the syntax, processing for the command is terminated. If there are no errors, processing continues.
 - CA-Endevor/DB then performs the processing requested by the command.
 - CA-Endevor/DB prints a confirmation message.
3. CA-Endevor/DB produces an End-of-Job (EOJ) report. The next section discusses this report.
4. The system then terminates CA-Endevor/DB processing.

EOJ Report

CA-Endevor/DB produces an End-of-Job (EOJ) report every time it is run. This report describes the commands that were entered and whether the commands were performed or failed.

The EOJ Report records every type of command (for example, ADD ENTITY, MODIFY ENTITY, etc.) that was entered for this particular batch job. Each line of the report reflects the number of commands read, per type, and the number of commands within that type that failed processing. Note the EOJ report below, which was produced for a test batch run.

COMMAND	READ	FAILED
ADD CCID	0000005	0000001
MODIFY ENTITY	0000004	0000001
ADD USER	0000001	0000000
SIGNOUT	0000001	0000000
UNKNOWN	0000001	0000001
TOTAL	0000012	0000003

NDVRMISB: E099 PROCESSING TERMINATED WITH ERRORS

Each line reflects the command type, number of records read for the command type, and number of records for the command type which failed. Statistics are produced only for those command types which are processed. In the top line of the report, for example, you can see that CA-Endevor/DB read 5 ADD CCID command records and that one of them failed. This means that only four ADD CCID commands were processed in the batch run. Similarly, looking at the second line, you can see that CA-Endevor/DB read 4 MODIFY ENTITY commands and one of them failed. In this case, only three MODIFY ENTITY commands were processed. If there are no failures, the FAILED values will be all zeroes.

Note that the UNKNOWN command line lists the number of commands received that CA-Endevor/DB could not identify. The parser looks for certain 'key' words when recording each command. If the parser does not recognize a command, for whatever reason (such as a misspelling), the information is recorded as an UNKNOWN command and is considered failed.

The total number of commands read and failed appears at the end of the report.

Commands and Clauses

This manual refers to the terms *command* and *clause*. For Batch purposes, these terms are defined as follows:

- A *command* begins with a verb (e.g., ADD, MODIFY, DELETE, DISPLAY, or PUNCH) and ends with a period (.). A command consists of one or more clauses, depending on how you code the Batch syntax.
- A *clause* is an individual unit of information within each command (e.g., TYPE = PRIVATE or VERSION = 1). Any number of clauses may be contained within one command.

In the following example:

1. MODIFY USER
2. NAME = JSB
3. PASSWORD = MICKEY
4. SECURITY CLASS = NDVR-GLOBAL
5. COMMENT = "JEFF BUCKSER X2224".

Lines 1-5 form a command. Line 1 begins with a verb (MODIFY) and line 5 ends with a period.

Lines 2-5 are individual clauses. Each of these clauses provides information essential to the command.

Batch commands do not have to be entered in any particular format. For instance, you could also specify the example above as follows:

```
MODIFY USER NAME = JSB PASSWORD = MICKEY  
COMMENT = "JEFF BUCKSTER X2224"  
SECURITY CLASS = NDVR-GLOBAL.
```

Batch Coding Conventions

This section of the chapter details Batch attributes as well as Batch syntax and coding conventions. Although selected information is repeated within the description of each command (Chapter 2), refer to this chapter as often as necessary for a quick review of coding information.

Batch Syntax Conventions

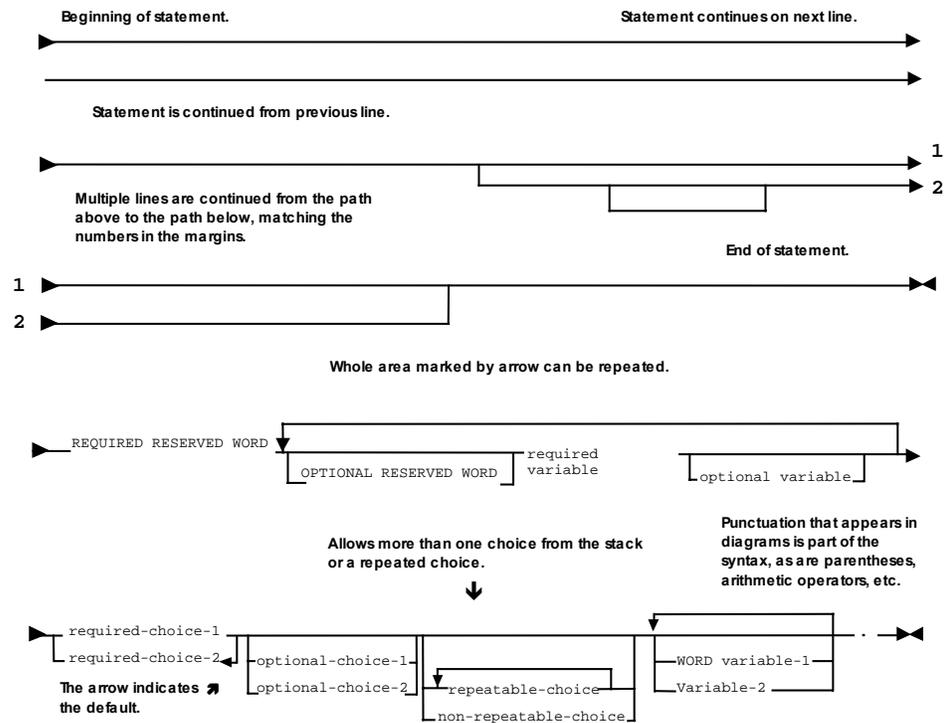
The syntax diagrams are in railroad track format. Here are some rules of thumb:

- Read the diagrams from left to right and from top to bottom.
- The required portion of reserved words is CAPITALIZED. Optional keywords are in lowercase standard. Variables are in *lowercase italic*.
- A required clause rests on the line, an optional clause, below the line. When you encounter a stack of choices, if the top item rests on the line, choose one of the items in the stack. If the top item swings below the line, the stack is optional.
- You must enter a period at the end of every command. Anything entered after the period will not be part of that command. If no period is found, the system generates an error message. In the Batch syntax, a period indicates that you should enter a period at the end of this clause.
- If you enter a value with an embedded period or space, or the value is a keyword, you must enclose that value with quotes. Also note that if you want to enter a value for *version*, you must enter a non-negative integer. If you want to enter a value for *version=**, you must enter a non-negative integer or an asterisk (*).

You cannot use quotation marks in the VERSION field of any commands in the Batch facility.

- Used within a choice stack, an arrow indicates that the value indicated (pointed at) is the default. This value is used in processing the clause if none of the values in the bracket are specified.
- Parenthesis () indicate a compound list. This enables you to specify more than one value for a selection. For example, in the clause below:
SIGNOUT FUNCTIONS (Y Y Y)

The three flags are all set to Y. Any parentheses that are included in a syntax diagram must be coded.



Use of Quotation Marks

You must always use quotation marks (either single or double as explained below) when a user-defined value contains an embedded space, leading digit, or one of the following characters:

- Period (.)
- Comma (,)
- Semi-colon (;)
- Colon (:)
- Equal sign (=)
- Parentheses (or)
- Greater or Less Than signs (< or >)
- Single Quotation mark (')
- Double Quotation mark (")

If the user-defined value contains a double quotation mark ("), the value must be surrounded by single quotation marks ('). Conversely, if the user-defined value contains a single quotation mark, the value must be surrounded by double quotation marks. If you enter a double quotation mark as the first quotation mark, you must enter a double quotation mark as the ending quotation mark. The same rule applies for single quotation marks.

Note: You cannot use quotation marks in the VERSION field of any commands in the CA-Endevor/DB Batch facility.

Use of Equal Sign (=)

You can use an equal sign (=) in place of the literal IS anytime IS is specified in command syntax. For example:

```
ENTITY NAME = entity-name
```

would produce the same results as:

```
ENTITY NAME IS entity-name
```

Field Length

Listed below are the fields requiring user-defined entries and the number of characters allowed for each:

- Entity name - up to **40** characters
- Type - up to **16** characters
- Version - up to **4** digits
- User Name - up to **32** characters
- CCID name - up to **12** characters
- Comment - up to **60** characters
- Status - up to **16** characters
- Security Class - up to **16** characters
- Dictionary name (dictname) - up to **8** characters
- System Name - up to **8** characters
- Estimated Work Completion - **8** characters
- Actual Work Completion - **8** characters
- Password - up to **8** characters
- DBNAME - up to **8** characters
- Management Group - up to **16** characters

Using a Mask Character For All Names Except Entity Names

This section applies to the following types:

- CCID
- Preauthorization
- Status
- Management group
- Security class
- User

You can use an asterisk (*) instead of a literal to specify that all names be considered when performing a command. An asterisk is acceptable for all commands that list an asterisk as acceptable in the Syntax Rules. For example, in the clause:

NAME is *status*

You can enter either a full status name or an "*" for the variable *status*. Entering a status name will specify that particular status for this command. Entering an asterisk will specify all statuses that meet the rest of the selection criteria for this command.

Using a Mask Character For Entity Names

The mask character provides a means of identifying CCDB entities that match specified criteria. You can specify as much or as little of the entity name as you want. The more information you give, the more specific the set of names that are identified.

Use the name mask when you want to enter partial entity names or name segments. For example, if you have a series of entities whose names all start with the characters ABC, you could specify "ABC*" in a command and thus identify all of them.

A name mask enables you to specify that all names, or all names beginning with a particular series of letters, be considered when performing a command. A name mask is only acceptable for the ENTITY NAME value, where applicable clauses are shown with a "-mask" in the syntax. For example, for a MODIFY ENTITY command, in the clause:

ENTity NAME is *entity-name*

You can replace a name-mask with an asterisk (*), an entity name such as "XYX100", or enter a name mask such as "XYZ*".

Using Asterisks

Asterisks (*) are used as mask characters. The information that the system retrieves depends on where you place the asterisk and the additional information you supply. The system returns CCDB object names matching all the criteria you specify in your request.

To use the mask feature, type the beginning portion of the name segment followed immediately by an asterisk. You can place the asterisk in the following positions:

- In place of a literal. For example:

```
ENTITY NAME IS = *
```

In this example, *all* entity names that meet the other specified selection criteria are retrieved.

- As the last non-blank character in the name. For example:

```
ENTITY NAME IS XYZ*
```

In this example, all entity names beginning with the characters "XYZ" are retrieved, if they meet the other specified selection criteria.

Using Asterisks With Segmented Names

Some entity names in the CCDB are segmented. A segmented name is a name composed of the names of multiple dictionary objects. This information is concatenated to form the entity name used in the CCDB. For example, a logical record named CUSTOMER-ORDER defined in a subschema named SALESUB1 which is compiled within schema SALESCHM will appear in the CCDB as "CUSTOMER-ORDER" SALESUB1SALESCHM.

This is one way that you could identify the logical record in a command. Note the spaces after the LOGICAL-RECORD name. There is an easier way to specify segmented names, using "delimiters", which are explained later in this chapter.

The following entity types have segmented names:

- SUBSCHEMA
- SET
- MODULE
- LOGICAL-RECORD

Name Segment Requirements

Each name segment for the four different entity types has a required length. When not using delimiters, you need to pad the name segment with blanks to maintain the required length. The table below shows the number of characters required for each segment.

Entity Type	Name Segment	Length
SUBSCHEMA	Subschema	8
	Schema	8
SET	Set	32*
	Schema	8
MODULE	Module	32
	Language	8
LOGICAL-RECORD	Logical-record	16
	Subschema	8
	Schema	8

* Set name prior to CA-IDMS 12.0 is 16 characters.

Delimiters

When using CA-Endevor/DB, you can use commas or periods as name segment delimiters. The delimiters separate the name segments. For example, to use delimiters when identifying a logical record in the previous example, you would specify:

```
ENTITY NAME = "CUSTOMER-ORDER.SALESUB1.SALESCHM" TYPE = LR
```

You do not use spaces to pad name segments when you are delimiting with commas.

Note: You must specify TYPE = *type* when using delimiters. You cannot specify TYPE = *.

Using Asterisks For Segmented Names

You can use asterisks in different positions within a segmented name. To illustrate these positions, refer to the examples in this section. For these examples, we use entities that have a type of LOGICAL-RECORD. However, the same principles apply to the other three types of entities that have segmented names: SUBSCHEMA, SET, and MODULE.

All the examples in this section refer to the following set of entities of type LOGICAL-RECORD.

REC NO.	ENTITY NAME
1.	CUSTOMER-ORDER.SALESUB1.SALESCHM TYPE = LR
2.	CUSTOMER-ORDER.SALESUB2.SALESCHM TYPE = LR
3.	CUSTOMER-ORDER.SALESUB3.SALESCHM TYPE = LR
4.	SALESPERSON-CUST.SALESUB1.SALESCHM TYPE = LR
5.	SALESPERSON-CUST.SALESUB2.SALESCHM TYPE = LR
6.	SALESPERSON-CUST.SALESUB3.SALESCHM TYPE = LR
7.	SALESPERSON-ORD.SALESUB1.SALESCHM TYPE = LR
8.	SALESPERSON-ORD.SALESUB2.SALESCHM TYPE = LR
9.	SALESPERSON-ORD.SALESUB3.SALESCHM TYPE = LR

The identification of one or more of the entities in this list may be done in several ways. You can specify a whole name, whole name segments, asterisks and commas, or a combination of these.

You can place an asterisk in the following positions:

- As the last, non-blank character in a particular name segment. For example:

ENTITY NAME = "CUSTOMER-ORDER.SALESUB*.SALESCHM" TYPE = LR

This example identifies a specific logical record, CUSTOMER-ORDER, with each subschema. It requests the indicated logical record for all subschemas whose name begins with "SALESUB", but restricts the request to only those subschemas (and thus only those logical records) compiled against the SALESCHM schema.

In the table illustrated above, CA-Endevor/DB would process records 1, 2, and 3.

- As the last non-blank character in the full name. For example:

ENTITY NAME = "SALES*"

This example identifies all logical records whose names start with "SALES", regardless of the schema or subschema.

In the table illustrated above, CA-Endevor/DB would process records 4, 5, 6, 7, 8, and 9.

- In place of a name segment. For example:

SALESPERSON-ORD.*.SALESCHM

This example identifies the SALESPERSON-ORD LOGICAL-RECORD as defined in all subschemas compiled against the schema SALESCHM.

In the table illustrated above, CA-Endevor/DB would process records 7, 8, and 9.

Using Delimiters in Name Marking

You can use commas or periods as a mask placeholder in segmented names. For example, to retrieve all entities of logical-record SALESPERSON-ORD in schema SALESCHM, specify the following:

```
SALESPERSON-ORD, , SALESCHM
```

In this example, the first comma marks the end of the system name, the second comma designates the subschema name, and the name segment SALESCHM designates the schema name. In our example, CA-Endevor/DB would process records 7, 8, and 9.

You can combine the delimiter and asterisk to retrieve an even larger list of entities. For example, to identify the logical record CUSTOMER-ORDER in all subschemas compiled against all schemas whose names started with "SALES", specify:

```
CUSTOMER-ORDER.. SALES*
```

In the table illustrated above, CA-Endevor/DB would process records 1, 2, and 3.

Multiple asterisks are allowed in the clause. However, you can code only one asterisk per name segment.

Signon and Set Options Commands

This chapter describes the SIGNON and SET OPTIONS commands. These two commands operate in a different manner than the rest of the CA-Endevor/DB Batch commands described in this manual. The SIGNON command must be entered at the beginning of a batch job and the SET OPTIONS command designates the mode under which CA-Endevor/DB will operate.

