CA IDMS - 19.0
Administrating CA Endevor and DB for CA IDMS

Date: 15-Jan-2018
This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

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 YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2017 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.
# Table of Contents

## Monitoring Change Activity ............................. 11
- Change Manager Architecture ............................. 11
- Establishing a CCDB ........................................... 11
- Linking a Dictionary to the Change Monitor .............. 12
- DBName Table Coding Rules ............................... 13
- Seeding the CCDB ............................................... 14
- Change Monitor Configuration ............................. 14
- Configuration Table ........................................... 15
  - Sample JCL for NDVRCNFG ................................. 18
  - NDVRCNFG Coding Examples ............................... 19
- Change Monitor Operations ............................... 19
  - Current Monitor Access Mode ............................. 20
    - Pending Monitor Access Mode ............................. 21
  - Security Processing ......................................... 21
  - Change Logging ............................................... 21

## Security System Overview ............................. 22
- Security System Architecture ............................. 22
- Security Facilities ........................................... 23
- Security Components ........................................ 25
- Security Implementation .................................... 26
  - Entity Type Monitoring .................................... 26
    - Security Classifications ................................. 26
    - Use of the CA Endevor/DB Batch Front End .......... 27
    - AUTO SIGNOUT ............................................... 27
    - STATUS ......................................................... 28
    - Preauthorization ............................................ 28
    - LOCK .......................................................... 29
- Security Classes and Levels ............................. 29
  - Security Class Options ..................................... 30
- Sign-on ............................................................ 31
- Change Monitor Processing ............................... 33
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Security</td>
<td>35</td>
</tr>
<tr>
<td>Setting Up a Security Administrator</td>
<td>35</td>
</tr>
<tr>
<td>Disallowing Batch Processing</td>
<td>37</td>
</tr>
<tr>
<td>Setting Up Global Dictionary Options</td>
<td>38</td>
</tr>
<tr>
<td>Security Class Maintenance</td>
<td>41</td>
</tr>
<tr>
<td>Establishing and Maintaining Security Classes</td>
<td>41</td>
</tr>
<tr>
<td>NDVRUA10 Field Descriptions</td>
<td>42</td>
</tr>
<tr>
<td>NDVRMA10 Field Descriptions</td>
<td>42</td>
</tr>
<tr>
<td>Security Preauthorization</td>
<td>46</td>
</tr>
<tr>
<td>Restricting Users Through Preauthorization</td>
<td>47</td>
</tr>
<tr>
<td>Protecting Critical Entities Through Preauthorization</td>
<td>51</td>
</tr>
<tr>
<td>Restricting Access to a CCID Through Preauthorization</td>
<td>56</td>
</tr>
<tr>
<td>Assigning Status Privileges Through Preauthorization</td>
<td>58</td>
</tr>
<tr>
<td>Preparing for Derived CCID Processing</td>
<td>60</td>
</tr>
<tr>
<td>Lock Security</td>
<td>64</td>
</tr>
<tr>
<td>The Browse Option</td>
<td>65</td>
</tr>
<tr>
<td>The Lock Option</td>
<td>66</td>
</tr>
<tr>
<td>An Alternate Procedure</td>
<td>67</td>
</tr>
<tr>
<td>The Unlock Option</td>
<td>67</td>
</tr>
<tr>
<td>An Alternate Procedure</td>
<td>68</td>
</tr>
<tr>
<td>CCIDs and Dictionaries</td>
<td>68</td>
</tr>
<tr>
<td>Archiving and Compressing the CCDB</td>
<td>69</td>
</tr>
<tr>
<td>NDVRARCO Command Language</td>
<td>71</td>
</tr>
<tr>
<td>SIGNON</td>
<td>71</td>
</tr>
<tr>
<td>COMPRESS</td>
<td>72</td>
</tr>
<tr>
<td>ARCHIVE</td>
<td>73</td>
</tr>
<tr>
<td>ALTER CONFIRMATION</td>
<td>73</td>
</tr>
<tr>
<td>Running NDVRARCO</td>
<td>74</td>
</tr>
<tr>
<td>NDVRARCO Output</td>
<td>74</td>
</tr>
<tr>
<td>Sample JCL and Syntax</td>
<td>76</td>
</tr>
<tr>
<td>Sample z/OS and OS/390 JCL</td>
<td>76</td>
</tr>
</tbody>
</table>
Promotion Support Facilities ................................................................. 77
Selection for Migration ........................................................................ 78
Target System Impact Analysis ............................................................. 79
Entity Migration .................................................................................. 80
Target System Audit Trail Creation ....................................................... 81
Source System Audit Trail Creation ....................................................... 82
Programs and Security for Migration ................................................... 83
System Identification .......................................................................... 83
Source System Validation and Integrity Checking Rules ........................ 84
  NDVRDSEL Selection and Verification .............................................. 84
  STRATEGY 1 - CHANGE RELATIONSHIPS ..................................... 85
  STRATEGY 2 - HIERARCHY RELATIONSHIPS ................................. 85
  STRATEGY 3 - MINIMUM RELATIONSHIPS .................................... 86
Strategy Comparison .......................................................................... 86
NDVRDSEL Command Syntax .............................................................. 89
  NDVRDSEL Sample JCL ................................................................ 95
    Sample z/OS and OS/390 JCL ....................................................... 96
NDVRDSEL Outputs .......................................................................... 96
  NDVRDSEL Output File (ddname NDVRENO) .................................... 97
  NDVRDSEL Control Report (ddname NDVRLST) ............................... 98
  NDVRDSEL Input Command Listing ............................................... 99
  NDVRDSEL Compiled Command Listing ....................................... 99
  NDVRDSEL Entity List Exception Listing ......................................... 100
  Report Fields .............................................................................. 100
  NDVRDSEL End-of-Job Statistics .................................................... 101
  NDVRDSEL Detail Report (ddname NDVRDTL) ................................. 102
  NDVRDSEL Utility Report (ddname NDVRUTL) ............................... 103
NDVRDCOR Correlation ................................................................. 104
  Target System Impact Analysis Rules ............................................. 106
NDVRDCOR Command Syntax .......................................................... 107
NDVRDCOR Sample JCL ................................................................ 108
  Sample z/OS and OS/390 JCL ....................................................... 109
NDVRDCOR Outputs ...................................................................... 109
  NDVRDCOR Control Report (ddname NDVRLST) ............................ 109
  NDVRDCOR - Input Command Listing ............................................ 110
  NDVRDCOR - Compiled Command Listing .................................... 110
  NDVRDCOR - Migration Exception Report ..................................... 111
  Report Fields ............................................................................ 111
  NDVRDCOR - Expansion Exception Report .................................... 112
  NDVRDCOR - End-of-Job Statistics ................................................. 112
<p>| NDVRDCOR Detail Report (ddname NDVRDTL) | 113 |
| NDVRDCOR - End-of-Job Statistics | 114 |
| NDVRDCOR Utility Report (ddname NDVRUTL) | 114 |
| NDVRDCOR - Input Entity List File | 114 |
| NDVRDCOR - Target Entity Exceptions | 115 |
| NDVRDCOR - End-of-Job Statistics | 115 |
| NDVRDLVR Definition Delivery | 115 |
| NDVRDLVR Command Syntax | 116 |
| NDVRDLVR Sample JCL | 121 |
| Sample z/OS and OS/390 JCL | 121 |
| NDVRDLVR Outputs | 123 |
| NDVRDLVR Output Files | 123 |
| File Requirements | 123 |
| NDVRDLVR Control Report (ddname NDVRLST) | 124 |
| NDVRDLVR - Processing Summary | 125 |
| NDVRBOOK in Migration Mode | 126 |
| NDVRBOOK Command Syntax | 127 |
| NDVRBOOK Outputs | 128 |
| Importing Entities Exported by NDVRDLVR | 129 |
| Order of Compiler Execution | 129 |
| NDVRBOOK Migration JCL (Source) | 130 |
| Sample z/OS and OS/390 JCL | 130 |
| NDVRBOOK Migration JCL (Executable) | 134 |
| Sample z/OS and OS/390 JCL | 134 |
| NDVRBOOK Generic Migration JCL (any program) | 135 |
| Sample z/OS and OS/390 JCL | 135 |
| NDVRDCF1 Target Confirmation | 135 |
| NDVRDCF1 Command Syntax | 136 |
| NDVRDCF1 Sample JCL | 137 |
| Sample z/OS and OS/390 JCL | 137 |
| NDVRDCF1 Outputs | 137 |
| NDVRDCF1 Output File (ddname NDVRENO) | 138 |
| NDVRDCF1 Control Report (ddname NDVRLST) | 138 |
| NDVRDCF1 - Input Command Listing | 138 |
| NDVRDCF1 - Input Entity List Header Report | 139 |
| NDVRDCF1 - End-of-Job Statistics | 139 |
| NDVRDCF1 Detail Report (ddname NDVRDTL) | 140 |
| NDVRDCF1 - Output Confirmation File Report | 140 |
| NDVRDCF2 Source Confirmation | 141 |
| NDVRDCF2 Command Syntax | 141 |
| NDVRDCF2 Sample JCL | 142 |
| Sample z/OS and OS/390 JCL | 142 |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDVRDCF2 Outputs</td>
<td>143</td>
</tr>
<tr>
<td>NDVRDCF2- Input Command Listing</td>
<td>143</td>
</tr>
<tr>
<td>NDVRDCF2- Entity File Listing</td>
<td>143</td>
</tr>
<tr>
<td>NDVRDCF2- End-of-Job Statistics</td>
<td>144</td>
</tr>
<tr>
<td>The Source Code Comparator</td>
<td>144</td>
</tr>
<tr>
<td>Running the Comparator in Stand-alone Mode</td>
<td>145</td>
</tr>
<tr>
<td>JCL</td>
<td>146</td>
</tr>
<tr>
<td>Sample z/OS and OS/390 JCL</td>
<td>146</td>
</tr>
<tr>
<td>SAMPLE z/OS and OS/390 JCL</td>
<td>147</td>
</tr>
<tr>
<td>z/OS and OS/390 JCL</td>
<td>148</td>
</tr>
<tr>
<td>NDVRRJCL Command Syntax</td>
<td>148</td>
</tr>
<tr>
<td>NDVRRJCL Inputs</td>
<td>149</td>
</tr>
<tr>
<td>NDVRRJCL Outputs</td>
<td>150</td>
</tr>
<tr>
<td>Return Codes</td>
<td>150</td>
</tr>
<tr>
<td>Running NDVRCOMP</td>
<td>151</td>
</tr>
<tr>
<td>Control Card Specifications</td>
<td>151</td>
</tr>
<tr>
<td>Syntax</td>
<td>151</td>
</tr>
<tr>
<td>Input Parameters</td>
<td>152</td>
</tr>
<tr>
<td>Sample Outputs</td>
<td>153</td>
</tr>
<tr>
<td>Return Codes</td>
<td>156</td>
</tr>
<tr>
<td>Running the Comparator in Migration Mode</td>
<td>156</td>
</tr>
<tr>
<td>NDVRDCMP Command Syntax</td>
<td>156</td>
</tr>
<tr>
<td>JCL</td>
<td>161</td>
</tr>
<tr>
<td>NDVRDCMP Inputs</td>
<td>161</td>
</tr>
<tr>
<td>NDVRDCMP Outputs</td>
<td>162</td>
</tr>
<tr>
<td>NDVRDCMP - Input Command Listing</td>
<td>162</td>
</tr>
<tr>
<td>NDVRDCMP - Entity Comparison Listing</td>
<td>162</td>
</tr>
<tr>
<td>NDVRDCMP - Index to Entity Listing</td>
<td>163</td>
</tr>
<tr>
<td>NDVRDCMP - Processing Summary</td>
<td>164</td>
</tr>
<tr>
<td>The JCL Converter</td>
<td>165</td>
</tr>
<tr>
<td>Why JCL Needs To Be Converted</td>
<td>165</td>
</tr>
<tr>
<td>Security Menu Mask Definitions</td>
<td>166</td>
</tr>
<tr>
<td>Mask Values</td>
<td>166</td>
</tr>
</tbody>
</table>
Online/Batch Control Flags ........................................................................ 168
CA Endevor/DB Change Monitor authorizes and logs updates to any IDD in an accompanying Change Control Database (CCDB). Logically, the CCDB is an extension of the dictionary. Updates to the CCDB and the dictionary being monitored are kept in synchronization through the CA Endevor/DB system software. This section describes the administration of Endevor/DB for CA IDMS.

For more information, see the following topics:

- Monitoring Change Activity (see page 11)
- Security System Overview (see page 22)
- Global Security (see page 35)
- Security Class Maintenance (see page 41)
- Security Preauthorization (see page 46)
- Lock Security (see page 64)
- Archiving and Compressing the CCDB (see page 69)
- Promotion Support Facilities (see page 77)
- The Source Code Comparator (see page 144)
- The JCL Converter (see page 165)
- Security Menu Mask Definitions (see page 166)
- Online/Batch Control Flags (see page 168)
Monitoring Change Activity

**Change Monitor** operates as a standard CA IDMS database procedure through the subschema used to update the dictionary. The Change Monitor is “hooked” into a dictionary by mapping the CA Endevor/DB-supplied NDVRNWKA for the IDMSNWKA subschema. From this point forward all updates attempted through NDVRNWKA will be routed to the Change Monitor for logging and authorization. (Actual dictionary updates are performed by Computer Associates compilers.) For more information, see the following topics:

- Change Manager Architecture (see page 11)
- Establishing a CCDB (see page 11)
- Linking a Dictionary to the Change Monitor (see page 12)
- DBName Table Coding Rules (see page 13)
- Seeding the CCDB (see page 14)
- Change Monitor Configuration (see page 14)
- Configuration Table (see page 15)
- Change Monitor Operations (see page 19)

Change Manager Architecture

Once the Change Monitor receives control, it establishes the relationship between a dictionary and its CCDB through the DBNAME table. To achieve monitoring, the NDVRNWKA subschema, which describes the dictionary, as well as the NDVRSUBS subschema, which defines the CCDB, are mapped to a common DBNAME. The Change Monitor supports as many CCDBs as are defined in the DBNAME table. The Change Monitor decodes the DBNAME table at system startup to determine the proper pairing of CCDBs to dictionaries.

To summarize, change monitoring is initialized by mapping the IDMSNWKA subschema to the NDVRNWKA subschema and establishing a CCDB for the database names to be monitored.

**Note:** CA Endevor/DB cannot be used with Data Sharing and Dictionaries managed by CA Endevor/DB cannot be shared.

Establishing a CCDB

Installing a CCDB is analogous to installing a dictionary. JCL and parameters to accomplish these tasks are supplied with the installation media and are fully documented in the *CA IDMS Installation and Maintenance* -- z/OS section. Execute the following jobs from the installation procedure:
Job 6: Creates the segments, areas, and files for the Change Control Database (CCDB) and place in the Global DMCL. This step also specifies the IDMSNWKA to NDVRNWKA subschema mapping for each database name to be monitored.

Job 7: Allocates the Change Control Database.

Job 8: Punches and link-edits the local DMCL.

Job 9: Modifies the CV startup JCL to include the CA Endevor/DB load library.

Job 10: Assembles the Change Control Data Base server/data base name configuration table (NDVRCNFG) and links it to your CV load library.

Job 11: Cycles the CV.

Linking a Dictionary to the Change Monitor

The Change Monitor runs as a task named NDVRSERV under CV, or as a subtask of the operating system in local mode operation. The task definition and programs for the Change Monitor are provided on the CA Endevor/DB installation media. Once the installation procedure (shipped with the media) is executed, the Change Monitor is available to any dictionary for which the IDMSNWKA-to-NDVRNWKA subschema mapping has been specified (for the applicable database name in the table). It is only necessary to install the Change Monitor once per CV. Any number of dictionary/CCDB pairs can be run through the monitor.

Note: Disabling CA Endevor/DB is only recommended when unloading and reloading the dictionary for expansion purposes.

Figure 1.1 below depicts the flow of operation surrounding installation of CA Endevor/DB change monitoring on a dictionary.
DBName Table Coding Rules

During execution the Change Monitor detects the DICTNAME and SUBSHEMA name when an update to an IDD is requested. It then searches for this name in the DBNAME table to determine the appropriate CCDB to which activity should be logged.

The relationship between a dictionary (that is, an IDMSNWKA mapping) and a CCDB (that is, a xxxxCCDB segment) must be one-to-one. The same pair of mappings must always occur together under a DBNAME.

The DBName for the dictionary linked, or "hooked" to the Change Monitor should be the same as the Segment Name of the segment in the DMCL containing the dictionary's DDDLML area. Failure to do so may allow the Change Monitor to be bypassed if the segment name of the DDDLML is specified as a DBName or Dictname. A valid DBName for a "hooked" dictionary must include:

- Segment(s) describing the areas of a dictionary (DDDLML, DDLDCLOD, and, optionally, DDLDCMSG)
- Segment(s) describing all areas of a CCDB (NDVR-ADM, NDVR-LOG, and NDVR-PAK)
- Subschema mapping of IDMSNWKA to NDVRNWKA

For example, if the DMCL contains the following segment definitions:

```
SEGMENT APPLDICT
  AREA APPLDICT.DDLDML
  AREA APPLDICT.DDLDCLOD
SEGMENT SYSMSG
  AREA SYSMSG.DDLDCMSG
SEGMENT APPLCCDB
  AREA APPLCCDB.NDVR-ADM
  AREA APPLCCDB.NDVR-LOG
  AREA APPLCCDB.NDVR-PAK
```

The correct DBName definition to link the APPLDICT dictionary to the Change Monitor would be:

```
CREATE DBNAME dbtable-name.APPLDICT
  INCLUDE SEGMENT APPLDICT
  INCLUDE SEGMENT SYSMSG
  INCLUDE SEGMENT APPLCCDB
  SUBSCHEMA IDMSNWKA MAPS TO NDVRNWKA
  SUBSCHEMA NDVRSUBS MAPS TO NDVRSUBS;
```

If APPLDICT is the default dictionary for the system, in addition to the above, you must establish subschema mappings at the DBTable level as follows:

```
MODIFY DBTABLE dbtable-name
  SUBSCHEMA IDMSNWKA MAPS TO NDVRNWKA
  DBNAME APPLDICT
  SUBSCHEMA IDMSNWK? MAPS TO IDMSNWK?
  DBNAME APPLDICT
```

Note: Pointing two separate dictionaries at the same CCDB is not permitted. View the CCDB as a logical and physical extension to the dictionary.
Seeding the CCDB

When the CA Endevor/DB system encounters an empty Change Control Database (CCDB), it loads the first Dictionary Descriptor and Security Class records automatically. The initial CCDB is under the control of a Security Class named NDVR-GLOBAL. NDVR-GLOBAL enables all menu options, allows modification without a CCID or userid, and has preassignment and Auto-Signout turned off. The Dictionary record is named after the DBNAME of the dictionary for which the CCDB was established. The initial global dictionary options specify no UserID required, no password required, Auto-User, and a default Security Class of NDVR-DDA. Therefore, the first user will be running under the restrictions imposed by NDVR-DDA since Auto-User will add the first user. This prevents users from accidentally gaining access to the CA Endevor/DB Security System while the system is being set up by the installer.

After establishing a physical CCDB, the CA Endevor/DB Online front end must be used to assign a security administrator, assign a System Identifier name, and set default security options. The first time the CA Endevor/DB Online front end is used, the system will encounter the empty Change Control Database (CCDB) and load the dictionary and security class records described above.

To establish a security administrator, refer to Section3, "Global Security."

After setting up the security administrator, the dictionary and security options may be changed to conform to a specific installation. Options such as requiring a password can be specified for the dictionary, and permissions can be modified for each security class. The ability to use the CA Endevor/DB Batch front end is initially disabled for the loaded security classes and must be enabled to use this facility. To enable the Batch front end, refer to Section3, "Global Security."

Change Monitor Configuration

The Change Monitor runs with a driver task under CV, or as a subtask of the operating system in local mode operation. For CV, the task definition and programs for the driver are provided on the CA Endevor/DB installation media (in dictionary member CA-ENDEVOR-DB-SYSGEN). Once the installation procedure shipped with the media is executed, the Change Monitor and driver are available to any dictionary which has been "hooked" through the IDMSNWKA-to-NDVRNWKA mapping. Any number of dictionary/CCDB pairs can be run through the monitor.

During execution, the Change Monitor automatically attaches a driver task (named NDVRSERV under CV) when it is first invoked. The driver task opens one protected update run unit (on demand) for each CCDB under its control. While this run unit is active, NDVRSERV has exclusive control of the CCDB. This run unit remains active throughout the day and is responsible for all updates performed against the CCDB.

Figure 1.2 depicts the run time architecture of the CA Endevor/DB system.
IDMSDB--Change Monitor Configuration

Configuration Table

As part of the initial installation procedure (STEP 3), a null configuration table (load module NDVRCNFG) was placed in the CA Endevor/DB load library with other executable components of the system. The NDVRCNFG table can be optionally modified to:

- Assign base names when multiple DBNAMEs point to the same dictionary and CCDB.
- Cause the Change Monitor to attach more than one driver task.
- Configure CCDBs to driver tasks.
When running with the null NDVRCNFG table supplied at installation, all CCDBs defined to a CV run with default base names, and run under one driver task as depicted in Figure 1.2. This is more than adequate for the majority of installations and need not be modified. It must be kept in mind that CCDB activity is extremely minor in comparison to the work performed in the IDD.

The configuration table is a simple two-entry table with a Driver Number in column one and a DBNAME in column two. The DBNAME in the table is used as the base name (DICTNAME) for the CCDB/IDD pair to which it belongs. The Driver Number is a digit from 1 to 9 that specifies the driver task assigned to handle the CCDB for that DBNAME. If a Driver Number is not specified, it defaults to 0.

When the driver task detects a newly initialized CCDB, it automatically primes the database with the records required for execution. One of the records stored in the CCDB (the Dictionary Record) contains the DICTNAME of its associated IDD. However, if the DICTNAME is changed (by a new DBNAME table) after initial install, the CCDB is automatically updated by the driver to reflect the new name. These are normally straightforward operations, except where two or more DICTNAMEs point to the same IDMS dictionary at initial startup or when renaming (by a new DBNAME table).

**Note:** If a site does not use multiple DBNAMES to point to the same IDMS dictionary, base name setup can be ignored.

The *base name* is stored internally in the CCDB Dictionary Record by the driver at startup. Only one base name can exist per dictionary. The base name displays at the top of all CA Endevor/DB Online screens (in the header area) regardless of the DICTNAME used to gain access to that dictionary (for example, through DCUF SET DICTNAME or CA Endevor/DB Signon). The default base name for a CCDB will be the lowest in collating sequence unless the system is instructed otherwise (as explained below). It is important to note that the primary dictionary is always known as '' under these rules regardless of coding in the DBNAME table. Base names become important when using the Promotion Support Facilities (see Section8, "Promotion Support Facilities") to create migration audit trail records in the CCDB.

Figure 1.3 depicts a system running with a NDVRCNFG table.
Notice that DBNAMEs (DCT1 and DCT2) both point to the same CCDB. Since DCT2 is coded in the NDVRCNFG table, DCT2 is the base name. DCT1 becomes an alias name. All non-base names become alias names. DCT1 will function normally (in DCUF commands and CA Endevor/DB search criteria) in all respects. When an alias name is used to sign-on, or in DCUF commands prior to using CA Endevor/DB online screens, the base name will be echoed in the screen header DICTNAME field.

Also notice in Figure 1.3 that a driver number other than 0 was specified with DCT2. This will cause the Change Monitor to establish another task, and assign it DCT2's CCDB run unit. Although the example shows a separate task assigned to DCT2, this is not necessary. If the Driver Number for DCT2 was specified as 0 or omitted, DCT2 would have its CCDB run unit managed by driver 0.

The following rules apply to this table:

- The DBNAME specified in the NDVRCNFG table becomes the base name for that CCDB.
- A given DBNAME can only appear once in the table.
All CCDBs not assigned through the NDVRCNFG table are assigned to driver 0.

At system startup, the NDVRCNFG Table is loaded and analyzed along with the DBNAME table. The following actions take place:

- The table is edited for consistency, and appropriate diagnostics are produced.
- All CCDBs are assigned base names and/or assigned to drivers according to the table.
- If any CCDBs are unassigned after the NDVRCNFG table is processed (those coded in the DBNAME table but not entered in the NDVRCNFG table), they are allocated to driver 0 and assigned the default base name (lowest in collating sequence).
- All drivers specified and the default driver 0 (if needed) are started. If all the CCDBs are allocated to other drivers, driver 0 will not be started. Run units for CCDBs will be bound by the appropriate drivers when the first access request is made.

**Note:** Under local mode, one driver subtask is started for the CCDB being accessed. Be sure to supply the correct NDVRCNFG table and IDMSDBTB in the job step.

### Sample JCL for NDVRCNFG

To assign base names, and/or establish more than one driver and assign CCDBs to drivers, Job 10 is supplied on the CA Endevor/DB installation media. If the job from the install library is used, all substitutable parameters except for *driver no* and *base name* were established as part of the installation procedure.

```jcl
//JOBNAME JOB (AAA),'JOB 10 INSTALLATION',CLASS=A, 
// MSGCLASS=X 
//*****************************************************************************/ 
//* STEP 1: ASSEMBLE CCDB SERVER/DBNAME CONFIGURATION TABLE. */ 
//*****************************************************************************/ 
//STEP1 EXEC PGM=your.assembler,REGION=2048K, 
// PARM='DECK,NOLOAD,NORLD,NOXREF' 
//STEPLIB DD DSN=ndms.loadlib, 
// DISP=SHR 
//SYSPRINT DD SYSOUT=* 
//SYSLIB DD DSN=ndvrdb.srclib, 
// DISP=SHR 
//SYSPUNCH DD DSN=6.&ASMP.,DISP=(NEW,PASS),UNIT=TDISK, 
// DCB=BLKSIZE=80,SPACE=(TRK,(5,1)) 
//SYSUT1 DD DSN=6.&ASMWRK1.,UNIT=TDISK,SPACE=(TRK,(5,1)) 
//SYSUT2 DD DSN=6.&ASMWRK2.,UNIT=TDISK,SPACE=(TRK,(5,1)) 
//SYSUT3 DD DSN=6.&ASMWRK3.,UNIT=TDISK,SPACE=(TRK,(5,1)) 
//SYSIN DD DSN=ndvrdb.srclib, 
// DISP=SHR 
//*****************************************************************************/ 
//* LINK THE NDVRCNFG TABLE: */ 
//*****************************************************************************/ 
//STEP2 EXEC PGM=your.linkeditor,REGION=2048K,COND=(4,LE), 
// PARM='RENT,LET,LIST,MAP,XREF,SIZE=(256K,64K),NCAL' 
//SYSPRINT DD SYSOUT=* 
//SYSUT1 DD DSN=6.&LNKWORK.,UNIT=TDISK,SPACE=(CYL,(5,1)) 
//SYSLMOD DD DSN=usrvcv.loadlib (NDVRCNFG), 
// DISP=SHR 
```
NDVRCNFG Coding Examples

To establish base names for a dictionary pointed to by more than one DICTNAME, code:

```plaintext
//SYSIN DD *
NDVRSERV DBNAME=DBNAMEA set up first CCDB name
NDVRSERV DBNAME=DBNAMEB set up second CCDB name
NDVRSERV END=YES
END
```

To establish two driver tasks and run DBNAME DB1 and DB2 under a common separate driver, code:

```plaintext
//SYSIN DD *
NDVRSERV SERVER=1, DBNAME=DB1
NDVRSERV SERVER=1, DBNAME=DB2
NDVRSERV END=YES
END
```

All DBNAMEs not specified above will go to driver 0.

To establish three driver tasks and assign DB1 to one separate driver and DB2 to another separate driver, code:

```plaintext
//SYSIN DD *
NDVRSERV SERVER=1, DBNAME=DB1
NDVRSERV SERVER=2, DBNAME=DB2
NDVRSERV END=YES
END
```

All DBNAMEs not specified above, but defined to the DBNAME table will go to driver 0.

Change Monitor Operations

Contents
- Current Monitor Access Mode (see page 20)
  - Pending Monitor Access Mode (see page 21)
  - Security Processing (see page 21)
  - Change Logging (see page 21)

Under CV, an authorized user can employ the CA Endevor/DB online system to turn the Change Monitor, the CCDB, or the Security System on or off for a dictionary. This may be required in emergency situations when there is a critical need to bypass security.

Note: Since the NDVRSERV driver task(s) keeps an open run unit for each active CCDB, the areas in the CCDB cannot be varied offline by DCMT commands until the run unit bound by task NDVRSERV under program NDVRFLIO completes. The Control Functions Menu allows the user to release the CCDB run unit via an online command.

To turn off the monitor, security, or release a CCDB run unit:
1. Sign-on to CA Endevor/DB by entering NDVRMIS at the NEXT TASK CODE prompt.

2. Enter OPTION ==>10 (CONTROL FUNCTIONS) on the MAIN MENU.
   The System Control Functions menu appears:

   CA-E/DB nn.n volser    CA-ENDEVOR/DB SYSTEM CONTROL FUNCTIONS mm/dd
   /yy NDVRUA00
   USER ===> EDBADMIN    DICTNAME ===> SRCNDVR    MODE ===> UPDATE
   OPTION ===> 8
   1 - BROWSE CCDB DESCRIPTOR RECORD    2 - MODIFY CCDB DESCRIPTOR RECORD
   3 - BROWSE SECURITY DESCRIPTORS      4 - ADD A SECURITY DESCRIPTOR
   5 - MODIFY SECURITY DESCRIPTORS      6 - DELETE SECURITY DESCRIPTORS
   7 - BROWSE MONITOR DICT STAT BLOCKS  8 - MODIFY MONITOR DICT STAT BLOCKS
   SECURITY CLASS ===> (IF OPTIONS 3, 4, 5, 6)
   DICTNAME ===> SRCNDVR (IF OPTIONS 7, 8)

3. Enter OPTION ==>8 (MODIFY MONITOR DICT STAT BLOCKS).

4. Enter the base name of the CCDB to be varied in the DICTNAME field.
   The Monitor DICT Status Block Detail screen appears:

   CA-E/DB nn.n volser    MONITOR DICT STATUS BLOCK DETAIL    mm/dd
   /yy NDVRMA30
   USER ===> EDBADMIN    DICTNAME ===> SRCNDVR    ACTION ===> MODIFY
   DBNAME ===> SRCNDVR
   CURRENT MONITOR ACCESS MODE ===> U    PENDING MONITOR ACCESS MODE ===> O
   CA-E/DB SECURITY PROCESSING ===> Y    CA-E/DB CHANGE LOGGING ===> Y

   If you left the DICTNAME blank on the prior screen, a list of all the active CCDBs running under the
   driver task for the dictionary under which you are operating will appear. In this case, place a non-
   blank character next to the correct DICTIONARY and press ENTER to get to the screen shown above.
   This screen contains Change Monitor control information for the specified dictionary.