SIGNON Command

The SIGNON command identifies the user, enabling CA-Endevor/DB to access security restrictions for that user. It also determines whether that user is allowed to perform batch processing.

SET OPTIONS Command

CA-Endevor/DB Batch operates in two modes: PROCESS and PUNCH.

- PROCESS mode is the default. It enables you to process CA-Endevor/DB commands against the CCDB. If you want to execute any command in CA-Endevor/DB Batch, you must run in PROCESS mode. Chapter 3 describes in detail all the commands that you can execute in PROCESS mode.
- PUNCH mode enables you to look at what is in a CCDB before you process the contents of the CCDB. In PUNCH mode, CA-Endevor/DB takes the command, builds syntax for all the elements in the CCDB that meet the selection criteria, and then writes the detailed command syntax into a punch data set. Chapter 4 describes in detail all the commands that you can execute in PUNCH mode.

The SET OPTIONS command enables you to choose which mode, PROCESS or PUNCH, in which you want to operate.

Note: To process commands with the verb DISPLAY or PUNCH, you may run in either mode.

SIGNON

Purpose

The assigned userid identifies you to CA-Endevor/DB, enabling the product to access security restrictions for that user. It also determines whether you are allowed to do batch processing. In CA-Endevor/DB Batch, you can process under one signon ID at a time. If specified, it must be the first statement in the batch run. If you want to process under two different userids, you must submit two batch jobs.

When you enter a SIGNON command, you perform three functions:

- You identify the CCDB to be accessed (DICTNAME).
- You identify yourself (your userid).
- You identify the CCID(s) under which you are working.

The security rules that determine what functions you will be allowed to perform are determined by all three pieces of information. (Refer to the Security Class Maintenance chapter of the *CA-Endevor/DB Administrator Guide* for more information on security rules.) If any of the functions that you perform require change logging, then the change log entries are marked with the userid/CCID information.

CA-Endevor/DB "remembers" the CCIDs that you specify from one CA-Endevor/DB Online or CA-Endevor/DB Batch session to the next. If you are working under the same CCID(s) as in your last session, you can omit the CCID clause in this command.

Starting with CA-IDMS Release 12.0, all CA-IDMS processing runs under an assigned CA-IDMS/DC userid. In the case of:

- *Direct CA-IDMS/DC logon*, the userid is taken from the VTAM login.
- *Batch processing*, the userid is taken from the originating interactive session (TSO, CICS), or from the batch job (JOB card USER parameter).

The purpose of the CA-IDMS/DC SIGNON task is no longer to *establish* a CA-IDMS/DC userid, but to *switch* userids - all CA-IDMS/DC processing is now done under a specific userid.

When CA-Endevor/DB is used to monitor dictionary changes by any of these means, it will automatically determine the userid, and attribute the dictionary changes to that userid. It determines your userid at the instant of your first dictionary update, and "remembers" it for as long as you are connected to CA-IDMS. Thus, you can execute CA-IDMS dictionary utilities (ADSC, etc.), and the dictionary updates will automatically be attributed to your userid.

While the automatic usage of your DC userid is often useful, there are several situations where it is not appropriate:

- If your site requires the use of CA-Endevor/DB passwords.
- If you need to switch from working under one (set of) CCID(s) to another.
- If you change DC userids after making changes to a dictionary.

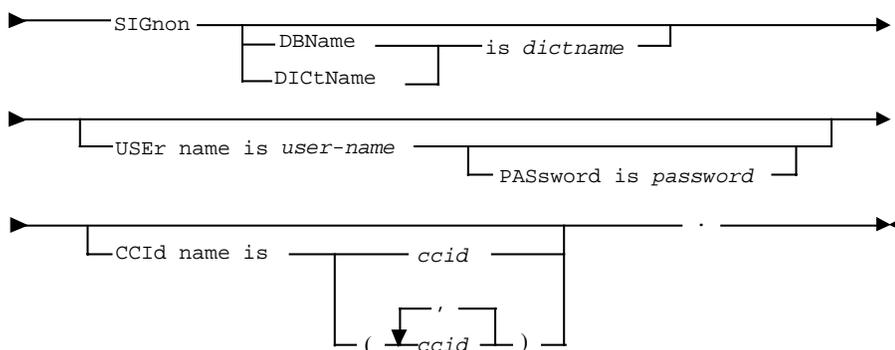
For the preceding cases, you use the SIGNON command to specify your userid.

Processing Flow

When you enter a SIGNON command, CA-Endevor/DB performs the following processing:

- A validation check of the input parameters to ensure that you are authorized to perform batch operations. If you are not, the system prints an error message and disallows the transaction.
 - You have a valid userid, or that one can be automatically created for you.
 - If passwords are required in the CCDB, you have a valid password.
 - If you enter a dictname, it must be valid.
 - If CCIDs are required for your userid, you must enter a valid CCID.
 - If the CCID(s) that you sign on under are PRIVATE, you are preauthorized to sign on to them.
- The system prints a confirmation message informing you that the signon was successfully completed.

Syntax



Syntax Rules

The rules pertaining to each clause in the syntax are listed below. Required and optional clauses are noted, as well as any other requirements specific to this command.

USER name is *user-name*

The 1-32 character userid of the user. It is used to identify the user in the CCDB. If you omit the USER NAME clause, changes that you make will not be attributed to any user in the CCDB.

PASsword is *password*

A 1-8 character password that is stored in the CCDB. If you omit the PASSWORD clause, it will default to spaces, the default password for all users. If you omit this clause and a password is required, you will not be able to sign on and CA-Endevor/DB will generate an error message.

DBName is *dicname*
DICname is *dicname*

The name of the dictionary against which you are processing. If you omit this clause, it defaults to the default dictionary for CA-IDMS.

CCId name is *ccid*

The CCID(s) under which you will be operating. You can sign on under one or more CCIDs, or you can specify NOCCID. If you enter NOCCID, CA-Endevor/DB clears any list of CCIDs from previous sessions and updates that you perform are not assigned to any specified CCIDs.

If you omit this clause, it will default to the last CCID you logged on under. Security rules that apply for that CCID are carried over to this session.

If you do not enter a NAME clause, the CCID clause defaults to "unspecified". If you process under an unspecified CCID clause, the changes that you make will not be logged to a CCID.

If, as a result of security class restrictions, you are in DERIVE CCID mode, any CCID clause you specify is ignored. SIGNON commands act as if you entered CCID = NOCCID.

Examples

The following examples illustrate the use of the SIGNON command.

Syntax Rules

The rules pertaining to each clause in the syntax are listed below. Required and optional clauses are noted, as well as any other requirements specific to this command.

SET options PROcess
SET options PUNCh

This clause will select the mode under which you will process. You can specify:

- **PROCESS** - to execute commands against the CCDB. If you omit the command, CA-Endevor/DB will default to PROCESS mode. Refer to Chapter 3 for a full description of PROCESS mode.
- **PUNCH** - to enter PUNCH mode. Refer to Chapter 4 for a full description of PUNCH mode.

STOP enables you to end the updating of the CCDB if CA-Endevor/DB Batch encounters a critical error. If you want to stop processing when such an error is found, you must enter **STOP** in the SET OPTIONS clause. If you specify **STOP**, and CA-Endevor/DB does stop processing due to a critical error, any remaining batch commands will be checked for syntax errors. If you specify **STOP** in your batch job, it will remain **STOP** until the job ends.

By default, if a critical error is detected, or when a command fails, CA-Endevor/DB Batch processes the rest of the commands in the batch run.

Examples

The following examples illustrate the use of the SET OPTIONS command.

Example 1

```
SET OPTIONS PUNCH.
```

This command will enable you to enter commands in PUNCH mode.

Example 2

```
SET OPTIONS PROCESS STOP.
```

This command will stop processing if CA-Endevor/DB encounters a critical error.

Overview

CA-Endevor/DB provides you with the capability of maintaining the Change Control Database (CCDB) through the use of English-like commands. This chapter describes these commands in detail, providing you with an explanation of command processing, the CA-Endevor/DB Batch syntax, and the coding rules specific to each command.

PROCESS mode causes the CA-Endevor/DB Batch program to update the CCDB for update type verbs. If the DISPLAY or PUNCH verb is specified in a supported command, the Batch processing outputs the contents of the CCDB as if operating in PUNCH mode with an *ADD CCDB-entity-type* request. If a DISPLAY verb is specified, Batch processing displays commands in a card-image format, preceded by "*" on the listing. If a PUNCH verb is specified, Batch processing displays the commands and also outputs the commands into a card image dataset, without "*" identifiers, for subsequent execution. Chapter 3 describes the command behavior when running in PROCESS mode.

The other option is PUNCH mode, which causes the CA-Endevor/DB Batch program to output commands into a card image dataset for subsequent execution. If DISPLAY or PUNCH verbs are specified on a supported command, the PUNCH mode processes these requests the same as described above for PROCESS mode. Chapter 4 describes the command behavior when running in PUNCH mode.

CA-Endevor/DB Batch operates by default in PROCESS mode. If you want to change modes, you can use the SET OPTIONS command, described in Chapter 2.

DISPLAY/PUNCH CCDB-entity-type

Purpose

This verb is supported for CA-Endevor/DB CCDB component types CCID, ENTITY, MANAGEMENT GROUP, STATUS, SECURITY CLASS, and USER.

Use `DISPLAY CCDB-entity-type` to view the contents of one or more CCDB-entity occurrences, in card-image command syntax form preceded by "*" before each line of syntax output to the NDVRLST. DISPLAY requests operate as if executing in PUNCH mode with an `ADD CCDB-entity-type` request.

Use `PUNCH CCDB-entity-type` to report the contents of one or more CCDB-entity occurrences, in card-image command syntax form preceded by "*" before each line of syntax output to the NDVRLST, and to output each line of syntax to the NDVRPCH file, without "*" preceding the syntax. This enables you to view or edit the command syntax. PUNCH requests operate as if executing in PUNCH mode with an `ADD CCDB-entity-type` request.

Processing Flow

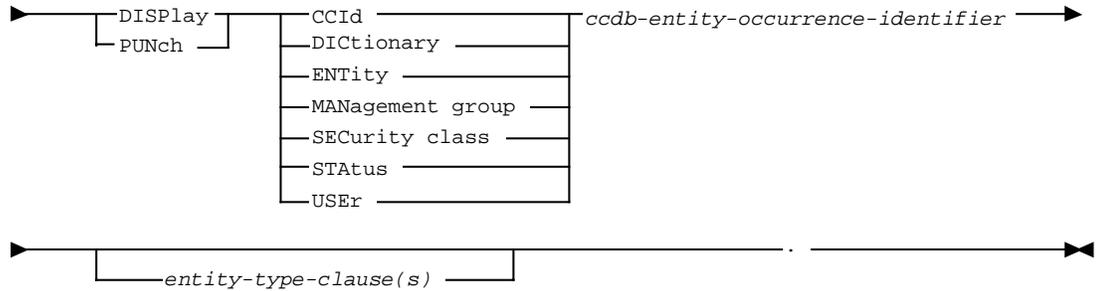
When you enter a `DISPLAY` or `PUNCH CCDB-entity-type` request, CA-Endevor/DB performs the same processing sequence for the specified CCDB-entity-type as if executing under PUNCH mode with an `ADD` command.

- Performs a validation check of the input parameters to ensure that:
 - You are authorized to browse the type of CCDB-entity specified in the request. If you are not, the system prints an error message and disallows the transaction.
 - If you identified a specific CCDB-entity-occurrence, it must exist in the CCDB.
- There are two ways that you can use the `DISPLAY` command:
 - **Specifying CCDB-entity-occurrence identifier clause(s).** If specified, CA-Endevor/DB reports the syntax for that specific occurrence. If you also specified any override clauses, the reported syntax contains the clause(s) you specified.
 - **Entering an asterisk in the CCDB-entity-occurrence identifier clause(s).** If you enter an asterisk in an identifier clause, or omit one or more of the identifier clauses, CA-Endevor/DB reports the syntax for all CCDB-entity occurrences meeting the masking requirements. If you omit all identifier clauses, CA-Endevor/DB reports the syntax for every CCDB-entity-occurrence for the `CCDB-entity-type`.

If you also specified any overriding clauses, the reported syntax contains the clauses you specified.

Refer to Chapter 4 for additional information on the processing flow for an ADD request for the specified CCDB-entity-type.

Syntax



Syntax Rules

For a DISPLAY or PUNCH request, only the verb and CCDB-entity-type are required. All other clauses in the command, including CCDB entity occurrence identifier and entity-type clauses, are the same as documented for ADD requests for the individual CCDB entity types.

Examples

The following examples illustrate the use of the DISPLAY and PUNCH commands.

Example 1

```
DISPLAY ENTITY CUSTOMER TYPE RECORD VERSION 1.
```

This command will produce the following output to the NDVRLST file:

```
*+ ADD ENTITY NAME IS "CUSTOMER"
*+ TYPE IS "RECORD"
*+ VERSION IS 0001
*+ COMMENT IS
*+ "EMPSCHM VER 100 RECORD ENTITY"
*+ STATUS IS "MIGRATE-TEST"
*+ .
```

Example 2

```
PUNCH CCID
COMMENT 'COMMENT OVERRIDE'.
```

This command will produce syntax for each CCID in the CCDB with any COMMENT values in the CCDB replaced by the value specified in the comment clause. The command will write the following to the NDVRLST file:

```
*+ ADD CCID NAME IS "EDB-DCADMIN"  
*+     TYPE IS PUBLIC  
*+     SECURITY CLASS IS "NDVR-GLOBAL"  
*+     COMMENT IS  
*+         "COMMENT OVERRIDE"  
*+     .  
*+ ADD CCID NAME IS "EDB-SYSADMIN"  
*+     TYPE IS PUBLIC  
*+     SECURITY CLASS IS "NDVR-GLOBAL"  
*+     COMMENT IS  
*+         "COMMENT OVERRIDE"  
*+     .
```

The command will also write the following to the NDVRPCH file:

```
ADD CCID NAME IS "EDB-DCADMIN"  
    TYPE IS PUBLIC  
    SECURITY CLASS IS "NDVR-GLOBAL"  
    COMMENT IS  
    "COMMENT OVERRIDE"  
    .  
ADD CCID NAME IS "EDB-SYSADMIN"  
    TYPE IS PUBLIC  
    SECURITY CLASS IS "NDVR-GLOBAL"  
    COMMENT IS  
    "COMMENT OVERRIDE"  
    .
```

CCID

Purpose

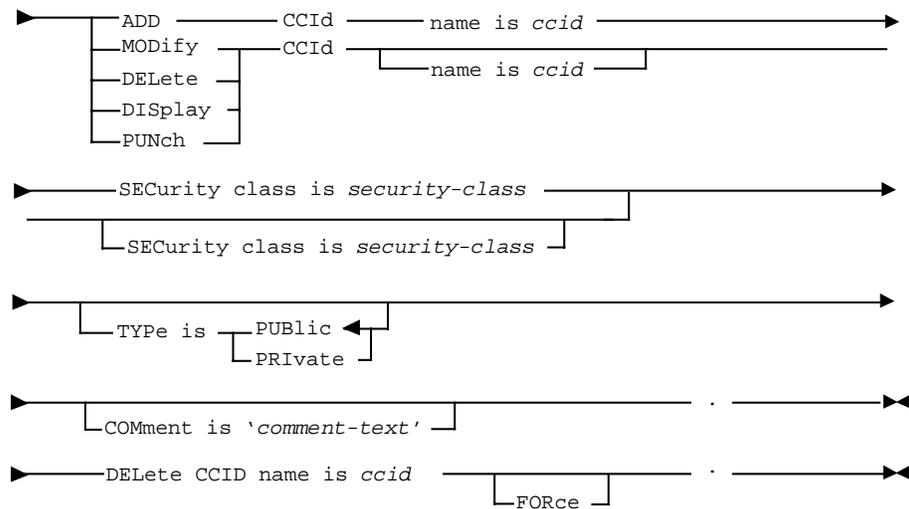
A CCID (Change Control Identifier) categorizes CCDB information for control and reporting purposes. The use of CCIDs is optional. For example, you might have one CCID for each project or, within each project, a separate CCID for design activities, detailed specification activities, implementation activities, and so forth.

Processing Flow

When you enter a CCID command, CA-Endevor/DB performs the following processing:

- Validates the input parameters to insure that:
 - You are authorized to perform the action (ADD, MODIFY, DELETE, or BROWSE) for a CCID. If you are not, the system will print an error message and disallow the transaction.
 - The values specified in the input parameters are valid and all required clauses are specified for the command.
- For ADD actions, insures that the CCID does not already exist in the CCDB; for all other actions, insures that the CCID does exist in the CCDB.
- Performs the requested action.
- Prints a confirmation message that the action was successfully completed.

Syntax



Syntax Rules

The rules pertaining to each clause in the syntax are listed below.

CCID name is *ccid*

Identifies a new CCID to be added to the CCDB or an existing CCID occurrence to be modified, deleted, displayed, or punched. The *ccid* must be a 1 to 12-character alphanumeric value. For ADD actions, this clause is required and must be unique within the CCDB; for DELETE actions, this clause is required and must identify an existing CCID within the CCDB. For all other operations, this clause is optional or may be specified with valid masking specifications as described in Chapter 1.

SECurity class is *security-class*

Identifies the Security Class to be associated with the CCID. The Security Class must already exist in the CCDB. This clause is required for ADD commands.

TYPE is *type*

Indicates whether the CCID is PUBLIC or PRIVATE. If omitted, defaults to PUBLIC.

COMment is *comment-text*

A 1 to 60-character, user-defined remark for the CCID.

FORce

DELETE command only. This clause enables you to delete a CCID even if it has Change Log Entries (CLEs) associated with it.

Usage

DELETE with FORCE

If FORCE is specified on a DELETE command, all Change Log Entries (CLEs) associated with the CCID are also deleted. A CLE is generated automatically anytime an entity is updated. CLEs are associated with a CCID when:

- A user signs on with the CCID then updates a monitored record type.
- A user operating in DERIVE CCID mode updates a monitored record type of which a specific entity has been preauthorized to the CCID with the DERIVE CCID option specified.

Examples

The following examples illustrate the use of CCID commands.

Example 1

```
ADD CCID NAME IS 9900521-0056 SECURITY CLASS IS DEVELOPER
TYPE IS PUBLIC.
```

The CCDB Administrator has added a new CCID to be used by a development team. Any user will be allowed to signon under this CCID.

Example 2

```
MODIFY CCID NAME IS 9900521-0056
TYPE IS PRIVATE
COMMENT "CCID IS NOW PRIVATE".
```

The CCDB Administrator has modified the type clause so that a user must be preauthorized to this CCID in order to use it and has documented the action in the comments.

Example 3

```
MODIFY CCID NAME IS *
TYPE IS PUBLIC
COMMENT "NO SIGNON RESTRICTIONS".
```

This command will change the comment in every public CCID in the CCDB.

Example 4

```
DELETE CCID NAME IS 9900521-0056 FORCE.
```

The CCDB Administrator has decided to delete the CCID and to force all CLEs associated with the CCID to also be deleted.

DICTIONARY

Purpose

The Dictionary Descriptor in the CCDB contains the global characteristics of the CCDB and is automatically created by CA-Endevor/DB.

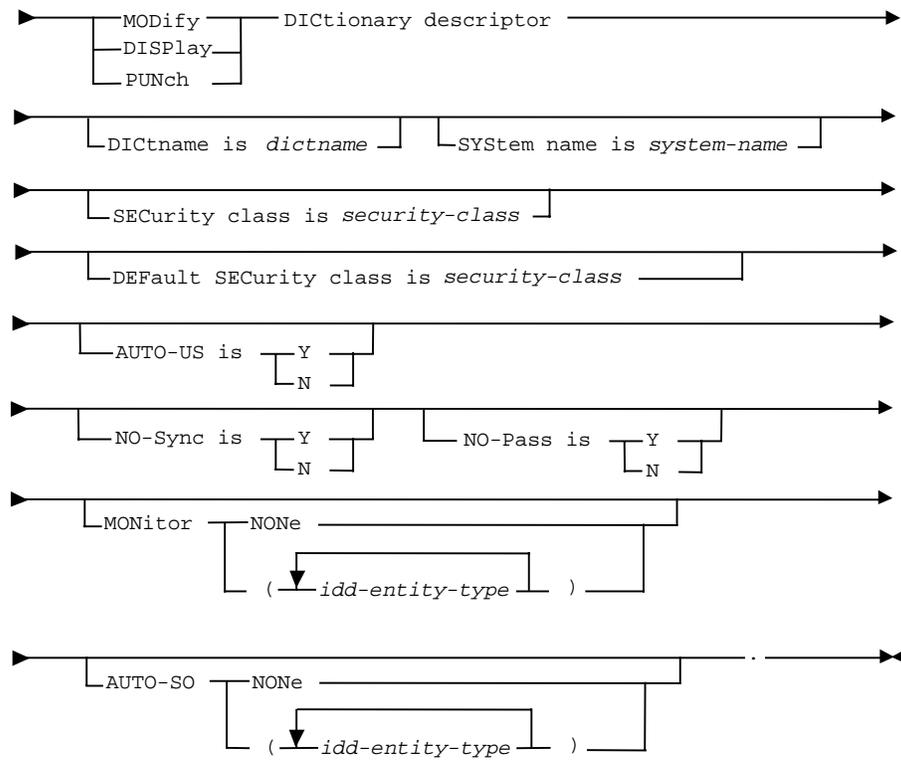
Use the DICTONARY command to modify or display (or punch) the dictionary descriptor. The clauses in this command specify global CA-Endevor/DB processing characteristics. For instance, whether or not users must supply a password when signing on to CA-Endevor/DB.

Processing Flow

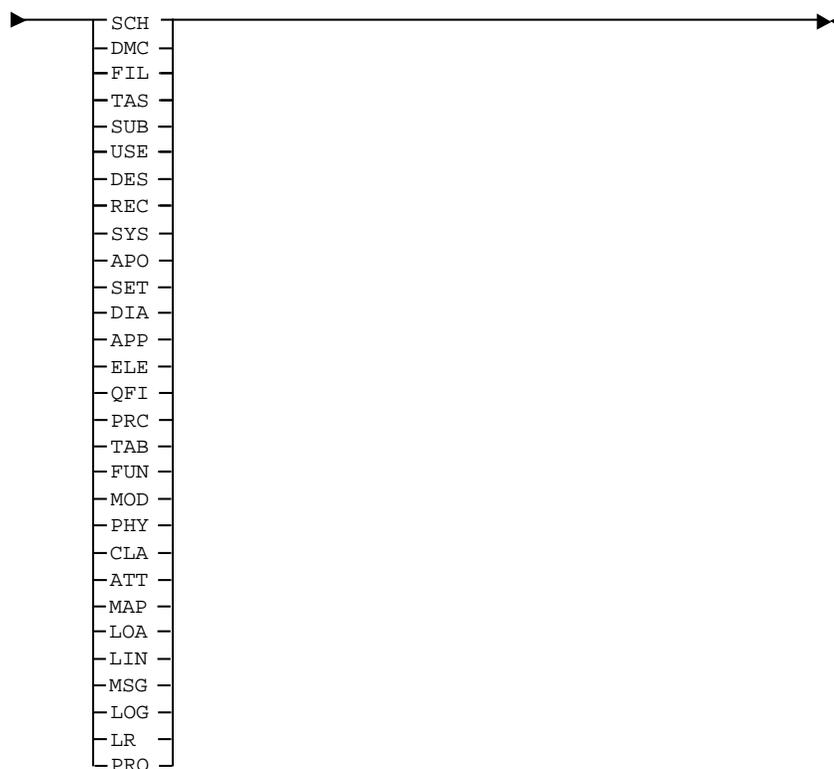
When you enter a DICTONARY command, CA-Endevor/DB performs the following processing:

- Validates the input parameters to insure that:
 - You are authorized to perform the action (MODIFY or BROWSE) for a DICTONARY. If you are not, the system will print an error message and disallow the transaction.
 - The values specified in the input parameters are valid and all required clauses are specified for the command.
- Performs the requested action.
- Prints a confirmation message that the action was successfully completed.

Syntax



idd-entity-type Clause



Syntax Rules

The rules pertaining to each clause in the syntax are listed below.

DICtname is *dictname*

The dictname value for the dictionary must be the same as the CA-IDMS DBName Table entry defining the CCDB/dictionary pair. This is the dictname to be specified in the SIGNON command.

SYStem name is *system-name*

The system name is used to uniquely identify the dictionary for migration record keeping purposes. Refer to Chapter 1 or the *CA-Endevor/DB Administrator Guide* for more information on establishing a system name.

SECurity class is *security-class*

Identifies the Security Class assigned to the dictionary descriptor. For MODIFY commands, if specified, the Security Class must already exist in the CCDB.

Initially, the Dictionary is set up with the NDVR-GLOBAL Security Class. It is recommended that this be left unaltered until a thorough understanding of the Security System is mastered.

Default SECURITY class is *security-class*

Identifies the default security class to be associated with users who are added via the Automatic User (AUTO-US) procedure. Initially, this is set to NDVR-DDA. For MODIFY commands, if specified the Security Class must already exist in the CCDB.

AUTO-Us is

Identifies the automatic actions to be taken when new users access CA-Endevor/DB:

Y – New users are automatically added by CA-Endevor/DB when encountered for the first time.

N – New users must be manually added to CA-Endevor/DB before the users will be able to sign on.

NO-Sync is

Establishes synchronization enforcement between CA-IDMS/DC and CA-Endevor/DB userids.

Y – Synchronization will not be enforced.

N – Synchronization will be enforced. CA-Endevor/DB will require that the userids must match.

NO-Pass is

Identifies password specification requirements for signing on to CA-Endevor/DB.

Y – Entering a password is optional.

N – Entering a password is required.

MONitor is

Identifies the CA-IDD entity types to be monitored by CA-Endevor/DB. On a MODIFY command, this clause replaces the monitor options stored in the CCDB's dictionary descriptor.

NONE – No entity-types are to be monitored.

(*idd-entity-type(s)*) – The *idd-entity-types* specified, and only those specified, are to be monitored. Refer to Appendix B for a description of the entity types that can be monitored by CA-Endevor/DB. See Usage below for additional information.

AUTO-SO is

Identifies the requirements for automatic signout of monitored CA-IDD entity occurrence when modified. On a MODIFY command, this clause replaces the auto-signout options stored in the CCDB's dictionary descriptor.

NOTE-CA-Endevor/DB will not automatically signout any entity occurrence.

(*idd-entity-type(s)*) – CA-Endevor/DB will automatically signout entities of the identified CA-IDD entity types when modified to the user performing the modification. Refer to Appendix B for a description of the entity types that can be automatically signed out by CA-Endevor/DB. See Usage below for additional information.

COMment is '*comment-text*'

A 1 to 60-character, user-defined remark for the dictionary descriptor.

Usage

MONITOR Clause

If the MONITOR clause is specified, you must identify each entity type to be monitored. The clause provides global replacement for all monitored entity types. For instance, if you did not want to monitor MESSAGE or SysGen entity types, you would specify the following:

```
MONITOR ( SCH FIL SUB USE REC SYS APO SET DIA APP ELE
          QFI PRC TAB FUN MOD CLA ATT MAP LOA LR  PRO )
```

omitting the TAS, DES, PHY, LIN, MSG and LOG *idd-entity-type(s)*.

AUTO-SO Clause

If the AUTO-SO clause is specified, you must identify each entity type to be automatically signed out. The clause provides global replacement for all automatically signed out entity types. For instance, if you wanted to automatically signout only Records, Schemas, and Applications as they are modified, you would specify the following:

```
AUTO-SO ( SCH REC APP )
```

If you wanted no automatic signout out of any CA-IDD entity type, you would specify the following:

```
AUTO-SO NONE
```

Examples

The following example illustrates the use of the MODIFY DICTIONARY command.

Example 1

```
MODIFY DICTIONARY  
  DICTNAME IS SRCNDVR SYSTEM IS SYSTEM81.
```

The CCDB Administrator modified the dictionary descriptor to identify its dictname and system name.

Example 2

```
DISPLAY DICTIONARY.
```

This command will output the current values in the dictionary descriptor to the NDVRLST file.

ENTITY

Purpose

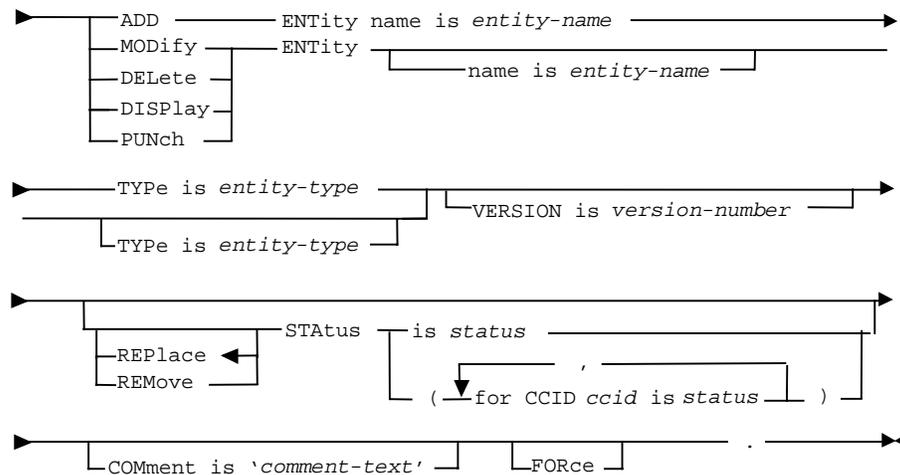
An entity is an object monitored by the Change Monitor in the CCDB. This object can be a CA-IDMS data dictionary (IDD) object like a dialog or a map. Normally, CCDB entity descriptor records for IDD entity occurrences are created automatically by CA-Endevor/DB the first time the Change Monitor encounters the monitored entity-type.

Processing Flow

When you enter an ENTITY command, CA-Endevor/DB performs the following processing:

- Validates the input parameters to insure that:
 - You are authorized to perform the action (ADD, MODIFY, DELETE, or BROWSE) for a ENTITY. If you are not, the system will print an error message and disallow the transaction.
 - The values specified in the input parameters are valid and all required clauses are specified for the command.
- For ADD actions, insures that the ENTITY does not already exist in the CCDB; for all other actions, insures that the ENTITY does exist in the CCDB.
- Performs the requested action.
- Prints a confirmation message that the action was successfully completed.

Syntax



Syntax Rules

name is *entity-name*

Identifies a new ENTITY to be added to the CCDB or an existing ENTITY occurrence to be modified, deleted, displayed, or punched. The *entity-name* must be a 1 to 40-character alphanumeric value. For ADD actions this clause is required, cannot contain masking specifications and the combined *entity-name*, *entity-type* and *version-number* must be unique within the CCDB. For all other operations, this clause is optional or may be specified with valid masking specifications as described in Chapter 1.