Current Monitor Access Mode

This field displays the current status of the Change Monitor.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>Update mode. In update mode the CCDB areas are readied in update mode.</td>
</tr>
<tr>
<td>R</td>
<td>Retrieval mode. In retrieval mode the CCDB areas are readied in retrieval. <strong>Note:</strong> This mode would only be used when an emergency offline process needs to be run against the CCDB while the dictionary is still available for update. No Change Logging is accomplished.</td>
</tr>
<tr>
<td>O</td>
<td>Offline. In offline mode the Change Monitor does not ready the CCDB areas. No CCDB or dictionary processing can take place in the CV. <strong>Note:</strong> This mode would only be used in an emergency situation when the CCDB is required of an offline updating process. O must be specified before the CCDB can be varied offline with DCMT commands.</td>
</tr>
</tbody>
</table>

The following table summarizes the actions, which the CA Endevor/DB Change Monitor performs for various combinations of the Dictionary Status Block settings:
<table>
<thead>
<tr>
<th>Access Mode</th>
<th>Security Processing</th>
<th>Change Logging</th>
<th>Action Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>- - -</td>
<td>- - -</td>
<td>Fail all requests</td>
</tr>
<tr>
<td>R</td>
<td>N</td>
<td>N</td>
<td>Allow all requests; log nothing</td>
</tr>
<tr>
<td>R</td>
<td>- - -</td>
<td>Y</td>
<td>This combination is not allowed by CA Endevor/DB</td>
</tr>
<tr>
<td>R</td>
<td>Y</td>
<td>N</td>
<td>Do security</td>
</tr>
<tr>
<td>U</td>
<td>N</td>
<td>N</td>
<td>Allow all requests; log nothing</td>
</tr>
<tr>
<td>U</td>
<td>N</td>
<td>Y</td>
<td>Allow all requests; log all updates</td>
</tr>
<tr>
<td>U</td>
<td>Y</td>
<td>N</td>
<td>Do security; log nothing</td>
</tr>
</tbody>
</table>

**Pending Monitor Access Mode**

Set this field to U (Update), R (Retrieval), or O (Offline) to change the CURRENT MONITOR ACCESS MODE status. The requested action will take effect as soon as work-in-progress is completed. Setting this field to R automatically turns off Change Logging. Setting this field to O automatically turns off Change Logging and Security Processing. It is highly recommended that the Data Dictionary be varied offline or retrieval before setting the CCDB to O or R.

**Important!** When the mode is reset to U, after being set to O or R, the Security and Logging switches are left at N. Remember to set these switches to Ys for full CA Endevor/DB services to be in effect. Vary the Data Dictionary to update mode after enabling CA Endevor/DB services.

**Security Processing**

Current status of the security system for this dictionary. To modify, set this field to Y or N.

- **Y** -- Security system is enabled
- **N** -- Security system is disabled. All updates are allowed to process regardless of signout, lock, or preauthorization. No signouts are attempted.

**Change Logging**

Current status of the change monitor for this dictionary. To modify, set this field to Y or N.

- **Y** -- Change Log Entries are created in the CCDB when entities are updated.
- **N** -- Change Log Entries are not created in the CCDB when entities are updated. Vary the Data Dictionary areas to retrieval mode before turning off logging.
Security System Overview

The CA Endevor/DB Security System enhances IDD-level security enforcement. All CA Endevor/DB users and the Change Monitor run under control of the CA Endevor/DB Security System. Conceptually, the CA Endevor/DB Security System resides between the CA IDMS compilers and the dictionary. At execution time, all requests to modify entities in the CCDB or the IDD are cleared through CA Endevor/DB security. After passing CA Endevor/DB security edits, a dictionary update request is then passed to the IDD to be handled in the usual manner.

- Security System Architecture (see page 22)
- Security Facilities (see page 23)
- Security Components (see page 25)
- Security Implementation (see page 26)
- Security Classes and Levels (see page 29)
- Sign-on (see page 31)
- Change Monitor Processing (see page 33)

Security System Architecture

If the user has already established IDD-level security, CA Endevor/DB adds a layer of additional capability on top of that already set up. If no IDD security is in effect, CA Endevor/DB security can be used exclusively to control updates to IDD entities. The security administrator can establish any combination of CA Endevor/DB security, as well as IDD options.

Please note: No security is enforced when dictionary entities are retrieved immediately prior to update. CA Endevor/DB security is enforced when the update actually occurs. If you fail IDD security, the system generates an IDD error message indicating that you cannot perform that action. If you fail CA Endevor/DB security, the system generates an CA Endevor/DB error message indicating that you cannot perform that action. Immediately after that message, the system displays an IDD error message telling you that the update has failed.

Figure 2.1 illustrates the conceptual flow of requests through the CA Endevor/DB Security System. Both the CA Endevor/DB Online System and the CA IDMS compilers gain authorization to update entities through CA Endevor/DB.
The CA Endevor/DB Security System determines overall processing options through procedure enforcement, entity attributes, and discrete function masks. These concepts are explained in more detail below.

Security Facilities

The CA Endevor/DB Security System is designed to prevent unauthorized users from modifying entities in the CCDB and/or IDD. There are several reasons for wanting to closely control who modifies what. For example:

- Some entities may be critical or sensitive, and should only be updated by specific users (for example, disbursement programs).

- Some users may be restricted to updating only specific types of entities (for example, programmers may not be allowed to modify schema definitions).

- Some users may be restricted to updating only specific entity occurrences (for example, trainee programmers may be restricted to only updating the classroom study applications).

- Some shops may want to enforce a rule allowing only one programmer (or one group of programmers) to work on an entity at a time. In this case, once someone modifies an entity, no one else is allowed to modify it until the "initial modifier" releases it.

- Certain actions may only be performed by appropriate personnel (for example, migration).

The CA Endevor/DB Security System provides these capabilities through special security facilities.
Note: No CA Endevor/DB security facility is required - they are all optional. If a facility is not needed at a site, it can be disabled.

- **Entity Type MONITOR Flags**
  For each entity type that CA Endevor/DB handles, there is a MONITOR flag. The effects of these flags are global: if you set a MONITOR flag to N, then CA Endevor/DB simply ceases to pay any attention to that entity type. At installation time, all MONITOR flags are Y.

- **SIGNON Rules**
  There are various rules that govern the CA Endevor/DB signon process. These rules can be determined on a user-by-user basis, or a CCID-by-CCID basis, or can be set globally. These rules determine:
  - Whether a USER must be predefined in the CCDB before SIGNON is allowed;
  - Whether a password must be correctly specified before SIGNON is allowed;
  - Whether the CA IDMS/DC userid must be used as the CA Endevor/DB userid;
  - Whether changes by the user to entities in the dictionary must be attributed to the user alone or to a CCID.
  - If changes must be attributed to a CCID, whether the user must identify that CCID at SIGNON time or if predefined CCID associations will be used.

- **SIGNOUT Processing**
  An entity can be signed out to a USER or a CCID. Once signed out, only users signed on as that USER or under that CCID can update the entity. An entity can be signed out manually or automatically. There is a set of entity type AUTO SIGNOUT flags. For each entity type that CA Endevor/DB handles, there is an AUTO SIGNOUT flag. The effects of these flags are global: if you set an AUTO SIGNOUT flag to Y, an entity is automatically signed out when it is modified. It is signed out to the USER or CCID who modified it. There is also a SIGNOUT function in the Promotion Support Selection utility - if used, all entities selected for migration are signed out when selected, and then signed in when the migration is confirmed.

- **PREAUTHORIZATION Processing**
  A USER or CCID can be preauthorized to an entity. Preauthorizations can be used in any or all of five places:
  - For specific entities - to set restrictions so that only preauthorized users can change them.
  - For specific users - to restrict which entities they are allowed to change.
  - To control which users are allowed to SIGNON or to make changes under given CCIDs.
  - To control which users are allowed to establish entity-status relationships for certain statuses. These relationships are used to control the promotion process.
  - To predefine the CCID to which changes will be attributed for certain entities.

- **LOCK Processing**
  A USER, a CCID, or the entire dictionary can be locked.
Once a USER is locked, no one can sign on as that user or perform any updates as that user.

Once a CCID is locked, no one can sign on under that CCID or make changes attributed to that CCID.

Once a DICTIONARY is locked, no one can update anything in that dictionary.

**ACTION Rules**

There are rules governing the ability to perform various CA Endevor/DB actions. These rules can be set on a user-by-user basis, or a CCID-by-CCID basis, or globally. These rules determine:

- Whether a user is allowed to run the Archive/Compress utility;
- Whether a user is allowed to run the Promotion Support utilities;
- Which functions of the Online front end a user is allowed to operate.
- Whether a user is allowed to run the Batch front end and if so, which functions of the Batch front end a user is allowed to operate.

**Security Components**

The CA Endevor/DB Security System consists of assigned security procedures in the CCDB, the security enforcement logic inside the Change Monitor itself, the portions of the MIS Front End used to maintain the CCDB, and the portions of the Promotion Support utilities that perform SIGNOUT and SIGNIN processing. The security administrator uses the MIS Front End to establish the security procedures in the CCDB, and the Change Monitor automatically watches over those procedures. To understand the CA Endevor/DB Security System, it is essential to understand the various security-related data structures in the CCDB, how they interrelate, and how the Change Monitor uses them to provide the features described above.

<table>
<thead>
<tr>
<th>Component Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary Descriptor</td>
<td>Contains control flags that affect all dictionary entities and users. Some of the SIGNON rule control flags are kept in the Dictionary Descriptor, along with the MONITOR flags and the AUTO SIGNOUT flags. This descriptor also contains a LOCK flag for the dictionary, the name of the Security Class for the dictionary, and the name of a Security Class to use when signon processing is performed without a specified user.</td>
</tr>
<tr>
<td>Security Class Descriptor</td>
<td>Contains the remainder of the control flags (not contained in the Dictionary Descriptor). There is a Security Class associated with the dictionary, with each USER, and with each CCID. During signon processing, the security administrator identifies a dictionary, and optionally a USER and up to 12 CCIDs -- each with an associated Security Class. It is possible, then, to have a total of up to 14 Security Class Descriptors as implied by the signon process. The control flags from these Security Class Descriptors are merged under the following rules: An N value for a flag in any Security Class means the user operates with N. A Y value for all security classes means the user operates with Y.</td>
</tr>
<tr>
<td>USER Descriptor</td>
<td>Contains the name, password, security class, and most recent list of CCIDs for a user. When a user signs on, if no CCIDs are specified, the list is assumed to identify the CCIDs to use; if one or more CCIDs are specified, they are assumed to replace the old list. An</td>
</tr>
</tbody>
</table>
## Component Description

exception to this use of Signon CCIDs is when DERIVED CCID processing is in effect for the user. Then Signon CCIDs are not used, and predefined CCID to entity associations are used to identify what CCIDs to associate as the updates to dictionary entities occur. The USER Descriptor also contains a LOCK flag for the user.

<table>
<thead>
<tr>
<th>CCID Descriptor</th>
<th>Contains the name and security class for a CCID. It also contains a LOCK flag for the CCID.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGNOUT and PREAUTH Record</td>
<td>Acts as a junction between ENTITY and either USER or CCID. It contains a signout flag, a PREAUTH flag, and a DERIVE CCID flag. A given SIGNOUT/ PREAUTHORIZATION record can serve three purposes: To record that an entity is signed out to the user or CCID involved in the junction To record that the user of the CCID involved in the junction is preauthorized to the entity. To record that changes made by any user running in DERIVED CCID mode will be attributed to the CCID that participates in the junction.</td>
</tr>
</tbody>
</table>

## Security Implementation

### Contents

- Entity Type Monitoring (see page 26)
  - Security Classifications (see page 26)
  - Use of the CA Endevor/DB Batch Front End (see page 27)
  - AUTO SIGNOUT (see page 27)
  - STATUS (see page 28)
  - Preauthorization (see page 28)
  - LOCK (see page 29)

This section describes the CA Endevor/DB security planning process, and the features that are used to implement each desired capability.

## Entity Type Monitoring

The security administrator's first consideration is to decide which entity types to monitor (and thus secure through the CA Endevor/DB Security System). If, for instance, you do not wish to control the update of ELEMENT entities, set the MONITOR flag for ELEMENTs to \texttt{N}.

For more information on entity types, see Appendix B, "CA Endevor/DB Entity Types," in the CA Endevor/DB for CA IDMS Batch Reference section.

### Security Classifications

The security administrator's next consideration is whether to establish security by USER, security by CCID, global security, or no security at all.

- **No security** is the easiest to implement. Simply change the name of the default user Security Class (in the Dictionary Descriptor) to the global privileges Security Class.
Global security is only slightly more difficult. Change the name of the default user Security Class to the global security class, and then disable (in that Security Class) those features, which no user is to be able to perform.

To establish security by USER or CCID, you must divide your user population into classes, based upon which actions users in a given class are to be allowed to perform. You then set up a Security Class Descriptor for each class. Choose one of these as the default class, and specify it as the default user Security Class in the Dictionary Descriptor. In order to guarantee that a user signs on under the proper USER or CCID, set the NO-PASS flag in the Dictionary Descriptor to N (thus forcing all users to specify a password when they sign on).

To establish security by USER, modify each USER Descriptor to specify the name of the appropriate Security Class. CCIDs may be left optional but, if defined, would have the same Security Class as the Dictionary.

To establish security by CCID, modify each USER Descriptor to specify the same Security Class as the Dictionary, set the appropriate Security Class in each CCID Descriptor, mark every CCID (that has special privileges) as PRIVATE, and establish the preauthorizations that determine which users are allowed to operate under each CCID.

Use of the CA Endevor/DB Batch Front End

The Batch front end provides a similar set of functions to the CA Endevor/DB Online front end. Through the use of a batch command language, massive updates can be issued from simple, powerful commands. Also, command syntax can be created using existing data from a CCDB. This can be very effective when cloning CCDB information. The Security Administrator must decide whether to disallow this facility and if so, what security classes to disable it for. To disable this facility, set the BATCH flag in the Security Class to N. A complete discussion of this facility is available in the CA Endevor/DB for CA IDMS Batch Reference section.

Note: Setting the Batch flag to Y or N will have no effect on the use of other CA Endevor/DB Batch utilities.

AUTO SIGNOUT

Another issue to be decided is whether you want to use AUTO SIGNOUT to prevent simultaneous update by multiple users. You control this feature on an entity-type by entity-type basis, by setting AUTO-SO flags in the Dictionary Descriptor. If you elect to use AUTO-SO, then you must also decide whether to have the entities automatically signed out to USER or CCID.

If you want AUTO SIGNOUT to USER, set the SO-USER flag in every Security Class to Y and the NO-USER flag in every Security Class to N.

If you want AUTO SIGNOUT to CCID, set the SO-USER flag in every Security Class to N, the SO-CCID flag to Y, and the NO-CCID flag in every Security Class to N.
STATUS

Next, you must decide whether you want to control the setting of STATUS. This utility allows only certain user classes to set the STATUS functions of the Online front end or the Batch front end. STATUS may be used to control the Promotion Support Selection utility and, thus, should be considered very sensitive. If you must distinguish between specific STATUS values (for Promotion or any other reason), then you will have to set up the preauthorizations that determine which USER is allowed to set the sensitive STATUS, and mark that STATUS as PRIVATE.

Preauthorization

The next five areas of concern are all addressed by the use of preauthorizations. They are:

- **Dangerous Users** - These dictionary users are to be restricted to only updating certain entities in the dictionary. For example, trainees would fit into this category. The restriction is specified through the use of the NO-AUTH, LIM-AUTH, and A-OPT flags in the appropriate SECURITY CLASS records. Set the flags in the SECURITY-CLASS record named in the USER descriptor for each "dangerous user" to:
  
  LIM-AUTH = N  
  NO-AUTH = N  

  Set the A-OPT flags in the same SECURITY-CLASS record to **N** for all entity types that are to be protected. Then establish a PREAUTHORIZATION junction between each "dangerous user" and all of the entities that the user is to be allowed to change.

- **Sensitive Entities** - These entities are to be updated by only certain users. For example, a disbursement dialog would fit into this category. This restriction is also specified through the use of the NO-AUTH, LIM-AUTH, and A-OPT flags in the SECURITY-CLASS records. Set the flags in every SECURITY-CLASS record to:
  
  LIM-AUTH = Y  
  NO-AUTH = N  

  Set the A-OPT flags in every SECURITY-CLASS record to **N** for those entity types that are to be protected. Then establish a PREAUTHORIZATION junction between each sensitive entity and each user that is to be allowed to modify that entity.

  **Note:** The protection requires at least one PREAUTHORIZATION junction for each sensitive entity. If an entity participates in NO PREAUTHORIZATION junctions, it is assumed by the system not to be sensitive.

- **Derived CCID** - In some shops, it may be infeasible to require that all users sign on to CA Endevor/DB each time they switch from one CCID to another. For example, if a unique CCID is established for every change for every DIALOG, programmers would be issuing CA Endevor/DB signons on a frequent basis. To circumvent this problem, the administrator can predefine the relationships between CCIDs and dictionary entities, and the programmers can run in "DERIVED CCID" mode. When doing so, they only signon to CA Endevor/DB to specify their userid. The CCID to which a given change is attributed will be determined by the presence of a PREAUTHORIZATION junction. This processing mode is also specified through the SECURITY-CLASS record. In the SECURITY-
CLASS records named in each DERIVED CCID user descriptor record, set the DE-CCID flag to Y. Then establish a PREAUTHORIZATION junction between each entity to be changed and the CCID to which changes are to be attributed. In each of those PREAUTHORIZATION junctions, set the DE-CCID flag to Y.

- **Private CCID** - You may need to make CCIDs private for several reasons. Perhaps you have established security by CCID or you manage "Sensitive Entities" by CCID. In these (and other) cases, you will need to control which users are allowed to signon or make changes under a CCID. The restriction is specified by setting the TYPE of each restricted CCID to PRIVATE. Then establish a PREAUTHORIZATION junction between each USER that is to have access to a given CCID and the following entity:

  ENTITY NAME = ccid-name
  TYPE = CCID
  VERSION = 1

- **Private Status** - In promotion processing, the NDVRDSEL program EXCLUDE command will exclude any entity associated with a given STATUS. The ability to associate entities with the STATUSes used in your shop's promotion processing is therefore important. To control that ability, set the TYPE of each STATUS used in promotion processing to PRIVATE. Then establish a PREAUTHORIZATION junction between each USER that is to have the ability and the following entity:

  ENTITY NAME = status-name
  TYPE = STATUS
  VERSION = 1

**LOCK**

The last area of concern in the implementation of the CA Endevor/DB Security System is LOCK. If a USER or CCID is to be disabled for any reason, lock it. If all updates to a dictionary are to be suspended, lock it.

**Note:** Locking a dictionary does not affect the ability to update the corresponding CCDB.

**Security Classes and Levels**

The CA Endevor/DB Security System is structured to allow global security options that apply to all users within a Dictionary/CCDB pair, as well as to allow tailored specifications to individual users and/or CCIDs. Global options for a Dictionary are maintained and established in the CCDB record that represents the Dictionary being monitored. Individual users and CCIDs cannot override options specified in this global record.

In addition to global security options specified only at the Dictionary level, each Dictionary, User, and CCID can be associated with a Security Class. The Security Class is a named set of rules that apply to the CCDB entity with which it is associated. In general, there is one Security Class for each unique set of permissions allowed. An installation usually sets up a separate Security Class for DBAs, CCDB Administrators, and general users. The CCDB Administrator simply relates the desired Security Class to Users, CCIDs, and Dictionaries. The administrator can establish installation standards in the Dictionary Security Class and be assured that no other Security Class can override it. Security Classes are also used to restrict the activities of individual users, or individuals working under a given CCID.
Security Class Options

<table>
<thead>
<tr>
<th>Procedures</th>
<th>Entity Attributes</th>
<th>Function Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign-in</td>
<td>Update</td>
<td>All menu masks</td>
</tr>
<tr>
<td>Auto signout</td>
<td>Authorization</td>
<td>All Batch commands</td>
</tr>
<tr>
<td>CCID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Limited Preauthorization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full Preauthorization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Archive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Migrate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NM-Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Batch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Derived CCID Mode</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Options specified exclusively at the Dictionary level only. (These cannot be varied by individual users):

<table>
<thead>
<tr>
<th>Procedures</th>
<th>Entity Attributes</th>
<th>Function Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto-User</td>
<td>Monitor</td>
<td>Non</td>
</tr>
<tr>
<td>Synch</td>
<td>Sign-out</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following diagrams shows the use of Dictionary and Security Class options in combination.
CA Endevor/DB Sign-on is a two phase process. Phase one establishes the userid, while phase two establishes the CCID(s) and Security Class options under which the user will be operating.

- **Phase 1**
  
  CA Endevor/DB performs two types of Phase 1 Sign-on:

<table>
<thead>
<tr>
<th>Sign-on Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explcit</td>
<td>Performed as a result of an CA Endevor/DB Sign-on screen or a SIGNON command. An CA Endevor/DB Sign-on element is built with the userid keyed into the screen (if any) or the CA IDMS/DC userid (if any). If Auto-User is in effect, the userid will be added to the CCDB automatically. If the global options require a password or a userid, it will be requested by the system if omitted.</td>
</tr>
<tr>
<td>Implicit</td>
<td>Performed internally by the Change Monitor when it first encounters an attempt to update an entity in a Dictionary or CCDB. If no prior CA Endevor/DB Sign-on element exists for the user, the system will attempt an Implicit Sign-on. The userid in the CA IDMS/DC sign-on element will be used to obtain the CA Endevor/DB user definition. If the global options require a userid, and none can be obtained from a sign-on element, the attempted Dictionary update will be disallowed, and the user will be requested to perform an Explicit Sign-on. If Auto-User is in effect, the userid will be added to the CCDB automatically. If the global options require a password or a userid, it will be requested by the system if omitted.</td>
</tr>
</tbody>
</table>
effect, the userid will be automatically added to the CCDB.

When running CA IDMS compilers in CA Endevor/DB Batch mode, an Implicit Signon occurs if the Batch compiler is not run under NDVRBOOK, or if no userid was used in the SIGNON statement. The Implicit Signon will attempt to acquire a userid associated with the batch job. If no userid is available, the Implicit Signon will be for an unspecified userid. If the global options require a userid, the attempted dictionary update will be disallowed.

- **Phase 2**
  Once phase one of SIGNON is complete, phase two processing begins. The USER descriptor for the userid determined in phase one is fetched from the CCDB and the security class named therein is fetched. If no userid was determined, the DEFAULT SECURITY CLASS named in the DICTIONARY descriptor is fetched. In either case, the dictionary security class is also fetched, and the two sets of security flags are merged. At this point, CA Endevor/DB SIGNON processing checks the merged DE-CCID flag to determine if DERIVED CCID mode is in effect.

If DERIVED CCID mode is in effect (DE-CCID = Y), processing is as follows:

1. The CA Endevor/DB and CA IDMS/DC userids are compared. If SYNCH = N and the userids are different, the session is terminated.
2. The USER descriptor is checked. If it is LOCKED, the session is terminated.
3. The DICTIONARY descriptor is checked. If it is LOCKED, the session is terminated.
4. If there is not an existing USER descriptor for the user in the CCDB and AUTO-US = Y, CA Endevor/DB creates a USER descriptor with the security class set to the default security class and the password set to blanks.
5. If the userid is not specified and NO-USER = N, the session is terminated.
6. If the password is not specified and NO-PASS = N, the session is terminated.
7. If all the above checks are passed, the session is started in "DERIVED CCID" mode.

If DERIVED CCID mode is not in effect (DE-CCID = N), processing is as follows:

1. The CA Endevor/DB and CA IDMS/DC userids are compared. If SYNCH = N and the userids are different, the session is terminated.
2. The USER descriptor is checked. If it is LOCKED, the session is terminated.
3. The DICTIONARY descriptor is checked. If it is LOCKED, the session is terminated.
4. If a userid is specified and a USER descriptor exists in the CCDB, and if CCIDs were specified in the SIGNON, the CCID list in the USER descriptor is replaced. If no CCIDs were specified in the signon, the CCID list in the USER descriptor is picked up and processing proceeds as if those CCIDs had been specified.
5. For each CCID specified in the signon, the CCID descriptor is fetched. If the CCID is locked, the session is terminated. If the CCID is PRIVATE and the user is not preauthorized, the session is terminated. The security class named in the CCID descriptor is merged with the dictionary and user security classes.

6. If there is not an existing USER descriptor for the user in the CCDB and AUTO-US = Y, CA Endevor/DB creates a USER descriptor with the security class set to the default security class and the password set to blanks.

7. If the userid is not specified and NO-USER = N, the session is terminated.

8. If the password is not specified and NO-PASS = N, the session is terminated.

9. If no CCIDs were specified in the signon or existing USER descriptor and NO-CCID = N, the session is terminated.

10. If all above checks are passed, the session is started in "Normal" mode.

Once the user session is started, each change made to a dictionary or CCDB entity is subject to security checking and logging.

The options in the Dictionary level Security Class apply to all users, including the security administrator. Be very careful when modifying the Security Class associated with the Dictionary. The options in the CCID Security Class apply to all users who are working with that CCID. The User Security Class options apply to the individual user. Security Class restrictions at the Dictionary level cannot be overridden. When no userid or CCID is required, or when a user is initially added with Auto-User, a default Security Class is assigned by CA Endevor/DB to cover the session. The CCDB administrator specifies the default mask in the CCDB Dictionary definition.

Change Monitor Processing

Every change made by every user in either the dictionary or CCDB is seen by the CA Endevor/DB change monitor. For every change, the processing takes place in two phases:

1. Before the change is applied to the dictionary/CCDB, it is checked by the security system.

2. After the change occurs, it is logged in the CCDB.

The security checking is as follows:

- If the CCDB is offline, the update request is disallowed.

- If security or monitoring is turned off, no other security validations are performed, and the update request is allowed.

- If the dictionary descriptor for the dictionary indicates that the dictionary is locked, the update request is disallowed.

- If the dictionary descriptor for the dictionary indicates that monitoring is not required for this record type, no other security validations are performed, and the update request is allowed.
If the change is being made in DERIVED CCID mode, the CCID validations which would be performed at signon time must be performed for each CCID which had been preauthorized to the entity with the DERIVE CCID option. As the associated CCIDs (the ones related to the entity through DE-CCID = Y preauthorization junctions) are scanned, locked CCIDs are skipped, as are PRIVATE CCIDs to which the user is not preauthorized. If the security permission flags indicate that the user is not allowed to process without a CCID, and there are no associated CCIDs, the update request is disallowed. The authorization flags for each derived CCID’s security class are merged into a net set of permission flags. If the change is being made in the “normal” mode (not DERIVED CCID mode), the net set of permission flags were built in phase 2 of signon processing and this step is skipped.

- The security permission flags must indicate that the user is allowed to update this record type. If not, the update request is disallowed.

- If the entity is signed out to another user or a CCID, the update request is disallowed.

- If the A-OPT flag for this record type is set to Y, the update is allowed and the next two checks for FULL AUTH and LIM AUTH classifications are skipped.

- If the security permission flags indicate that the user is "FULL AUTH" classification and a preauthorization for either the user or associated CCIDs does not exist, the update request is disallowed. This rule implements the "Dangerous User" function described earlier in the Preauthorization section of this section.

- If the security permission flags indicate a user is "LIM AUTH" classification, and the entity is preauthorized to another user or CCID, but not this user or CCID, the update request is disallowed. This rule implements the "Sensitive Entity" function described earlier in the Preauthorization section of this section.

At this point, if all validations have been successfully completed by the security system, the update takes place. After the change has taken place, it is logged in the CCDB. The log processing is as follows:

No logging of the update will be done if any of the following conditions are true:

- The CCDB is not in update mode.

- Logging is turned off.

- If the dictionary descriptor for the dictionary indicates that monitoring for this record type is not required.

If the request is to be logged, CA Endevor/DB performs the following processing:

- If the entity does not already exist, it will be added to the CCDB.

- A Change Log Entry (CLE) will be created, which is linked to all associated CCIDs (either the user's signon or the derived CCIDs for the entity).
Global Security

A security administrator with authority to set global options and maintain Security Classes can be set up through the CA Endevor/DB Online facility or the CA Endevor/DB Batch facility. To do this, you must accomplish the following tasks:

1. Set up a user description for the security administrator.

2. Associate the security administrator with an appropriate Security Class.

To do so using the Batch facility, submit a Batch job containing the following CA Endevor/DB commands:

SIGNON USER NAME IS user-name DICTNAME IS dictname.
MOD USER EDBADMIN SECURITY CLASS IS NDVR-GLOBAL.

To do so using the Online facility, sign on to DC/UCF. Then sign on to CA Endevor/DB by entering NDVR at the ENTER NEXT TASK CODE: prompt.

The Signon Function menu displays as follows:

<table>
<thead>
<tr>
<th>CA-E/DB nn.n volser</th>
<th>SIGNON FUNCTION</th>
<th>mm/dd/yy</th>
<th>NDVRM000</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTION ====&gt; 2</td>
<td>DICTNAME ====&gt; SRCNDVR</td>
<td>MODE ====&gt; UPDATE</td>
<td></td>
</tr>
<tr>
<td>1 - SIGNON AND RETURN TO IDMS</td>
<td>2 - SIGNON AND GO TO FUNCTION MENU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENDEVOR/DB USER:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAME ====&gt; EDBADMIN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PASSWORD ====&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCID(S):</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(*NOCCID ====&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO CLEAR)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONLINE SYSTEM PARAMETERS:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBNAME ====&gt; SRCNDVR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USAGE MODE ====&gt; UPDATE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Setting Up a Security Administrator (see page 35)
- Disallowing Batch Processing (see page 37)
- Setting Up Global Dictionary Options (see page 38)

Setting Up a Security Administrator

Perform the following tasks:

1. Enter 2 (SIGNON AND GO TO FUNCTION MENU) after OPTION ====>

2. Enter the Userid of the security administrator in the NAME field.

3. Enter the Dictionary name associated with this CCDB after DBNAME ====>.
   Since Auto-User is in effect, the CA Endevor/DB system will automatically create a user record for the security administrator. This user will be automatically associated with the Security Class NDVR-DDA that is insufficient for security administration. In a later step, the Security Class will be changed. The Main Function Menu displays as follows:
CA-E/DB nn.n volser MAIN FUNCTION MENU mm/dd/yy NDVRU700
USER ==> EDBADMIN DICTNAME ==> SRCNDVR MODE ==> UPDATE
NDVRM710: I001 ENDEVOR/DB SIGNON PROCESSING COMPLETED
OPTION ==> 7

1 - SIGNIN/SIGNOUT FUNCTIONS
2 - AUTHORIZATION FUNCTIONS
3 - LOCK FUNCTIONS
4 - ENTITY AND ENTITY CHANGE HISTORY
5 - CCID AND CCID CHANGE HISTORY
6 - STATUS AND STATUS ASSOCIATIONS
7 - USER AND USER CHANGE HISTORY
8 - DICTIONARY AND DICTIONARY HISTORY
9 - MANAGEMENT GROUPS AND CCIDS
10 - ENDEVOR/DB CONTROL FUNCTIONS
11 - ENDEVOR/DB SIGNON FUNCTION
12 - RETURN TO IDMS/DC

Enter (USER AND USER CHANGE HISTORY) after OPTION ==>. Press ENTER.