TYPE is *entity-type*

Specifies the type of the entity. For ADD actions, this clause is required and must identify a valid entity type as listed in Appendix B. For all other actions, clause is optional and defaults to all entity-types. See Usage below for additional information.

VERSION is *version-number*

The version number of the entity occurrence in the CCDB. For IDD entity types, this value is the same as the IDD Version Number. If specified, *version-number* must be a valid integer in the range 0 thru 9999. For ADD actions, if omitted, the *version-number* defaults to 1.

STATUS

Reflects associations of each selected entity with Status definitions. For MODIFY actions: REPLACE (default) replaces all existing Status associations with those specified in the remainder of the clause; REMOVE removes the existing status(es) specified in the remainder of the clause. See Usage below for additional information.

is status

Identifies a global status for the entity. The Status must already be defined in the CCDB.

for CCId *ccid* is status

Identifies a Status to be associated with the entity in the context of the indicated *ccid*. The CCID and Status must already be defined in the CCDB.

COMment is '*comment-text*'

A 1 to 60-character, user-defined remark for the ENTITY.

FORce

DELETE command only. This clause enables you to delete an ENTITY even if it has Change Log Entries (CLEs) associated with it.

Usage

Entity Descriptor Identification

MODIFY, DELETE, DISPLAY, and PUNCH entity identification considerations. If any part of the entity-identification (NAME, TYPE and VERSION) is not specified, or is specified with name masking, CA-Endevor/DB processing will be accomplished against all occurrences meeting the criteria of the information specified. The combinations can be:

- Name, type and version all specified exactly. The combination identifies one (or zero) specific entity descriptor.
- Name and type specified, version defaults to *. This combination identifies all entity descriptors in the CCDB which match the specified name and type, regardless of the version number.
- Name specified, type defaults to *, version defaults to *. This combination identifies all entity descriptors in the CCDB which match the specified name, regardless of the type and version.

- Type specified, name defaults to *, version defaults to *. This combination identifies all entity descriptors in the CCDB which match the specified type, regardless of the name and version.
- No specification. Name defaults to *, type defaults to *, version defaults to *. This combination identifies all entity descriptors in the CCDB.
- Name mask specified instead of name in any of the cases above will identify a set of entity names. Refer to Chapter 1 for a description of the name-mask rules.

STATUS

CA-Endevor/DB will execute the Status clause against each entity selected by the command. It is used to establish a status for the entity occurrence at a global level and/or within the context of a CCID. For instance, you may wish to assign separate status associations for an entity to control its migration selections. For example,

```
ADD ENTITY SELECTION-DATE TYPE ELEMENT VER 100
      STATUS ( FOR CCID "INSTALL" IS "ALWAYS-MIGRATE"
              CCID "PROMOTE" IS "NEVER-MIGRATE").
```

DELETE with FORCE

If FORCE is specified on a DELETE command, all Change Log Entries (CLEs) associated with the ENTITY are also deleted. A CLE is generated automatically anytime an entity is updated.

Examples

The following examples illustrate the use of ENTITY commands:

Example 1

```
ADD ENTITY SELECTION-DATE TYPE ELEMENT VERSION 1
  STATUS (FOR CCID "ADMINISTER" IS "MANUAL-ENTITY")
  COMMENT "PRE-DEFINED ELEMENT ENTITY".
```

CA-Endevor/DB will store an entity named SELECTION-DATE with an entity type of ELEMENT, a version number of 100 and the indicated remarks in the CCDB and associate it with a status of MANUAL-ENTITY within the context of the ADMINISTER CCID.

Example 2

```
MODIFY ENTITY * TYPE ELEMENT VER 100
  STATUS IS "NEVER-MIGRATE".
```

CA-Endevor/DB will assign a global status of NEVER-MIGRATE to all entities in the CCDB with a type of ELEMENT and a version number of 100.

Example 3

```
MODIFY ENTITY SELECTION-DATE TYPE ELE VER 100
  REMOVE STATUS IS "NEVER-MIGRATE"
  COMMENTS "SPECIAL ELEMENT".
```

CA-Endevor/DB will remove the NEVER-MIGRATE status from the version 100 of the SELECTION-DATE element entity and update the comments.

MANAGEMENT GROUP

Purpose

Management Groups classify CCIDs for reporting purposes. A single management group might include all CCIDs for a particular project, for example, or for several related projects.

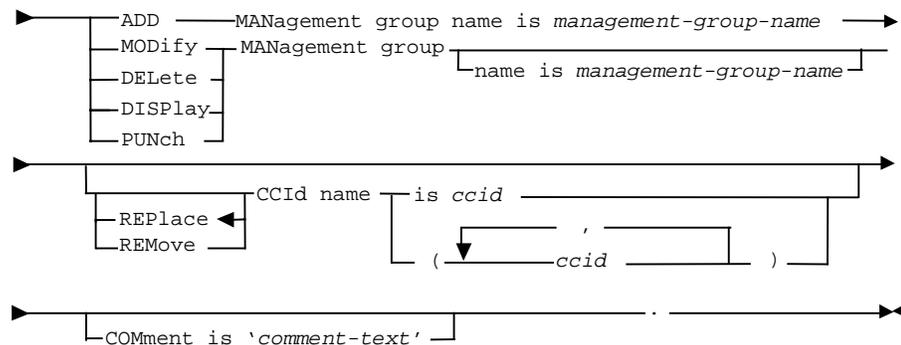
You can associate a CCID with an existing management group, to categorize that CCID within the group. This does not change the data stored for either the CCID or the management group but rather creates a relationship between the two in the CCDB.

Processing Flow

When you enter a MANAGEMENT GROUP command, CA-Endevor/DB performs the following processing:

- Validates the input parameters to insure that:
 - You are authorized to perform the action (ADD, MODIFY, DELETE, or BROWSE) for a MANAGEMENT GROUP. If you are not, the system will print an error message and disallow the transaction.
 - The values specified in the input parameters are valid and all required clauses are specified for the command.
- For ADD actions, insures that the MANAGEMENT GROUP does not already exist in the CCDB; for all other actions, insures that the MANAGEMENT GROUP does exist in the CCDB.
- Performs the requested action.
- Prints a confirmation message that the action was successfully completed.

Syntax



Syntax Rules

The rules pertaining to each clause in the syntax are listed below.

name is management-group-name

Identifies a MANAGEMENT GROUP to be added to the CCDB or an existing MANAGEMENT GROUP descriptor to be modified, deleted, displayed, or punched.

For ADD actions, this clause is required. *management-group-name* is a 1 to 16-character alphanumeric value and must be a unique management group name within the CCDB.

For all other operations, this clause is optional or may be specified with valid masking specifications as described in Chapter 1. All operations will be affected on each management group meeting the selection criteria.

CCId name is ccid

For an ADD command, this clause specifies the name of the CCID(s) you want to associate with the management group.

For MODIFY commands, if REPLACE (default) is specified, all existing CCID associations are removed and new CCID(s) named in this clause are associated with each selected management group; if REMOVE is specified, existing associations with the CCID(s) named in the clause are removed.

The CCID(s) specified in the clause must already be defined in the CCDB. Up to 1000 CCIDs may be associated with a management group.

COMment is 'comment-text'

A 1 to 60-character, user-defined remark for the management group.

Examples

The following examples illustrate the use of MANAGEMENT GROUP commands.

Example 1

```
ADD MANAGEMENT GROUP NAME "DEV140"  
  CCID NAME = (CAABF0-9703 CAABF0-9705)  
  COMMENT "MANAGEMENT GROUP FOR DEVELOPMENT"
```

CA-Endevor/DB will store a new management group in the CCDB and associate this management group with CCIDs CAABF0-9703 and CAABF0-9705.

Example 2

```
MODIFY MANAGEMENT GROUP  
  REMOVE CCID CAABF0-9703.
```

CA-Endevor/DB will modify all management groups in the CCDB to remove associations with CCID CAABF0-9703.

Example 3

```
DISPLAY MANAGEMENT GROUP NAME IS DEV*
```

CA-Endevor/DB will output the contents of each management group in the CCDB whose name begins with "DEV" to the NDVRLST file.

PREAUTHORIZATION

Purpose

PREAUTHORIZATION is a CA-Endevor/DB Security function which associates a user or CCID with one or more entity descriptor records in the CCDB. This association may serve any of the following purposes:

- To protect certain IDD entities from update by anyone other than preauthorized users or projects.
- To insure that certain users or projects only update specific preauthorized IDD entities.
- To restrict specific CCIDs so that only preauthorized users may sign on to them.
- To preregister identified IDD entities so that the Change Log Entries (CLEs) created when those entities are modified will be recorded against specific CCIDs.
- To restrict selected statuses so that only preauthorized users can associate those statuses with entities.

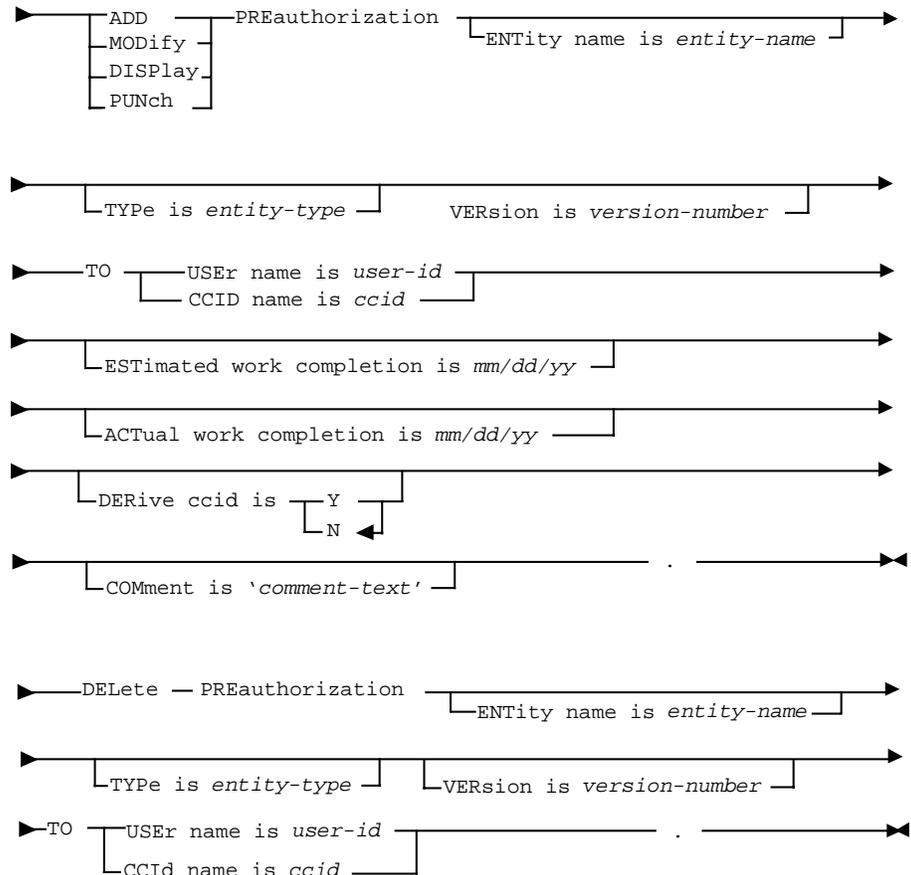
Refer to the *CA-Endevor/DB Administrator Guide* for more information on preauthorization.

Processing Flow

When you enter a PREAUTHORIZATION command, CA-Endevor/DB performs the following processing:

- Validates the input parameters to insure that:
 - You are authorized to perform the action (ADD, MODIFY, DELETE, or BROWSE) for a PREAUTHORIZATION. If you are not, the system will print an error message and disallow the transaction.
 - The values specified in the input parameters are valid and all required clauses are specified for the command.
 - For ADD commands, insures that the PREAUTHORIZATION does not already exist in the CCDB; for all other commands, insures the PREAUTHORIZATION does exist in the CCDB.
 - Performs the requested action.
 - Prints a confirmation message that the action was successfully completed.

Syntax



Syntax Rules

The rules pertaining to each clause in the syntax are listed below.

ENTity name is *entity-name*

Specifies the name of the entity(s) to which the preauthorization command applies. The *entity-name* can be up to 40 alphanumeric characters and may be specified with name masking as described in Chapter 1. See Usage below for additional information.

TYPe is *entity-type*

Specifies the type of the entity(s) to which the preauthorization command applies. If specified, *entity-type* must identify a valid entity type as listed in Appendix B. See Usage below for additional information.

VERsion is *version-number*

The version number of the entity(s) to which the preauthorization command applies. For IDD entity types, this value is the same as the IDD Version Number. If specified, *version-number* must be a valid integer in the range 0 thru 9999. For ADD actions, *version-number* defaults to 1.

TO

The user or CCID associated with the preauthorization command. You must specify either an existing user or an existing CCID, but not both.

USER is *user-id*

Identifies a User preauthorization to the selected entity(s). The *user-id* is 1 to 32 characters identifying an existing CCDB User descriptor.

CCId is *ccid*

Identifies a CCID preauthorization to the selected entity(s). The *ccid* is 1 to 12 characters identifying an existing CCDB CCID descriptor.

ESTimated work completion is *estimated-date*

Identifies the estimated completion date of this project. If specified, *estimated-date* must be entered in *mm/dd/yy* format. This date is maintained as part of project control functionality.

ACTual work completion is *actual-date*

Identifies the actual completion date of this project. If specified, *actual-date* must be entered in *mm/dd/yy* format. This date is maintained as part of project control functionality.

DERive ccid is

This clause is valid only if the TO CCID clause is specified in the PREAUTHORIZATION command.

Y – This preauthorization is to be used for CCID derivation. Preauthorization containing DERIVE CCID = Y is also used to permit access to the entity for those users not running in derived CCID mode.

N – This preauthorization is not to be used for CCID derivation.

COMment is '*comment-text*'

A 1 to 60-character, user-defined remark for the PREAUTHORIZATION.

Usage

Entity Descriptor Identification

If any part of the entity-identification (ENTITY NAME, TYPE Descriptor and VERSION) is omitted or is specified with name masking, CA-Endevor/DB processing will be accomplished against all occurrences meeting the criteria of the information specified. The combinations can be:

- **Name, type and version all specified exactly.** The combination identifies one (or zero) specific entity descriptor.
- **Name and type specified, version defaults to *.** This combination identifies all entity descriptors in the CCDB which match the specified name and type, regardless of the version number.
- **Name specified, type defaults to *, version defaults to *.** This combination identifies all entity descriptors in the CCDB which match the specified name, regardless of the type and version.
- **Type specified, name defaults to *, version defaults to *.** This combination identifies all entity descriptors in the CCDB which match the specified type, regardless of the name and version.
- **No specification.** Name defaults to *, type defaults to *, version defaults to *. This combination identifies all entity descriptors in the CCDB.
- **Name mask specified instead of name in any of the cases above** will identify a set of entity names. Refer to Chapter 1 for a description of the name-mask rules.

Changing Users/CCIDs Preauthorized to an Entity

To change a user or CCID preauthorized to an entity, use a DELETE PREAUTHORIZATION command followed by an ADD PREAUTHORIZATION command.

Examples

The following examples illustrate the use of PREAUTHORIZATION commands.

Example 1

```
ADD PREAUTHORIZATION
ENTITY NAME IS "EMPSCHM" TYPE SCHEMA VERSION 100
TO USER DATABASE-ADMIN
ESTIMATED COMPLETION DATE 06/30/97
COMMENT "BETA RELEASE".
```

CA-Endevor/DB will add this preauthorization to the CCDB for the identified entity. This command allows user DATABASE-ADMIN to perform Schema compiler updates against the EMPSCHEM Version 100 schema.

Example 2

```
MODIFY PREAUTHORIZATION
  ENTITY * TYPE RECORD VERSION 100
  TO CCID DBADMIN.
```

This command modifies all preauthorizations defined for record entity types with version 100, regardless of the name, to the DBADMIN CCID and to allow that CCID to be available as a derived CCID.

Example 3

```
DELETE PREAUTHORIZATION
  TO CCID DEVREVIEW.
```

This command removes all preauthorizations associated with the CCID "DEVREVIEW" from the CCDB.

SECURITY CLASS

Purpose

Security classes are the most important part of CA-Endevor/DB security. Security classes define restrictions that apply to dictionaries, CCIDs and users. Each of these entities can be associated with a different security class. At execution time (during stage two of the signon process), the security system combines all the security classes referenced and arrives at a resultant set of "permission flags". If an activity controlled by a permission flag is disallowed at any level (dictionary, CCID or user), the user signed on to that dictionary and working under that CCID will not be allowed to perform that activity.

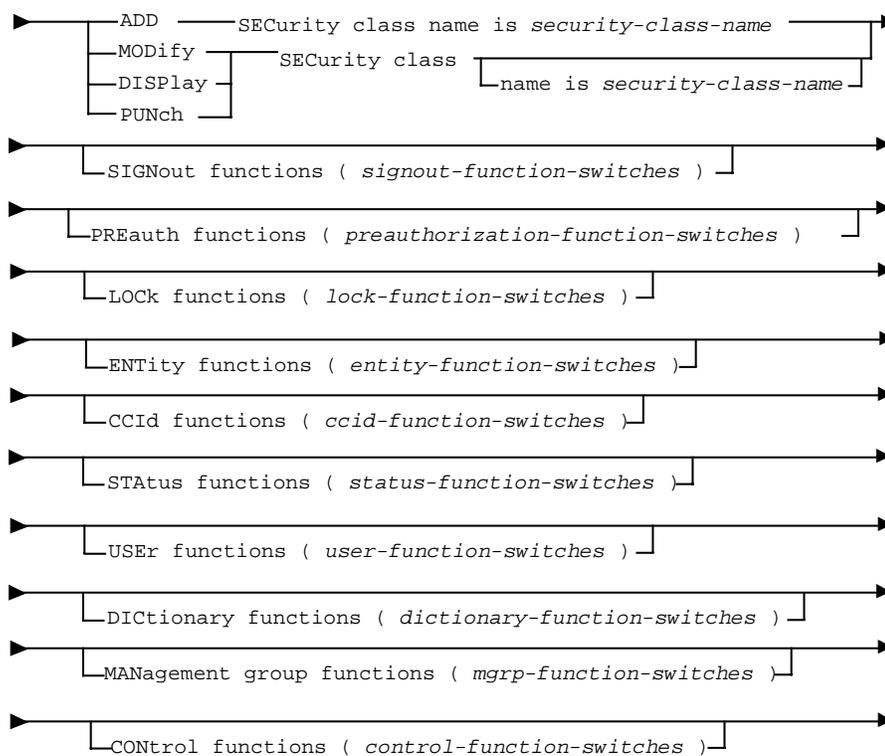
The dictionary, CCID and user definitions each contain a reference to a security class name. There will usually be few security classes in a CCDB. Many installations set up one for the Security Administrator (usually NDVR-GLOBAL), one for DBAs, one for development leaders and one for general application developers.

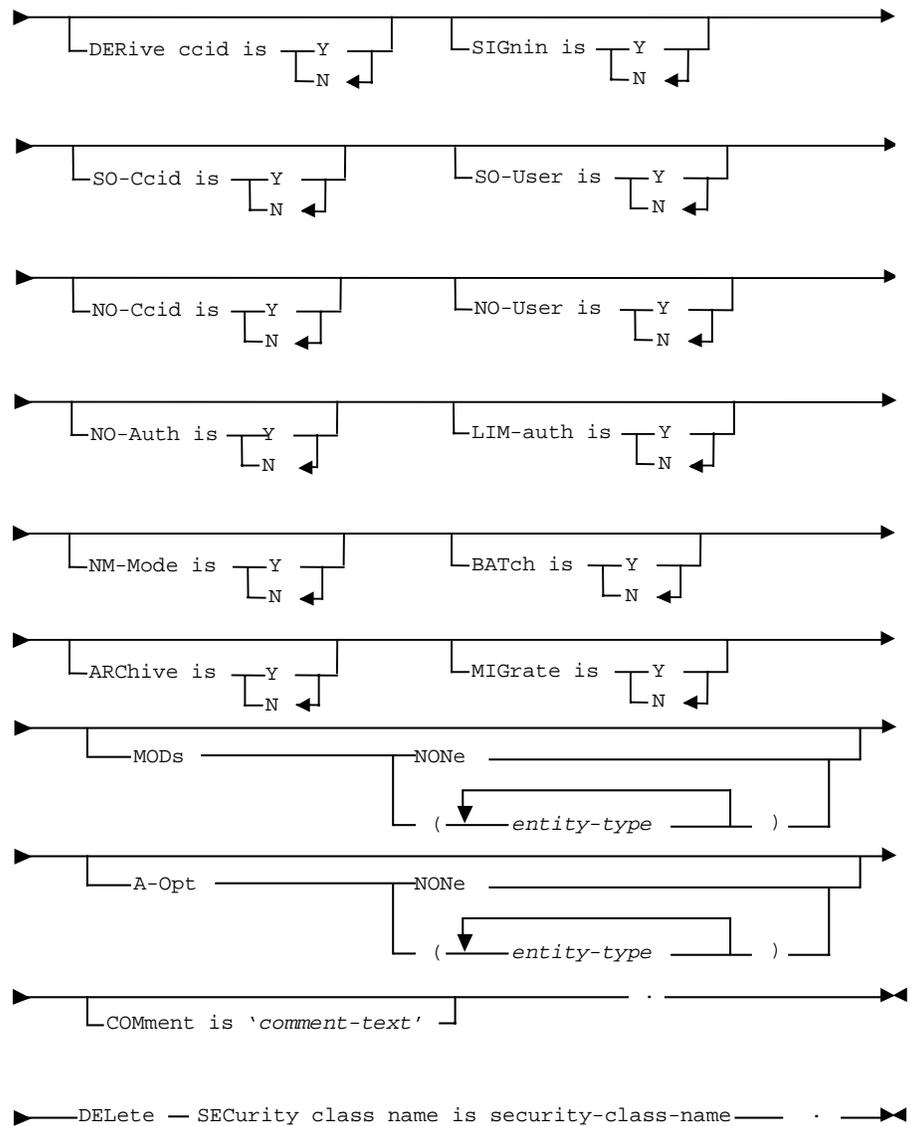
Processing Flow

When you enter a SECURITY CLASS command, CA-Endevor/DB performs the following processing:

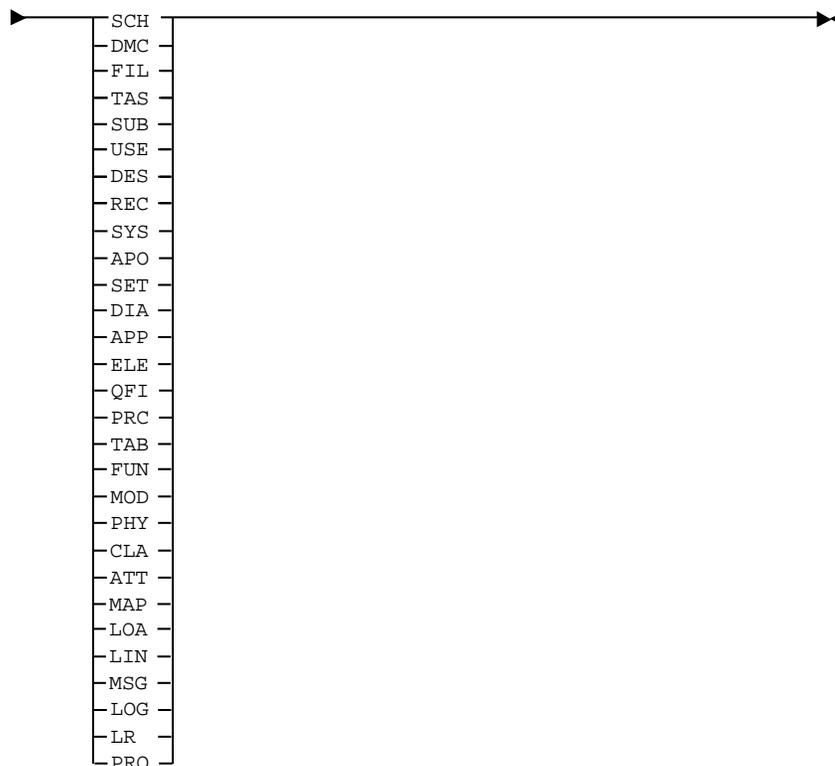
- Validates the input parameters to insure that:
 - You are authorized to perform the action (ADD, MODIFY, DELETE, or BROWSE) for a SECURITY CLASS. If you are not, the system will print an error message and disallow the transaction.
 - For ADD actions, insures that the SECURITY CLASS does not already exist in the CCDB; for all other actions, insures that the SECURITY CLASS does exist in the CCDB.
 - The values specified in the input parameters are valid and all required clauses are specified for the command.
- Performs the requested action.
- Prints a confirmation message that the action was successfully completed.

Syntax





entity-type Clause



Syntax Rules

The rules pertaining to each clause in the syntax are listed below.

name is *security-class-name*

Identifies a new SECURITY CLASS to be added to the CCDB or an existing SECURITY CLASS to be modified, deleted, displayed, or punched. The *security-class-name* is a 1 to 16-character alphanumeric value. For ADD and DELETE commands, this clause is required and must specify a unique security class within the CCDB; for all other operations, this clause is optional or may be specified with valid name masking specifications as described in Chapter 1.

Important! Do not delete either of the security classes named in the dictionary description, nor the one identified in the CCDB Administrator's User descriptor record.

authorization functions

The function clauses define from three to nine switches each, which can be turned on by specifying a Y to indicate permission granted for the function or turned off by specifying an N to indicate permission is denied for the function. Note that Browse authority controls both MIS Online selection of Browse options and MIS Batch Display and Punch commands, where applicable. See Usage below for additional information.

Authorization Function	Switch	Function
SIGNout	1	Browse signed out entities
	2	Signout entities
	3	Signin entities
PREauthorization functions	1	Browse Preauthorizations
	2	Add Preauthorizations
	3	Delete Preauthorizations
	4	Modify Preauthorizations
LOCK functions (MIS Online only)	1	Browse locked users
	2	Lock users
	3	Unlock users
	4	Browse locked CCIDs
	5	Lock CCIDs
	6	Unlock CCIDs
	7	Browse locked dictionaries
	8	Lock dictionaries
	9	Unlock dictionaries
ENTity functions	1	Browse entity descriptors
	2	Add entity descriptor
	3	Modify entity descriptor
	4	Delete entity descriptor
	5	Browse entity change history
	6	Browse entity status history

Authorization Function	Switch	Function
CCId functions	1	Browse CCID descriptors
	2	Add CCID descriptor
	3	Modify CCID descriptor
	4	Delete CCID descriptor
	5	Browse CCID/change associations
	6	Add CCID/change associations
	7	Modify CCID/change associations
	8	Delete CCID/change associations
	9	Browse entity status for CCID
STAtus functions	1	Browse Status descriptors
	2	Add status descriptor
	3	Modify status descriptor
	4	Delete status descriptor
	5	Browse status/entity associations
	6	Add status/entity associations
	7	Modify status/entity associations
	8	Delete status/entity associations
USer functions	1	Browse User descriptors
	2	Add User descriptor
	3	Modify user descriptor
	4	Delete User descriptor
	5	Browse User/change associations
	6	Add User/change associations
	7	Modify User/change associations
	8	Delete User/change associations
DICTionary functions	1	Browse dictionary descriptors
	2	Modify dictionary descriptors
	3	Delete dictionary descriptors
	4	Browse change log entries
	5	Modify change log entries

Authorization Function	Switch	Function
	6	Delete change log entries
MANagement group functions	1	Browse management group descriptors
	2	Add management group descriptors
	3	Modify management group descriptors
	4	Delete management group descriptors
	5	Browse management group/CCID associations
	6	Add management group/CCID associations
	7	Modify management group/CCID associations
	8	Delete management group/CCID associations
CONTRol functions	1	Browse CCDB descriptors
	2	Modify CCDB descriptors
	3	Browse security descriptors
	4	Add security descriptors
	5	Modify security descriptors
	6	Delete security descriptors
	7	Browse monitor dictionary status blocks
	8	Modify monitor dictionary status blocks

DERive ccid is

Determines how CA-Endevor/DB will handle CCIDs and change logging.

Y – CCIDs are ignored at CA-Endevor/DB signon. When an entity is modified, the CCDB is checked to see if a CCID is associated with the entity (through the PREAUTHORIZATION DERIVE CCID IS Y clause); if so, the change log entry created in the CCDB will be marked with the associated CCID.

N – User must perform a CA-Endevor/DB signon and specify a CCID in order to cause change log entries to be marked with a CCID.

SIGnin is

Establishes permission level for entity signout/signin for other users and/or CCIDs.

Y – Signout/Signin entities for other users and/or CCIDs is allowed.

N – Signout/Signin entities for other users and/or CCIDs is not allowed.

SO-Ccid is

Used in conjunction with the dictionary descriptor's AUTO-SO indicator to control automatic signout of entities to a CCID. If AUTO-SO is not in effect (N), this clause has no effect; if AUTO-SO is in effect (Y), this switch is used in conjunction with the SO-USER switch to determine to whom an entity is signed out.

Y – If SO-USER is N, signout the entity to the first CCID that the user who modified the entity was signed on under; If SO-USER is Y, this switch is ignored.

N – No automatic signout to a CCID is in effect.

SO-User is

Used in conjunction with the dictionary descriptor's AUTO-SO indicator to control automatic signout of entities to a User. If AUTO-SO is not in effect (N), this clause has no effect; if AUTO-SO is in effect (Y), this switch is used in conjunction with the SO-CCID switch to determine to whom an entity is signed out.

Y – Automatic signout is in effect to the user who modified the entity.

N – If SO-CCID is Y, automatic signout is in effect to the CCID; if SO-CCID is N, automatic signout is in effect to the user who modified the entity.

NO-Ccid is

Determines the requirements for specifying a CCID.

Y – Changes may be made without a known CCID.

N – A CCID must be specified in a CA-Endevor/DB SIGNON or the user must run in Derive CCID processing mode.

NO-User is

Determines the requirements for specifying a userid.

Y – Changes may be made without a known userid; if specified, NO-USER = Y must be specified in the dictionary descriptor's security classes.

N – Changes will not be logged unless either a CA-IDMS/DC or a CA-Endevor/DB userid is specified. SIGNON is required.

NO-Auth is

In conjunction with the LIM-AUTH clause, establishes preauthorization restrictions for a user making changes. Refer to the *CA-Endevor/DB Administrator Guide* for a full description of the use of NO-AUTH and LIM-AUTH.

Y – User is not subject to preauthorization rules.