The system will respond with the USER FUNCTIONS menu as follows:

CA-E/DB nn.n volser USER FUNCTIONS mm/dd/yy NDVRU700
USER ==> EDBADMIN DICTNAME ==> SRCNDVR MODE ==> UPDATE
OPTION ==> 3
1 - BROWSE USER DESCRIPTORS 2 - ADD A USER DESCRIPTOR
3 - MODIFY USER DESCRIPTORS 4 - DELETE USER DESCRIPTORS
5 - BROWSE USER/CHANGE ASSOCIATIONS 6 - ADD A USER/CHANGE ASSOCIATION
7 - MODIFY USER/CHANGE ASSOCIATIONS 8 - DELETE USER/CHANGE ASSOCIATIONS
USER ==> EDBADMIN (IF OPTIONS 1-8 )
ENTITY:
NAME ===> 
TYPE ===> 
VERSION ===> 
CHANGE-LOG SELECTION CRITERIA: (IF OPTIONS 5, 6, 7, 8 )
START DATE ===> END DATE ===> mm/dd/yy
START TIME ===> END TIME ===> 
ACTION CODE ===> 

5. Enter 3 (MODIFY USER DESCRIPTIONS) in the OPTIONS field.

6. Enter the security administrator’s userid in the USER field.
Press ENTER.

The User Detail screen for the Security Administrator displays as follows:

CA-E/DB nn.n volser USER DETAIL mm/dd
/yy NDVRM710
USER ==> EDBADMIN DICTNAME ==> SRCNDVR MODE ==> UPDATE
ACTION ==> MODIFY
******************************************************************************
USER ==> EDBADMIN PASSWORD ===> 
SECURITY CLS ==> NDVR-GLOBAL
CURRENT CCID ===> 
COMMENT ===> CCDB ADMINISTRATOR
LOCKED ===> N LOCK DATE ===> LOCK TIME ===> 

This user definition was added by the system automatically at Sign-on.

7. Tab down to SECURITY CLS ===>.

8. Enter NDVR-GLOBAL.
Press ENTER.

The system will now associate the userid for the security administrator with the NDVR-
GLOBAL Security Class. From this point forward, the security administrator will have the
Disallowing Batch Processing

To disallow performing Batch processing, the NDVR-GLOBAL security class must be modified so that the CA Endevor/DB Batch facility is disabled. To disable this processing using the Online facility:

1. Enter =10 in the OPTION field on the Dictionary Function menu. This gives you direct access to the System Control Functions menu without going through the Main Function menu. The System Control Functions menu displays as follows:

   CA-E/DB nn.n volser   CA-ENDEVOR/DB SYSTEM CONTROL FUNCTIONS mm/dd/yy NDVRA00
   USER ===> EDBADMIN    DICTNAME ===> SRCNDVR    MODE ===> UPDATE
   OPTION ===> 5
   1 - BROWSE CCDB DESCRIPTOR RECORD         2 - MODIFY CCDB DESCRIPTOR RECORD
   3 - BROWSE SECURITY DESCRIPTORS           4 - ADD A SECURITY DESCRIPTOR
   5 - MODIFY SECURITY DESCRIPTORS           6 - DELETE SECURITY DESCRIPTORS
   7 - BROWSE MONITOR DICT STAT BLOCKS       8 - MODIFY MONITOR DICT STAT BLOCKS
   SECURITY CLASS ===>  (IF OPTIONS 3, 4, 5, 6 )
   DICTNAME ===> SRCNDVR            (IF OPTIONS 7, 8 )

2. Enter 5 in the OPTION field on the System Control Functions menu. Enter NDVR-GLOBAL in the Security Class field. Then press ENTER.

   Note: We are going to update the security class settings for the NDVR-GLOBAL security class, which is used by the security administrator. If we did not enter a security class, the system would provide a list of security classes from which we could select one or more for modifications. Refer to Section 4, "Security Class Maintenance," for further information on security class maintenance.
3. Enter N to the right of the BATCH: option. Press ENTER.

The System Control Functions screen displays as follows:

```
3.
1.
2.
```

Enter to the right of the BATCH: option. Press ENTER.

The System Control Functions screen displays as follows:

```
CA-E/DB nn.n volser CA-ENDEVOR/DB SYSTEM CONTROL FUNCTIONS mm/dd/yy NDVRUA00
USER ===> EDBADMIN DICTNAME ===> SRCNDVR MODE ===> UPDATE
NDVRMA10: I001 ACTION COMPLETED NORMALLY: SECURITY-CLASS MODIFY
OPTION ===>
  1 - BROWSE CCDB DESCRIPTOR RECORD  2 - MODIFY CCDB DESCRIPTOR RECORD
  3 - BROWSE SECURITY DESCRIPTORS  4 - ADD A SECURITY DESCRIPTOR
  5 - MODIFY SECURITY DESCRIPTORS  6 - DELETE SECURITY DESCRIPTORS
  7 - BROWSE MONITOR DICT STAT BLOCKS  8 - MODIFY MONITOR DICT STAT BLOCKS
SECURITY CLASS ===> NDVR-GLOBAL
DICTNAME ===> SRCNDVR
```

```
Setting Up Global Dictionary Options
```

To set up the global dictionary options, you must modify the default settings which were automatically created when the CCDB was first opened. These global dictionary options can be set up through the CA Endevor/DB Online facility or the Batch facility.

To do so using the Batch facility, execute the following MODIFY DICTIONARY command in a Batch job:

```java
MODIFY DICTIONARY dictname
SECURITY CLASS IS NDVR-GLOBAL
DEFAULT SECURITY CLASS IS NDVR-DDA
SYSTEM NAME IS system-name
AUTO-US IS Y
NO-SYNC IS Y
NO-PASS IS Y
MONITOR (SCH FIL TAS SUB USE DES REC SYS APO SET DIA
  APP ELE QFI PRC TAB FUN MOD PHY CLA ATT MAP
  LOA LIN MSG LOG LR PRO)
AUTO-SO NONE.
```

Refer to the following description of the Online screen for more information on the DICTIONARY option settings.

To modify the global dictionary options using the Online facility:

1. Enter NDVRMIS at the Next Task Code prompt to sign on to CA Endevor/DB.

2. Enter 8 in the OPTION field on the Main Menu.

   The Dictionary Functions menu screen displays:

```
CA-E/DB nn.n volser DICTIONARY FUNCTIONS mm/dd/yy NDVRUA00
USER ===> EDBADMIN DICTNAME ===> SRCNDVR MODE ===> UPDATE
OPTION ==> 2
  1 - BROWSE DICTIONARY DESCRIPTORS  2 - MODIFY DICTIONARY DESCRIPTORS
  3 - DELETE DICTIONARY DESCRIPTORS  4 - BROWSE CHANGE-LOG ENTRIES
  5 - MODIFY CHANGE-LOG ENTRIES  6 - DELETE CHANGE-LOG ENTRIES
DICTIONARY NAME ===> SRCNDVR
ENTITY:           TYPE:           VERSION:           CHANGE-LOG SELECTION CRITERIA:
  NAME ===>       TYPE ===>       VERSION ===>         (IF OPTIONS 1-6 )
  START DATE ===>   END DATE ===> mm/dd/yy
  START TIME ===>   END TIME ===>
ACTION CODE ===>  
```
3. Enter 2 (MODIFY DICTIONARY DESCRIPTORS) after OPTION ===>.  
   The following screen displays:
   CA-E/DB nn.n volser  DICTIONARY DESCRIPTOR DETAIL  mm/dd
   USER ===> EDBADMIN  DICTNAME ===> SRCNDVR  MODE ===> UPDATE
   ACTION ===> MODIFY
   ***************************  DICTIONARY INFORMATION  ***************************
   NAME ===> SRCNDVR  SYSTEM IDENTIFIER ===> SYSTEM81
   SEC. CLASS ===> NDVR-GLOBAL  DEFAULT USER CLASS ===> NDVR-DDA
   ORG. NAME ===>  DICTIONARY TYPE ===> N
   LOCKED ===> N  LOCK DATE ===>  LOCK TIME ===>  
   COMMENT ===> SOURCE DEMONSTRATION DICTIONARY
   AUTO-US: Y  NO-SYNC: Y  NO-PASS: Y
   ENTITY: SCH DMC FIL TAS SUB USE DES REC SYS APO SET DIA APP ELE OFI PRC TAB FUN
   AUTO-SO: N N N N N N N N N N N N N N N N N N
   ENTITY: MOD PHY CLA ATT MAP LOG LIN MSG LOA LR PRO
   AUTO-SO: Y Y Y Y Y Y Y Y Y Y Y Y

   This screen contains the global definition for the Dictionary displayed.

   Fields are described below.

   • NAME
     Dictionary for which the CCDB was set up.

   Note: This name will change automatically if the DBNAME table entry for this dictionary is modified to a new or different name.

   • SYSTEM IDENTIFIER
     Enter a 1-8 character value to be used to identify the system in which this dictionary resides. This identifier may represent the system's external or internal name, a node name, or a data sharing group name. The System Identifier is used with the name of the dictionary to uniquely identify the dictionary for migration oriented Change Log entries which contain identifying information relating to the destination and origin of entities that have migrated from one dictionary to the next. System Identifier should be unique across your CA IDMS environment regardless of CV, CPU, or data center.

   • SEC. CLASS
     The Security Class associated with the Dictionary. Initially, the Dictionary is set up with NDVR-GLOBAL as a Security Class.

   Note: It is recommended that this be left unaltered until a thorough understanding of the Security System is established.

   The restrictions set up in the Dictionary Security Class (see Section4, "Security Class Maintenance") apply to all users in the Dictionary.

   To change the Dictionary Security Class, enter the new Security Class name to be used for the Dictionary.

   • DEFAULT USER CLASS
     This is the Security Class that will be associated with users who are added using the Auto-User procedure. Initially, this is set to NDVR-DDA. The default Security Class can be changed by keying the new Security Class name desired.
- **ORIGINAL NAME**
  Not supported in this release. Leave blank.

- **DICTIONARY TYPE**
  Not supported in this release. Always = N.

- **LOCKED**
  Informational. Values are Y or N. Indicates if the Dictionary is locked.

- **LOCK DATE**
  Informational. Date the Dictionary was locked.

- **LOCK TIME**
  Informational. Time the Dictionary was locked.

- **AUTO-US**
  Values are Y or N:
  - Y -- Users are added automatically by CA Endevor/DB when encountered the first time.
  - N -- Users are not automatically added to the system.

- **NO-SYNC**
  Values are Y or N.
  - Y -- When signing on, the CA Endevor/DB and CA IDMS/DC userids can be different.
  - N -- When signing on, the CA Endevor/DB and CA IDMS/DC userids cannot be different.

- **NO-PASS**
  Values are Y or N.
  - Y -- When signing on, the CA Endevor/DB password is optional.
  - N -- When signing on, the CA Endevor/DB password is required.

- **ENTITY**
  The three-character entity name abbreviations used by the CA Endevor/DB and CA IDMS/DC system for IDD entity types. They act as column headings for the next two fields.

- **MONITOR**
  Values are Y or N.
  - Y -- Monitor the modifications to entities of the type specified in the column heading (ENTITY:).
  - N -- Do not monitor or create Change Log Entries for modifications to the entity type in the column heading. When monitoring is turned off for an entity type, none of the other CA Endevor/DB attributes have significance.

- **AUTO-SO**
  Values are Y or N.
Security Class Maintenance

Security Classes are a central part of CA Endevor/DB security. Within the Security Class, restrictions are defined which apply to Dictionaries, CCIDs, and Users. Each one of these entities can be associated with a different Security Class. At execution time, the Security System combines all the Security Classes referenced and arrives at a resultant Security Class. If a permission is disallowed at any level (Dictionary, CCID, or User), the resultant Security Class disallows the action.

Contents

- Establishing and Maintaining Security Classes (see page 41)
- NDVRUA10 Field Descriptions (see page 42)
- NDVRMA10 Field Descriptions (see page 42)

Establishing and Maintaining Security Classes

The Dictionary, CCID, and User definitions each contain a reference to a Security Class name. There is usually no more than five or six Security Classes in a CCDB. Many installations set up one for the security administrator (usually NDVR-GLOBAL), one for the Dictionary (usually NDVR-GLOBAL), one for the DBA, one for development leaders, and one for general application developers.

Security classes can be defined and maintained using either the Online front end or the Batch front end. The commands used in the Batch front end are ADD, MODIFY, and DELETE SECURITY CLASS. Refer to the CA Endevor/DB for CA IDMS Batch Reference section for further information on using them. The meanings of the various options for the ADD SECURITY CLASS or MODIFY SECURITY CLASS commands are discussed in the following description of the Online screens.

When using the Online front end, Security Classes are maintained under the Control Functions screen. To access this screen:

1. Enter 10 in the OPTION field on the Main Function menu.

   The System Control Functions screen displays as follows:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BROWSE CCDB DESCRIPTOR RECORD</td>
</tr>
<tr>
<td>2</td>
<td>MODIFY CCDB DESCRIPTOR RECORD</td>
</tr>
<tr>
<td>3</td>
<td>BROWSE SECURITY DESCRIPTORS</td>
</tr>
<tr>
<td>4</td>
<td>ADD A SECURITY DESCRIPTOR</td>
</tr>
<tr>
<td>5</td>
<td>MODIFY SECURITY DESCRIPTORS</td>
</tr>
<tr>
<td>6</td>
<td>DELETE SECURITY DESCRIPTORS</td>
</tr>
<tr>
<td>7</td>
<td>BROWSE MONITOR DICT STAT BLOCKS</td>
</tr>
<tr>
<td>8</td>
<td>MODIFY MONITOR DICT STAT BLOCKS</td>
</tr>
<tr>
<td>SECURITY CLASS        =&gt; SRCNDVR     (IF OPTIONS 3, 4, 5, 6)</td>
<td></td>
</tr>
<tr>
<td>DICTNAME            =&gt; SRCNDVR     (IF OPTIONS 7, 8)</td>
<td></td>
</tr>
</tbody>
</table>

   To add a new Security Class, enter 4 in the OPTION field.
   To modify an existing Security Class, enter 5 in the OPTION field.
   To delete a Security Class, enter 6 in the OPTION field.
To go directly to the Security Class Detail screen (NDVRMA10), enter the name of the Security Class to be processed in the Security Class field. If you clear that field, you will first go to the Security Class List screen (NDVRUA10).

To go directly to the Security Class Detail screen (NDVRMA10), enter the name of the Security Class to be processed in the Security Class field. If you clear that field, you will first go to the Security Class List screen (NDVRUA10).

** NDVRUA10 Field Descriptions **

** ACTION **

Description of the current processing function: ADD, MODIFY, or DELETE.

** Field Description **

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no title)</td>
<td>Place any non-blank character beside an entry and press Enter to display the Security Class Detail screen (NDVRMA10) for that particular security class.</td>
</tr>
</tbody>
</table>

** SECURITY CLASS **

A list of the security classes that you are able to select.

** COMMENT **

A comment describing the security class.

** NDVRMA10 Field Descriptions **

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>=== NDVR-DDA</td>
</tr>
</tbody>
</table>
CA Endevor/DB discrete function switches are at the top of the screen, procedures are in the middle, and entity attributes are at the bottom. All fields are defined so that a value of N denotes a restriction and Y denotes a permission. Remember that an N causes the restriction to take effect regardless of other Security Classes that are merged to arrive at the result. In other words, restrictions cannot be overridden.

Fields are described below:

- **ACTION**
  Action selected on the previous menu.

- **NAME**
  The name of the Security Class.

- **COMMENT**
  Any comment may be entered.

- **MENU**
  The six rows directly underneath the COMMENT field on this screen are used to control the Online and Batch front end functions that will be available to a user logged on under this Security Class. There are ten subfunction menus in the Online front end, and so there are ten groups of menu flags on the screen. Each group is labeled with the name for the corresponding Online front end subfunction screen. For example:

<table>
<thead>
<tr>
<th>MENU</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTROL</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

  The eight Y/N flags to the right of "CONTROL: " are used to allow or disallow the use of the eight options on the System Control Functions (NDVRUA00) screen. Thus, in the example above, a user signed on under this security class would have the use of options 1, 3, and 7 on the NDVRUA00 screen. The other options would be denied if attempted, and would not appear on that screen. These same flags control a user’s ability to use Batch front end commands and command options. In the example above, the user would be able to use the ADD SECURITY CLASS, MODIFY SECURITY CLASS, and DELETE SECURITY CLASS commands in PUNCH mode, but not in PROCESS mode. PUNCH mode is the CA Endevor/DB Batch equivalent of Browse actions in the CA Endevor/DB Online front end.

  The full breakdown of MENU flags and the Online and Batch functions that they control is fully described in Appendix B, "Online/Batch Control Flags."

- **SIGNIN**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>A user signed on under this security class can sign entities in and out for other users.</td>
</tr>
<tr>
<td>N</td>
<td>A user signed on under this security class can sign entities in and out only for him/her self.</td>
</tr>
</tbody>
</table>

- **SO-CCID**

- **SO-USER**
  These options control the behavior of the security monitor when it is performing automatic signout (DICTIONARY descriptor AUTO-SO = Y). The AUTO-SO flag has precedence. If AUTO-SO is set to Y, automatic signout will occur, regardless of the SO-CCID and SO-USER flag settings. If
AUTO-SO = N, automatic signout will not occur. SO-CCID and SO-USER determine whether automatic signout signs an entity out to the userid or the CCID under which the change occurs. The combinations are as follows:

<table>
<thead>
<tr>
<th>SO-CCID</th>
<th>SO-USER</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>N</td>
<td>Automatic signout to CCID</td>
</tr>
<tr>
<td>Y or N</td>
<td>Y</td>
<td>Automatic signout to user</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Automatic signout to user</td>
</tr>
</tbody>
</table>

- **NO-CCID**

  **Option Description**
  
  | Y | The Change Monitor will not require a CCID before modifying an entity. |
  | N | The Change Monitor will require a CCID before modifying an entity. Note: If this is the Security Class for the Dictionary, make sure the security administrator is associated with a CCID before turning this field to N. Failure to do so may result in a universal inability to access the Dictionary.

- **NO-USER**

  **Option Description**
  
  | Y | The Change Monitor will not require a USER before modifying an entity. |
  | N | The Change Monitor will require a USER before modifying an entity. Note: If this is the Security Class for the Dictionary, make sure the security administrator USER has been added before turning this field to N.

- **NO-AUTH**

- **LIM-AUTH**

  These are used in conjunction to control the preauthorization procedures to be applied as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO-AUTH=N, LIM-AUTH=Y</td>
<td>User can modify entities that have been preauthorized to that user and entities that have not been preauthorized to anybody else.</td>
</tr>
<tr>
<td>NO-AUTH=N, LIM-AUTH=N</td>
<td>User must be preauthorized for update to all entities.</td>
</tr>
<tr>
<td>NO-AUTH=Y, LIM-AUTH=Y or N</td>
<td>Preauthorization is ignored for this user. Any entity may be modified regardless of authorization.</td>
</tr>
</tbody>
</table>

- **NM-MODE**

  **Option Description**
  
  | Y | User can use the NDVRDLVR Utility command, which runs in "no-monitor" mode. |
  | N | User cannot use the NDVRDLVR Utility command in "no-monitor" mode. |
### ARCHIVE

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>User can run the NDVRARCO utility.</td>
</tr>
<tr>
<td>N</td>
<td>User cannot run the NDVRARCO utility.</td>
</tr>
</tbody>
</table>

### MIGRATE

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>User can run the NDVRBOOK utility with OPTION=MIGRATE to create migrate in CLEs on the target system, and the user can run program NDVRDCF2 to create migrate out CLEs on the source system.</td>
</tr>
<tr>
<td>N</td>
<td>User cannot run with migration-level authority.</td>
</tr>
</tbody>
</table>

### DE-CCID

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>User will run in DERIVED CCID mode.</td>
</tr>
<tr>
<td>N</td>
<td>User will not run in DERIVED CCID mode.</td>
</tr>
</tbody>
</table>

### BATCH

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>User can run the NDVRMISB Batch Management Facility.</td>
</tr>
<tr>
<td>N</td>
<td>User cannot run the NDVRMISB Batch Management Facility.</td>
</tr>
</tbody>
</table>

### ENTITY

The three-character entity abbreviations used by CA Endevor/DB to represent DICTIONARY and CCDB entity types. These act as column headings for the next two fields. Refer to Appendix B, "CA Endevor/DB Entity Types," in the CA Endevor/DB for CA IDMS Batch Reference section for more information on entity types.

### MODS

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>User can modify entities of that type.</td>
</tr>
<tr>
<td>N</td>
<td>Users cannot update entities of that type.</td>
</tr>
</tbody>
</table>

### A-OPT

This option takes precedence over the NO-AUTH and LIM-AUTH flags. If A-OPT = Y for a particular entity type, you may modify all entities of that type, regardless of preauthorization. Use this field to tailor preauthorization to entity types.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>User can modify entities of that type without preauthorization.</td>
</tr>
</tbody>
</table>
Option | Description
--- | ---
N | Preauthorization is required for users to modify this entity type.

### Security Preauthorization

The next five areas of concern are all addressed by the use of preauthorizations. They are:

- **Dangerous Users.** These dictionary users are to be restricted to updating only certain entities in the dictionary. For example, trainees would fit into this category. The restriction is specified through the use of the NO-AUTH, LIM-AUTH, and A-OPT flags in the appropriate SECURITY CLASS records. Set the flags in the SECURITY-CLASS record named in the USER descriptor for each "dangerous user" to:
  
  \[
  \begin{align*}
  \text{LIM-AUTH} &= N \\
  \text{NO-AUTH} &= N
  \end{align*}
  \]

  Set the A-OPT flags in the same SECURITY-CLASS record to N for all entity types that are to be protected. Then establish a PREAUTHORIZATION junction between each "dangerous user" and all of the entities that the user is to be allowed to change.

- **Sensitive Entities.** These entities are to be updated only by certain users. For example, a disbursement dialog would fit into this category. This restriction is also specified through the use of the NO-AUTH, LIM-AUTH, and A-OPT flags in the SECURITY-CLASS records. Set the flags in every SECURITY-CLASS record to:
  
  \[
  \begin{align*}
  \text{LIM-AUTH} &= Y \\
  \text{NO-AUTH} &= N
  \end{align*}
  \]

  and set the A-OPT flags in every SECURITY-CLASS record to N for those entity types that are to be protected. Then establish a PREAUTHORIZATION junction between each sensitive entity and each user that is to be allowed to modify that entity.

  **Note:** The protection requires at least one PREAUTHORIZATION junction for each sensitive entity. If an entity participates in NO PREAUTHORIZATION junctions, it is assumed by the system not to be sensitive.

- **Derived CCID.** In some shops, it may be infeasible to require that all users sign on to CA Endevor/DB each time they switch from one CCID to another. For example, if a unique CCID is established for every change for every DIALOG, then programmers would be issuing CA Endevor/DB signons all day. To circumvent this problem, the CA Endevor/DB administrator can predefine the relationships between CCIDs and dictionary entities, and the programmers can run in "DERIVED CCID" mode. When doing so, they only signon to CA Endevor/DB to specify their userid - the CCID to which a given change is attributed will be determined by the presence of a PREAUTHORIZATION junction. This processing mode is also specified through the SECURITY-CLASS record. In the SECURITY-CLASS records named in each DERIVED CCID user descriptor record, set the DE-CCID flag to Y. Then establish a PREAUTHORIZATION junction between each entity to be changed and the CCID to which changes are to be attributed. In each of those PREAUTHORIZATION junctions, set the DE-CCID flag to Y.
1. **Private CCID.** You may need to make CCIDs private for several reasons: if you have established security by CCID or if you manage "Sensitive Entities" by CCID. In these (and other) cases, you will need to control which users are allowed to signon or make changes under a CCID. The restriction is specified by setting the TYPE of each restricted CCID to PRIVATE. Then establish a PREAUTHORIZATION junction between each USER that is to have access to a given CCID and the following entity:

```
ENTITY NAME = ccid-name
TYPE = CCID
VERSION = 1
```

2. **Private Status.** In promotion processing, the NDVRDSEL program EXCLUDE command will exclude any entity associated with a given STATUS. The ability to associate entities with the STATUSes used in your shop's promotion processing is therefore important. To control that ability, set the TYPE of each STATUS used in promotion processing to PRIVATE. Then establish a PREAUTHORIZATION junction between each USER that is to have the ability and the following entity:

```
ENTITY NAME = status-name
TYPE = STATUS
VERSION = 1
```

This section provides a step-by-step approach to assigning preauthorization for each of the objectives stated above.

You can do this through the Online facility or Batch facility. In Batch, you would use the ADD, MODIFY, and DELETE PREAUTHORIZATION commands. Refer to the *CA Endevor/DB for CA IDMS Batch Reference section* for more information on Batch. In Online, you would select option 2 from the Main Function Menu.

**Restricting Users Through Preauthorization**

When using preauthorization to restrict a user, the CCDB administrator defines preauthorization relationships between that user and the limited dictionary entities, which s/he is allowed to modify.

For example, you may want to restrict a programmer trainee (EDBADMIN) to modify only a limited set of training entities (beginning, in this example, with the characters DEPT). Once a list of preauthorized entities has been established for that user, the user is automatically denied the ability to modify any other dictionary entities.

To accomplish this, you can use the Online front end or Batch front end. In Batch, you would use the ADD PREAUTHORIZATION command. The Batch commands that equate to the next eight Online screens would be as follows:

```
ADD PREAUTHORIZATION ENTITY NAME = DEPT* TO USER EDBADMIN.
MOD SECURITY CLASS DEFAULT-SECURITY NO-AUTH = N LIM-AUTH = N A-OPT NONE.
```

To accomplish this through the Online front end, perform the following:

1. Enter option 2 on the Pre-authorization Functions menu (2-ADD PREAUTHORIZATIONS). Next, specify the entities which the user (EDBADMIN) will be able to modify (ENTITY NAME=DEPT*).
OPTION ==> 2
1 - BROWSE PRE-AUTHORIZATIONS
2 - ADD PRE-AUTHORIZATIONS
3 - DELETE PRE-AUTHORIZATIONS
4 - MODIFY PRE-AUTHORIZATIONS

ENTITY:
  NAME ==> DEPT*
  TYPE ==> 
  VERSION ==> 
  USER ==> EDBADMIN
  CCID ==> 

Press ENTER.

The system responds with a Pre-authorization List screen, which identifies all available entities as specified (DEPT). In the following example, all available programmer training entities (beginning with the characters DEPT) have been listed since the wildcard (*) was specified as part of the ENTITY name qualifier.

CA-E/DB nn.n volser 
/yy NDVRU210
USER ==> EDBADMIN
ACTION ==> AUTHORIZE

Where AUTH is Y, the entity is already preauthorized. Where DER is Y, one or more preauthorizations exist for the entity to a CCID with the DERIVE CCID option specified. Where OUT is Y, the entity is signed-out. The USER and CCID fields indicate the number of Users and CCIDs to which an entity is preauthorized. In the above example, the last listed entity (DEPTUPD-PREMAP) is preauthorized to one CCID (0001 CCID) with the DERIVE CCID option specified. For more information, use the Browse Preauthorization function for this entity.

Note: This screen lists all entities that have not been preauthorized to user EDBADMIN. User EDBADMIN, as specified on the Pre-authorization Functions screen, will reappear on the Pre-authorization Detail screen on the next page.

2. Press ENTER.

The system responds with a Pre-authorization Detail screen for each selected entity. A sample detail screen is shown below.

3. Using this screen, you further define the list of entities, which can be modified by this user.
Select the entities that you wish to preauthorize to the user (EDBADMIN) by entering any non-blank character to the left of the desired entries. In our example, all entities beginning with DEPTUPD have been selected.

4. Press ENTER.
The system responds with a Pre-authorization Detail screen for each selected entity. A sample detail screen is shown below.
4. **Estimated Work Completion** => **Actual Work Completion**

**Comment** =>

**Entity Information**

**Name** => DEPTUPD
**Type** => DIALOG
**Comment** =>

**User Information**

**Name** => EDBADMIN
**Security Class** => NDVR-GLOBAL
**Current CCID** =>
**Comment** => CCDB Administrator

**CCID Information**

**Name** =>
**Security Class** =>
**Locked** =>
**Comment** =>

Within the Pre-authorization Information section of the screen, fill in the appropriate fields to document preauthorization.

User information has already been filled in, based on earlier input from the Pre-authorization Functions screen. This information can be changed to preauthorize a different user, or it can be "spaced out" and replaced with CCID information to preauthorize a CCID.

By pressing ENTER after each detail screen as it appears, you're building a list of the entities preauthorized to the restricted user (EDBADMIN).

By pressing PF3, the system cancels your preauthorization request.

When all selected entities (DEPTUPD) have been entered, the system responds with a final list of all "leftover" (not preauthorized) entities remaining from the previous list. This enables you to double-check the list for any entities you may have missed.

**Note:** To preauthorize the same entities to another user, follow the same procedure as above. Another method is to preauthorize entities to a CCID, and then preauthorize users to that CCID. Entities may be preauthorized to single or multiple CCIDs, single or multiple users, or a combination of CCIDs and users. When entities are preauthorized to both Users and CCIDs, this does not force the preauthorized user to use one of the preauthorized CCIDs.

Now that you've built the preauthorization list for the user (EDBADMIN), "alert" the CA Endevor/DB Security System to heed that list. To do this:

1. Select option 10 (CA-Endevor/DB CONTROL FUNCTIONS) from the Main Function Menu.
1 - SIGNIN/SIGNOUT FUNCTIONS
2 - AUTHORIZATION FUNCTIONS
3 - LOCK FUNCTIONS
4 - ENTITY AND ENTITY CHANGE HISTORY
5 - CCID AND CCID CHANGE HISTORY
6 - STATUS AND STATUS ASSOCIATIONS
7 - USER AND USER CHANGE HISTORY
8 - DICTIONARY AND DICTIONARY HISTORY
9 - MANAGEMENT GROUPS AND CCIDS
10 - ENDEVOR/DB CONTROL FUNCTIONS
11 - ENDEVOR/DB SIGNON FUNCTION
12 - RETURN TO IDMS/DC

Press ENTER.
The system responds with the CA-Endevor/DB SYSTEM CONTROL FUNCTIONS screen.