N – User is subject to preauthorization rules.

LIM-auth is

In conjunction with the NO-AUTH clause, establishes preauthorization restrictions for a user making changes. Refer to the *CA-Endevor/DB Administrator Guide* for a full description of the use of NO-AUTH and LIM-AUTH.

Y – Limited preauthorization applies to this user.

N – Full preauthorization applies to this user.

NM-Mode is

Controls use of TAG commands in the Migration facility, NDVRDLVR process.

Y – TAG commands may be specified.

N – TAG commands may not be specified.

BATCh is

Establishes permission to execute the MIS Batch facility, NDVRMISB.

Y – NDVRMISB execution is allowed.

N – NDVRMISB execution is disallowed.

ARChive is

Establishes authority to execute the Archive and Compress utility, NDVRARCO, to archive Change Log Entries.

Y – NDVRARCO execution is allowed.

N – NDVRARCO execution is disallowed.

MIGRate is

Establishes authority to execute the Migration facility, including the migration booking, NDVRBOOK with OPTION = MIGRATE.

Y – Migration facility utility execution is allowed.

N – Migration facility utility execution is disallowed.

MODs

Identifies the CCDB and CA-IDD entity types which users under this security class are allowed to modify. Omitting an entity type means no update is allowed for that entity type.

NONE

Updates are disallowed for all entity-types.

(entity-type(s))

Update is allowed for the entity-types specified, and only those specified. Refer to Appendix B for a definition of the entity types.

A-OPT

Identifies the CCDB and CA-IDD entity types for which preauthorization rules apply before updates are allowed. This facility is used in conjunction with the NO-AUTH and LIM-AUTH values in the security class and by the existence of preauthorizations in the CCDB.

NONE

Preauthorization rules will be applied to all entity types.

(entity-type(s))

For each entity-type specified, preauthorization rules will not be applied.

COMment is 'comment-text'

A 1 to 60-character, user-defined remark for the security class descriptor.

Usage

DELeTe SEcUrity Class

Important! Do not delete either of the security classes named in the dictionary description, nor the one identified in the CCDB Administrator's User descriptor record.

Authorization functions

Each of the authorization function clauses has from three to nine switches apiece. For example, the SIGNOUT clause has three switches representing, in order, the authority to Browse signed out entities, Signout entities, and Signin entities. To turn on all the switches, enter the following syntax:

```
SIGNOUT ( Y Y Y )
```

To turn on only the third switch, Signin entity authority, enter the following syntax:

```
SIGNOUT ( N N Y )
```

To turn on only the first switch, Browse signed out entities authority, enter the following syntax:

```
SIGNOUT ( Y )
```

Notice that entering the trailing N's for the second and third authority switches is not required.

This logic applies for all the authorization function clauses.

Examples

The following examples illustrate the use of SECURITY CLASS commands:

Example 1

```
ADD SECURITY CLASS NAME IS TRAINEE  
COMMENT "SECURITY CLASS FOR ALL TRAINEES"  
SIGNOUT FUNCTIONS ( N Y N )
```

```

PREAUTH FUNCTIONS ( Y N N N )
LOCK   FUNCTIONS ( Y N N Y N N Y N N )
ENTITY FUNCTIONS ( Y Y Y N Y Y )
CCID   FUNCTIONS ( Y N N N Y N N N Y )
STATUS FUNCTIONS ( Y N N N Y N N N )
USER   FUNCTIONS ( Y N N N Y N N N )
DICTIONARY FUNC ( Y N N Y N N )
MANAGEMENT FUNC ( Y N N N Y N N N )
CONTROL FUNCTIONS ( Y N Y N N N Y N )
DERIVE CCID      N
SIGNIN           N
SO-CCID          Y
SO-USER          N
NO-CCID          N
NO-USER          N
NO-AUTH          N
LIM-AUTH         N
NM-MODE          N
BATCH            Y
ARCHIVE          N
MIGRATE          N
MODS             ( DIA PRC MAP LOA ).

```

CA-Endevor/DB will add a security class to the CCDB to be used for Trainees. This security class does not allow the trainees any privileges that could cause damage to the CCDB. Trainees will be allowed to browse information in the CCDB but will only be allowed to ADD or MODIFY ENTITY information. Trainees will be only allowed to update dialogs, processes, maps, and load modules in the CA-IDD. Trainees will be allowed to use the MIS Batch facility but will not be allowed to run the Archive or Migrate utilities.

Example 2

```
DISPLAY SECURITY CLASS.
```

CA-Endevor/DB will output the definitions for each Security Class in the CCDB to the NDVRLST file.

SIGNIN

Purpose

The SIGNIN command enables you to sign in entities that have been signed out. Sign in of an entity releases exclusive rights to that entity, allowing another user or CCID to modify it. In addition, when an entity is signed out, preauthorization commands cannot be executed for that entity. An entity must be signed in before it can be preauthorized.

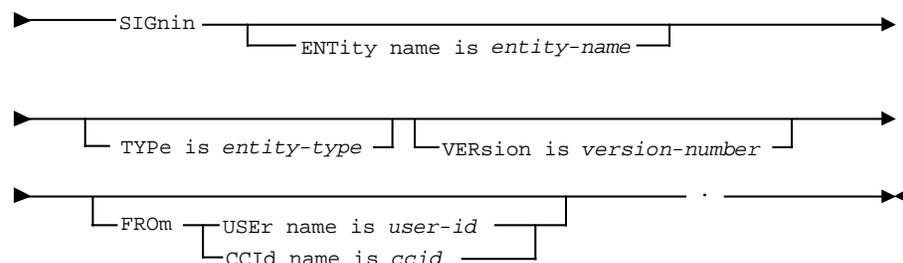
For example, if you want to modify a map and another user has signed out one of that map's work records, you would not be allowed to regenerate the map. The other user, or someone authorized to sign in entities for that user, would have to do so before you could generate the map.

Processing Flow

When you enter a SIGNIN command, CA-Endevor/DB performs the following processing:

- Validates the input parameters to insure that:
 - You are authorized to execute SIGNIN commands. If you are not, the system will print an error message and disallow the transaction.
 - The values specified in the input parameters are valid and all required clauses are specified for the command.
- Insures the selected entity exists in the CCDB and is signed out to the designated user or CCID.
- Signs in the entity.
- Prints a confirmation message that the action was successfully completed.

Syntax



Syntax Rules

The rules pertaining to each clause in the syntax are listed below.

ENTity name is *entity-name*

Identifies the entity(ies) to be signed in. If specified, *entity-name* must be a 1 to 40-character alphanumeric value and may contain valid name masking specifications as described in Chapter 1.

TYPe is *entity-type*

Specifies the type of the entity. If specified, it must identify a valid entity type as listed in Appendix B. If omitted, *entity-type* defaults to all entity-types. See Usage below for additional information.

VERsion is *version-number*

The version number of the entity occurrence in the CCDB. For IDD entity types, this value is the same as the IDD Version Number. If specified, *version-number* must be a valid integer in the range 0 thru 9999. If omitted, *version-number* defaults to all versions.

FROM

Identifies the user or CCID whose entities are to be signed in. If omitted, SIGNIN action depends on the entity information specified. See Usage below for additional information.

USER name is *user-id*

The user whose entities are to be signed in. The *user-id* must be the 1 to 32-character name of an existing user.

CCId name is *ccid*

The CCID whose entities are to be signed in. The *ccid* must be the 1 to 12-character name of an existing CCID.

Usage

Entity Descriptor Identification

If any part of the entity-identification (NAME, TYPE, and VERSION) is not specified, or is specified with name masking, SIGNIN entity selection will be accomplished for all entity occurrences meeting the criteria of the information specified.

Name, type and version all specified exactly. The combination identifies one (or zero) specific entity descriptor.

Name and type specified, version defaults to *. This combination identifies all entity descriptors in the CCDB which match the specified name and type, regardless of the version number.

Name specified, type defaults to *, version defaults to *. This combination identifies all entity descriptors in the CCDB which match the specified name, regardless of the type and version.

Type specified, name defaults to *, version defaults to *. This combination identifies all entity descriptors in the CCDB which match the specified type, regardless of the name and version.

No specification. Name defaults to *, type defaults to *, version defaults to *. This combination identifies all entity descriptors in the CCDB.

Name mask specified instead of name in any of the cases above will identify a set of entity names. Refer to Chapter 1 for a description of the name-mask rules.

FROM clause

If the FROM clause is omitted, all selected entities will be signed in, regardless of the user or CCID to which they have been signed out.

If the FROM clause is specified, selected entities will be signed in only for the specific user or CCID.

Examples

The following examples illustrate the use of the SIGNIN commands.

Example 1

```
SIGNIN ENTITY NAME DEPARTMENT TYPE REC VERSION 1  
FROM CCID DBADMIN.
```

CA-Endevor/DB will signin the specified single entity which was signed out to the "DBADMIN" CCID.

Example 2

```
SIGNIN ENTITY NAME * TYPE * VERSION 100.
```

CA-Endevor/DB will signin all entities in the CCDB containing version number 100 from any user or CCID to which the occurrence has been signed out.

SIGNOUT

Purpose

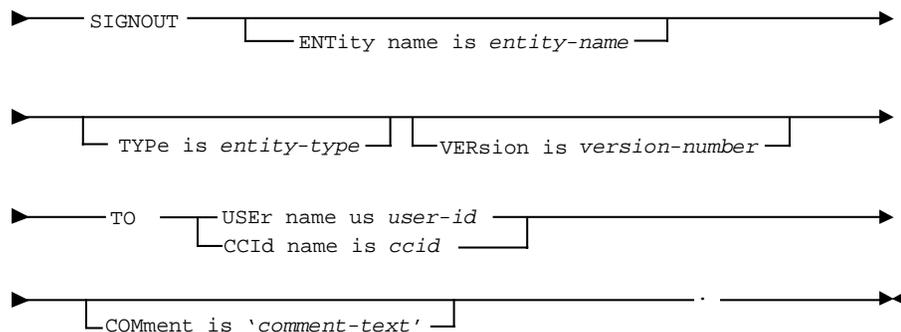
The SIGNOUT command enables you to explicitly reserve entities for the exclusive use of a particular CCID or user.

Processing Flow

When you enter a SIGNOUT command, CA-Endevor/DB performs the following processing:

- Validates the input parameters to insure that:
 - You are authorized to execute SIGNOUT commands. If you are not, the system will print an error message and disallow the transaction.
 - The values specified in the input parameters are valid and all required clauses are specified for the command.
- Insures that the entity exists in the CCDB and is not already signed out.
- Signs out the entity.
- Prints a confirmation message that the action was successfully completed.

Syntax



Syntax Rules

The rules pertaining to each clause in the syntax are listed below.

ENTity name is *entity-name*

Identifies the entity(ies) to be signed out. If specified, *entity-name* must be a 1 to 40-character alphanumeric value and may contain valid name masking specifications as described in Chapter 1.

TYPE is *entity-type*

Specifies the type of the entity. If specified, it must identify a valid entity type as listed in Appendix B. If omitted, *entity-type* defaults to all entity-types. See Usage below for additional information.

VERSION is *version-number*

The version number of the entity occurrence in the CCDB. For IDD entity types, this value is the same as the IDD Version Number. If specified, *version-number* must be a valid integer in the range 0 thru 9999. If omitted, *version-number* defaults to all versions.

TO

Identifies the user or CCID to whom the selected entities are to be signed out. SIGNOUT must be accomplished to either a user or a CCID, but not both.

USER name is *user-id*.

The user to whom the entities are to be signed out. The *user-id* must be the 1 to 32-character name of an existing user.

CCID name is *ccid*

The CCID to whom the entities are to be signed out. The *ccid* must be the 1 to 12-character name of an existing CCID.

COMMENT is '*comment-text*'

A 1-60 character, user-defined remark for the SIGNOUT.

Usage

Entity Descriptor Identification

If any part of the entity-identification (NAME, TYPE, and VERSION) is not specified, or is specified with name masking, SIGNOUT entity selection will be accomplished for all entity occurrences meeting the criteria of the information specified.

Name, type and version all specified exactly. The combination identifies one (or zero) specific entity descriptor.

Name and type specified, version defaults to *. This combination identifies all entity descriptors in the CCDB which match the specified name and type, regardless of the version number.

Name specified, type defaults to *, version defaults to *. This combination identifies all entity descriptors in the CCDB which match the specified name, regardless of the type and version.

Type specified, name defaults to *, version defaults to *. This combination identifies all entity descriptors in the CCDB which match the specified type, regardless of the name and version.

No specification. Name defaults to *, type defaults to *, version defaults to *. This combination identifies all entity descriptors in the CCDB.

Name mask specified instead of name in any of the cases above will identify a set of entity names. Refer to Chapter 1 for a description of the name-mask rules.

Examples

The following examples illustrate the use of the SIGNOUT commands.

Example 1

```
SIGNOUT ENTITY NAME DEPARMENT TYPE REC VERSION 1  
      TO CCID DBADMIN.
```

CA-Endevor/DB will sign out the specified single entity to the "DBADMIN" CCID.

Example 2

```
SIGNOUT ENTITY NAME * TYPE * VERSION 100  
      TO USER DATABASE-ADMINISTRATOR  
      COMMENT "PRODUCTION FREEZE"
```

CA-Endevor/DB will signout all entities in the CCDB containing version number 100 to the "DATABASE-ADMINISTRATOR" user and document each SIGNOUT.

STATUS

Purpose

A status is similar to a CA-IDD attribute. It is a "tag" used to annotate, identify, or classify entities and is often used to include or exclude entities for migration.

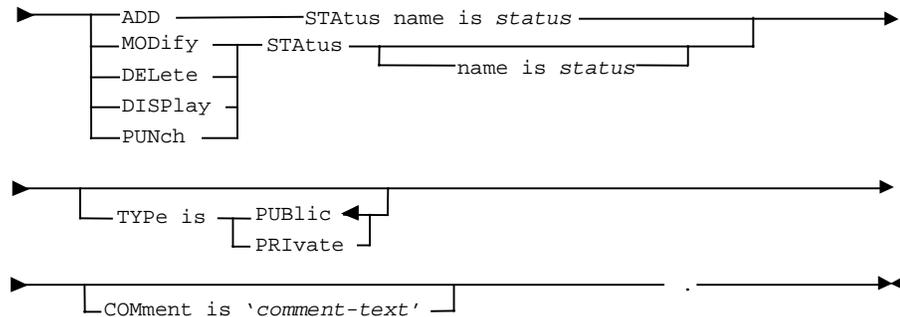
Status identifies can be associated with an entity or with an entity within the context of a particular CCID. Status names are user-defined and might identify a "state" such as DESIGNED, "IN DEVELOPMENT", COMPLETED, PROGRAMMED, and so forth. A status can be assigned by itself ("globally") or within the context of a CCID. For example, a given entity might have the status "IN DEVELOPMENT" in CCID ACCT-9706 and the status TESTED in CCID ACCT-9703. An entity may have a global status and different status within a given CCID.

Processing Flow

When you enter a STATUS command, CA-Endevor/DB performs the following processing:

- Validates the input parameters to insure that:
 - You are authorized to perform the action (ADD, MODIFY, DELETE, or BROWSE) for a STATUS. If you are not, the system will print an error message and disallow the transaction.
 - The values specified in the input parameters are valid and all required clauses are specified for the command.
- For ADD actions, insures that the STATUS does not already exist in the CCDB; for all other actions, insures that the STATUS does exist in the CCDB.
- Performs the requested action.
- Prints a confirmation message that the action was successfully completed.

Syntax



Syntax Rules

The rules pertaining to each clause in the syntax are listed below.

name is status

Identifies a new STATUS to be added to the CCDB or an existing STATUS occurrence to be modified, deleted, displayed, or punched. The *status-name* must be a 1 to 16-character alphanumeric value. For ADD actions this clause is required, cannot contain masking specifications, and must be unique within all statuses in the CCDB. For all other operations, this clause is optional or may be specified with valid masking specifications as described in Chapter 1.

TYPE is type

Indicates whether the STATUS is PUBLIC or PRIVATE. If omitted, defaults to PUBLIC.

COMment is 'comment-text'

A 1 to 60-character, user-defined remark for the STATUS.

Examples

The following example illustrates the use of the ADD STATUS command.

Example 1

```

ADD STATUS NEVER-MIGRATE
  TYPE IS PUBLIC
  COMMENT "STATUS FOR ENTITIES TO NEVER MIGRATE".
  
```

CA-Endevor/DB will create the NEVER-MIGRATE status in the CCDB with a type of PUBLIC and the indicated comments for documentation purposes.

Example 2

```
MODIFY STATUS NEVER*  
TYPE IS PRIVATE.
```

CA-Endevor/DB will modify all statuses in the CCDB which match the masked name "NEVER" to reflect a type of PRIVATE. This will require that a user be preauthorized to assign any status whose name matches "NEVER" to an entity.

USER

Purpose

User information describes individual CA-Endevor/DB users or user groups to the CA-Endevor/DB system. Each user descriptor contains a name, optional password, security restrictions, current CCID(s), and descriptive comments. The user name and password are used during SIGNON to identify the user to the system and to verify that the user is authorized to use CA-Endevor/DB. The security class identifies the set of security restrictions in effect for the user (further described in the *CA-Endevor/DB Administrator Guide*. The CCIDs identify the CCIDs under which the user is working (or under which the user last worked).

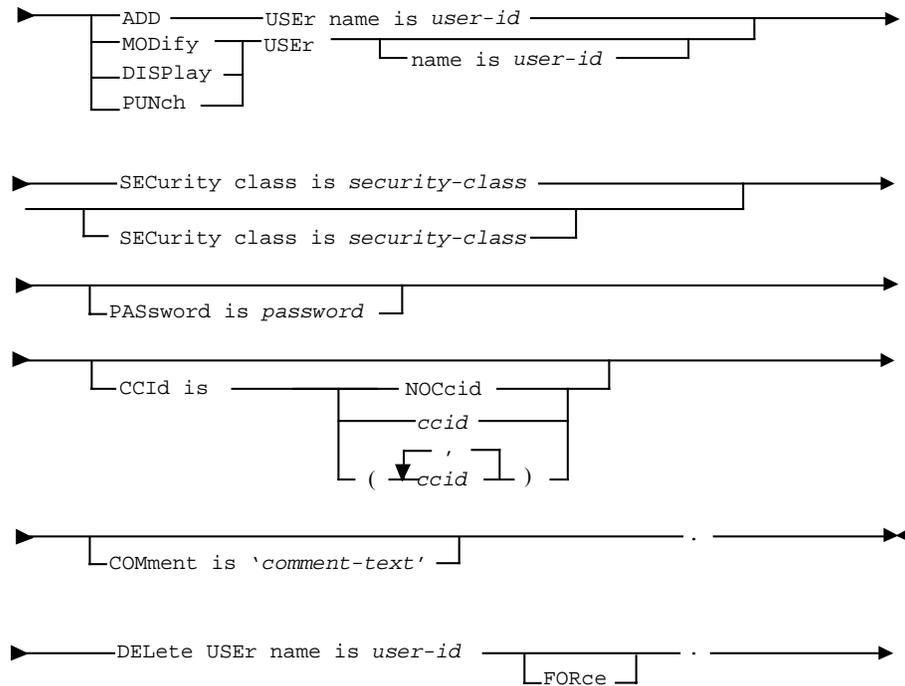
If necessary, a particular user might be restricted from dictionary access altogether. This can effectively secure your dictionary and CCDB from persons no longer permitted access.

Processing Flow

When you enter a USER command, CA-Endevor/DB performs the following processing:

- Validates the input parameters to insure that:
 - You are authorized to perform the action (ADD, MODIFY, DELETE, or BROWSE) for a user. If you are not, the system will print an error message and disallow the transaction.
 - The values specified in the input parameters are valid and all required clauses are specified for the command.
- For ADD actions, insures that the user does not already exist in the CCDB; for all other actions, insures that the user does exist in the CCDB.
- Performs the requested action.
- Prints a confirmation message that the action was successfully completed.

Syntax



Syntax Rules

The rules pertaining to each clause in the syntax are listed below.

name is *user-id*

Identifies a new user to be added to the CCDB or an existing user descriptor to be modified, deleted, displayed, or punched. The *user-id* must be a 1 to 32-character alphanumeric value. For ADD actions, this clause is required and must be unique within all users in the CCDB; for DELETE actions, this clause is required and must identify an existing user within the CCDB. For all other operations, this clause is optional or may be specified with valid masking specifications as described in Chapter 1.

SECurity class is *security-class*

Identifies the Security Class to be associated with the user. The Security Class must already exist in the CCDB. This clause is required for ADD commands.

PASsword is *password*

Specifies a password for the user. A *password* can be any 1 to 8-character alphanumeric value and is stored in the CCDB in an encrypted form to further secure the user. Password values are never displayed.

CCID Name is *ccid*

Identifies the name of any CCID associations for this particular user. If specified, *ccid* must be a 1 to 12-character alphanumeric value and identifying a previously defined CCID. If association is desired between the user and multiple CCIDs, enter the CCIDs using the list format shown in the syntax diagram. See Usage below for additional information.

NOCCid

No CCIDs are associated with this user.

COMment is '*comment-text*'

A 1 to 60-character, user-defined remark for the user.

FORce

DELETE command only. This clause enables you to delete a user even if it has Change Log Entries (CLEs) associated with it.

Usage

CCIDs

If a user has associated CCIDs, CA-Endevor/DB will "remember" those CCIDs from session to session. On a MODIFY request, to associate the user with a different set of CCIDs, you must first enter the NOCCID option to remove any associations between the user and any CCIDs. You would then specify the new set of CCIDs, if any, to be associated with the user.

DELETE with FORCE

If FORCE is specified on a DELETE command, all Change Log Entries (CLEs) associated with the user are also deleted. A CLE is generated automatically anytime an entity is updated.

Examples

The following examples illustrate the use of the ADD USER command.

Example 1

```
ADD USER NAME IS DATABASE-ADMINISTRATOR
      PASSWORD IS DBA
      CCID (DBADMIN, SYSADMIN)
      COMMENT DATABASE ADMINISTRATOR FOR PROJECT'.
```

The CCDB Administrator has added a new user to be used by the database administrator and has associated this user with both the DBADMIN and SYSADMIN CCIDs.

Example 2

```
MODIFY USER NAME IS DATABASE-ADMINISTRATOR  
  CCID IS NOCCID.  
MODIFY USER NAME IS DATABASE-ADMINISTRATOR  
  CCID IS DBADMIN.
```

The CCDB Administrator has modified the user to remove all previous CCID associations and then to associate the user with only the DBADMIN CCID.

Example 3

```
DISPLAY USER NAME IS DEV*.
```

This command will display all users in the CCDB whose name matches the "DEV" name-mask.

Example 4

```
DELETE USER NAME IS DEVUSER FORCE.
```

The CCDB Administrator has decided to delete the USER and to force all CLEs associated with the USER to also be deleted.

What Is PUNCH Mode?

PUNCH mode does not execute commands that you enter, but takes the commands, builds syntax for all of the elements in the CCDB which meet the selection criteria, and then punches the detailed command syntax into a punch dataset without echoing the syntax on the Batch process listing. Using a PUNCH *<CCDB-entity-type>* command for the supported CCDB-entity-types, under PROCESS mode, achieves the same results as if an ADD *<CCDB-entity-type>* command were issued under PUNCH mode, except that the syntax is echoed on the Batch process listing, prefixed with an “*+”. In either case, the command syntax written to the punch dataset may be viewed or edited.

Why Use PUNCH Mode?

While most of your CA-Endevor/DB Batch work will be done in the PROCESS mode, the PUNCH mode can be used. Examples of when you might want to use PUNCH mode are:

- You want to produce syntax for SIGNIN or SIGNOUT commands or you want the syntax to reflect operations other than ADD.
- You do not want the syntax to be echoed on the Batch listing

If you enter a command in PUNCH mode, CA-Endevor/DB writes syntax to a punch dataset for all objects in the CCDB that meet the selection criteria. The generated syntax is not echoed on the CA-Endevor/DB Batch listing. You can review the punched syntax for the command(s) you entered and, if needed, edit it. You can run the edited commands in PROCESS mode to execute the commands against the CCDB.

If, for example, you wanted to produce syntax for all the entities in the CCDB, you would enter the following commands:

```
SET OPTION PUNCH.  
ADD ENTITY.
```

CA-Endevor/DB would then punch the following:

```
ADD ENTITY NAME IS "COEDASO      "
      TYPE IS "DIALOG           "
      VERSION IS 00001
      COMMENT IS "CLAP ON       "
      .
ADD ENTITY NAME IS "COEDASO-PREMAP "
      TYPE IS "PROCESS          "
      VERSION IS 00001
      COMMENT IS "CLAP OFF      "
      .
ADD ENTITY NAME IS "COEDASO-RESPONSE"
      TYPE IS "PROCESS          "
      VERSION IS 00001
      COMMENT IS "CLAP ON       "
      .
ADD ENTITY NAME IS "COEDSHIP      "
      TYPE IS "DIALOG           "
      VERSION IS 00001
      COMMENT IS "CLAP OFF      "
      .
```

In the example above, CA-Endevor/DB punched four ADD ENTITY commands, one for each entity in the CCDB. As you can see, CA-Endevor/DB included the NAME, TYPE, VERSION, and COMMENT clauses in the syntax. You can now review all the entities in the CCDB, along with all the information for those entities located in the CCDB.

If you wanted to change any of the commands and then execute them against a CCDB, you can use your standard editor.

In this example CCDB, there are only four entities. If there were more entities in the CCDB, they would also be punched.

PUNCH Mode Syntax

The syntax for the commands in PUNCH mode is the same as described in the Batch Syntax chapter. You have to enter all the required clauses in each syntax and end each command with a period.

When you have CA-Endevor/DB Batch process a command in PUNCH mode, the punched output will reflect the contents of the CCDB. There will be one command punched for each description (entity, CCID, etc.) in the CCDB that matches the selection criteria in your original command. The punched commands will each include some combination of the clauses in your original command and the information in the CCDB. The rules for this combination depend on the command verb you use in your original command (ADD, MODIFY, or DELETE).

The following sections describe the output clause combinations for each of the three command verbs.

Processing Flow

When you enter a command in PUNCH mode, CA-Endevor/DB performs the following processing:

- Validates the input parameters to insure that:
 - You are authorized to BROWSE the CCDB object named in the request. If you are not, the system will print an error message and disallow the transaction.
 - The values specified in the input parameters are valid and all required clauses are specified for the command.
- Produces syntax based on the input command for the specified CCDB object to the output NDVRPCH file.
- Prints a confirmation message that the action was successfully completed.

The ADD Verb

The ADD verb in PUNCH mode uses information from the CCDB to produce command clause values. If you do not enter selection criteria in the commands, the ADD verb will punch all the information located in the CCDB for that command. For example, if you enter the command:

```
SET OPTIONS PUNCH.  
ADD ENTITY.
```

CA-Endevor/DB would punch the syntax for every entity in the CCDB. The punched syntax would include the NAME, TYPE, VERSION, COMMENT, and STATUS clauses for each entity in the CCDB. If you were to include selection criteria in the command, such as the following:

```
SET OPTIONS PUNCH.  
ADD ENTITY TYPE IS DIALOG.
```

CA-Endevor/DB would punch the syntax for only those entities whose type is dialog. The punched syntax would again include the NAME, TYPE, VERSION, COMMENT, and STATUS clauses for each entity. Applicable selection criteria is explained in the description for each command.

If you specify an "attribute" clause (i.e., a clause that describes the CCDB object rather than identifies it), the clause you specify in your original command will appear in place of any CCDB data in every punched command. For example, if you enter the following command:

```
SET OPTIONS PUNCH.  
ADD ENTITY TYPE IS RECORD STATUS IS NEVERMIGRATE.
```

CA-Endevor/DB will punch the following syntax:

```
ADD ENTITY NAME IS COL-COLLIER
      TYPE IS RECORD
      VERSION IS 00001
      COMMENT IS
      "PEA PORR HT"
      STATUS IS NEVERMIGRATE
.
ADD ENTITY NAME IS FL-GRAFIX
      TYPE IS RECORD
      VERSION IS 00002
      COMMENT IS
      "WKS IN BOSTON"
      STATUS IS NEVERMIGRATE
.
```

CA-Endevor/DB punched ADD ENTITY commands for all entities that were of type RECORD. Each punched command included the STATUS clause as entered in the original command.

The Modify Verb

The MODIFY verb will punch the MODIFY command syntax for the objects that meet the selection criteria in the CCDB. CA-Endevor/DB will punch the identifying clauses in each command. The identifying clauses are comprised of the information necessary to identify the CCDB object. For example, an entity is identified by its name, type, and version. If you enter the command:

```
SET OPTIONS PUNCH.
MODIFY ENTITY.
```

CA-Endevor/DB would punch the following MODIFY ENTITY syntax:

```
MOD ENTITY NAME IS "COEDASO           "
      TYPE IS "DIALOG           "
      VERSION IS 00001
.
MOD ENTITY NAME IS "COEDASO-PREMAP "
      TYPE IS "PROCESS           "
      VERSION IS 00001
.
MOD ENTITY NAME IS "COEDASO-RESPONSE"
      TYPE IS "PROCESS           "
      VERSION IS 00001
.
MOD ENTITY NAME IS "COEDSHIP           "
      TYPE IS "DIALOG           "
      VERSION IS 00001
.
```

Any additional clauses that you enter will also be included in every punched MODIFY command. Only the clauses that you enter in the command, in addition to the identifying clauses, will be punched. If, for example, you do not enter a COMMENT clause in the MODIFY command, CA-Endevor/DB will not include that clause in the punched command.

For example, if you want to punch a MODIFY ENTITY command with a specific COMMENT clause for every entity whose name begins with JSB, you would enter the following command:

```
SET OPTIONS PUNCH.
MOD ENTITY NAME JSB*
COMMENT = "MODIFY JSBTEST".
```

CA-Endevor/DB Batch will punch the following:

```
MOD ENTITY NAME IS JSBTEST1
      TYPE IS ELEMENT
      VERSION IS 1
      COMMENT IS "MODIFY JSBTEST"
.
MOD ENTITY NAME IS JSBTEST2
      TYPE IS RECORD
      VERSION IS 1
      COMMENT IS "MODIFY JSBTEST".
.
MOD ENTITY NAME IS JSBTEST2
      TYPE IS ELEMENT
      VERSION IS 2
      COMMENT IS "MODIFY JSBTEST"
.
```

As you can see, CA-Endevor/DB punched a MODIFY ENTITY command for every entity in the CCDB whose name began with JSB. The identifying clauses, NAME, TYPE, and VERSION were automatically included in the command. The COMMENT clause that you entered was also included in each command that was punched. Comments (if there were any) in the CCDB were ignored.