2. Select option 5 (MODIFY SECURITY DESCRIPTORS).

   CA-E/DB nn.n volser    CA-ENDEVOR/DB SYSTEM CONTROL FUNCTIONS mm/dd/yy   NDVRUA00
   USER ===> EDBADMIN    DICTNAME ===> SRCNDVR    MODE ===> UPDATE
   OPTION ===> 5
   1 - BROWSE CCDB DESCRIPTOR RECORD  2 - MODIFY CCDB DESCRIPTOR RECORD
   3 - BROWSE SECURITY DESCRIPTORS    4 - ADD A SECURITY DESCRIPTOR
   5 - MODIFY SECURITY DESCRIPTORS    6 - DELETE SECURITY DESCRIPTORS
   7 - BROWSE MONITOR DICT STAT BLOCKS 8 - MODIFY MONITOR DICT STAT BLOCKS
   SECURITY CLASS ===>   DICTNAME ===> SRCNDVR
   (IF OPTIONS 3, 4, 5, 6)
   (IF OPTIONS 7, 8)

Press ENTER.
The system then provides a list of all the Security Classes in the CCDB on the SECURITY CLASS
LIST screen.

3. Select all items on the list by typing any non-blank character to the left of each Security Class
entry. This lets you "zoom in" on each Security Class in order to set security flags as needed.

   CA-E/DB nn.n volser    SECURITY CLASS LIST    mm/dd
   /yy NDVRUA10
   USER ===> EDBADMIN    DICTNAME ===> SRCNDVR    MODE ===> UPDATE
   ACTION ===> MODIFY
   SECURITY CLASS    COMMENT
   s DEFAULT-SECURITY SECURITY CLASS FOR RESTRICTED CAPABILITIES
   s QA SECURITY CLASS FOR QUALITY ASSURANCE
   s DEVELOPMENT SECURITY CLASS FOR DEVELOPMENT
   s SUPPORT SECURITY CLASS FOR TECHNICAL SUPPORT
   s NDVR-DDA DICTIONARY ADMINISTRATION CAPABILITIES
   s NDVR-GLOBAL UNIVERSAL ENDEVOR/DB AND DICTIONARY CAPABILITIES
   ** END **

Press ENTER.
The system responds with a SECURITY CLASS DETAIL screen for each Security Class selected on
the above list.

4. On the SECURITY CLASS DETAIL screens that apply to all other (non-restricted) users, set the
Preauthorization flags on the Security Class screens as follows (No Preauthorization required):

   LIM-AUTH=Y
   NO-AUTH=Y

Also, set all A-Opt flags to Y.

5. On the SECURITY CLASS DETAIL screen that applies to restricted user EDBADMIN (in our
example, the Security Class is DEFAULT-SECURITY), set the Preauthorization flags as follows
(Full Preauthorization):

   LIM-AUTH=N
LIM-AUTH=N
NO-AUTH=N

Also, set all A-Opt flags to N.

**Important!** Do not set both LIM-AUTH and NO-AUTH to N for the dictionary Security Class NDVR-GLOBAL unless your intentions are to preauthorize all users to every entity before they update it.

Note: If there are specific entity types for which preauthorization rules are to be ignored, set those individual A-Opt flags to Y.

Now that the preauthorization list has been built, and the Security Class flags have been set, user EDBADMIN’s access (and all other users with Security Class DEFAULT-SECURITY) is restricted to only those preauthorized entities. If user EDBADMIN attempts to modify an entity to which s/he is not preauthorized, the CA Endevor/DB Security System will prevent access and display an error message to that effect.

To remove preauthorization, select option 3 (DELETE PREAUTHORIZATION) on the Pre-authorization Functions screen.

### Protecting Critical Entities Through Preauthorization

When using preauthorization to protect critical or sensitive entities, the CCDB administrator is restricting those entities from modification by the general user population. Typical examples would be company payroll and personnel programs that contain confidential information. In this case, users would be preauthorized to make modifications to those sensitive programs. Dictionary entities that were not deemed to be sensitive could be modified by any user.
To accomplish this, the entity can be preauthorized to the user or CCID that will make modifications to the entity. The security classes for the general user population would be modified to disallow modifications to entities, which have been preauthorized to another user. You can do this by using either the Online front end or the Batch front end. In Batch, you would use the ADD PREAUTHORIZATION and MODIFY SECURITY CLASS commands. The Batch commands that equate to the next eight Online screens would be as follows:

ADD PREAUTHORIZATION ENTITY NAME = DEPT* TO USER EDBADMIN.
MOD SECURITY CLASS NDVR-DDA NO-AUTH = N LIM-AUTH = Y A-OPT NONE.
MOD SECURITY CLASS SUPPORT NO-AUTH = N LIM-AUTH = Y A-OPT NONE.
MOD SECURITY CLASS PROJECT-LEADER NO-AUTH = N LIM-AUTH = Y A-OPT NONE.
MOD SECURITY CLASS GMG-SECURITY NO-AUTH = N LIM-AUTH = Y A-OPT NONE.

The first command preauthorizes the user EDBADMIN to make modifications to all entities, which begin with DEPTUPD. The MODIFY SECURITY commands modify all the security classes for the general user population to disallow modifications to entities for which they have not been preauthorized.

To accomplish this through the Online front end, perform the following:

1. Select option 2 (ADD PREAUTHORIZATIONS) on the Pre-authorization Functions menu. Next, specify the entities which the user (EDBADMIN) will be able to modify (ENTITY NAME=DEPT*).

Press ENTER.

The system responds with a Pre-authorization List screen, which identifies all available entities as specified (DEPT). In the following example, the entities beginning with the characters DEPT have been listed since the wildcard (*) was specified as part of the ENTITY name qualifier.

Press ENTER.

* END *
Where AUTH is Y, the entity is already preauthorized. Where DER is Y, one or more preauthorizations exist for the entity to a CCID with the DERIVE CCID option specified. Where OUT is Y, the entity is signed-out. The USER and CCID fields indicate the number of users and CCIDs to which an entity is preauthorized. In the above example, the last listed entity (DEPTUPD-PREMAP) is preauthorized to one CCID (0001 CCID) with the DERIVE CCID option specified. For more information, use the Browse Preauthorization function for this entity.

**Note:** This screen lists all entities that have not been preauthorized to User EDBADMIN. User EDBADMIN, as specified on the Pre-authorization Functions screen, will reappear on the Pre-authorization Detail screen on the next page.

2. To further define the list of entities which can be modified by this user, select the entities that you wish to preauthorize to the user (EDBADMIN) by entering any non-blank character to the left of the desired entries. In our example, all entities named DEPTUPD have been selected. Press ENTER.

The system responds with a Pre-authorization Detail screen for each selected entity. A sample detail screen is shown below.

```
CA-E/DB nn.n volser PRE-AUTHORIZATION DETAIL mm/dd
/yy NDVRM210
USER ===> EDBADMIN DICTNAME ===> SRCNDVR MODE ===> UPDATE
ACTION ===> AUTHORIZE
*************** PRE-AUTHORIZATION INFORMATION ***********************
DERIVE CCID ===> N SIGNED OUT ===> N PRE-AUTHORIZED ===> N
EST. WORK COMPLETION ===> ACT. WORK COMPLETION ===> COMMENT ===>
*************** ENTITY INFORMATION ***************
NAME ===> DEPTUPD VERSION ===> 1
TYPE ===> DIALOG
COMMENT ===>
*************** USER INFORMATION ***************
NAME ===> EDBADMIN SECURITY CLS ===> NDVR-GLOBAL
CURRENT CCID ===> LOCKED ===> N
COMMENT ===> CCDB ADMINISTRATOR
*************** CCID INFORMATION ***************
NAME ===> SECURITY CLASS ===> LOCKED ===> COMMENT ===>
```

3. Within the Pre-authorization Information section of the screen, fill in the appropriate fields to document preauthorization.

User information has already been filled in, based on earlier input from the Pre-authorization Functions screen. This information can be changed to preauthorize a different user, or it can be "spaced out" and replaced with CCID information to preauthorize a CCID.

By pressing ENTER after each detail screen as it appears, you're building a list of the entities preauthorized to the restricted user (EDBADMIN).

To cancel your preauthorization request, press PF3.

When all selected entities (DEPTUPD) have been entered, the system responds with a final list of all "leftover" (not preauthorized) entities remaining from the previous list. This enables you to double-check the list for any entities you may have missed.

```
CA-E/DB nn.n volser PRE-AUTHORIZATION LIST mm/dd
/yy NDVRU210
USER ===> EDBADMIN DICTNAME ===> SRCNDVR MODE ===> UPDATE
NDVRM210: I002 ALL SELECTED RECORDS PROCESSED
```
4. Return to the Main Function Menu by pressing CLEAR or PF3.

Note: To preauthorize the same entities to another user, follow the same procedure as above. Another method is to preauthorize entities to a CCID, and then preauthorize users to that CCID. Entities may be preauthorized to single or multiple CCIDs, single or multiple Users, or a combination of CCIDs and Users. When entities are preauthorized to both Users and CCIDs, this does not force the preauthorized user to use one of the preauthorized CCIDs.

Now that you’ve built the preauthorization list for the user (EDBADMIN), "alert" the CA Endevor/DB Security System to heed that list. To do this:

1. Select option 10 (ENDEVOR/DB CONTROL FUNCTIONS) from the Main Function Menu.

Press ENTER.
The system responds with the CA-ENDEVOR/DB SYSTEM CONTROL FUNCTIONS screen.

2. Select option 5 (MODIFY SECURITY DESCRIPTORS).

Press ENTER.
The system then provides a list of all the Security Classes in the database on the SECURITY CLASS LIST screen.
3. Select all items on the list by typing any non-blank character to the left of each Security Class entry. This lets you "zoom in" on each Security Class in order to set security flags as needed.

```
CA-E/DB nn.n volser SECURITY CLASS LIST mm/dd
/yy NDVRUA10
USER ===> EDBADMIN DICTNAME ===> SRCNDVR MODE ===> UPDATE
ACTION ===> MODIFY

SECURITY CLASS COMMENT
s DEFAULT-SECURITY SECURITY CLASS FOR RESTRICTED CAPABILITIES
s QA SECURITY CLASS FOR QUALITY ASSURANCE
s DEVELOPMENT SECURITY CLASS FOR DEVELOPMENT
s SUPPORT SECURITY CLASS FOR TECHNICAL SUPPORT
s NDVR-DDA DICTIONARY ADMINISTRATION CAPABILITIES
s NDVR-GLOBAL UNIVERSAL ENDEVOR/DB AND DICTIONARY CAPABILITIES
** END **
```

Press ENTER.
The system responds with a SECURITY CLASS DETAIL screen for each Security Class selected on the above list.

4. On all of the SECURITY CLASS DETAIL screens that follow, set the Preauthorization flags as follows (Limited Preauthorization):

- **LIM-AUTH=Y**
- **NO-AUTH=N**

Also, set all A-Opt flags to **N**.
These screens will set the flags in all the security classes that you have selected.

---

**Important!** Do not set both LIM-AUTH and NO-AUTH to **N** for the dictionary Security Class NDVR-GLOBAL unless your intentions are to preauthorize all users to every entity before they update it.

```
CA-E/DB nn.n volser SECURITY CLASS DETAIL mm/dd/yy NDVRMA10
USER ===> EDBADMIN DICTNAME ===> SRCNDVR MODE ===> UPDATE
ACTION ===> MODIFY

************************** SECURITY CLASS INFORMATION ****************************
NAME ===> DEFAULT-SECURITY
COMMENT ===> SECURITY CLASS FOR RESTRICTED CAPABILITIES
MENU 1 2 3 4 5 6 7 8 9 MENU 1 2 3 4 5 6 7 8
CONTROL: Y N Y N N N N N SIGNOUT: Y Y Y
LOCK: N N N N N N N N N AUTH: N N N N
CCID: Y N N N Y Y Y Y Y ENTITY: Y Y Y Y Y Y Y
STATUS: Y N N N Y Y Y Y Y USER: Y N N N Y N N N
M-GRP: Y N N N Y N N N DICT: Y N N Y Y Y
SIGNIN: Y SO-CCID: N SO-USER: Y NO-USER: Y NO-CCID: Y NO-AUTH: N LIM-AUTH: Y
MM-MODE: N ARCHIVE: N Migrate: N DE-CCID: N BATCH: N
ENTITY: SCH DMC FIL TAS SUB USE DES REC SYS APO SET DIA APP ELE QFI PRC TAB FUN
MODS: Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
A-OPT: N N N N N N N N N N N N N N N N N N N N N N N
ENTITY: MOD PHY CLA ATT MAP LOG LIN MSG LOA LR PRO CCD DIC EUS CCI MGR STA SEC
MODS: Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
A-OPT: N N N N N N N N N N N N N N N N N N
```

If there are specific entity types for which preauthorization rules are to be ignored, set those individual A-Opt flags to **Y**.
Now that the preauthorization list has been built, and the Security Class flags have been set, user EDBADMIN's access (and all other users with Security Class DEFAULT-SECURITY) is restricted to only those preauthorized entities. If user EDBADMIN attempts to modify an entity to which s/he is not preauthorized, the CA Endevor/DB Security System will prevent access and display an error message to that effect.

To remove preauthorization, select option 3 (DELETE PREAUTHORIZATION) on the Pre-authorization Functions screen.

Restricting Access to a CCID Through Preauthorization

The CA Endevor/DB Security System allows an administrator to restrict the users that are allowed to sign on or make changes under a CCID. This is necessary to insure security integrity in the following situations:

- Security management by CCID. If the security class restrictions are administered by CCID (instead of by user).
- If sensitive entities are preauthorized to CCIDs.

To control which users are allowed to SIGNON to a given CCID or to make changes under that CCID, you must mark the CCID as PRIVATE and then establish preauthorizations between each entitled user and the following entity:

```
ENTITY NAME = ccid
VERSION = 1
TYPE = CCID
```

Both of these actions can be performed with either the Online front end or the Batch front end. For example, the Batch commands would be:

```plaintext
MOD CCID EDB-SYSADMIN TYPE PRIVATE.
ADD PREAUTHORIZATION ENTITY NAME EDB-SYSADMIN
TYPE CCID VERSION 1 TO USER EDBADMIN.
```

To accomplish this with the Online front end:

1. Select option 2 on the Pre-authorization Functions screen (ADD PREAUTHORIZATIONS).
   Specify an ENTITY NAME (in our example, EDB-SYSADMIN), type CCID in the ENTITY TYPE field, and type a 1 in the ENTITY VERSION field. In the USER field, enter the userid for the user (EDBADMIN in our example) you are preauthorizing.

```
CA-E/DB nn.n volser        PRE-AUTHORIZATION FUNCTIONS        mm/dd/yy  NDVRU200
USER ===> EDBADMIN        DICTNAME ===> SRCNDVR        MODE ===> UPDATE
OPTION ===> 2
  1 - BROWSE PRE-AUTHORIZATIONS          2 - ADD PRE-AUTHORIZATIONS
  3 - DELETE PRE-AUTHORIZATIONS          4 - MODIFY PRE-AUTHORIZATIONS
ENTITY:
  NAME ===> EDB-SYSADMIN
  TYPE ===> CCID
  VERSION ===> 1
USER ===> EDBADMIN
CCID ===> EDBADMIN
```

Press ENTER.
Press ENTER. Since Full Preauthorization qualification information has been specified, the system bypasses 
the list screen and responds directly with a Pre-authorization Detail screen.

Press ENTER. User EDBADMIN is now preauthorized to use the CCID named EDB-SYSADMIN. 
Follow this same procedure for each user to whom you want to grant preauthorization for 
that CCID. The end result is a group of users that is now preauthorized to a specific CCID.

Return to the Main Function Menu and select option 5 (CCID AND CCID CHANGE HISTORY).

Select option 3 (MODIFY CCID DESCRIPTORS). Enter the name of the CCID (EDB-SYSADMIN in 
our example).
4. **CHANGE-LOG SELECTION CRITERIA:**

   (IF OPTIONS 5 - 8 )
   
   **START DATE ===> END DATE ===> 04/30/97**
   
   **START TIME ===> END TIME ===>**
   
   **ACTION CODE ===>**

   Press ENTER.

   The system responds with a CCID DETAIL screen, which shows the selected CCID.

5. In order to restrict access to a CCID (in our case, EDB-SYSADMIN), change the CCID TYPE to PRIVATE.

   CA-E/DB nn.n volser      CCID DETAIL      mm/dd/yy      NDVRM510
   USER ===> EDBADMIN     DICTNAME ===> SRCNDVR     MODE ===> UPDATE
   ACTION ===> MODIFY
   ****************************************** CCID INFORMATION ****************************************
   NAME ===> EDB-SYSADMIN    SEC. CLASS ===> NDVR-GLOBAL    TYPE ===> PRIVATE
   COMMENT ===> EDB SYSTEM ADMINISTRATION
   LOCKED ===> N             LOCK DATE ===>            LOCK TIME ===> 

   Press ENTER.

   **Important!** Use of that CCID is now restricted to only those users who are specifically preauthorized.

   To remove preauthorization, select option 3 (DELETE PREAUTHORIZATION) on the Pre-authorization Functions screen.

### Assigning Status Privileges Through Preauthorization

The CA Endevor/DB Security System allows an administrator to restrict the users that are allowed to assign a status to an entity. This is necessary when controlling what entities are included in a migration using status(es).

To control which users are allowed to assign a status to an entity(ies), you must mark the status as PRIVATE and then establish preauthorizations between each entitled user and the following entity:

   ENTITY NAME = status-name
   VERSION = 1
   TYPE = STATUS

Both of these actions can be performed with either the Online front end or the Batch front end. For example, the Batch commands would be:

   MODIFY STATUS NEVER-MIGRATE TYPE PRIVATE.
   ADD PREAUTHORIZATION ENTITY NAME IS NEVER-MIGRATE
   TYPE IS STATUS VERSION IS 1 TO USER EDBADMIN.

To accomplish this with the Online front end:
1. Select option 2 on the Pre-authorization Functions screen (ADD PREAUTHORIZATIONS).
Specify an ENTITY NAME (in our example, NEVER-MIGRATE), type STATUS in the ENTITY TYPE field, and 1 in the ENTITY VERSION field. Then, specify the user to be allowed status privileges (in our case, EDBADMIN).

```
CA-E/DB nn.n volser   PRE-AUTHORIZATION FUNCTIONS   mm/dd/yy   NDVRU200
USER ===> EDBADMIN   DICTNAME ===> SRCNDVR   MODE ===> UPDATE
OPTION ===> 2
1 - BROWSE PRE-AUTHORIZATIONS 2 - ADD PRE-AUTHORIZATIONS
3 - DELETE PRE-AUTHORIZATIONS 4 - MODIFY PRE-AUTHORIZATIONS
ENTITY:
   NAME ===> NEVER-MIGRATE
   TYPE ===> STATUS
   VERSION ===> 1
   USER ===> EDBADMIN
   CCID ===> (IF OPTIONS 1 - 4)
```

Press ENTER.
The system responds with a Pre-authorization Detail screen.

```
CA-E/DB nn.n volser   PRE-AUTHORIZATION DETAIL   mm/dd/yy   NDVRM210
USER ===> EDBADMIN   DICTNAME ===> SRCNDVR   MODE ===> UPDATE
ACTION ===> AUTHORIZE
PRE-AUTHORIZATION INFORMATION
-----------------------------
NAME ===> NEVER-MIGRATE
TYPE ===> STATUS
VERSION ===> 1
USER ===> EDBADMIN
CCID ===> (IF OPTIONS 1 - 4)
```

2. Press ENTER. The designated user (EDBADMIN) is now preauthorized to set or turn off the specified Status (NEVER-MIGRATE). Follow the above procedure for every user to whom you want to assign status setting privileges. The end result is a group of users that is able to set and remove the status for entities.

3. Return to the Main Function Menu and select option 6 (STATUS AND STATUS ASSOCIATIONS).

```
CA-E/DB nn.n volser   MAIN FUNCTION MENU   mm/dd/yy   NDVRU000
USER ===> EDBADMIN   DICTNAME ===> SRCNDVR   MODE ===> UPDATE
OPTION ===> 6
1 - SIGNIN/SIGNOUT FUNCTIONS
2 - AUTHORIZATION FUNCTIONS
3 - LOCK FUNCTIONS
4 - ENTITY AND ENTITY CHANGE HISTORY
5 - CCID AND CCID CHANGE HISTORY
6 - STATUS AND STATUS ASSOCIATIONS
7 - USER AND USER CHANGE HISTORY
8 - DICTIONARY AND DICTIONARY HISTORY
9 - MANAGEMENT GROUPS AND CCIDS
10 - ENDEVOR/DB CONTROL FUNCTIONS
11 - ENDEVOR/DB SIGNON FUNCTION
12 - RETURN TO IDMS/DC
```

4. Press ENTER.
The system responds with the STATUS FUNCTIONS screen.
5. Select option 3 (MODIFY STATUS DESCRIPTORS) and, in the STATUS field, fill in the name of the STATUS to which status privileges will be applied (NEVER-MIGRATE).

6. Press ENTER.
   The system responds with a STATUS DETAIL screen.

7. Change the TYPE from PUBLIC to PRIVATE.

   Press ENTER. The user (EDBADMIN) is now preauthorized to set or turn off NEVER-MIGRATE status for any entity.

   To remove preauthorization, select option 3 (DELETE PREAUTHORIZATION) on the Pre-authorization Functions screen.

---

**Preparing for Derived CCID Processing**

When using DERIVED-CCID processing, the CCID(s) associated with a change are determined when the change is made to an entity. The DERIVED CCID processing allows a user to associate changes to one or more CCIDs without the user having to signon to CA Endevor/DB with the CCID(s). To turn on DERIVED CCID processing, the dictionary's and user's security class must specify DE-CCID = Y. If they are not, DERIVED CCID processing is not active. In addition, entities are preauthorized to the derived CCIDs with the DE-CCID flag set to Y.

To accomplish this, you can either use the Online front end or the Batch front end. In Batch, you would use the ADD PREAUTHORIZATION and the MODIFY SECURITY CLASS commands. The Batch commands to set up CCID EDB-QA as a derived CCID for all entities whose name begins with "DEPT" and to assign derived CCID processing for all security classes would be as follows:

```
ADD PREAUTHORIZATION ENTITY NAME = DEPT*
TO CCID EDB-QA DERIVE CCID = Y.
MOD SECURITY CLASS = * DERIVE CCID = Y.
```

To accomplish this through the Online front end, perform the following:
1. Select option 2 on the Pre-authorization Functions menu (2-ADD PREAUTHORIZATIONS). Next, specify the entities to which the CCID EDB-QA is to be preauthorized.

    CA-E/DB nn.n volser  PRE-AUTHORIZATION FUNCTIONS  mm/dd/yy NDVRU200
    USER ===⇒ EDBADMIN   DICTNAME ===⇒ SRCNDVR   MODE ===⇒ UPDATE
    OPTION ===⇒ 2
    1 - BROWSE PRE-AUTHORIZATIONS
    2 - ADD PRE-AUTHORIZATIONS
    3 - DELETE PRE-AUTHORIZATIONS
    4 - MODIFY PRE-AUTHORIZATIONS
    ENTITY: (IF OPTIONS 1 - 4)
    NAME ===⇒ DEPT*
    TYPE ===⇒
    VERSION ===⇒
    USER ===⇒
    CCID  EDB-QA (IF OPTIONS 1 - 4)

Press ENTER.

The system responds with a Pre-authorization List screen, which identifies all available entities whose name begins with DEPT. In the following example, all entities (beginning with the characters DEPT) not already preauthorized to CCID EDB-QA have been listed (since the wildcard (*) was specified as part of the ENTITY name qualifier).

2. Using this screen, you further define the list of entities that are to be preauthorized to this CCID. Select the entities that you wish to preauthorize to the CCID (EDB-QA) by entering any non-blank character to the left of the desired entries. In our example, all entities whose name begins with "DEPT-" have been selected.

    CA-E/DB nn.n volser  PRE-AUTHORIZATION LIST  mm/dd/yy NDVRU210
    USER ===⇒ EDBADMIN   DICTNAME ===⇒ SRCNDVR   MODE ===⇒ UPDATE
    ACTION ===⇒ AUTHORIZE
    USER  CCID  OUT  AUTH  DER  ENTITY  NAME  TYP  VERS
    _    N    N    N    DEPTINQ
    _    N    N    N    DEPTINQ-ENTER
    _    N    N    N    DEPTINQ-PREMAP
    _    N    N    N    DEPTMAP
    _    N    N    N    DEPTMAP
    s    N    N    N    DEPTMAP
    s    N    N    N    DEPTMAP
    s    001 CCIDS N    Y    N    DEPTUPD-ENTER
    s    001 CCIDS N    Y    N    DEPTUPD-PREMAP

    * END *

Where AUTH is Y, the entity is already preauthorized. Where DER is Y, DERIVE CCID processing is in effect for the preauthorization. Where OUT is Y, the entity is signed-out. The USER and CCID fields indicate the number of USERs and CCIDs to which an entity is preauthorized. In the above example, the last listed entity (DEPTUPD-PREMAP) is preauthorized to one CCID (0001 CCID). For more information, use the Browse Preauthorization function as applied to this entity.

⚠️ **Note:** This screen lists all entities that have not already been preauthorized to CCID EDB-QA. CCID EDB-QA, as specified on the Pre-authorization Functions screen, will reappear on the Pre-authorization Detail screen on the next page.

3. Press ENTER.

The system responds with a Pre-authorization Detail screen for each selected entity. A sample detail screen is shown below.
Within the Pre-authorization Information section of the screen, fill in the appropriate fields to document preauthorization:

1. Enter a Y in the DERIVE CCID field to specify that this CCID is to be derived. The CCID information has already been filled in, based on earlier input from the Pre-authorization Functions screen. This information can be changed to preauthorize a different CCID.

2. By pressing ENTER after each detail screen as it appears, you're building a list of the entities preauthorized to the CCID EDB-QA.

3. By pressing PF3, the system will cancel your preauthorization request. When all selected entities whose name begins with DEPTUPD have been entered, the system responds with a final list of all "leftover" (not preauthorized) entities remaining from the previous list. This enables you to double-check the list for any entities you may have missed.

4. Return to the Main Function Menu by pressing CLEAR or PF3.

Note: To preauthorize the same entities to another CCID, follow the same procedure as above. Entities may be preauthorized to single or multiple CCIDs.

Now that you've built the preauthorization list for the CCID EDB-QA, you need to modify the security descriptors for the dictionary descriptor and the users who will be using DERIVED CCID processing. To do this:

1. Select option 10 (ENDEVOR/DB CONTROL FUNCTIONS) from the Main Function Menu.
2. Press ENTER.
The system responds with the CA-ENDEVOR/DB SYSTEM CONTROL FUNCTIONS screen.

3. Select option 5 (MODIFY SECURITY DESCRIPTORS).

4. Press ENTER.
The system then provides a list of all the Security Classes in the database on the SECURITY CLASS LIST screen.

5. This action changes all security classes to use derived CCID processing. Select all items on the list by typing any non-blank character to the left of each Security Class entry. This lets you "zoom in" on each Security Class in order to set security flags as needed.

6. Press ENTER.
The system responds with a SECURITY CLASS DETAIL screen for each Security Class selected on the above list.

7. On the Security Class Detail screen, enter a Y in the DE-CCID field. This will enable DERIVED CCID processing for users and the dictionary descriptors, which have these security classes.
Note: If the security class for the dictionary descriptor (NDVR-GLOBAL) does not specify DERIVED CCID processing (DE-CCID = Y), DERIVED CCID processing will not be active even if a user's security class in that dictionary specified CCID processing.

By pressing ENTER after each detail screen as it appears, you are building a group of security classes which specify derived CCID processing. Press PF3 to cancel your MODIFY SECURITY CLASS request.

Lock Security

In addition to preauthorizing users and CCIDs, security measures can be taken one step further using a level of control called lock. The lock function is used to prevent use of an CA Endevor/DB userid or CCID, and/or to prevent a dictionary from being updated. Lock is a temporary condition; when you want to allow access again, you simply unlock the entity.

You can lock:

- An CA Endevor/DB User (userid) - which prevents signon and updates by that userid.
- A CCID - which prevents signon under that CCID and updates logged to that CCID.
- A Dictionary - which prevents any modifications from being done in the dictionary.

This section provides a step-by-step approach to locking and unlocking users, CCIDs, and the dictionary.

Lock processing begins when you select option 3 from the Main Function Menu:

The system returns the Lock/Unlock Functions menu below:
The Browse Option

Prior to locking or unlocking a particular user(s), you may want to determine who has already been locked. Use the browse option (1) to retrieve a list of these users.

Note: Because the procedures are the same for all three entities, the examples described and illustrated on the following pages pertain to only one entity - users. To browse, lock, or unlock a CCID or dictionary, you would simply select the appropriate option number and follow these same procedures.

1. Select option 1 on the Lock/Unlock Functions menu (1 - BROWSE LOCKED USERS).

2. Press ENTER.

3. Select the userid you want to review by entering any non-blank character to the left of the desired entry. In this example, userid EDBADMIN has been selected.

4. Press ENTER.

The remainder of this section discusses the three options and related procedures used in LOCK processing. Read the section(s) appropriate to your needs.

- The Browse Option (see page 65)
- The Lock Option (see page 66)
- The Unlock Option (see page 67)
5. Press ENTER and then PF3 to return directly to the Lock/Unlock Functions menu.

If you selected more than one userid for review, the User Lock Detail screen for the next user appears. Continue pressing ENTER until all detail screens have been displayed; the last ENTER brings you back to the User Lock List. Press PF3 to return to the Lock/Unlock Functions screen.

The Lock Option

When you lock a particular user, you are preventing that userid from logging onto the CA Endevor/DB system, as well as preventing IDD updates by that user. This restriction remains in place until you unlock the userid.

1. Select option 2 on the Lock/Unlock Functions menu (2 - LOCK USERS). In addition, enter the userid you want to lock; in this example, userid SYSADMIN is designated.

2. Press ENTER.

   The User Lock Detail screen displays for userid SYSADMIN.

3. To lock userid SYSADMIN, press ENTER. Then, press PF3 to return to the Lock/Unlock Functions menu.

   Note the LOCKED field on the bottom line of the screen. When you first see the screen, the letter N appears in this field. Locking the userid automatically changes the entry to Y, which appears in the field when you next access this screen. In addition, userid SYSADMIN will be listed on the User Lock List the next time that screen appears.
An Alternate Procedure

If for some reason you did not indicate a specific user on the Lock/Unlock Functions menu, the User Lock List appears when you press ENTER. This list contains all CA Endevor/DB users who are not already locked. You can now select the userid(s) you want to lock by entering any non-blank character next to the desired entry(ies). Press ENTER; when the User Lock Detail screen appears for the user, you can perform any of the following activities:

- **LOCK this user.** Simply press ENTER; if you selected more than one userid, the User Lock Detail screen for the next userid displays.