The Delete Verb

The DELETE verb in the PUNCH mode builds a DELETE command for each CCDB object that meets the selection criteria. CA-Endevor/DB will punch the identifying clauses for every command. The identifying clauses are comprised of the information necessary to identify the CCDB object. For example, an entity is identified by its name, type, and version.

For example, if you entered the following command:

```
SET OPTIONS PUNCH.
DEL ENTITY NAME JSB*.
```

CA-Endevor/DB Batch would punch the following commands for all the entities that met the selection criteria, in this case being the NAME clause:

```
DEL ENTITY NAME IS JSBTEST1
      TYPE IS ELEMENT
      VERSION IS 1
.
DEL ENTITY NAME IS JSBTEST2
      TYPE IS RECORD
      VERSION IS 1
```

```
      .  
DEL ENTITY NAME IS JSBTEST2  
      TYPE IS ELEMENT  
      VERSION IS 2  
      .
```

In the example above, CA-Endevor/DB punched DELETE syntax for each entity in the CCDB whose name began with JSB.

Batch Execution JCL

Introduction

The JCL used to execute CA-Endevor/DB Batch under OS/390 is shown below. A sample of this JCL, SAMPMISB, is provided on the installation tape and loaded during installation.

OS/390 JCL

The JCL used to run NDVRMISB under the CA-IDMS Central Version and in local mode follows:

CA-IDMS/CV Batch JCL

```

//JOBNAME      JOB YOUR.JOBCARD.INFORMATION
//JOBLIB       DD DISP=SHR,DSN=usercv.loadlib
//            DD DISP=SHR,DSN=ndvrdb.loadlib
//            DD DISP=SHR,DSN=idms.loadlib
//*
//*****
//*
//* JOB:        SAMPMISB
//*
//* PURPOSE:    RUN CA-ENDEVOR/DB BATCH COMMANDS.
//*
//* STEP:       FUNCTION:
//* =====
//*
//* GOMISB     RUNS CA-ENDEVOR/DB BATCH UPDATE.
//*
//*****
//*
//GOMISB       EXEC PGM=NDVRMISB,REGION=640K
//SYSCTL       DD DISP=SHR,DSN=idms.sysctl
//NDVRPCH      DD DISP=OLD,DSN=your.punch.dataset
//NDVRLST      DD SYSOUT=*
//NDVRERR      DD SYSOUT=*
//SYSIDMS      DD *
DMCL=dmcl-name
DICTNAME=dictionary-name
Other Optional SYSIDMS Parameters

```

```

/*
//NDVRIPT DD *
SIGNON
USER = youruserid PASSWORD = yourpswd
      DICTNAME userdict.
*+ PUT YOUR CA-ENDEVOR/DB BATCH COMMANDS HERE. +*
/*

```

The following table describes the substitution parameters above.

Field	Description
<i>usercv.loadlib</i>	Dataset name of user loadlib. This library contains DMCL and subschema load modules. It is not needed for CA-IDMS/CV, but is required for local mode.
<i>ndvrdb.loadlib</i>	Dataset name of loadlib containing CA-Endevor/DB software.
<i>idms.loadlib</i>	Dataset name of loadlib containing CA-IDMS software.
640k	Size of region to run CA-Endevor/DB Batch under the CA-IDMS/CV.
<i>your.punch.dataset</i>	Dataset name of the CA-Endevor/DB Batch punched output. This dataset will be used to write detail CA-Endevor/DB Batch command syntax in PUNCH mode. This dataset will contain 80 character card images and must be large enough to contain all commands punched by CA-Endevor/DB.
<i>youruserid</i>	Your userid.
<i>yourpswd</i>	Your password.
<i>idms.sysctl</i>	Dataset name of the CA-IDMS/CV SYSCTL file.
<i>userdict</i>	Name of the dictionary to which you are signing on.
<i>dmcl-name</i>	Name of the DMCL used by the CA-IDMS/CV.
<i>dictionary-name</i>	DICTNAME of the CCDB against which you are running the batch update.

Local Mode JCL

To execute NDVRMISB in local mode:

1. Remove the SYSCTL statement.
2. Replace with the following statements:

```
//CDB1 DD DSN=your.dbcdb1.dataset.name,DISP=OLD
```

```
//CDB2    DD  DSN=your.dbcdb2.dataset.name,DISP=OLD
//CDB3    DD  DSN=your.dbcdb3.dataset.name,DISP=OLD
//SYSJRNL DD  DSN=your.idms.journal(+1),DISP=(NEW,CATLG,CATLG),
//          UNIT=TAPE
```

3. Increase the region size value as needed to accommodate standard CA-IDMS local mode overhead.

Note: When running in local mode, ensure that the CCDB is restored following an abend. For this reason, it is highly recommended that CA-IDMS/CV mode be used whenever possible.

The following table describes the substitution parameters above.

Field	Description
<i>your.dbcdb1.dataset.name</i>	Dataset name for CA-Endevor/DB CCDB ADM area
<i>your.dbcdb2.dataset.name</i>	Dataset name for CA-Endevor/DB CCDB LOG area
<i>your.dbcdb3.dataset.name</i>	Dataset name for CA-Endevor/DB CCDB PAK area
<i>your.idms.journal</i>	Dataset name of the CA-IDMS/DB journal file

Note: If your DMCL has been defined with dynamic dataset allocation for the CDB1, CDB2, and CDB3 files, the DD statements may be omitted from your local mode JCL.

CA-Endevor/DB Batch Reserved Words

Within CA-Endevor/DB Batch, there are several reserved words - parameters or keywords (for example, type or system) that should not or cannot be used as qualifiers or identifiers within the syntax. If you do use such a word when coding the syntax, you must surround the word in quotes; either single or double quotes can be used. For example, assume you are adding an entity called ADD. If you coded

```
ADD ENTITY ADD.
```

you would receive an error message. ADD is a reserved word and must be entered with quotes. The statement:

```
ADD ENTITY 'ADD'.
```

would be accepted by CA-Endevor/DB. Likewise, a clause such as MODIFY ENTITY 'ENTITY' would be accepted by the system.

CA-Endevor/DB Batch Reserved Words

The list below contains the fully-spelled reserved words. Any partial spelling of a particular word (three characters or more) is also considered reserved. For example, "COMMENTS", "COMMENT", and "COM" are all reserved within CA-Endevor/DB.

<u>A-OPT</u>	<u>DICTNAME</u>
<u>ACTUAL</u>	<u>DMCL</u>
<u>ADD</u>	<u>ECHO</u>
<u>APO</u>	<u>ELE</u>
<u>APPLICATION</u>	<u>ENTITY</u>
<u>ARCHIVE</u>	<u>ERUS (EUS)</u>
<u>ATTRIBUTE</u>	<u>ESTIMATED</u>
<u>AUTO-SO</u>	<u>FILE</u>
<u>AUTO-US</u>	<u>FOR</u>
<u>BATCH</u>	<u>FROM</u>
<u>CCDB</u>	<u>FUNCTIONS</u>
<u>CCID</u>	<u>GROUP</u>
<u>CLASS</u>	<u>IS</u>
<u>COMMENT</u>	<u>LIM-AUTH</u>
<u>COMPLETION</u>	<u>LINE</u>
<u>CONTROL</u>	<u>LOAD-MODULE</u>
<u>DBNAME</u>	<u>LOCK</u>
<u>DEFAULT</u>	<u>LOG</u>
<u>DELETE</u>	<u>LOGICAL-RECORD (LR)</u>
<u>DERIVE</u>	<u>MANAGEMENT (MGR)</u>
<u>DESCRIPTION</u>	<u>MAP</u>
<u>DESCRIPTOR</u>	<u>MESSAGE</u>
<u>DIALOG</u>	<u>MIGRATE</u>
<u>DICTIONARY</u>	<u>MODIFY</u>

<u>MODS</u>	<u>REMOVE</u>
<u>MONITOR</u>	<u>REPLACE</u>
<u>MSG</u>	<u>SCHEMA</u>
<u>NAME</u>	<u>SCL</u>
<u>NM-MODE</u>	<u>SECURITY</u>
<u>NO-AUTH</u>	<u>SET</u>
<u>NO-CCID</u>	<u>SIGNIN</u>
<u>NO-PASS</u>	<u>SIGNON</u>
<u>NO-SYNC</u>	<u>SIGNOUT</u>
<u>NO-USER</u>	<u>SO-CCID</u>
<u>NOECHO</u>	<u>SO-USER</u>
<u>NOMESSAGE</u>	<u>STATUS</u>
<u>OPTIONS</u>	<u>STOP</u>
<u>PASSWORD</u>	<u>SUBSCHEMA</u>
<u>PHYSICAL-TERMINA</u> <u>(PTERM)</u>	<u>SYSTEM</u>
<u>PREAUTHORIZATION</u>	<u>TABLE</u>
<u>PRINT</u>	<u>TASK</u>
<u>PRIVATE</u>	<u>TO</u>
<u>PRO</u>	<u>TYPE</u>
<u>PROCESS (PRC)</u>	<u>USER</u>
<u>PUBLIC</u>	<u>VERSION</u>
<u>PUNCH</u>	<u>WORK</u>
<u>QFILE</u>	
<u>RECORD</u>	

CA-Endevor/DB Entity Types

Dictionary Entity Types

These types can be specified in the MONITOR and AUTO-SO clauses of the MODIFY DICTIONARY command, in the MODS and A-OPT clauses of the ADD SECURITY CLASS and MODIFY SECURITY CLASS commands, and in the TYPE clause of all commands containing ENTITY NAME, TYPE, and VERSION clauses.

- **SCH** (SCHEMA) - physical database definition
- **FIL** (FILE) - non-database file definition as defined in the IDD ADD FILE
- **TAS** (TASK) - online processing unit identified by the ADD TASK command in SYSGEN
- **SUB** (SUBSCHEMA) - logical database definition
- **USE** (USER) - CA-IDMS/DC user definition
- **DES** (DESTINATION) - a SYSGEN component
- **REC** (RECORD) - grouping of data elements
- **SYS** (SYSTEM) - CA-IDMS/CV system-id built by the SYSGEN compiler
- **APO** (APPLICATION - OLD) - old (pre-CA-IDMS Release 12.0) ADSA application definition
- **SET** (SET) - schema compiler set definition
- **DIA** (DIALOG) - as built by ADS/O
- **APP** (APPLICATION) - ADSA application definition
- **ELE** (ELEMENT) - unit of data; data element
- **QFI** (QFILE) - predefined group of OLQ commands
- **PRC** (PROCESS) - ADS program unit
- **TAB** (TABLE) - as defined in the IDD ADD TABLE command
- **FUN** (FUNCTION) - ADSA internal transaction-id
- **MOD** (MODULE) - As defined in the IDD ADD MODULE command

- **PHY** (PHYSICAL-TERMINAL) - CA-IDMS/DC terminal definition as defined in SYSGEN
- **CLA** (CLASS) - as defined in the IDD ADD CLASS
- **ATT** (ATTRIBUTE) - as defined in the IDD ADD ATTRIBUTE
- **MAP** (MAP) - CA-IDMS/DC screen format definition
- **LOG** (LOGICAL-TERMINA) - CA-IDMS/DC logical terminal as defined in SYSGEN
- **LIN** (LINE) - line as defined in SYSGEN
- **MSG** (MESSAGE) - as defined in the IDD ADD MESSAGE
- **LOA** (LOAD MODULE) - executable group of machine instructions stored in the dictionary load area
- **LR** (LOGICAL-RECORD) - group of physical database records tied together by predefined CA-IDMS DML commands, as defined in the subschema
- **PRO** (PROGRAM) - as defined in the SYSGEN ADD PROGRAM statement

CCDB ENTITY TYPES

These types can be specified in the MODS and A-OPT clauses of the ADD SECURITY CLASS and MODIFY SECURITY CLASS commands, and in the TYPE clause of all commands containing ENTITY NAME, TYPE, and VERSION clauses.

- **CCD** (CCDB) - change control data base
- **DIC** (DICTIONARY) - CA-Endevor/DB Batch DICTIONARY command, MIS Online=8 function
- **EUS** (ENDEVOR-USER) - CA-Endevor/DB Batch USER command, MIS Online=7 function
- **CCI** (CCID) - CA-Endevor/DB Batch CCID command, MIS Online=5 function
- **MGR** (MANAGEMENT GROUP) - CA-Endevor/DB Batch MANAGEMENT GROUP command, MIS Online=9 function
- **STA** (STATUS) - CA-Endevor/DB Batch STATUS command, MIS Online=6 function
- **SEC** (SECURITY CLASS) - CA-Endevor/DB Batch SECURITY CLASS command, MIS Online=10 function

Index

B

Batch coding conventions, 1-5
Batch syntax conventions, 1-5

C

CA-Endevor/DB
description of, 1-1
CA-Endevor/DB Batch
description of, 1-2
features, 1-2
processing flow, 1-3
reserved words, A-1
CA-IDMS/CV Batch JCL, 5-1
CCDB entity types, B-2
CCID command, 3-5
processing flow, 3-5
syntax, 3-6
syntax rules, 3-6
usage, 3-7
Coding conventions
batch, 1-5
Commands and clauses
description of, 1-4

D

DICTIONARY command, 3-9
processing flow, 3-9
syntax, 3-10
syntax rules, 3-11

usage, 3-13

Dictionary entity types, B-1

DISPLAY/PUNCH CCDB-entity-type, 3-2
processing flow, 3-2
syntax, 3-3
syntax rules, 3-3

E

End-of-Job (EOJ) report, 1-3
ENTITY command, 3-15
processing flow, 3-15
syntax, 3-16
syntax rules, 3-16
usage, 3-17

L

Local mode JCL, 5-2

M

MANAGEMENT GROUP
processing flow, 3-20
MANAGEMENT GROUP command, 3-20
syntax, 3-21
syntax rules, 3-21

O

OS/MVS JCL, 5-1

P

PREAUTHORIZATION command, 3-23

processing flow, 3-23

syntax, 3-24

syntax rules, 3-24

usage, 3-26

PROCESS mode, 3-1

PUNCH mode, 3-1

ADD verb, 4-3

DELETE verb, 4-5

description of, 4-1

MODIFY verb, 4-4

processing flow, 4-3

syntax, 4-2

DICTIONARY command, 3-10

DISPLAY/PUNCH CCDB-entity-type verb, 3-3

ENTITY command, 3-16

MANAGEMENT GROUP command, 3-21

PREAUTHORIZATION command, 3-24

SECURITY CLASS command, 3-29

SET OPTIONS command, 2-5

SIGNIN command, 3-40

SIGNON command, 2-3

SIGNOUT command, 3-43

STATUS command, 3-47

USER command, 3-50

Syntax conventions

batch, 1-5

S

SECURITY CLASS command, 3-28

processing flow, 3-28

syntax, 3-29

syntax rules, 3-31

usage, 3-38

SET OPTIONS command, 2-5

syntax, 2-5

syntax rules, 2-5

SIGNIN command, 3-40

processing flow, 3-40

syntax, 3-40

syntax rules, 3-41

usage, 3-41

SIGNON command, 2-2

processing flow, 2-3

syntax, 2-3

syntax rules, 2-4

SIGNOUT command, 3-43

processing flow, 3-43

syntax, 3-43

syntax rules, 3-43

usage, 3-44

STATUS command, 3-46

processing flow, 3-46

syntax, 3-47

syntax rules, 3-47

Syntax

CCID command, 3-6

U

USER command, 3-49

processing flow, 3-49

syntax, 3-50

syntax rules, 3-50

usage, 3-51