- **NOT lock this user.** Simply clear the ACTION field and press ENTER. Again, if you indicated more than one userid to be locked, the next User Lock Detail screen displays.

- Return directly to the User Lock List. Press PF3.

Continue locking or not locking until all detail screens have been displayed. The last ENTER brings you back to the User Lock List. Press PF3 to return to the Lock/Unlock Functions menu.

The Unlock Option

**Contents**

- An Alternate Procedure (see page 68)
- CCIDs and Dictionaries (see page 68)

Unlocking a userid allows users to signon under that userid once again.

1. Select option 3 on the Lock/Unlock Functions menu (3-UNLOCK USERS) and enter the userid you want to unlock. In this example, userid SYSADMIN will be unlocked.

   ```
   CA-E/DB nn.n volser LOCK/UNLOCK FUNCTIONS mm/dd/yy NDVRU300
   USER ==> EDBADMIN DICTNAME ==> SRCNDVR MODE ==> UPDATE
   OPTION ==> 3
   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
   USER ==> SYSADMIN
   CCID ==> (IF OPTIONS 4, 5, 6)
   DICTIONARY ==> SRCNDVR (IF OPTIONS 7, 8, 9)
   ```

2. Press ENTER.
   The system again responds with the User Lock Detail screen for userid SYSADMIN. Now, however, the ACTION field states unlock rather than lock.

   ```
   CA-E/DB nn.n volser USER LOCK DETAIL mm/dd/yy NDVRM310
   USER ==> EDBADMIN DICTNAME ==> SRCNDVR MODE ==> UPDATE
   ACTION ==> UNLOCK
   ********************************************************** USER INFORMATION **********************************************************
   USER ==> SYSADMIN PASSWORD ==> SECURITY CLS ==> NDVR-GLOBAL
   CURRENT CCID ==> COMMENT ==> LOCKED ==> Y LOCK DATE ==> 04/30/97 LOCK TIME ==> 10:02:12
   ```
3. To unlock userid SYSADMIN, press ENTER. Then press PF3 to return to the Lock/Unlock Functions menu.

On this display, the entry in the LOCKED field is Y. As with the LOCK option, the entry automatically changes when you press ENTER - in this case, to N. And, userid SYSADMIN will no longer be on the User Lock List screen.

An Alternate Procedure

If you did not indicate a specific userid on the Lock/Unlock Functions menu, the User Lock List appears when you press ENTER. This list consists of all CA Endevor/DB users currently in a locked condition. Select the userid(s) you want to unlock by entering any character next to the desired entry (ies). Press ENTER; when the USER Lock Detail screen appears, you can perform any of the following activities:

- **UNLOCK this user.** Press ENTER; if you selected more than one userid, the User Lock Detail screen for the next userid is returned.

- **NOT unlock this userid.** Simply clear the ACTION field and press ENTER. Again, if you indicated more than one userid, the next User Lock Detail screen displays.

- Return directly to the User Lock List by pressing PF3.

Continue locking or unlocking until all detail screens have been displayed. The last ENTER brings you back to the User Lock List. Press PF3 to return to the Lock/Unlock Functions menu.

CCIDs and Dictionaries

As mentioned above, the procedures for browsing, locking, and unlocking CCIDs and dictionaries are the same as those for users. Select the appropriate options for these entities, as listed below:

- To **BROWSE LOCKED CCIDS** - select option 4.

- To **LOCK CCIDS** - select option 5.

- To **UNLOCK CCIDS** - select option 6.

- To **BROWSE LOCKED DICTIONARIES** - select option 7.

- To **LOCK DICTIONARIES** - select option 8.

- To **UNLOCK DICTIONARIES** - select option 9.
Archiving and Compressing the CCDB

CA Endevor/DB's archive and compress facility, NDVRARCO, allows the CCDB administrator to eliminate obsolete information from the CCDB, to optimize disk space, to transfer essential data onto tape or disk for future reference, and to maintain essential information in special Configuration Change Log entries within CCDB. The archive and compress facility accomplishes this by performing the following tasks:

- **Removing Uncommitted Change Log Entries from the CCDB.**
  If an updating program (such as IDD) runs a job which is abended or interrupted, that incomplete job is marked as "uncommitted" in the CCDB database. CA Endevor/DB ignores uncommitted CLEs for change control purposes, and disposes of them when the ARCHIVE/COMPRESS program is run.

- **Compressing Data in the CCDB.**
  The goal of compression is to eliminate "noise" records in the CCDB that add little value to the audit trail or change control process. During compression, contiguous strings of CLEs, which relate to the same user and CCID entity are summarized into a single compressed CLE with a net action code (depending on the sequence of activity). When the Change Log History for dictionaries, CCIDs, users or entities is displayed, compressed CLEs appear in their original time relative positions. In order to do compress processing, the user must enter the COMPRESS command and specify a compress age.

- **Archiving Data from the CCDB to a Sequential File.**
  Archive processing enables the CCDB administrator to transfer data from the CCDB into a sequential file (tape or disk). Typically, these records contain obsolete change information, which is no longer needed on a regular basis, yet warrants being preserved for future reference. Such would be the case, for example, when a migration is completed and the associated change information is no longer needed in the online database for future migrations, regression detection, or audit trail purposes. Archiving that information would free up space in the CCDB while maintaining a "pre-migration" copy of that change log information on tape for historical purposes. In order to invoke archive processing, the user must specify the ARCHIVE command and specify the name of the CCID, USER, or MANAGEMENT-GROUP to be archived.

- **Modifying Confirmation Log Entries**
  This function allows the CCDB administrator to modify the dictionary name and/or system identifier information contained in Confirmation Change Log entries written to the CCDB during the promotion, or migration, process.

- **Converting Secondary Commit Groups**
  When Derived CCID processing is used, secondary commit groups are sometimes created to associate Change Log Entries (CLEs) to CCIDs. The ARCHIVE/COMPRESS program converts them to ordinary commits.

**Note:** Because secondary commit records contain a dbkey as data, the ARCHIVE/COMPRESS program must be run prior to running any CA IDMS utility (such as UNLOAD/RELOAD) which will change dbkeys for existing CCDB NDVRCOMT records. The ARCHIVE/COMPRESS program will eliminate the stored dbkey values.
Installations commonly apply both compression and archive to the same CCDB, using these facilities for two basic purposes:

- For routine cleanup/maintenance, where uncommitted jobs are automatically deleted and "noise" information is systematically compressed.

- For archiving the change log history used to drive a migration process. The same USER/CCID /MANAGEMENT-GROUP used for the NDVRDSEL SELECT commands would generally be used for the ARCHIVE commands.

The use of CA Endevor/DB to control the promotion or migration process depends completely on the contents of the CCDB. The use of NDVRARCO's ARCHIVE processing removes Change Log records from the CCDB; the use of NDVRARCO's ALTER CONFIRMATION processing modifies the dictionary identification information contained in Confirmation Change Log records in the CCDB.

There are three major guidelines to be followed when using NDVRARCO's ARCHIVE processing.

- **Multiple Migration Targets.**
  If you intend to migrate dictionary entities to more than one target, you should never use NDVRARCO to archive the source dictionary development change history until the migration is complete to all targets. Do not archive a CCID used to trigger the selection of entities until it has been received by all target systems.

- **Multiple Migration Sources.**
  If you make changes to dictionary entities at two or more source dictionaries, you should never use NDVRARCO to archive change history until all affected dictionaries are synchronized. For example, let's say that development is done at dictionary A, QA at dictionary B, and production at dictionary C. If a migration from dictionary A to dictionary B is followed by a QA-related change at B, then the history of change at B cannot be archived until migrated to, or reproduced at, both A and C. Use the Post Migration Activity Report (NDVRPT15) to identify QA and production fixes.

- **Vendor-Supplied Source.**
  If you make changes to vendor-supplied dictionary entities, you should never use NDVRARCO to archive the history of those changes. The audit trail of these changes will always be needed to upgrade to new maintenance levels and releases of the vendor product. Regular and exclusive use of compression in these cases will keep the growth of the CCDB at moderate and predictable levels.

NDVRARCO's ALTER CONFIRMATION processing requires some thought before its use. Since selection of entities for promotion is based on "the last time the entity was promoted to the target dictionary" and the target dictionary is identified by the Dictionary Name and System Identifier in the Confirmation Change Log records in the CCDB, accurate dictionary identification information is crucial. If a "hooked," or monitored, dictionary is moved from one system to another, or the system it resides in is renamed, or if the segment name of the dictionary is changed (causing a change in the Database Name Table DBName entry for the monitored dictionary), this information needs to be updated in the Confirmation Change Log entities in each applicable CCDB. The following guidelines should be followed:
Source Dictionary Identification Changes
If the dictionary name and/or system identifier of a source dictionary is changed, after updating
the source CCDB’s Dictionary Descriptor to reflect the change, ALTER CONFIRMATION commands
should run against each target CCDB into which entities from the source dictionary have been
promoted to reflect the new source dictionary identification information. This information is
contained in "Migrate In" (Action Code "V") Confirmation Change Log records in the target CCDB.

Target Dictionary Identification Changes
If the dictionary name and/or system identifier of a target dictionary is changed, after updating
the target CCDB’s Dictionary Descriptor to reflect the change, ALTER CONFIRMATION commands
should be run against each source CCDB from which entities have been promoted to reflect the
new target dictionary identification information. This information is contained in "Migrate Out"
(Action Code "C") Confirmation Change Log records in the source CCDB.

NDVRARCO Command Language
NDVRARCO is driven via three separate commands: SIGNON, COMPRESS, and ARCHIVE.

SIGNON
The syntax for SIGNON is as follows:

```
SIGNON DBName is dictname
SIGNON DictName
SIGNON USEr name is user-name
SIGNON PAssword is password
SIGNON CCId name is ccid
SIGNON ( ccid )
```

The SIGNON command identifies the user responsible for the archive/compress processing and
optional password and CCID list. If no CCID list is specified, the default CCIDs for the user are assigned
from the CCDB.
Note: If the user runs NDVRARCO but doesn't specify COMPRESS, ARCHIVE or ALTER CONFIRMATION, the utility will clean up uncommitted work only.

COMPRESS

In order to perform compress processing, the user must explicitly instruct NDVRARCO to do so. The syntax for COMPRESS is as follows:

```
►►────── COMpress ────────────────────────────────────────────────────────────►
►────┬───────────────────────────────────────────────────────┬───── - ───►◄
└─ TO ──┬─ DATE is ──┬─ TIME is ──│─ AGE is ──┘
          mm/dd/yy       hh:mm:ss     age
```

- The TO DATE value is used to specify the month, day, and year prior to which the compress processing is to be performed. This is followed by an optional TIME value, which defaults to 23:59:59 if not specified.
- The TO AGE value is used to specify the number of days prior to which compression is to be performed.

If neither TO DATE nor TO AGE is specified, the system defaults to an age of 1 (day). NDVRARCO will not touch any Change Log Entries in the database that are less than 24 hours old.

Note: You are limited to one COMPRESS command per NDVRARCO run. If you specify more than one COMPRESS command, only the last applies.

There is a specific set of rules that CA Endevor/DB applies during compress processing:

- **For A (Add), D (Delete), or M (Modify) actions** -- A series of consecutive Adds that have the same entity description by the same USER/CCID will compress to a single Add; the same holds true for consecutive series of Modifies or Deletes. An Add followed by a series of Modifies will compress to an Add. A series of Modifies followed by a Delete will compress to a Delete. Finally, when Adds and Deletes alternate, the net result will be a Delete if the last action was a Delete, and a Modify if the last action was an Add. For A/M/D action compression, the resultant database record is also modified to show the count of the original change log record, plus the date/time of the oldest compressed record in the series.

- **For C (migrate out) and V (migrate in) actions** -- These actions are involved in migration as records are migrated out (C) of the source dictionary and migrated into (V) the target dictionary. When compress processing is performed, the most recent C and V record for each entity, for each source/target dictionary are preserved. Thus, the level of the last migration from or to, any dictionary is always preserved.
• Note that since the last C and V actions relative to each source or target are critical to future migration, NDVRARCO considers these records to be inviolate and does not remove them from the database, even if they meet an ARCHIVE command criteria.

• For I (Signin) and O (Signout) actions -- the most recent I (signin) and O (signout) records are preserved for each entity during compression.

### ARCHIVE

In order to perform archive processing, the user must explicitly instruct NDVRARCO to do so. The syntax for ARCHIVE is as follows:

```plaintext
ARCHive

TO

DATE is mm/dd/yy

TIME is hh:mm:ss

AGE is age

USER

CCId

MANagement-group

is name

ALL

.```

• The TO DATE value is used to specify the month, day, and year prior to which archive processing is to begin. This is followed by an optional TIME value, which defaults to 23:59:59 if not specified.

• The TO AGE value is used to specify the number of days prior to which archive processing is to be performed.

The USER/CCID/MANAGEMENT-GROUP clause specifies the name of the group to be archived.

• When specifying a management group, the user can either key in the word MANAGEMENT-GROUP in its entirety or use the MGR abbreviation.

• Quotes are used only if the name contains embedded special characters such as commas or blanks. It is important to note that when archiving change history that hasn't been booked to any user or CCID, it is specified as " " (quote-blank-quote), blank meaning no user or no CCID.

• When you specify USER, CCID, or MANAGEMENT GROUP IS ALL, all Change Log Entries qualifying under the TO DATE/AGE range will be archived.

If neither TO DATE nor TO AGE is specified, the system defaults to an age of 1 (day). NDVRARCO will not touch any Change Log Entry in the database that is less than 24 hours old.

You can specify up to 43 ARCHIVE commands per NDVRARCO run.

### ALTER CONFIRMATION

In order to perform Confirmation Change Log record modification, the user must explicitly instruct NDVRARCO to do so. The syntax for ALTER CONFIRMATION is as follows:

```plaintext
ALTER

MODify

CONfirmation

change log change-log

entries entries
```

15-Jan-2018
The FROM DICTNAME value is used to specify the dictname, or dbname, and system identifier contained in a Confirmation Change Log record that you wish to alter.

The TO DICTNAME value is used to specify the dictname, or dbname, and system identifier that you want to be placed in the Confirmation Change Log record for each record that matches the value specified in the FROM DICTNAME parameter. Those Confirmation Change Log records matching the dictname/dbname and system identifier specified in the FROM DICTNAME clause will be modified to reflect the dictname/dbname and system identifier specified in the TO DICTNAME clause. Statistics identifying the number of Confirmation Change Log records modified for each ALTER CONFIRMATION command will be reported at the end of the job.

You can specify up to 100 ALTER CONFIRMATION commands per NDVRARCO run.

Running NDVRARCO

One NDVRARCO execution is required for each CCDB to be processed. These jobs are generally run during periods of low CV activity, such as immediately after startup or before shutdown. However, they may be efficiently run at any point during the day. Upon completion, a Change Log Entry that reflects the NDVRARCO run is associated with the Dictionary record in the CCDB with an action code of P. This record contains CLE purge statistics.

Archived CLEs are placed in the SYS020 file. These records can be used as input into the CA Endevor /DB reporting input module (See the section on Reporting in the CA Endevor/DB for CA IDMS Using section.). It is suggested that the SYS020 file be set up as a generation data group to allow easy access to detailed historical information.

NDVRARCO Output

NDVRARCO output provides the user with four types of information: a standard input command listing, followed by compiled input commands, running information, and summary statistics.

As a matter of course, NDVRARCO echoes the user’s input commands at the beginning of the output listing. Following the initial input command listing, NDVRARCO provides a series of informational messages that apply specifically to compiled input.
If a COMPRESS command was specified, NDVRARCO informs the user that compress processing will be performed, along with the date and time prior to which everything will be compressed.

If ARCHIVE commands were specified, NDVRARCO provides a message for each archive that is to be performed, along with the name of each program to be archived and a PRIOR TO date and time.

If ALTER CONFIRMATION commands were specified, NDVRARCO provides a message for each command indicating the number of Confirmation Change Log records that were altered.

The final message in this listing block refers to the aborted updates, which were removed from the database as NDVRARCO performed its routine cleanup of uncommitted work.

- **Note:** If the user specified a MANAGEMENT-GROUP name, NDVRARCO provides the names of associated CCIDs in the compiled command listing, not the management group.

Following the compiled input listing, NDVRARCO provides run-related information for job monitoring purposes. As indicated in the bolded area below, the first listings in this block indicate aborted update sessions. These are the incomplete jobs that are rolled out by the updating program (for example, IDD). One of these messages appear for each aborted session. The final information in this block serves as a "counter" for the passing of each 100 CLEs. These listings indicate the date/time stamp in that particular CLE, along with that CLE’s associated entity name. This is NDVRARCO’s way of letting administrators observe execution progress.

The final block of information provided in NDVRARCO output is the summary statistics listing.

- The first message in this block reflects the number of aborted update sessions removed from the database. This single message occurs with every NDVRARCO run.
The second message reflects the number of change-log records removed from the database by compression. This single message occurs only if compress processing was requested.

The third type of message is archive-related and will consequently appear only if archive processing was requested. Each archive request produces a list of summary statistics indicating the number of change-log records removed from the database, and the number of records written to an archive file.

The fourth type of message will appear only if ALTER CONFIRMATION processing was requested. Each ALTER CONFIRMATION request produces summary statistics indicating the number of Confirmation Change Log records altered in the CCDB for the request.

Sample JCL and Syntax

Use the following JCL to run NDVRARCO. It is contained in member SAMPARCO on the installation media JCL library:

Sample z/OS and OS/390 JCL

```plaintext
//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: SAMPARCO */ 
/* PURPOSE: ARCHIVE, COMPRESS AND PURGE OLD CHANGE LOG ENTRIES. */ 
/* */ 
/* STEP: FUNCTION: */ 
/* ===== ======== */ 
/* */ 
/* ARCHCOMP RUNS PROGRAM NDVRARCO. USERID MUST BE AUTHORIZED. */ 
/* */ 
****************************************************************************** 
/* ARCHCOMP EXEC PGM=NDVRARCO,REGION=1000K */ 
/* */ 
/* OUTPUT ARCHIVE FILE: COMPATIBLE WITH CULPRIT REPORTING. */ 
/* EXAMPLE ASSUMES YOU WILL USE GENERATION DATA GROUPS. */ 
/* */ 
//SYS020 DD DSN=user.xlsxccdb.archgdg (+1), 
//DISP=(,CATLG),UNIT=tape, 
//DCB=(user.gdgmodel,BLKSIZ=14400,LRECL=288,RECFM=FB) 
//SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRLST DD SYSOUT=* 
//NDVRERR DD SYSOUT=* 
//SYSDUMP DD DUMMY 
//SYSIDMS DD * 
//DMCL=dmcl-nameDICTNAME=dictionary-nameOther Optional SYSIDMS Parameters/* 
//NDVRIPT DD * 
/* SIGNON */ 
/* USER = youruserid PASSWORD = yourpswd DICTNAME = userdict. */ 
/* COMpress TO AGE nn. */ 
```
Promotion Support Facilities

In a typical promotion, entities are copied from a source data dictionary to a target data dictionary. CA Endevor/DB Facilities perform the following functions:

- Selection of entities to migrate based upon input selection criteria, IDD relationships, and Change Log information.

- Integrity checking the source system prior to migration for complete development activity (for example, dialogs generated when a subordinate process or map is changed, records updated when subordinate elements are changed, maps generated when work records are modified, subschemas recompiled when schemas are changed, etc.). A complete description of integrity checks is contained later in this section.

- Impact analysis prior to migration on the target dictionary by examining the IDD relationships and the CCDB history of changes on that system. During impact analysis, migrating entities and related entities in the target system are examined for update activity since they last migrated from the source system. Any update usually represents an inconsistency between the source and target systems.

- Migration of changed entities from the source to the target system using the CA Endevor/DB native mode migrator.

- Automatic Signout of migrating entities during impact analysis to prevent update until the migration process is completed.

- Building a standard IDD Class/Attribute structure in the source dictionary as an interface to other vendors' migration facilities at an installation, if desired.

- Audit trail creation on the source and target systems reflecting the origin and destination of migrated entities.

The Promotion Support architecture, in conjunction with the Dynamic Change Monitor and the CCDB, gives CA Endevor/DB the unique capability to assure accurate promotions while eliminating a majority of the time-consuming tasks normally associated with migration. CA Endevor/DB facilities can be used to perform all migration tasks or to interface with an existing migrator in use at an installation.

**Important!** CA Endevor/DB Promotion Support does not currently migrate CLASS/ATTRIBUTE or USER definitions. If you have entities with ATTRIBUTES or any of the ATTRIBUTE-related characteristics (USER-DEFINED-COMMENTS, USER-DEFINED-ENTITIES, simple ATTRIBUTES), then you must ensure that the CLASSES and ATTRIBUTES are defined at the target dictionary before running the target-dictionary portions of promotion. For instance, if a source-dictionary entity is defined with MYCLASS=MYATTRIBUTE, then that characteristic will be migrated. If MYCLASS and MYATTRIBUTE are not already defined at the target, then do so manually before running NDVRBOOK (OPTION=MIGRATE).
The promotion process begins by executing the **SELECTION/VERIFICATION PROCESSOR** (program NDVRDSEL). Entities are selected to be migrated from the information contained in the CCDB Change Log, input selection criteria, and the IDD. As a result, only those entities, which have been actually modified in the source system since the last migration to the target system, are initially eligible. Criteria, which are used to select entities for promotion, include the target system identifier, CCIDs, users, status, date and time ranges, management group, or a combination thereof.

Using the entities selected from the CCDB as a starting point, the data dictionary is interrogated and relationships expanded to determine if any closely related entities were modified since their last migration to the target system. Closely related entities are migrated if they have been modified.

Upon selection, a machine-readable and user-editable candidate entity list and control file is created. This list is then used as input to other promotion support utilities. Entities in the candidate list are also optionally signed-out at selection time by the Verification Processor. This assures that no unintended changes will be made to the entities selected until the promotion process is completed.
Following successful selection and verification of the candidate entity list on the source system, the **CORRELATION PROCESSOR** (program NDVRDCOR) performs a comprehensive impact analysis on the target system. As input, the Correlation Processor uses a sequential file containing a list of element names and control information produced by NDVRDSEL. Figure 8.2 illustrates the Correlation /Verification cycle.
Impact analysis is performed independently of the actual entity migration. Each entity in the candidate list is checked against the CCDB and the target IDD to ensure that no update activity has occurred to the entities migrating (or to closely related entities) since those entities were last promoted from the source system. In this manner, reversion of applied fixes or parallel development conflicts are captured prior to receiving unexpected results. Any discrepancies can be identified and repaired prior to migration. The process of executing Verification and Correlation may be safely and quickly iterated in the quality assurance phase of promotion.

Entity Migration

Source System

Change Control Database

Data Dictionary

DDL

Delivery

Select

Verify

Batch Compiler

Entity List

Correlate

Data Dictionary

Target System

Change Control Database

IDMSDB--Entity Migration
After quality assurance, the finalized entity list is used to migrate either source statements or executable modules (load module only migration) from the source system by the **DELIVERY PROCESSOR**. Optionally, a Class/Attribute structure can be built to serve as a mechanism for tagging those entities, which have been marked for promotion for use by another migration package.

When using another vendor’s migration package, migrate using the Class/Attribute path after the Delivery Processor creates it from the selection list. Figure 8.3 represents the Delivery process. Detailed information on the exact outputs produced is contained later in this section.
At this point in the process, all entities committed to promotion are updated in the target system via standard compilers running stand-alone or indirectly through a migrator. During migration, the Dynamic Change Monitor on the target system is run in a special migration mode to reflect that the data dictionary modifications are the result of migration. This is accomplished by passing a special command to the Change Monitor through program NDVRBOOK in a procedure described later in this section. Resultant migration Change Log Entries (Migrate-in CLEs) on the target system (CLE action code = V) are stamped with a "footprint" containing the exact date and time received, the source system identification, and the time the entity was selected on the source system. Figure 8.4 reflects this process. The Migrate-in CLEs can be created regardless of the migration procedure used at an installation.

Source System Audit Trail Creation

![Diagram of the source system audit trail creation process]

IDMSDB--Source System Audit Trail Creation

Figure 8.5 - NDVRDCF1 and NDVRDCF2 - Promotion Support Cycle Completion.
After a successful migration is committed and tested out on the target system, a confirmation file is extracted from the target system by program NDVRDCF1 and sent back to the source system via program NDVRDCF2. Migration Change Log Entries are created in the source system CCDB (CLE action code = C), reflecting the exact time the entries were received on the target system and the target system identification. At this point, a complete audit trail of the migration exists on the source and target systems.

Migration Change Log Entries are used by future Correlation and Verification executions as part of the on-going quality assurance/maintenance cycle. As an added feature, the Change Log Entries created by the Reception Processor are used for comprehensive migration activity reports. The Reception Processor is independent of the quality assurance processors and can be run stand-alone or in conjunction with any migration procedure or program.

Programs and Security for Migration

The logical processes depicted in Figure 8.5 are implemented with the following utility programs:

<table>
<thead>
<tr>
<th>Process</th>
<th>Program Name</th>
<th>Run on System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td>NDVRDSEL</td>
<td>Source</td>
</tr>
<tr>
<td>Validation (Source)</td>
<td>NDVRDSEL</td>
<td>Source</td>
</tr>
<tr>
<td>Impact Analysis (Target)</td>
<td>NDVRDCOR</td>
<td>Target</td>
</tr>
<tr>
<td>Exportation</td>
<td>NDVRDLVR</td>
<td>Source</td>
</tr>
<tr>
<td>Importation w/Audit Trail</td>
<td>NDVRBOOK</td>
<td>Target</td>
</tr>
<tr>
<td>Source Audit Trail (extract)</td>
<td>NDVRDCF1</td>
<td>Target</td>
</tr>
<tr>
<td>Source Audit Trail (update)</td>
<td>NDVRDCF2</td>
<td>Source</td>
</tr>
</tbody>
</table>

Any user executing the CA Endevor/DB promotion support programs will require the following procedures enabled in the Security Class (or use NDVR-GLOBAL):

NM-MODE = Y  
MIGRATE = Y

For instructions on modifying Security Class parameters and a full explanation of these options, See Section 4, "Security Class Maintenance."

System Identification

CA Endevor/DB creates audit trails and cross reference records to reflect migration activity in the CCDBs of the target and source systems. Each dictionary involved in migration as a SOURCE or TARGET is identified by SYSTEM NAME and a DBNAME in the dictionary descriptor record contained in the CCDB. The SYSTEM NAME is used to provide unique system identification when the source and target dictionaries share the same DICTNAME. This often happens when the dictionaries involved reside on separate CVs.
Set up SYSTEM NAMEs using the DICTIONARY FUNCTIONS online submenu. See Section1 in this document for full instructions on establishing a SYSTEM NAME.

When more than one DBNAME is used to point to the same physical data dictionary within a CV, CA Endevor/DB uses the DBNAME supplied in promotion support input command files for migration activity checking.

**Note:** A consistent DBNAME should be used for each physical dictionary for all promotion support activity. Failure to use a consistent name causes the selection and impact analysis routines to produce excessive warnings and selections. It is recommended that the base name be used.

Source System Validation and Integrity Checking Rules

**NDVRDSEL Selection and Verification**

The function of NDVRDSEL is to select and verify the entities to be migrated from the source to the target system. NDVRDSEL reads the Sign-on and input command syntax from the NDVRIPT file and creates a sequential NDVRENO (Entity Out) file. The NDVRENO file contains control information, which identifies the source and target system and a list of entity statements. This file is in display format and is suitable for editing in the event that subsequent NDVRDSEL or NDVRDCOR runs identify changes to be made to the entity list. Make these changes with any standard source code editor.

**Contents**

- NDVRDSEL Selection and Verification (see page 84)
- STRATEGY 1 - CHANGE RELATIONSHIPS (see page 85)
- STRATEGY 2 - HIERARCHY RELATIONSHIPS (see page 85)
- STRATEGY 3 - MINIMUM RELATIONSHIPS (see page 86)
- Strategy Comparison (see page 86)

During the validation process, all modified entities, which satisfy the selection criteria, are initially selected based on CCDB information. The IDD is then optionally interrogated to determine if any related entities have been modified since last migration to the specified target system.

Entities, which are examined in this process, fall into four general classes:

- Entities with CCDB change activity according to the selection criteria. These entities are migrated to the target system.
- Entities related to changed entities, which are unconditionally migrated with the changed entity.
• Entities related to changed entities which are migrated only if modified in the source system since its last migration to the target, even though the modification occurred outside of the CCDB selection criteria (another Management Group, CCID, userid, time range, or Status).

• Related entities, which should have been modified in the source system based on the modification of a subordinate entity, but were not. For example, if a map was modified but its related Dialog was not regenerated, that dialog would fall into this category. These entities are listed on an exception report for investigation.

Three different strategies can be employed to select IDD entities related to those that were logged in the CCDB. These strategies are specified as part of the input command file to program NDVRDSEL.

**STRATEGY 1 - CHANGE RELATIONSHIPS**

The EXPAND IDD CHANGE RELATIONSHIPS selection strategy is used when:

• All promoting development activity has taken place under CA Endevor/DB.

• Only changed entities are to be promoted.

It operates under the following assumptions:

• All development in the source dictionary has taken place under CA Endevor/DB.

• Only the changes undertaken since the Change Monitor was installed are to be promoted.

• The target dictionary contains the complete application, part of which will be replaced by the promotion process.

**STRATEGY 2 - HIERARCHY RELATIONSHIPS**

The EXPAND IDD HIERARCHY RELATIONSHIPS selection strategy is used when:

• All promoting development activity has *not* taken place under CA Endevor/DB.

• All related entities are to be promoted regardless of change activity.

• The application is not currently present in the target dictionary.

It is intended to operate under the following conditions:

• CA Endevor/DB was installed or interrupted in the middle of a development cycle.

• Major entities have change activity against them in the CCDB, but modified subordinate entities may not have had any change activity.
STRATEGY 3 - MINIMUM RELATIONSHIPS

The EXPAND IDD MINIMUM RELATIONSHIPS selection strategy is used when:

- All promoting development activity has taken place under CA Endevor/DB.
- Only changed entities are to be promoted.
- The expansion processing is to be limited. For example, if the NDVRDSEL selection criteria chooses a DIALOG which is used in several APPLICATIONs, then the EXPAND IDD MINIMUM RELATIONSHIPS will only inspect entities “lower” in the application definition, such as MAPs and PROCESSES, and will not include the APPLICATIONs because they are “higher” in the hierarchy.

It is intended to operate under the following condition:

- Only changes undertaken since the Change Monitor was installed are to be promoted.

Strategy Comparison

The following table identifies, by entity type, the selection of entities related to the promoting entity depending on the type of EXPAND IDD RELATIONSHIPS clause specified to NDVRDSEL (see syntax below).

It is important to note that the Promoting Entity is selected based on CCDB information (Change Log Entries and any selection criteria specified in the NDVRDSEL syntax), but during EXPAND IDD processing, related entities are selected based on their relationship to the promoting entity in the IDD.

Entities related to a selected entity, which should have been modified, but were not, may be listed on the EXCEPTION LISTING. For example, an APPLICATION selected for promotion is related to a RECORD, which is not selected, but the date last updated of the RECORD is more recent than the date last updated of the APPLICATION.

<table>
<thead>
<tr>
<th>Promoting Entity Type</th>
<th>Related Entity Type</th>
<th>EXPAND IDD RELATIONSHIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION</td>
<td>LOAD MODULE</td>
<td>Always</td>
</tr>
<tr>
<td></td>
<td>RECORD</td>
<td>Modified</td>
</tr>
<tr>
<td>DIALOG</td>
<td>LOAD MODULE</td>
<td>Always</td>
</tr>
<tr>
<td></td>
<td>MAP</td>
<td>Modified</td>
</tr>
<tr>
<td></td>
<td>PROCESS</td>
<td>Modified</td>
</tr>
<tr>
<td></td>
<td>RECORD</td>
<td>Modified</td>
</tr>
<tr>
<td></td>
<td>SUBSCHEMA</td>
<td>Modified</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>ELEMENT</td>
<td>Modified</td>
</tr>
<tr>
<td></td>
<td>RECORD</td>
<td>Modified</td>
</tr>
</tbody>
</table>

15-Jan-2018
<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Modified</th>
<th>Always</th>
<th>Never</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>FILE</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
</tr>
<tr>
<td>MAP</td>
<td>DIALOG</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
</tr>
<tr>
<td>LOAD MODULE</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>MODULE</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>RECORD</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>TABLE</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>MODULE MAP</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>PROGRAM</td>
<td>Always</td>
<td>Never</td>
<td>Never</td>
<td></td>
</tr>
<tr>
<td>PROCESS DIALOG</td>
<td>Modified</td>
<td>Never</td>
<td>Never</td>
<td></td>
</tr>
<tr>
<td>PROCESS</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>PROGRAM MAP</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>MODULE</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>RECORD</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>Modified</td>
<td>Never</td>
<td>Never</td>
<td></td>
</tr>
<tr>
<td>QFILE</td>
<td>QFILE</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
</tr>
<tr>
<td>RECORD</td>
<td>APPLICATION</td>
<td>Modified</td>
<td>Never</td>
<td>Never</td>
</tr>
<tr>
<td>DIALOG</td>
<td>Modified</td>
<td>Never</td>
<td>Never</td>
<td></td>
</tr>
<tr>
<td>ELEMENT</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>MAP</td>
<td>Modified</td>
<td>Never</td>
<td>Never</td>
<td></td>
</tr>
<tr>
<td>PROGRAM</td>
<td>Modified</td>
<td>Never</td>
<td>Never</td>
<td></td>
</tr>
<tr>
<td>SCHEMA</td>
<td>Modified</td>
<td>Never</td>
<td>Never</td>
<td></td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>Modified</td>
<td>Never</td>
<td>Never</td>
<td></td>
</tr>
<tr>
<td>SCHEMA</td>
<td>RECORD</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>SET</td>
<td>SCHEMA</td>
<td>Modified</td>
<td>Never</td>
<td>Never</td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>DIALOG</td>
<td>Modified</td>
<td>Never</td>
<td>Never</td>
</tr>
<tr>
<td>LOAD MODULE</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>PROGRAM</td>
<td>Modified</td>
<td>Never</td>
<td>Never</td>
<td></td>
</tr>
<tr>
<td>RECORD</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>SCHEMA</td>
<td>Modified</td>
<td>Never</td>
<td>Never</td>
<td></td>
</tr>
<tr>
<td>TABLE LOAD MODULE</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>MAP</td>
<td>Modified</td>
<td>Always</td>
<td>Modified</td>
<td></td>
</tr>
</tbody>
</table>

### Promoting Entity Type

<table>
<thead>
<tr>
<th>Promoting Entity Type</th>
<th>Related Entity Type</th>
<th>DATE/TIME EXCEPTION LISTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION</td>
<td>LOAD MODULE</td>
<td>APPL &gt; LOAD MODULE</td>
</tr>
<tr>
<td></td>
<td>RECORD</td>
<td>RECORD &gt; APPLICATION</td>
</tr>
<tr>
<td>Promoting Entity Type</td>
<td>Related Entity Type</td>
<td>DATE/TIME EXCEPTION LISTED</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>DIALOG</td>
<td>LOAD MODULE</td>
<td></td>
</tr>
<tr>
<td>MAP</td>
<td>MAP &gt; DIALOG</td>
<td></td>
</tr>
<tr>
<td>PROCESS</td>
<td>PROCESS &gt; DIALOG</td>
<td></td>
</tr>
<tr>
<td>RECORD</td>
<td>RECORD &gt; DIALOG</td>
<td></td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>SUBSCHEMA &gt; DIALOG</td>
<td></td>
</tr>
<tr>
<td>ELEMENT</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>RECORD</td>
<td>ELEMENT &gt; Record</td>
<td></td>
</tr>
<tr>
<td>FILE</td>
<td>FILE</td>
<td></td>
</tr>
<tr>
<td>MAP</td>
<td>DIALOG</td>
<td></td>
</tr>
<tr>
<td>LOAD MODULE</td>
<td>MAP &gt; LOAD MODULE</td>
<td></td>
</tr>
<tr>
<td>MODULE</td>
<td>MODULE &gt; MAP</td>
<td></td>
</tr>
<tr>
<td>RECORD</td>
<td>RECORD &gt; MAP</td>
<td></td>
</tr>
<tr>
<td>TABLE</td>
<td>TABLE &gt; MAP</td>
<td></td>
</tr>
<tr>
<td>MODULE</td>
<td>MAP</td>
<td></td>
</tr>
<tr>
<td>MODULE</td>
<td>MODULE &gt; MAP</td>
<td></td>
</tr>
<tr>
<td>MODULE</td>
<td>PROGRAM</td>
<td>MODULE &gt; PROGRAM</td>
</tr>
<tr>
<td>PROCESS</td>
<td>DIALOG</td>
<td>PROCESS &gt; DIALOG</td>
</tr>
<tr>
<td>PROCESS</td>
<td>PROCESS</td>
<td></td>
</tr>
<tr>
<td>PROGRAM</td>
<td>MAP</td>
<td>MAP &gt; PROGRAM</td>
</tr>
<tr>
<td>MODULE</td>
<td>MODULE &gt; PROGRAM</td>
<td></td>
</tr>
<tr>
<td>RECORD</td>
<td>RECORD &gt; PROGRAM</td>
<td></td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QFILE</td>
<td>QFILE</td>
<td></td>
</tr>
<tr>
<td>RECORD</td>
<td>APPLICATION</td>
<td>RECORD &gt; APPLICATION</td>
</tr>
<tr>
<td>DIALOG</td>
<td>RECORD &gt; DIALOG</td>
<td></td>
</tr>
<tr>
<td>ELEMENT</td>
<td>ELEMENT &gt; RECORD</td>
<td></td>
</tr>
<tr>
<td>MAP</td>
<td>RECORD &gt; MAP</td>
<td></td>
</tr>
<tr>
<td>PROGRAM</td>
<td>RECORD &gt; PROGRAM</td>
<td></td>
</tr>
<tr>
<td>SCHEMA</td>
<td>RECORD &gt; SCHEMA</td>
<td></td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>RECORD &gt; SUBSCHEMA</td>
<td></td>
</tr>
<tr>
<td>SCHEMA</td>
<td>RECORD &gt; SCHEMA</td>
<td></td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>SCHEMA &gt; SUBSCHEMA</td>
<td></td>
</tr>
<tr>
<td>SET</td>
<td>SCHEMA &gt; SCHEMA</td>
<td></td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>DIALOG</td>
<td>SUBSCHEMA &gt; DIALOG</td>
</tr>
<tr>
<td>LOAD MODULE</td>
<td>SSC &gt; LOAD MODULE</td>
<td></td>
</tr>
<tr>
<td>PROGRAM</td>
<td>SUBSCHEMA &gt; PROGRAM</td>
<td></td>
</tr>
</tbody>
</table>
Promoting Entity Type | Related Entity Type | DATE/TIME EXCEPTION LISTED
--- | --- | ---
RECORD | RECORD > SUBSCHEMA |  
SCHEMA | SCHEMA > SUBSCHEMA |  
TABLE | LOAD MODULE | TABLE > LOAD MODULE  
MAP | TABLE > MAP |  

Generally for new development projects there is little difference between the three strategies, since all entities have been added and are selected in any case.

When maintenance promotions are undertaken, Strategy 1 will usually yield a much smaller and more efficient promotion list since only the changes will be promoted.

When maintenance promotions are undertaken in the situation where multiple maintenance teams were working on overlapping sets of entities, Strategy 3 may prevent the work of all the maintenance teams from being gathered together, and allow independent promotion. Note, however, that this may lead to inconsistencies in the target dictionary. In the case where the work of two teams is co-dependent, you must promote them together.

### NDVRDSEL Command Syntax

**Contents**

- NDVRDSEL Sample JCL (see page 95)
  - Sample z/OS and OS/390 JCL (see page 96)

The following command syntax, specified in the NDVRRIPT file, is accepted by NDVRDSEL:

```plaintext
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGNon</td>
<td>DBName is dictname</td>
</tr>
<tr>
<td>USER name</td>
<td>is user-id</td>
</tr>
<tr>
<td>PASSword</td>
<td>is password</td>
</tr>
<tr>
<td>CCId name</td>
<td>is ccid</td>
</tr>
<tr>
<td>TARget</td>
<td>SYStem is target-system-name</td>
</tr>
<tr>
<td>NODEname</td>
<td>is target-node-name</td>
</tr>
<tr>
<td>MODE</td>
<td>is EXECute</td>
</tr>
<tr>
<td>INPut</td>
<td>is DATABASE</td>
</tr>
<tr>
<td>SIGNout to</td>
<td>USER name is signout name</td>
</tr>
<tr>
<td>CCId</td>
<td></td>
</tr>
</tbody>
</table>
```

---

15-Jan-2018 89/171
The SIGNON command identifies the user responsible for the migration and optional password and CCID list. If no CCID list is specified, the default CCIDs for the user are assigned from the CCDB.

The TARGET command identifies the target system and base dictionary name to be migrated to. This identifier is used to drive the verification process. All activity against an entity since it last migrated will be used for integrity checking for WARN conditions and for INCLUDE conditions (See below).

Note: The target system name and dictname specified in this statement must match the values in the Dictionary record in the CCDB for the target dictionary. If an erroneous target dictionary and system name are given, NDVRDSEL will select many more entities than are necessary to achieve source and target synchronization. This is because the software bases its selection on change activity and last migration date.

The last migration date is determined from the Migrate out CLEs created by NDVRDCF2 when the migration is confirmed on the source system. The Migrate out CLEs contain the date of migration and the target system name and dictname. Since there will be no Migrate out CLEs on the source system for a non-existent target, giving the wrong target name will erroneously trigger the selection of all entities modified on the source system.

The Verification and Selection program can be run in three modes. They are:

<table>
<thead>
<tr>
<th>Mode Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIAL</td>
<td></td>
</tr>
</tbody>
</table>

The VERIFICATION and SELECTION program can be run in three modes. They are:

- **SIGNON**
  The SIGNON command identifies the user responsible for the migration and optional password and CCID list. If no CCID list is specified, the default CCIDs for the user are assigned from the CCDB.

- **TARGET**
  The TARGET command identifies the target system and base dictionary name to be migrated to. This identifier is used to drive the verification process. All activity against an entity since it last migrated will be used for integrity checking for WARN conditions and for INCLUDE conditions (See below).

Note: The target system name and dictname specified in this statement must match the values in the Dictionary record in the CCDB for the target dictionary. If an erroneous target dictionary and system name are given, NDVRDSEL will select many more entities than are necessary to achieve source and target synchronization. This is because the software bases its selection on change activity and last migration date.

The last migration date is determined from the Migrate out CLEs created by NDVRDCF2 when the migration is confirmed on the source system. The Migrate out CLEs contain the date of migration and the target system name and dictname. Since there will be no Migrate out CLEs on the source system for a non-existent target, giving the wrong target name will erroneously trigger the selection of all entities modified on the source system.

- **MODE**
  The Verification and Selection program can be run in three modes. They are:

<table>
<thead>
<tr>
<th>Mode Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIAL</td>
<td></td>
</tr>
</tbody>
</table>

The VERIFICATION and SELECTION program can be run in three modes. They are:

- **SIGNON**
  The SIGNON command identifies the user responsible for the migration and optional password and CCID list. If no CCID list is specified, the default CCIDs for the user are assigned from the CCDB.

- **TARGET**
  The TARGET command identifies the target system and base dictionary name to be migrated to. This identifier is used to drive the verification process. All activity against an entity since it last migrated will be used for integrity checking for WARN conditions and for INCLUDE conditions (See below).

Note: The target system name and dictname specified in this statement must match the values in the Dictionary record in the CCDB for the target dictionary. If an erroneous target dictionary and system name are given, NDVRDSEL will select many more entities than are necessary to achieve source and target synchronization. This is because the software bases its selection on change activity and last migration date.

The last migration date is determined from the Migrate out CLEs created by NDVRDCF2 when the migration is confirmed on the source system. The Migrate out CLEs contain the date of migration and the target system name and dictname. Since there will be no Migrate out CLEs on the source system for a non-existent target, giving the wrong target name will erroneously trigger the selection of all entities modified on the source system.

- **MODE**
  The Verification and Selection program can be run in three modes. They are:

<table>
<thead>
<tr>
<th>Mode Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIAL</td>
<td></td>
</tr>
</tbody>
</table>

The VERIFICATION and SELECTION program can be run in three modes. They are:

- **SIGNON**
  The SIGNON command identifies the user responsible for the migration and optional password and CCID list. If no CCID list is specified, the default CCIDs for the user are assigned from the CCDB.

- **TARGET**
  The TARGET command identifies the target system and base dictionary name to be migrated to. This identifier is used to drive the verification process. All activity against an entity since it last migrated will be used for integrity checking for WARN conditions and for INCLUDE conditions (See below).

Note: The target system name and dictname specified in this statement must match the values in the Dictionary record in the CCDB for the target dictionary. If an erroneous target dictionary and system name are given, NDVRDSEL will select many more entities than are necessary to achieve source and target synchronization. This is because the software bases its selection on change activity and last migration date.

The last migration date is determined from the Migrate out CLEs created by NDVRDCF2 when the migration is confirmed on the source system. The Migrate out CLEs contain the date of migration and the target system name and dictname. Since there will be no Migrate out CLEs on the source system for a non-existent target, giving the wrong target name will erroneously trigger the selection of all entities modified on the source system.

- **MODE**
  The Verification and Selection program can be run in three modes. They are:

<table>
<thead>
<tr>
<th>Mode Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIAL</td>
<td></td>
</tr>
</tbody>
</table>

The VERIFICATION and SELECTION program can be run in three modes. They are:

- **SIGNON**
  The SIGNON command identifies the user responsible for the migration and optional password and CCID list. If no CCID list is specified, the default CCIDs for the user are assigned from the CCDB.

- **TARGET**
  The TARGET command identifies the target system and base dictionary name to be migrated to. This identifier is used to drive the verification process. All activity against an entity since it last migrated will be used for integrity checking for WARN conditions and for INCLUDE conditions (See below).

Note: The target system name and dictname specified in this statement must match the values in the Dictionary record in the CCDB for the target dictionary. If an erroneous target dictionary and system name are given, NDVRDSEL will select many more entities than are necessary to achieve source and target synchronization. This is because the software bases its selection on change activity and last migration date.

The last migration date is determined from the Migrate out CLEs created by NDVRDCF2 when the migration is confirmed on the source system. The Migrate out CLEs contain the date of migration and the target system name and dictname. Since there will be no Migrate out CLEs on the source system for a non-existent target, giving the wrong target name will erroneously trigger the selection of all entities modified on the source system.

- **MODE**
  The Verification and Selection program can be run in three modes. They are:

<table>
<thead>
<tr>
<th>Mode Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIAL</td>
<td></td>
</tr>
</tbody>
</table>

The VERIFICATION and SELECTION program can be run in three modes. They are:

- **SIGNON**
  The SIGNON command identifies the user responsible for the migration and optional password and CCID list. If no CCID list is specified, the default CCIDs for the user are assigned from the CCDB.

- **TARGET**
  The TARGET command identifies the target system and base dictionary name to be migrated to. This identifier is used to drive the verification process. All activity against an entity since it last migrated will be used for integrity checking for WARN conditions and for INCLUDE conditions (See below).

Note: The target system name and dictname specified in this statement must match the values in the Dictionary record in the CCDB for the target dictionary. If an erroneous target dictionary and system name are given, NDVRDSEL will select many more entities than are necessary to achieve source and target synchronization. This is because the software bases its selection on change activity and last migration date.

The last migration date is determined from the Migrate out CLEs created by NDVRDCF2 when the migration is confirmed on the source system. The Migrate out CLEs contain the date of migration and the target system name and dictname. Since there will be no Migrate out CLEs on the source system for a non-existent target, giving the wrong target name will erroneously trigger the selection of all entities modified on the source system.

- **MODE**
  The Verification and Selection program can be run in three modes. They are:

<table>
<thead>
<tr>
<th>Mode Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIAL</td>
<td></td>
</tr>
</tbody>
</table>
Mode Description

This mode produces the NDVRENO file but does not perform the bulk sign-out of entities, even if specified in the input command syntax. All other edits are performed. This mode can be used by individual project leaders or development teams independently of the person responsible for migration, to check the integrity of work performed on the target and source dictionaries without affecting the state of the CCDB.

**Note:** Users executing NDVRDSEL in TRIAL mode do not require security authorization.

**EXEC**
This mode performs all edits and selection and will sign-out the entities upon selection if SIGNOUT is specified in the input command file.

**UTE**

**Note:** Any user operating in this mode must be authorized for MIGRATE = Y in his/her Security Class.

**BACK**
This mode signs-in all elements that were signed-out during a NDVRDSEL run.

**OFF**

**Note:** Any user operating in this mode must be authorized for MIGRATE = Y in his/her Security Class.

- **INCLUDE**
  Entities may be selected for migration based on CCID, user, Status, Date, or Management Group (MGRP). MGRP, CCID, and user are mutually exclusive. Only one of these conditions should be specified in a single INCLUDE statement, and may be combined with Date and Status conditions. Conditions within an INCLUDE statement are logically ANDed with other conditions within that statement. Each INCLUDE statement is logically ORed with other INCLUDE statements. Specify one INCLUDE for each set of modifications to be migrated.

  To perform this function, NDVRDSEL reads Change Log Entries after the last Migrate-out CLE (action code = C) to the target system. If no Migrate-out CLE exists, it assumes the last migration to the target took place at the beginning of the recorded history for that entity.

  When MGRP is specified, all IDD work performed under the CCIDs in that Management Group is selected.

  When STATUS is given as the only selector within an INCLUDE statement, or when STATUS is given in combination with USER, only the base status (status set without a CCID context in the CCDB) is examined.

  When STATUS is given as a selector in combination with MGRP or CCID, the base status or the status within the context of the included CCIDs is considered.

  When ALL is specified, all updated entities since the last migration to the target are included. If the last activity against an entity is a migrate out (CLE action = C) to the target dictionary, it will not be included. All other entities in the CCDB are included.

- **EXCLUDE**
  Multiple EXCLUDE statements may be specified. Excluded from migration are those items which have been selected by an INCLUDE or through IDD expansion, but have a particular Status. This mechanism is employed as a way of excluding individual entities, even though all other criteria would have resulted in selection. For example, it can be used to accommodate last minute project decisions to migrate only part of the work performed under a CCID, or to exclude untested work.

**Note:** The IDD selection logic will not expand through (or validity edit) an entity that has been excluded. To prevent wholesale selection when a global record has changed in a compatible manner, EXCLUDE the global record from migration.
When the WITHIN clause is specified, the status under the context of the CCID specified is examined. When no WITHIN clause is present, only the base status is considered (status set without a CCID context in the CCDB).

**Note:** This is intentionally made more restrictive than the INCLUDE STATUS clause which will select a base or CCID status when specified.

- **SIGNOUT**
  All entities selected for migration will be signed out. Existing Signouts are overridden by NDVRDSEL. It is therefore possible to allow individual programmers the ability to Signout entities via Auto-Sign or explicit Signout to preserve the integrity of entities during development. When selected for migration, the user responsible for migration gains control of the entity regardless of prior signout status.
  When the migration is completed, the signout is restored to the developer. Signout prevents modification by individuals other than the person responsible for the migration once an entity is selected. All entities will be signed out to the user or the CCID specified in the command.
  Signouts remain in effect until the entities are received on the target system (through NDVRDCF2) or until MODE=BACKOFF is executed. After migration, Signouts are automatically signed back in.

- **EXPAND**
  NDVRDSEL determines the entities to be migrated through two passes:

  - **Pass 1** reads the CCDB and selects changed entities, according to specified selection criteria, which have been modified since their last migration to the target system. Alternatively, Pass 1 reads the input Entity List File (NDVRENI) and selects all entities named in the file.
  
  - **Pass 2** examines the IDD and optionally selects and validates entities related to those, which have changed.

  When the EXPAND option is not in effect (default setting), the CCDB (or input Entity List File) is the sole source for entities which satisfy the selection criteria. No Pass 2 is performed. No consistency validation on the source dictionary is performed. In this case, the selection list will consist only of the entities modified in the source system without regard to affected entities in the source IDD.

  When the EXPAND IDD RELATIONSHIPS clause is in effect, the Pass 2 selection and verification process will consistency-edit the IDD to determine if any potentially dangerous conditions exist in the source dictionary (e.g., Element changes with no corresponding Record modification). Entities, which violate validation conditions, are reported in the NDVRDSEL Exception Report. See the "Source System Validation and Integrity Checking Rules" section in this section for a complete description of the validation and selection rules.

  **Note:** If the CA Endevor/DB migrator is being used, always specify the EXPAND clause.

When EXPAND is in effect, NDVRDSEL will place additional entities (beyond those contained in the CCDB) into the NDVRENO file. The strategy used to select additional entities will depend upon the EXPAND option. CHANGE is the default strategy when the EXPAND clause is specified.
### Option Meaning

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| CHANGE | CHANGE examines all relevant IDD relationships during Pass 2 in the dictionary according to the relationships described earlier in this section under Strategy 1. It will then select for migration all dictionary entities related to the entities logged to the CCDB if they have changed since they last migrated to the target system and dictionary. Any new item selected will start a new selection path. Selection paths stop when an entity is reached along a path, which has not been modified in the source system.  
**Note:** The object of this technique is to migrate only the minimum set of entities required to bring the source and target into synchronization. |
| HIERARCHY | HIERARCHY examines only hierarchical IDD relationships during Pass 2 in the dictionary according to the relationships described earlier in this section under Strategy 2. It will then select for migration all dictionary entities that are subordinate to any one already chosen. Any new item selected for migration will later be used to search for other entities.  
**Note:** When HIERARCHY is specified, related entity types are unconditionally migrated regardless of change activity. Also note that the object of this technique is to migrate complete application systems, especially in situations where CA Endevor/DB is being used to migrate an application system for the first time. |
| MINIMUM | MINIMUM examines only hierarchical IDD relationships, and then select only those subordinate entities that have changed since they last migrated to the target system and dictionary. Any new item selected for migration will later be used to search for other entities.  
**Note:** The object of this technique is to allow the separate migration of application systems that share components. It should be used with great caution, because such application systems may be so thoroughly inter-dependent that they cannot be migrated independently. |

### WARN

All entities selected from the CCDB for migration can be optionally edited to produce warnings for each of the following exception conditions. Warnings appear on the NDVRDSEL Exception Report.

#### Warning Description

| CCID IS MULTIPLE | A modification under more than one CCID has occurred to this entity since it last migrated to the target system. To perform this function, NDVRDSEL reads Change Log Entries after the last Migrate-out CLE (action code = C) to the target system. If no Migrate-out CLE exists, it assumes the beginning of the recorded history for that entity. By examining the detail change history associated with entities flagged with this warning, it is possible to identify the users involved for follow-up questioning, if necessary. Multiple updates can be prevented at modification time by the use of the CA Endevor/DB Signout facility. |
| CCID IS NULL | A modification with no CCID has occurred against this entity since it last migrated to the target system. This warning will flag unclassified work. This condition can be prevented at modification time through the use of the NO-CCID Security Class Option on the dictionary Security Class. |
| USER IS MULTIPLE | More than one user has modified this entity since it last migrated to the target system. To perform this function, NDVRDSEL reads Change Log Entries after the last Migrate-out CLE (action code = C) to the target system. If no Migrate-out CLE exists, it assumes the beginning of the recorded history for that entity. This edit is intended as an aid to insure that multiple users were aware of and tested each other’s updates. By examining the detail change history associated with entities flagged with this warning, it is possible to identify the users involved for follow-up questioning, if necessary. Multiple updates can be prevented at modification time by the use of the CA Endevor/DB Signout facility. |
Warning Description

**USER IS NULL**
A modification with no userid has occurred against this entity since it last migrated to the target system. This warning will flag unclassified work after the fact. This condition can be prevented at modification time through the use of the NO-USER Security Class Option on the dictionary Security Class.

**INPUT**
Two primary input sources can be used for entity editing:

- **INPUT=DATABASE**
- **INPUT=FILE**

**DATABASE**: The CCDB is used with any INCLUDE/EXCLUDE statements to produce the NDVRENO file. The NDVRENO file contains a user-editable entity list and control information. From this point forward, the NDVRENO data set is used as input to other promotion support programs. DATABASE is used for the initial run of NDVRDSEL when change history is used to select entities for migration. After externally editing the entity list in the NDVRENO file (if necessary), subsequent runs for integrity edits would be done with INPUT=FILE and an NDVRENI file. Figure 8.6 depicts a run with INPUT=DATABASE.

**FILE**: A previously created NDVRENO file can be read in through the NDVRENI DD statement. When this occurs, all entities in the list are re-edited for WARN conditions, selection criteria, and validation rules. A new NDVRENO file is always produced reflecting the most recent execution time. Entities manually added to the list are signed out if SIGNOUT is specified in the syntax. INPUT=FILE is used when iterating the Verification and Correlation process to obtain a "safe" or "improved" entity list based on validation edits and impact analysis.

Another use of INPUT=FILE is when a known entity list is migrating. Manually coding ENT statements eliminates the overhead of scanning the CCDB (Pass 1) for all change activity. In this case, the advantages of cross entity validation and selection (Pass 2) is still obtained even though the CCDB was not used for initial list creation.
**Note:** When doing this, make sure that a LIST FOLLOWS statement immediately precedes your ENT statement.

INPUT=FILE can also be used in BACKOFF processing. If you have run NDVRDSEL in EXECUTE mode and had it perform SIGNOUT processing, the most expedient way to undo the SIGNOUT processing is to submit a job that specifies MODE=BACKOFF and INPUT=FILE, and use the previous job's NDVRENO file as NDVRENI.

When INPUT=FILE is specified, the following additional edits are performed:

- A warning is produced when no change history for an entity exists in the CCDB.
- A warning is produced when an entity in the NDVRENI file would have been EXCLUDEd or failed to be INCLUDEd based on optionally supplied INCLUDE/ EXCLUDE commands. If no commands are supplied in conjunction with NDVRENI, there will be no warnings of this type.

Figure 8.7 depicts a run with INPUT=FILE. The NDVRENI statement is required when INPUT=FILE is coded.

### NDVRDSEL Sample JCL

Use the following JCL to run NDVRDSEL. It is contained in member SAMPDSEL on the CA Endevor/DB installation media JCL library:
Sample z/OS and OS/390 JCL

```plaintext
// JOBNAME JOB YOUR.JOBCARD.INFORMATION//JOBLIB DD DISP=SHR,DSN=usercv.loadlib/
// DD DISP=SHR,DSN=ndvrd赡.mailib/ DD DISP=SHR,DSN=ids.loadlib/
/*
*/ JOB: SAMPDSEL
/*
*/ PURPOSE: SELECT IDD ENTITIES FOR MIGRATION.
/*
*/ STEP: FUNCTION:
/* ===== ========
*/ SELECT GENERATE ENTITY LIST TO DRIVE THE REST OF MIGRATION
/* OPTIONALLY, RE-EDIT PRIOR MODIFIED/RECYCLED LIST.
/*

NDVRDSEL Outputs

Contents
- NDVRDSEL Output File (ddname NDVRENO) (see page 97)
- NDVRDSEL Control Report (ddname NDVRLST) (see page 98)
- NDVRDSEL Input Command Listing (see page 99)
- NDVRDSEL Compiled Command Listing (see page 99)
- NDVRDSEL Entity List Exception Listing (see page 100)
- Report Fields (see page 100)
- NDVRDSEL End-of-Job Statistics (see page 101)
- NDVRDSEL Detail Report (ddname NDVRDTL) (see page 102)
NDVRDSEL Utility Report (ddname NDVRUTL) (see page 103)

NDVRDSEL produces three report files and one output file:

- A control report (ddname NDVRLST). This report echoes the user-specified syntax, and itemizes all entities that have had CCDB-related warnings or exceptions issued against them.

- A detail report (ddname NDVRDTL). This report itemizes each entity selected or excluded from the NDVRENO file, the reason for selection or exclusion, and any warning or exceptions that were generated.

- A utility report (NDVRUTL). This report lists the input Entity List file (NDVRENI). If IDD expansion is performed, it will also contain a Validation Exception listing.

- A sequential output file (ddname NDVRENO). This data set is the file, which controls all subsequent steps in the promotion process.

NDVRDSEL Output File (ddname NDVRENO)

The NDVRENO file contains 80 character records in display format. It is the hub of the CA Endevor/DB Promotion Support System and is used in all subsequent promotion support facilities for control, detail, and backoff information. *Italic* items are generated internally by NDVRDSEL and placed in the output file. Other items are passed into the output file from the input specifications. The NDVRENO file contains the following generated syntax:

- **SOURCE**  
  SYSTEM [NAME] [IS | =] source system name  
  {DBNAME | DICTNAME} [IS | =] dictname  
  VERIFY DATE = mm/dd/yy  
  TIME = hh:mm:ss

- **TARGET**  
  SYSTEM [NAME] [IS | =] target system name  
  {DBNAME | DICTNAME} [IS | =] dictname

- **MODE**  
  [IS | =] {TRIAL | EXECUTE | BACKOFF}

- **INPUT**  
  [IS | =] {FILE | DATABASE}

- **SIGNOUT**  
  [TO] {USER | CCID} [NAME] [IS | =] signout name

- **INCLUDE**  
  [FROM [DATE] [IS | =] mm/dd/yy ]  
  [THRU [DATE] [IS | =] mm/dd/yy ]  
  [ALL]  
  [(MGRP | CCID | USER} [NAME] [IS | =] select name | '']  
  [WHERE STATUS [NAME] [IS | =] status value | '']

- **EXCLUDE**  
  [WHERE] STATUS [NAME] [IS | =] status value  
  [WITHIN CCID [NAME] [IS | =] status value ]

- **EXPAND**  
  IDD {CHANGE} | HIERARCHY} [RELATIONSHIPS].

- **WARN**  
  [WHERE]  
  [CCID [IS | =] MULTIPLE]  
  [CCID [IS | =] NULL]  
  [USER [IS | =] MULTIPLE]  

- **LIST**  
  FOLLOWS .

- **ENT**  
  type entity name vvvv.
  
  ENT type entity name vvvv.

---

15-Jan-2018 97/171
One entity or ENT statement is created for each entity INCLUDEd. If that entity was also EXCLUDEd, the statement would be commented out with an "**" in column 1.

**Note:** The comment causes the Correlation Processor (NDVRDCOR) and the Delivery Processor (NDVRLDLVR) to ignore the statement. If you want to include any excluded entity, simply remove the leading "**". To manually include an entity, edit the file and add an ENT statement according to the free form syntax above.

When running the NDVRENO file back into NDVRDSEL as NDVRENI (with INPUT=FILE), the generated syntax preceding LIST FOLLOWS is ignored. The ENT statements are processed against the selection criteria and the CCDB. New ENT statements may be added or old ones deleted as required, through standard editors. Selection criteria are always taken from the NDVRIPT file and placed in the output file regardless of the INPUT option. The most recently generated syntax is therefore echoed by subsequent migration utilities for informational purposes.

If you construct an NDVRENI file manually, be sure to start with a LIST FOLLOWS command. The ENT statements are formatted as follows:

<table>
<thead>
<tr>
<th>Columns</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Comment Indicator (blank or *)</td>
</tr>
<tr>
<td>2-4</td>
<td>ENT literal</td>
</tr>
<tr>
<td>5</td>
<td>blank</td>
</tr>
<tr>
<td>6-21</td>
<td>Entity Type</td>
</tr>
<tr>
<td>22</td>
<td>blank</td>
</tr>
<tr>
<td>23</td>
<td>Single Quote (')</td>
</tr>
<tr>
<td>24-63</td>
<td>Full Entity Name (segmented as required with spaces)</td>
</tr>
<tr>
<td>64</td>
<td>Single Quote (')</td>
</tr>
<tr>
<td>65</td>
<td>blank</td>
</tr>
<tr>
<td>66-69</td>
<td>Entity Version Number (nnnn)</td>
</tr>
<tr>
<td>70</td>
<td>Period(.)</td>
</tr>
<tr>
<td>71-72</td>
<td>blanks</td>
</tr>
<tr>
<td>73-80</td>
<td>Sequence Number of statement within NDVRENO file.</td>
</tr>
</tbody>
</table>

**NDVRDSEL Control Report (ddname NDVRLST)**

Program NDVRDSEL produces a four-part NDVRLST control report:

- An input command listing.
- A compiled command listing.
An entity list exception report.

An End-of-Job statistics summary.

A description of each part and a detailed explanation follow.

**NDVRDSEL Input Command Listing**

The Input Command Listing simply reflects the record images the user specified in the NDVRIPT file.

```
volser DATE TIME PAGE
RELEASE nn.n CA - ENDEVOR
/ D B mm/dd/yy 08:40:07 00001
NDVRDSEL CONTROL REPORT MIGRATION SELECT
/VERIFY PROCESSOR INPUT COMMAND LISTING
SIGNON DBNAME SRCNDVR
USER EDB-SYSTEM-ADMINISTRATOR
CCID EDB-SYSADMIN
TARGET SYSTEM SYSTEM81 DBNAME TGTNDVR.
MODE = TRIAL. INPUT IS DATABASE.
SIGNOUT TO CCID EDB-SYSADMIN.
INCLUDE WHERE STATUS IS MIGRATE-TEST.
INCLUDE FROM DATE mm/dd/yy.
EXCLUDE WHERE STATUS IS NEVER-MIGRATE.
EXPAND IDD HIERARCHY RELATIONSHIPS.
```

**NDVRDSEL Compiled Command Listing**

The Compiled Command Listing displays the input command file as seen by the parser and command interpreter within NDVRDSEL. When the program encounters a Signon command, it echoes the CCIDs under which the user is executing in the source CCDB. If SIGNOUT is in effect, CLEs created to reflect the signout (type = O) will be cataloged under the CCID(s) listed.

It is recommended that a unique CCID be created and used for each migration. In this way, all the entities signed out as a result of the Promotion Support System are easily identified by displaying the Change Log Entries by CCID through the CA Endevor/DB Online facility.

All INCLUDE and EXCLUDE statements are assigned rule numbers that are used in the Detail and Utility reports. Rule numbers appear above each INCLUDE and EXCLUDE in the report. When an INCLUDE MGRP statement is encountered, a display of all the CCID names included in that Management Group will accompany the statement listing.

```
volser DATE TIME PAGE
RELEASE nn.n CA, INC.
/ D B mm/dd/yy 08:40:08 00002
NDVRDSEL CONTROL REPORT **** TRIAL **** MIGRATION SELECT
/VERIFY PROCESSOR COMPILED COMMAND LISTING
SOURCE SYSTEM = 'SYSTEM81' DBNAME = 'SRCNDVR' 00000001
VERIFY DATE = mm/dd/yy TIME = 08:40:08 00000002
USER = 'EDB-SYSTEM-ADMINISTRATOR' 00000003
CCID = ('EDB-SYSADMIN', ' ' ' ' ' 00000004
' ' ' ' ' ' ' ' ' ' 00000005
' ' ' ' ' ' ' ' ' ' 00000006
' ' ' ' ' ' ' ' ' ' 00000007
' ' ' ' ' ' ' ' ' ' 00000008
' ' ' ' ' ' ' ' ' ' 00000009

TARGET SYSTEM = 'SYSTEM81' DBNAME = 'TGTNDVR'
```

---

15-Jan-2018 99/171
NDVRDSEL Entity List Exception Listing

The Entity List Exception Listing contains a tabular list of migrating entities which violated WARN CCDB conditions or were specifically EXCLUDEd from the migration.

All entities INCLUDEd in a migration receive an ENT statement in the NDVRENO file (even if it is later EXCLUDEd). Exception conditions flagged in the exception listing report display the ENT command as it appears in the NDVRENO file with a summary of the error conditions to the right. Entities which were EXCLUDEd have an "*" in front of the ENT statement. The "*" causes the entity to be ignored by subsequent programs in the promotion process. If the user decides to INCLUDE a specific entity that had been previously EXCLUDEd, he/she merely removes the '*' in the NDVRENO file using a standard source editor.

<table>
<thead>
<tr>
<th>volser</th>
<th>DATE</th>
<th>TIME</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA, INC.</td>
<td>08:40:35</td>
<td>00003</td>
<td></td>
</tr>
</tbody>
</table>

NDVRDSEL CONTROL REPORT **** TRIAL **** MIGRATION SELECT /VERIFY PROCESSOR
ENTITY LIST EXCEPTION LISTING

<table>
<thead>
<tr>
<th>ENTITY TYPE</th>
<th>ENTITY NAME</th>
<th>VERS</th>
<th>NUMBER</th>
<th>RULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>*ENT RECORD</td>
<td>'EMPL-RECORD-1'</td>
<td>0001</td>
<td>00000232</td>
<td>00000001</td>
</tr>
</tbody>
</table>

Report Fields

Column headings in this report are used in other NDVRDSEL outputs. They are described below:

<table>
<thead>
<tr>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type of the entity INCLUDEd.</td>
</tr>
<tr>
<td>Name of the entity INCLUDEd.</td>
</tr>
</tbody>
</table>
### Field Description

**Entit**

**y**

**Nam**

**e**

**Vers** Version Number of the entity INCLUDED.

**Seq** Sequence number in the statement in the NDVRENO file. This will assist the user in editing the unc file if it becomes necessary.

**Num**

**ber**

**INCL** The first rule number that caused this entity to be INCLUDED in the migration. A rule number of Rule "IDDX" indicates an item in the report that originated from the IDD expansion processing. Items with an INCL rule of IDDX would not have been selected under any of the user-supplied selection criteria. If the Change Count is not zero on these items, they are related to migrating entities but include changes outside of the supplied selection criteria.

**EXCL** The first rule number that caused this entity to be EXCLUDED from the migration.

**Rule Note:** EXCLUDEs always supersede INCLUDEs. If an EXCLUDE rule applies to the entity, its corresponding ENT statement will be commented out with an "*" in column 1.

**Fou** N = Entity occurred on the NDVRENI file or in the CCDB but does not exist in the IDD. These entities generate E-level errors when an attempt is made to migrate them.

**IDD Note:** This report column is meaningful only if EXPAND is in effect.

**Fou** N = Entity occurred on the NDVRENI file or was found in IDD but does not exist in the CCDB. This is an informational exception. According to the CCDB, this entity has not been modified.

**CCD**

**B**

**Change Count** The number of times this entity has been modified since it last migrated to the target system.

**Mult Y =** A modification has been made to this entity under more than one CCID since it last migrated CCID to the target.

**Null Y =** A modification has been made to this entity with no CCID since it last migrated to the target.

**Mult Y =** A modification has been made to this entity under more than one userid since it last User migrated to the target.

**Null Y =** A modification has been made to this entity with no userid since it last migrated to the User target.

---

### NDVRDSEL End-of-Job Statistics

The End-of-Job Statistics report concludes each of the report files produced by NDVRDSEL. Contained within it are informational statistics relating to the work that was performed during execution.
NDVRDSEL Detail Report (ddname NDVRDTL)

The Detail Report contains a listing of all the ENT statements written to the NDVRENO file on the left-hand side, and a summary of rules and statistics on the right-hand side. Use this report as reference when editing the NDVRENO file (if necessary). A copy of the End-of-Job statistics is printed after the detail report for reference.
NDVRDSEL Utility Report (ddname NDVRUTL)

The Utility report itemizes all closely related entities, which were modified improperly (out of sequence). Entities in this list represent potential problem areas that require further investigation. The major expert validation rules used to derive this report are graphically depicted in the "Source System Validation and Integrity Checking Rules" section in this section.

Each VALIDATION EXCEPTION ENTITY and its RELATED ENTITY is displayed with the last date and time updated (if available). Last update dates are extracted from the dictionary record as opposed to the CCDB to allow for cases where no CCDB record exists, and to provide meaningful validation reports for new installations. In some cases, the last update time is not available in the dictionary record. When no time is available, only the date is used for purposes of validation. When two related entities are updated on the same date, and no update time is available from the dictionary, CA Endevor/DB assumes that the order of update was correct. In all cases, the entity in the left-hand column is the most recently updated. The related entity that was updated in improper sequence is in the right-hand column.

This Entity Type Is in the Right Column of This Report Because . . .

Record

An Element within this Record has changed and is being migrated as part of another Record. The Element has not been replaced in this Record on the source system. If applicable, modify the Record and rerun NDVRDSEL. If the Record is not modified in the source system, it will not be rebuilt on the target system and will remain unchanged.
<table>
<thead>
<tr>
<th>This Entity Type</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialog</td>
<td>A Record, Map, Process, or Subschema used in this Dialog has been modified but the Dialog has not been modified on the source system. Regenerate the Dialog on the source system, test the changes, and rerun NDVRDSEL. <strong>Important!</strong> If a Map is involved (in the left column), failure to regenerate this dialog causes an CA IDMS/DC date mismatch on the target system when this dialog is executed. If a Record or Process is involved, the changes will be migrated to the target but no generation of the Dialog will occur.</td>
</tr>
<tr>
<td>Map</td>
<td>A Record contained in this Map has been changed but the Map has not been regenerated on the source system. <strong>Important!</strong> If this condition is left unchecked, the Record included in this Map will not successfully import into the target system if this Map exists on the target system. Regenerate the Map on the source system, test the changes, and rerun NDVRDSEL.</td>
</tr>
<tr>
<td>Subschema</td>
<td>Either a Record within the Subschema, or the Schema containing this Subschema, has changed since the Subschema was last generated. <strong>Important!</strong> If this condition is left unchecked, the Record included in this Subschema will not successfully import into the target system if this Subschema exists on the target system. Regenerate the Subschema on the source system, test the changes, and rerun NDVRDSEL.</td>
</tr>
<tr>
<td>Schema</td>
<td>A Set or Record that appears in this Schema has been changed since the Schema was last validated. <strong>Important!</strong> If this condition is left unchecked, the Record included in this Schema will not successfully import into the target system if this Schema exists on the target system. Validate the Schema, regenerate all Subschema on the source system, test the changes, and rerun NDVRDSEL.</td>
</tr>
<tr>
<td>Program</td>
<td>A Map or Record used by this program has changed and is migrating, but the program has not been regenerated to reflect the change. <strong>Important!</strong> Failure to regenerate this program causes an CA IDMS/DC date mismatch on the target system when this program is executed. Regenerate the program, test the changes, and rerun NDVRDSEL.</td>
</tr>
<tr>
<td>Load Module</td>
<td>A Dialog, Subschema, Map, Application, or Table was updated, but no Load Module was generated to reflect the update. <strong>Important!</strong> If a SOURCE FORM migration is taking place, failure to generate a Load Module will result in exportation and regeneration of untested components. If an EXECUTABLE FORM migration is taking place, out-of-date Load Modules (no corresponding source code in the dictionary) will be exported. Regenerate the component, test the changes, and rerun NDVRDSEL.</td>
</tr>
</tbody>
</table>

**NDVRDCOR Correlation**

The Correlation processor reads the entity list file produced by NDVRDSEL (NDVRENO) as ddname NDVRENI for entity names and control information. The CCDB and IDD on the target system is examined for modifications made to the entities in the list after they were last migrated into the target system dictionary from the source system dictionary.

- Target System Impact Analysis Rules (see page 106)
- NDVRDCOR Command Syntax (see page 107)
Any modifications made to the entities on the list are reported as potential candidates for problems since corresponding changes may not have been made in the source system. The user is also warned when updates are made without a user or CCID, or when multiple user or CCID updates have been made to an entity on the target since last migration from the source dictionary. A warning is issued to an entity that has been migrated in (CLE Action = V) from a dictionary other than the source dictionary since its last migration from the source dictionary.

**Note:** If entities are routinely renamed when they are migrated to QA or Production, or if their version number is set to a value other than the source system, make sure to change those values in the ENT statements contained in the NDVRENI file before running the Correlation Processor (NDVRCOR).

NDVRCOR can be instructed to ignore a warning for an entity if its Status has been set to a particular value. Using this mechanism, it is possible to suppress warnings in those cases where the user has prior knowledge as to its state on the sending system. For example, this technique can be used to mark entities for which all updates made to the Production or QA system were also made on the source system.

**Note:** Since NDVRCOR is a read-only process, no special security is required to execute it.

The Correlation Processor can be run in conjunction with NDVRDSEL executed in TRIAL mode by unauthorized application developers to qualify entities for migration before involving the DA or DBA. This process can be run to identify potential regression problems arising from multiple update or from updates occurring independently in the QA or Production dictionaries that did not originate in the source system (i.e., emergency fixes, QA fixes). Development teams can be given the capability to suppress unnecessary warnings through the use of installation standard Status codes.
Target System Impact Analysis Rules

During execution of the Correlation Processor (program NDVRDCOR), the target system CCDB and IDD are examined to determine if any modifications have been made to the target dictionary that might conflict with the entity changes migrating in. The entities examined fall into three general categories:

- Entities contained in the migration list, which have been modified at the target since last being migrated.
- Entities, which are closely related to, those contained in the entity list, and which have changed since last migrated from the source.
- Entities, which are closely related to those, contained in the entity list, and whose presence will prevent the migration from succeeding (BUILDER CODE violations).

NDVRDCOR always checks for entities in the first category. It does so by inspecting the target CCDB. NDVRDCOR accepts an EXPAND IDD RELATIONSHIPS command. If specified, the program will check for entities in the last two categories. Entities in the first category are printed in the MIGRATION ENTITY EXCEPTIONS report produced by NDVRDCOR, entities in the second category are printed in the EXPANSION ENTITY EXCEPTIONS report, and those in the third category are printed in the TARGET ENTITY EXCEPTIONS report.

The following table identifies the relationships examined in the target system for impact analysis when the EXPAND IDD RELATIONSHIPS command is specified. For each Entity Type, the Related Entity will be Exception Listed if it has been modified in the target system since it was last migrated from the source system.

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Related Entity Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION</td>
<td>LOAD MODULE</td>
</tr>
<tr>
<td>DIALOG</td>
<td>LOAD MODULE</td>
</tr>
<tr>
<td></td>
<td>MAP</td>
</tr>
<tr>
<td></td>
<td>PROCESS</td>
</tr>
<tr>
<td></td>
<td>RECORD</td>
</tr>
<tr>
<td></td>
<td>SUBSCHEMA</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>RECORD</td>
</tr>
<tr>
<td>MAP</td>
<td>DIALOG</td>
</tr>
<tr>
<td></td>
<td>LOAD MODULE</td>
</tr>
<tr>
<td></td>
<td>MODULE</td>
</tr>
<tr>
<td></td>
<td>PROGRAM</td>
</tr>
<tr>
<td></td>
<td>RECORD</td>
</tr>
<tr>
<td></td>
<td>TABLE</td>
</tr>
<tr>
<td>MODULE</td>
<td>MAP</td>
</tr>
<tr>
<td></td>
<td>PROGRAM</td>
</tr>
</tbody>
</table>
### NDVRDCOR Command Syntax

The parameters to execute NDVRDCOR are contained in the front of the NDVRENI file as created automatically by NDVRDSEL. The following command syntax specified in the NDVRRIPT file is used to supplement the automatic parameters:

- **SIGNon**
  - `DBName` is `dictname`
  - `DICTName` is `dictname`

- **USER name** is `user-id`
  - `PASsword` is `password`

- **CCId name** is `ccid`
  - `( ccid )`

- **IGNORE**
  - where `STATus` name is `status`
  - `WITHin CCId` name is `ccid`

- **EXPand IDD relationships**
**SIGNON**
Identifies the user responsible for the migration and optional password and CCID list. If no CCID list is specified, the default CCIDs for the user are assigned from the CCDB.

**IGNORE**
Multiple IGNORE statements can be specified. Ignored from warnings are those items, which have a particular Status. This mechanism is employed as a way of excluding individual entities, even though all other criteria would have resulted in warning.

⚠️ **Note:** When the WITHIN clause is specified, the status under the context of the CCID name is examined. When no WITHIN clause is present, only the base status is considered. If the same base and CCID status values are to be ignored, two IGNORE statements are necessary. Multiple IGNORE statements are ORed.

**EXPAND**
NDVRDCOR performs its correlation processing in two passes:

- Pass 1 reads the input Entity List file and checks the target CCDB to see if any of those entities have been changed at the target since last migration.

- Pass 2 examines the IDD and checks entities related to those which will be migrating.

When the EXPAND option is not in effect (default setting), the input Entity List File is the sole source for entities to be checked. No Pass 2 is performed. In this case, no information will be provided about target changes to related entities and no information will be provided about "migration stopper" entities (See below).

When the EXPAND IDD RELATIONSHIPS clause is in effect, the Pass 2 expansion and checking will be performed. Target dictionary entities that are not migrating, but that are directly related to the migrating entities and that have changed since last migrated from the source (if ever), will be reported in the Expansion Entity Exceptions report. See the "Target System Impact Analysis Rules" section for a complete description of the correlation rules. Also, the Pass 2 processing will produce the Target Entity Exceptions report. This is a listing of all MAPs, SUBSCHEMAS and SCHEMAS that exist at the target dictionary, and include records coming in the migration but are not themselves coming in the migration. Because of the BUILDER CODE rules enforced by CA-IDD, these entities will prevent the import of the new record definitions. You must remove these "migration stopper" entities from the target dictionary before attempting to run the migration import job, or else add them to the set of entities being migrated.

⚠️ **Note:** If the CA Endevor/DB migrator is being used, always specify the EXPAND clause.

**NDVRDCOR Sample JCL**
Use the following JCL to run NDVRDCOR. It is contained in member SAMPDCOR on the CA Endevor/DB installation media JCL library:
Sample z/OS and OS/390 JCL

```apl
//JOBNAME JOB YOUR.UJOBCARD.INFORMATION JOBLIB DD DISP=SHR,DSN=usercv.loadlib/
// DD DISP=SHR,DSN=nndvrb.loadlib/
//******************************************************************************
// JOB: SAMPDCOR
// PURPOSE: CORRELATES SELECTED ENTITIES AGAINST TARGET CCDB/DICT
//******************************************************************************
// STEP: FUNCTION:
//******************************************************************************
// CORRELAT EXAMINES TARGET CCDB/DICT FOR ENTITIES ON SELECTION FILE
// FROM NDVRDSEL RUN. WARNS OF POSSIBLE REGRESSION ERRORS.
//******************************************************************************
//******************************************************************************
// NDVRDCOR Outputs
NDVRDCOR produces three report files:

- A control report (NDVRLST). This report echoes the expanded control syntax from the NDVRENI and NDVRRIPT files, and itemizes all entities from IDD expansion and NDVRENI that have had warnings or exceptions issued against them.

- A detail report (NDVRDTL). This report itemizes all entities examined in the impact analysis process - both the entities named in the NDVRENI file and those found during IDD expansion.

- A utility report (NDVRLUTL). This report lists the input Entity List file (NDVRENI). If IDD expansion is performed, it will also contain a "migration stopper" report showing all MAPs, SCHEMAs, and SUBSCHEMAs that exist at the target and will prevent the migration from succeeding.

When sending an NDVRENO file from NDVRDSEL to a remote location for impact analysis, it is not necessary to include a listing of that file. NDVRDCOR will echo the original selection syntax, as well as ENT statements originally generated.

Each report is now shown in detail.

NDVRDCOR Control Report (ddname NDVRLST)
The control report file is composed of five parts:
An input command listing.
A compiled command listing.
A migration exception report.
An IDD expansion exception report.
An End-of-Job statistics summary.

**NDVRDCOR - Input Command Listing**

The Input Command Listing echoes the input statements contained in the NDVRIPT file.

```plaintext
volser DATE TIME PAGE
RELEASE nn.n mm/dd/yy 07:14:48 00001
NDVRDCOR CONTROL REPORT MIGRATION TARGET CORRELATION
INPUT COMMAND LISTING
SIGNON DBNAME TGTNDVR
USER EDB-SYSTEM-ADMINISTRATOR
CCID EDB-SYSADMIN.
IGNORE WHERE STATUS 'NEVER-MIGRATE'.
```

**NDVRDCOR - Compiled Command Listing**

The Compiled Command Listing is a combination of statements contained on the NDVRENI file merged with additional statements from the NDVRIPT file. All statements preceding the INPUT clause are taken from the NDVRENI file. Since NDVRDCOR always takes its input from the NDVRENI file, the compiled command indicates that. The EXPAND IDD command appears only if specified in the input commands. IGNORE RULES only appear if specified in the input.

```plaintext
volser DATE TIME PAGE
RELEASE nn.n mm/dd/yy 07:14:52 00002
NDVRDCOR CONTROL REPORT MIGRATION TARGET CORRELATION
COMPILED COMMAND LISTING
SOURCE SYSTEM = 'SYSTEM81' DBNAME = 'SRCNDVR '
VERIFY DATE = mm/dd/yy TIME = 09:00:18
USER = 'EDB-SYSTEM-ADMINISTRATOR '
CCID =( 'EDB-SYSADMIN',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
       ' ',
�
TARGET SYSTEM = 'SYSTEM81' DBNAME = 'TGTNDVR '
MODE = EXECUTE
INPUT = FILE
************** IGNORE RULE NUMBER 0001
IGNORE WHERE STATUS = 'NEVER-MIGRATE '
LIST FOLLOWS .
```
NDVRDCOR - Migration Exception Report

The Migration Exception Report contains a listing of all the ENT statements from the NDVRENI file for migrating entities which have been modified in the target system since they were last migrated from the source system. If no migration activity is logged in the CCDB, the last migration is assumed to have occurred at the beginning of recorded history for the dictionary. This report will not appear if there were no exception entities.

Important! If the changes made from these modifications have not been reflected in the migrating entities, possible reversion of independently applied changes to the target system will occur.

Report Fields

The column headings reflected in this report are used for other NDVRDCOR reports. They are explained below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>Type of entity migrating in.</td>
</tr>
<tr>
<td>Entity Name</td>
<td>Name of migrating entity.</td>
</tr>
<tr>
<td>Vers</td>
<td>Version number of migrating entity.</td>
</tr>
<tr>
<td>IGNR Rule</td>
<td>On detail reports, the IGNORE rule applied to overriding the exception condition. This will not appear on the exception report.</td>
</tr>
</tbody>
</table>

N = Entity could not be found in the target system IDD.
Field | Description
---|---
Found IDD | N = Entity could not be found in the target system CCDB.

Change Count | Number of times this entity was modified in the target system since it was last migrated from the source system as specified in the NDVRENI file.

Mult CCID | Y = A modification has been made to this entity under more than one CCID since it last migrated to the target.

Null CCID | Y = A modification has been made to this entity with no CCID since it last migrated to the target.

Mult User | Y = A modification has been made to this entity under more than one userid since it last migrated to the target.

Null User | Y = A modification has been made to this entity with no userid since it last migrated to the target.

NDVRDCOR - Expansion Exception Report

This report itemizes those entities, which are closely related to the entities migrating in, and which have been modified in the target system since their last migration.

⚠️ Note: Each of the entities in this list may not be synchronized with their counterparts in the source system. Therefore, when the pending migration is completed, an untested combination of entities will result. Possible consequences are date/time mismatches between Dialogs, Programs and Maps, and/or other unpredictable results. The expert rules used to determine the participants in this report are graphically depicted in the "Impact Analysis Rules" section in this section.

NDVRDCOR - End-of-Job Statistics

The End-of-Job Statistics report concludes each of the report files produced by NDVRDCOR. Contained within it are informational statistics relating to the work that was performed during execution.
NDVRDCOR Detail Report (ddname NDVRDTL)

The detail report file is composed of three parts:

- A compiled command listing.
- A listing of all entities inspected during the processing.
- An End-of-Job statistics summary.

Compiled Command Listing The Compiled Command Listing in the NDVRDTL file is identical to the one in the NDVRLST file, and is not shown here.

Correlation Detail Listing The Correlation Detail Listing itemizes all entities examined on the target IDD and CCDB that were involved in the impact analysis. If IGNORE rules were applied to any exception conditions, they will be shown in this report.

<table>
<thead>
<tr>
<th>ENTITY TYPE</th>
<th>ENTITY NAME</th>
<th>VERS</th>
<th>RULE</th>
<th>IDD</th>
<th>CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECOR</td>
<td>'COVERAGE'</td>
<td>0100.</td>
<td>000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECOR</td>
<td>'EMPLOYEE'</td>
<td>0100.</td>
<td>000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECOR</td>
<td>'EMPMAP-WORK-RECORD'</td>
<td>0001.</td>
<td>000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAP</td>
<td>'EMPMAPP1'</td>
<td>0001.</td>
<td>000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCHE</td>
<td>'EMPSCHM'</td>
<td>0100.</td>
<td>000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCHE</td>
<td>'EMPSS01 EMPSCHM'</td>
<td>0100.</td>
<td>000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODU</td>
<td>'EMPMAPP1'</td>
<td>0001.</td>
<td>000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODU</td>
<td>'MAP-FIELD-HELP'</td>
<td>0100.</td>
<td>000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*ENT</td>
<td>'SELECTION-DATE'</td>
<td>0100.</td>
<td>0001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**END-OF-JOB STATISTICS**

<table>
<thead>
<tr>
<th>ENTITY TYPE</th>
<th>INPUT</th>
<th>Matcher</th>
<th>Changed</th>
<th>Not Found</th>
<th><strong>Signed</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD MODULE</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>MAP</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>MODULE</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RECORD</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

TOTAL: 28 | 0 | 0 | 23 | 5 | 0 | 0 | 23 | 0
NDVRDCOR - End-of-Job Statistics

The End-of-Job Statistics listing in the NDVRDTL file is identical to the one in the NDVRLST file, and is not shown here.

NDVRDCOR Utility Report (ddname NDVRUTL)

The utility report file is composed of three parts:

- A listing of the input NDVRENI file.
- A target entities exception report.
- An End-of-Job statistics summary.

NDVRDCOR - Input Entity List File

The Input Entity List File report echoes all the control information and ENT statements contained in the NDVRENI file.

```
volser     CA, INC. D
ATE       C A - E N D E V O R / D B
RELEASE/07:14:52 00001
NDVRDCOR UTILITY REPORT MIGRATION TARGET CORRELATION
INPUT ENTITY LIST FILE
SOURCE SYSTEM = 'SYSTEM81' DBNAME = 'SRCDVR '
VERIFY DATE = mm/dd/yy TIME = 09:00:18
USER = 'EDB-SYSTEM-ADMINISTRATOR'
CCID = ('EDB-SYSADMIN','
EXPAND IDD CHANGE RELATIONSHIPS
*********** INCLUDE RULE NUMBER 0001
INCLUDE ALL
WHERE STATUS = 'MIGRATE-TEST'
*********** INCLUDE RULE NUMBER 0002
INCLUDE ALL
FROM DATE = mm/dd/yy
*********** EXCLUDE RULE NUMBER 0001
EXCLUDE WHERE STATUS = 'NEVER-MIGRATE'
LIST FOLLOWS.
```

ENT RECORD 'COVERAGE' '0100. 0001
ENT RECORD 'EMPLOYEE' '0100. 00031
ENT RECORD 'EMPMAP-WORK-RECORD' '0001. 000032
NDVRDCOR - Target Entity Exceptions

The Target Entity Exceptions listing is only produced if EXPAND IDD processing has been requested, and if any "migration stopper" entities have been found.

<table>
<thead>
<tr>
<th>volser</th>
<th>DATE</th>
<th>TIME</th>
<th>PAGE</th>
<th>RELEASE nn.n</th>
<th>CA, INC.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mm/dd/yy</td>
<td>07:14:52</td>
<td>00001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NDVRDCOR UTILITY REPORT
MIGRATION TARGET CORRELATION

************ MIGRATION STOPPING ENTITY ************

MIGRATING ENTITY *************** VERS ***************

<table>
<thead>
<tr>
<th>TYP</th>
<th>NAME</th>
<th>VERS</th>
<th>TYP</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCH</td>
<td>EMPSCHM</td>
<td>0100</td>
<td>REC</td>
<td>EMPLOYEE</td>
</tr>
<tr>
<td>SUB</td>
<td>EMPSS01</td>
<td>0100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NDVRDCOR - End-of-Job Statistics

The End-of-Job Statistics listing in the NDVRUTL file is identical to the one in the NDVRLST file, and is not shown here.

NDVRDLVR Definition Delivery

- After NDVRDSEL is run (to select and validate the source system) and NDVRDCOR is run (to perform impact analysis on the target), NDVRDLVR performs one or both of the following functions:
  - Migrates full Source or Executable (Load Module only) definitions from the source dictionary for all items contained in the NDVRENI file created by NDVRDSEL.
  - Builds a Class/Attribute Structure in the source dictionary for all items selected by NDVRDSEL. When using an alternate vendor’s migrator, use the Class/Attribute migration path to export entities selected by CA Endevor/DB. The Class/Attribute structure is also useful as an audit trail when copying the entire dictionary to a system where CA Endevor/DB is not installed.

The NDVRDLVR program performs its function by first reading the NDVRIPT file with the command syntax described below. It then invokes as many of the following CA IDMS compilers as are necessary for the action being performed:

- ADSOBCOM
- ADSOBTAT
- IDMSDDDL
The DDDL syntax required to add the CLASS and ATTRIBUTE structure (if needed) and extract the source statements and/or load modules is fed to the compiler internally by NDVRDLVR.

Contents

- NDVRDLVR Command Syntax (see page 116)
- NDVRDLVR Sample JCL (see page 121)
  - Sample z/OS and OS/390 JCL (see page 121)
- NDVRDLVR Outputs (see page 123)
  - NDVRDLVR Output Files (see page 123)
  - File Requirements (see page 123)
  - NDVRDLVR Control Report (ddname NDVRLST) (see page 124)
- NDVRDLVR - Processing Summary (see page 125)

No Change Log Entries are created as a result of NDVRDLVR activity. This program has the capability to instruct the Change Monitor to refrain from logging. This is done to prevent entities in dictionaries, which are intermediate stops in the migration path from appearing as if they were modified since the last migration. Thus, NDVRDCOR will not falsely warn against modification overlay to entities, which were migrated in and migrated out without intervening updates except by NDVRDLVR. For this reason, any user executing this program will need to be authorized for NM-MODE=Y in the Security Class.

NDVRDLVR Command Syntax

```
SIGnon
  └─ DBNAme is dictname  ─┤
     │ Dictionary Name    │
    └─ DICtName ──────────┘

USEr name is user-id  ─┤
  └─ PASsword is password ─

CCId name is ccid  ─┤
  └─ ( ccid ) ─

IDMs
  └─ TARget ─ USer name is dc-user-name ─
    └─ SOURCE ─ PASsword is dc-password ─

TAG with CLAss class-name ATTrIBUTE attribute-name ─
```
Note: The option text referred to in the syntax above identifies option values that are not validated by CA Endevor/DB. When the option values are specified, CA Endevor/DB simply inserts them into the appropriate CA IDMS compiler command. Also, if a SET OPTIONS command is repeated, only the last one specified will be used.

- **SIGNON**
  The SIGNON command identifies the user responsible for the migration and optional password and CCID list. If no CCID list is specified, the default CCIDs for the user are assigned from the CCDB.

  Note: For this utility to operate, the user must be authorized in the Security Class for MIGRATE=Y and (optionally) NM-MODE=Y if TAG is selected.

- **IDMS USER**
  The IDMS command is used to specify the IDD USER and (optionally) PASSWORD information used to access the migration dictionaries. The IDMS USER statement and the SOURCE/TARGET USER statements are mutually exclusive: if you specify IDMS USER, you cannot specify either SOURCE or TARGET USER. If you specify either the SOURCE or TARGET USER command, you cannot specify the IDMS USER command. If you specify the IDMS USER command, the user and password on that command will be used for the dictionary processing at both the source and target dictionaries. If omitted, you may specify either SOURCE and/or TARGET USER commands. If you specify none of these, no CA IDMS user identification will be used at either the target or the source dictionaries.

- **SOURCE USER**
  The SOURCE command is used to specify the IDD USER and (optionally) PASSWORD information used to access the source dictionary. The SOURCE USER statement and the IDMS USER statements are mutually exclusive: if you specify SOURCE USER, you cannot specify IDMS USER. Likewise, if you specify the IDMS USER command, you cannot specify the SOURCE USER command. You may specify or omit the TARGET USER command if you specify SOURCE USER. The SOURCE USER command must be used when a source IDD user name is needed and either a different or no target IDD user name is needed.
TARGET USER
The TARGET command is used to specify the IDD USER and (optionally) PASSWORD information used to access the target dictionary. The TARGET USER statement and the IDMS USER statements are mutually exclusive: if you specify TARGET USER, you cannot specify IDMS USER. Likewise, if you specify the IDMS USER command, you cannot specify the TARGET USER command. You may specify or omit the SOURCE USER command if you specify TARGET USER. The TARGET USER command must be used when a target IDD user name is needed and either a different or no source IDD user name is needed.

TAG
The TAG command is used to instruct NDVRDLVR to build a CLASS and ATTRIBUTE structure to tag the entities contained in the NDVRENI file.

EXPORT
The EXPORT command causes NDVRDLVR to extract the entities contained in the NDVRENI file from the source dictionary. Two mutually exclusive options are available:

- EXECUTABLE FORM -- whenever possible, only LOAD MODULES for entities will be extracted from the dictionary. See the "NDVRDLVR Output Files" section for a description of generated output.
- SOURCE FORM -- whenever possible, source descriptions of the entities will be extracted for regeneration on the target system. See the "NDVRDLVR Output Files" section for a description of generated output.

MODE
The MODE command determines whether to add CLASS/ATTRIBUTE relationships or remove them. To create a CLASS/ATTRIBUTE relationship on each entity in the NDVRENI file list, run with MODE = EXECUTE. To reverse the effects of a prior run, run the same NDVRENI file back into NDVRDLVR with MODE = BACKOFF. Only the CLASS/ATTRIBUTE relationships are deleted by BACKOFF. If TAG has not been run, there is nothing to back-off.

SET SOURCE IDMSDDDL SESSION OPTIONS
This command allows changes to the default session options used by CA Endevor/DB for the source dictionary. CA Endevor/DB sets the source IDMSDDDL session options using the following DDDL command:

```
SET SESSION OPTIONS
QUOTE IS '
INPUT 1 THRU 72 OUTPUT 80
DISPLAY AS SYNTAX VERB REPLACE
WITH DETAILS MODULE SOURCE PICTURE OVERRIDES SYNONYMS
ATTRIBUTES ALL COMMENT TYPES.
```

These options may be overridden or added to by specifying the SET SOURCE IDMSDDDL SESSION OPTIONS command. For example, the command:

```
SET SOURCE IDMSDDDL SESSION OPTIONS REGISTRATION OVERRIDE.
```

causes a second DDDL SET SESSION OPTIONS command to be used, which turns off entity occurrence security. Refer to the CA IDMS IDD DDDL Reference section for a full discussion of DDDL session options.
**SET SOURCE IDMSDDDL RECORD OPTIONS**

This command allows changes to the default punch record options specified by CA Endevor/DB when punching record definitions from the source dictionary. CA Endevor/DB sets the following options by default when punching record definitions from the source dictionary.

\[
\text{PUNCH RECORD record-name VERSION version-nr ALSO WITH FILES ELEMENTS SUBORDINATE ELEMENTS.}
\]

Note that the 'ALSO WITH' indicates that these options are to be added to the source IDMSDDDL session options.

The punch record options can be overridden by specifying the SET SOURCE IDMSDDDL RECORD OPTIONS command. If you specify any options here, you must specify them all. The option text specified must begin with the 'ALSO WITH' or 'WITHOUT' or 'WITH' text. For example, the command:

\[
\text{SET SOURCE IDMSDDDL RECORD OPTIONS ALSO WITH OLQ HEADERS.}
\]

includes OLQ headers in the punched record output.

With this command, following punch record command is issued:

\[
\text{PUNCH RECORD record-name VERSION version-nr ALSO WITH OLQ HEADERS.}
\]

Refer to the CA IDMS IDD DDDL Reference section for a full discussion of DDDL PUNCH RECORD options.

**SET SOURCE IDMSDDDL ELEMENT OPTIONS**

This command allows changes to the default options specified by CA Endevor/DB when punching element definitions from the source dictionary. CA Endevor/DB sets the following options by default when punching element definitions from the source dictionary:

\[
\text{PUNCH ELEMENT element-name VERSION version-nr.}
\]

The element options can be overridden by specifying the SET SOURCE IDMSDDDL ELEMENT OPTIONS command. If you specify any options here, you must specify them all. The options text specified must begin with the 'ALSO WITH' or 'WITHOUT' or 'WITH' text. For example, the command:

\[
\text{SET SOURCE IDMSDDDL ELEMENT OPTIONS WITHOUT SYNONYMS.}
\]

excludes element synonyms from the punched element output.

With this command, the following punch element command is issued:

\[
\text{PUNCH ELEMENT element-name VERSION version-nr WITHOUT SYNONYMS.}
\]

Refer to the CA IDMS IDD DDDL Reference section for a full discussion of DDDL PUNCH ELEMENT options.

**SET TARGET IDMSDDDL SESSION OPTIONS**

This command allows changes to the default session options used by CA Endevor/DB for the target dictionary.

CA Endevor/DB sets the following target IDMSDDDL session options:

\[
\text{QUOTE IS ' DEFAULT IS ON INPUT 1 THRU 72 OUTPUT 80}
\]

These options can be overridden or added to by specifying the SET TARGET IDMSDDDL SESSION OPTIONS command. For example, the command:
SET TARGET IDMSDDL SESSION OPTIONS DECIMAL-POINT IS COMMA.

establishes the comma (,) as the default decimal-point character. 
Refer to the CA IDMS IDD DDDL Reference section for a full discussion of DDDL session options.

- **SET SOURCE SCHEMA OPTIONS**
  This command allows changes to the default options specified by CA Endevor/DB when punching schema definitions from the source dictionary. 
  CA Endevor/DB sets the following options by default when punching schema definitions from the source dictionary: 

  PUNCH SCHEMA schema-id VERSION version-nr.

  The punch schema options can be overridden by specifying the SET SOURCE SCHEMA OPTIONS command. If you specify any options here, you must specify them all. The options text specified must begin with 'ALSO WITH', 'WITHOUT', or 'WITH' text. For example, the command:

  SET SOURCE SCHEMA OPTIONS ALSO WITH HISTORY.

  includes all history associated with the schema. 
  With this command, the following punch schema command is issued:

  PUNCH SCHEMA schema-id VERSION version-nr ALSO WITH HISTORY.

  Refer to the CA IDMS Database Administering section for a full discussion of PUNCH SCHEMA options.

- **SET SOURCE SUBSCHEMA OPTIONS**
  This command allows changes to the default options specified by CA Endevor/DB when punching subschema definitions from the source dictionary. 
  CA Endevor/DB sets the following options by default when punching subschema definitions from the source dictionary: 

  PUNCH SUBSCHEMA subschema-id.

  The punch subschema options can be overridden by specifying the SET SOURCE SUBSCHEMA OPTIONS command. If you specify any options here, you must specify them all. The options text specified must begin with the 'ALSO WITH' or 'WITHOUT' or 'WITH' text. For example, the command:

  SET SOURCE SUBSCHEMA OPTIONS ALSO WITH HISTORY.

  includes the date and time the subschema was created or last modified. 
  With this command, the following punch subschema command is issued:

  PUNCH SUBSCHEMA subschema-id ALSO WITH HISTORY.

  Refer to the CA IDMS Database Administering section for a full discussion of PUNCH SUBSCHEMA options.

- **SET SOURCE RHDCMPUT OPTIONS**
  This command allows changes to the default RHDCMPUT process options used by CA Endevor/DB for the source dictionary. By default, the following process options are set:

  PROCESS=TERSE
This option can be overridden by specifying the SET SOURCE RHDCMPUT PROCESS OPTIONS command. For example, the command:

```
SET SOURCE RHDCMPUT PROCESS OPTIONS PROCESS=TERSE,DATETIME=YES.
```

produces the syntax to define a map in terse form with the date/time stamp preserved. Refer to the CA IDMS Reference section for a full discussion of RHDCMPUT PROCESS options.

**TARGET RHDCMPUT OPTIONS**

This command allows changes to the default RHDCMPUT process options used by CA Endevor/DB for the target dictionary. By default, the following process options are set:

```
PROCESS=LOAD
```

This option can be overridden by specifying the SET TARGET RHDCMPUT PROCESS OPTIONS command. For example, the command:

```
SET TARGET RHDCMPUT PROCESS OPTIONS PROCESS=LOAD,REPORT.
```

includes a report of all maps generated. Refer to the CA IDMS Reference section for a full discussion of RHDCMPUT process options.

**NDVRDLVR Sample JCL**

Use the following JCL to run NDVRDLVR. It is contained in member SAMPDLVR on the CA Endevor/DB installation media JCL library:

```z/os and os/390 jcl
//JOBNAME JOB YOUR.JOBCARD.INFORMATION//JOBLIB DD DISP=SHR,DSN=usercv.loadlib/
   DD DISP=SHR,DSN=ndvrdb.loadlib/
   //******************************************************************************
   //*** THE FOLLOWING FILES NEEDED FOR EXPORT PROCESSING (EITHER TYPE) ***
   //*** ADSA STATEMENTS *** (SMALL)
   //NDVRAVEN DD DS=DSN=NDVRDLVR,DISP=(CATLG,DELETE),
   //UNIT=disk,vol=SER=vols,space=(TRK,(1,5),RLSE),
   //DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
   //******************************************************************************
```
//NDVRDVAR DD DSN=user.ndvrdlvr.dsdvar,DISP=(,CATLG,DELETE),
  //UNIT=disk, VOL=SER=volser, SPACE=(TRK,(1,5),RLSE),
  //DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//*** ADD ENTITY STATEMENTS *** (LARGE)
//NDVRDUPD DD DSN=user.ndvrdlvr.dsdupd,DISP=(,CATLG,DELETE),
  //UNIT=disk, VOL=SER=volser, SPACE=(CYL,(5,5),RLSE),
  //DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//*** ADD LOAD MODULES STMTS *** (LARGE)
//NDVRDLDD DD DSN=user.ndvrdlvr.dsdld,DISP=(,CATLG,DELETE),
  //UNIT=disk, VOL=SER=volser, SPACE=(CYL,(5,5),RLSE),
  //DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)

//********************************************************************
//* THE FOLLOWING FILES NEEDED FOR EXPORT SOURCE PROCESSING ONLY *
//********************************************************************
//*** ADSOB.COM SYNTAX STMTS *** (SMALL)
//NDVRDGEN DD DSN=user.ndvrdlvr.dsdgen,DISP=(,CATLG,DELETE),
  //UNIT=disk, VOL=SER=volser, SPACE=(TRK,(1,5),RLSE),
  //DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//*** SCHEMA DELETE STMTS *** (SMALL)
//NDVRCDDEL DD DSN=user.ndvrdlvr.dscdel,DISP=(,CATLG,DELETE),
  //UNIT=disk, VOL=SER=volser, SPACE=(TRK,(1,5),RLSE),
  //DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//*** ADD SCHEMA STMTS *** (LARGE)
//NDVRCUPD DD DSN=user.ndvrdlvr.dscupd,DISP=(,CATLG,DELETE),
  //UNIT=disk, VOL=SER=volser, SPACE=(CYL,(5,5),RLSE),
  //DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//*** DELETE MAP STMTS *** (SMALL)
//NDVRMDEL DD DSN=user.ndvrdlvr.dsmdel,DISP=(,CATLG,DELETE),
  //UNIT=disk, VOL=SER=volser, SPACE=(TRK,(1,5),RLSE),
  //DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//*** GENERATE MAP STMTS *** (SMALL)
//NDVRMGEN DD DSN=user.ndvrdlvr.dsmgen,DISP=(,CATLG,DELETE),
  //UNIT=disk, VOL=SER=volser, SPACE=(TRK,(1,5),RLSE),
  //DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//*** ADD MAP SYNTAX STMTS *** (LARGE)
//NDVRMUPD DD DSN=user.ndvrdlvr.dsmupd,DISP=(,CATLG,DELETE),
  //UNIT=disk, VOL=SER=volser, SPACE=(CYL,(5,5),RLSE),
  //DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)

//********************************************************************
//* IF CA-E/DB - CA-E/05/390 BRIDGE NOT USED, SPECIFY THESE FILES *
//* WITH DD DUMMY *
//********************************************************************
//*** SOURCE TRANSFER STMTS *** (SMALL)
//NDVRSC1 DD DSN=user.ndvrdlvr.dsscl1,DISP=(,CATLG,DELETE),
  //UNIT=disk, VOL=SER=volser, SPACE=(TRK,(1,5),RLSE),
  //DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//*** TARGET TRANSFER STMTS *** (SMALL)
//NDVRSC2 DD DSN=user.ndvrdlvr.dsscl2,DISP=(,CATLG,DELETE),
  //UNIT=disk, VOL=SER=volser, SPACE=(TRK,(1,5),RLSE),
  //DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//NDVRLST DD SYSOUT=*  
//NDVRERR DD SYSOUT=*  
//SYSLST DD SYSOUT=*  
//SYSUDUMP DD DUMMY  
//SYSIDMS DD *  
DMCL=dmcl-nameDICTNAME=dictionary-nameOther Optional SYSIDMS Parameters/*
//NDVRIPT DD *
SIGNON
  USER = youruserid PASSWORD = yourpswd DICTNAME userid.
  IDMS USER = iduserid PASSWORD = idpswd.
  EXPORT SOURCE FORM.
  MODE = EXECUTE.

15-Jan-2018  122/171
NDVRDLVR Outputs

NDVRDLVR produces one control report (NDVRLST) and up to 12 output files. The control report echoes the user-specified syntax and itemizes all entities and control information read from the NDVRENI file. Each CA IDMS compiler executed under NDVRDLVR also produces an output listing.

NDVRDLVR Output Files

The output files produced by NDVRDLVR are a function of the input commands specified and the entities being migrated. When no output is needed for a required file, it is opened and closed to create a null data set.

File Requirements

This section summarizes the contents of required output files for the following input commands:

- **TAG**
- **EXPORT - SOURCE OR EXEC. FORMS**
- **EXPORT SOURCE FORM**
- **TAG**
  - There is no output file for the TAG command. All class/attribute work is done through internal compiler invocation.
- **EXPORT - SOURCE OR EXECUTE. FORMS**
  - The following table summarizes the contents of required output files for the EXPORT - SOURCE OR EXEC. FORMS command.

<table>
<thead>
<tr>
<th>Output (Compiler and Content)</th>
<th>Required DDNAME</th>
</tr>
</thead>
</table>
| ADSOBTAT statements to define APPLICATIONS | NDVRA
| DCMT Vary commands to optionally run through the UCF batch simulator on the target system to pick up LOAD MODULE updates. | NDVRDA
| IDMSDDDL statements to update entities on the target system. When EXECUTABLE FORM is specified, this file will contain QFILES and MESSAGES. When SOURCE FORM is specified, this file will contain QFILES, MESSAGES, ELEMENTS, RECORDS, PROCESSES, TABLES, MODULES, and PROGRAM DEFINITIONS. | NDVRDU
| IDMSDDDL statements to add LOAD MODULES. | NDVRDLO
| **EXPORT SOURCE FORM** | |
  - The following table summarizes the contents of required output files for the EXPORT SOURCE FORM command. |
<table>
<thead>
<tr>
<th>Output (Compiler and Content)</th>
<th>Required DDNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADSOBCOM generate statements to define DIALOGS.</td>
<td>NDVRDGEN</td>
</tr>
<tr>
<td>IDMSCHEM statements to delete SCHEMAS.</td>
<td>NDVRCDEL</td>
</tr>
<tr>
<td>IDMSCHEM statements to add SCHEMAS.</td>
<td>NDVRCUPD</td>
</tr>
<tr>
<td>IDMSUBSC statements to delete SUBSCHEMAS.</td>
<td>NDVRUDEL</td>
</tr>
<tr>
<td>IDMSUBSC statements to add SUBSCHEMAS.</td>
<td>NDVRUUDP</td>
</tr>
<tr>
<td>RHDCMAP1 statements to delete MAPS</td>
<td>NDVRMDEL</td>
</tr>
<tr>
<td>RHDCMPUT &quot;PROCESS=LOAD&quot; statements to load MAPS</td>
<td>NDVRMGEN</td>
</tr>
<tr>
<td>RHDCMAP1 statements to add</td>
<td>NDVRMUPD</td>
</tr>
</tbody>
</table>

**NDVRDLVR Control Report (ddname NDVRLST)**

NDVRDLVR produces a three-part control report as follows:

- An input command listing.
- An entity file listing.
- A processing summary.

An explanation of each report follows.

**NDVRDLVR - Input Command Listing**

The input command listing echoes the user-specified syntax in the NDVRIPT file.

```
volser  DATE       TIME        PAGE
       mm/dd/yy  hh:mm:ss   0000
RELEASE nn.n  CA, INC.

NDVRDLVR INPUT COMMAND LISTING  MIGRATION DELIVERY PROCESSOR
SIGNON DBNAME SRCNDVR
USER EDB-SYSTEM-ADMINISTRATOR
CCID EDB-SYSADMIN.
IDMS USER DBADMIN PASSWORD ???????.
SET SOURCE IDMSDDL RECORD OPTIONS
   WITH ALL WITHOUT ATTRIBUTES HISTORY.
SET TARGET IDMSDDL SESSION OPTIONS DELETE IS ON.
EXPORT SOURCE FORM.
```

**NDVRDLVR - Entity File Listing**

The Entity File Listing contains a display of the NDVRENI file as it was seen by NDVRDLVR. This is an informational report included for reference.

```
volser  DATE       TIME        PAGE
       mm/dd/yy  hh:mm:ss   0000
RELEASE nn.n  CA, INC.

NDVRDLVR ENTITY FILE LISTING  MIGRATION DELIVERY PROCESSOR
SOURCE SYSTEM = 'SYSTEM81'  DBNAME = 'SRCNDVR'  00000001
VERIFY DATE = mm/dd/yy  TIME = 09:00:18  00000002
USER = 'EDB-SYSTEM-ADMINISTRATOR'  00000003
CCID =('EDB-SYSADMIN', ' ', ' ', ' ', 00000004
   ' ', ' ', ' ', ' ', 00000005
   ' ', ' ', ' ', ' ', 00000006
```
NDVRDLVR - Processing Summary

The Processing Summary is composed of two sections:

- A phase-level return code summary showing the return codes that resulted from the execution of each of the required compilers. The phases of execution are as follows:
  
  - Phase 1 - IDMSDDDL
  - Phase 2 - IDMSCHEM
  - Phase 3 - IDMSUBSC
  - Phase 4 - RHDCMPUT
  - Phase 5 - IDMSDDDL (LOAD MODULES ONLY)
    In the example below, phase 5 was not required and was omitted from execution.

- An End-of-Job statistics summary showing the number and types of output records produced to each file.
NDVRBOOK in Migration Mode

After NDVRDLVR has been run and all entities have been punched from the Source Dictionary, NDVRBOOK will be used to migrate the entities into the Target Dictionary. When this type of processing is being performed, NDVRBOOK is being executed in migration mode and must be run under Batch/CV.

When running DDDL compilers or other utilities that invoke compilers to perform the migration on the target system, NDVRBOOK is run in a special migrate mode.

Note: In order to run NDVRBOOK in Migration Mode, the user must be associated with a Security Class with MIGRATE=Y and DE-CCID (Derived CCID) = N. Derived CCID processing is not valid for migration processing.

When running in migrate mode, NDVRBOOK will read the NDVRENI file for control information before invoking the necessary compiler or utility to be executed. The SOURCE SYSTEM/DICTIONARY and VERIFY DATE/TIME will be extracted from the NDVRENI file and passed to the Change Monitor.

When processing the NDVRENI file for a migration at the target system, NDVRBOOK updates each entity record in the target CCDB applicable to the compiler named in the PROGRAM IS program-name statement with the migration-id. If the VERIFY ALL clause is specified on the OPTION IS MIGRATE statement, each entity record in the target CCDB is updated with the migration-id regardless of the compiler named in the PROGRAM IS program-id statement. NDVRBOOK performs the migration-id updated for the first applicable entity on the NDVRENI file last and checks for this entity's being updated with the migration ID first, ensuring that duplicate and incomplete updates are avoided.
When the compilers ADD or MODIFY any entity while under migrate mode, a special Change Log Entry will be produced which contains an action code of V (Migrate-in), the SOURCE SYSTEM /DICTIONARY, and the VERIFY DATE/TIME extracted from the NDVRENI file. This record serves as an audit trail or “footprint” to register the entity’s source origin and selection time, as well as the time it was received. CLEs are created when the Change Monitor detects an update by the executing compiler or migrator. When an update is performed by an executing compiler in migrator mode, the Change Monitor compares the migration-id for the entity in the CCDB (which had been previously updated by NDVRBOOK) with the migration-id for this session. If they are equal, a V type CLE is created. If they are not equal, no CLE is created. Migrate-in CLEs can be reported on in summary fashion by NDVRPT17, the TARGET MIGRATION REPORT (See Section12 of the CA Endevor/DB for CA IDMS Using section).

The Migrate-in CLEs created by the Change Monitor running in migrate mode are an essential part of the Correlation and Verification integrity checking process. They also provide an essential audit trail that assists in the resolution of production problems.

NDVRBOOK Command Syntax

To run any compiler or utility in migrate mode, specify the following syntax in NDVRIPT:

- **SIGNON**
  The SIGNON command identifies the user responsible for the migration and optional password and CCID list. If no CCID list is specified, the default CCIDs for the user are assigned from the CCDB.

  ![Note: The user must be authorized for MIGRATE=Y in the Security Class for this utility to operate.]

- **PROGRAM**
  The PROGRAM command provides the name of the program or compiler name to execute.

- **OPTION**
  The OPTION command instructs the Change Monitor to go into migration mode. All ADD and MODIFY actions made by the program executed in this step will create a CLE with action code = V,
as well as descriptive information from the control file such as the SOURCE SYSTEM/DICTIONARY and the SOURCE DATE/TIME.

When MIGRATE is specified, an NDVREN1 file, generally created by NDVRDSEL, must be included in the JCL.

- **VERIFY ALL**
  When processing the NDVREN1 file, this option instructs NDVRBOOK to update each entity record in the target CCDB that corresponds to an ENT entity in the Entity List File with the migration ID, regardless of the compiler named in the `PROGRAM IS program-name` statement. If omitted (default), only those entity records in the target CCDB applicable to the compiler named in the `PROGRAM IS program-name` statement will be updated with the migration-id.

## NDVRBOOK Outputs

When running under NDVRBOOK with OPTION=MIGRATE, the NDVRLST file will consist of three parts:

- The input syntax from the NDVRIPT file.
- The control commands extracted from NDVREN1 used to create the Migrate-in CLEs.
- The entity statements from the NDVREN1 file. You will only see this third portion for the first step executed for any given migration.
Importing Entities Exported by NDVRDLVR

Contents

When importing entities extracted by NDVRDLVR, examine the NDVRDLVR PROCESSING SUMMARY report (NDVRlst), Migration Delivery File Totals to determine which of the following files should be processed. If the Migration Delivery File Total for a file (i.e., NDVRMDEL) is 0, then the corresponding NDVRBOOK job should not be executed. CA IDMS compilers must be executed in the order listed for those files whose Migration Delivery File Total is not 0. All CA IDMS compilers should be run through NDVRBOOK, except as noted below.

**Important!** Failure to do so will result in unpredictable results.

- Order of Compiler Execution (see page 129)
- NDVRBOOK Migration JCL (Source) (see page 130)
  - Sample z/OS and OS/390 JCL (see page 130)
- NDVRBOOK Migration JCL (Executable) (see page 134)
  - Sample z/OS and OS/390 JCL (see page 134)
- NDVRBOOK Generic Migration JCL (any program) (see page 135)
  - Sample z/OS and OS/390 JCL (see page 135)

**Order of Compiler Execution**

<table>
<thead>
<tr>
<th>Compiler Input File Taken From NDVRDLVR</th>
<th>Compiler Input File Taken From NDVRDLVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHDCMA NDVRMDEL - Delete Maps</td>
<td>RHDCMA NDVRMDEL - Delete Maps</td>
</tr>
<tr>
<td>P1</td>
<td>P1</td>
</tr>
<tr>
<td>IDMSUB NDVRUDEL - Delete Subschemas</td>
<td>IDMSUB NDVRUDEL - Delete Subschemas</td>
</tr>
<tr>
<td>SC</td>
<td>SC</td>
</tr>
<tr>
<td>IDMSCH NDVRCDEL - Delete Schemas</td>
<td>IDMSCH NDVRCDEL - Delete Schemas</td>
</tr>
<tr>
<td>EM</td>
<td>EM</td>
</tr>
<tr>
<td>IDMSDD NDVRDUPD - Add Entities_</td>
<td>IDMSDD NDVRDUPD - Add Entities_</td>
</tr>
<tr>
<td>DL</td>
<td>DL</td>
</tr>
<tr>
<td>IDMSCH NDVRCUPD - Add Schemas</td>
<td>IDMSCH NDVRCUPD - Add Schemas</td>
</tr>
<tr>
<td>EM</td>
<td>EM</td>
</tr>
<tr>
<td>NDVRUUPD - Add Subschemas</td>
<td>NDVRUUPD - Add Subschemas</td>
</tr>
</tbody>
</table>
### NDVRBOOK Migration JCL (Source)

The following sample step illustrates the use of NDVRBOOK to create Migrate-in CLEs for source entities migrated out with CA Endevor/DB (MIGRATE SOURCE FORM to NDVRDLVR). It is contained in member SAMPMIGS on the CA Endevor/DB installation media JCL library:

**Sample z/OS and OS/390 JCL**

```
//JOBNAME JOB YOUR.JOBCARD.INFORMATION//JOBLIB DD DISP=SHR,DSN=usercv.loadlib/
// DD DISP=SHR,DSN=ndvrdm.loadlib/
//*********************************************************
//** JOB: SAMPMIGS
//** PURPOSE: MIGRATE-IMPORT JOB FOR SOURCE-FORM STYLE MIGRATION.
//** STEP: FUNCTION:
//** ====== =========
//** MAP1MDEL DELETES MAPS TO BE MIGRATED. (allows DDDL CHANGES)
//** UBSCUDEL DELETES SUBSCHEMAS TO BE MIGRATED.
//** CHEMCDEL DELETES SCHEMAS TO BE MIGRATED.
//** DDDLDUPD DELETE/ADD/MOD/REPLACE IDMSDDDL ENTITIES MIGRATING.
//** CHEMCUPD ADDS SCHEMAS TO BE MIGRATED.
```
UBSCUUPD ADDS SUBSCHEMAS TO BE MIGRATED.

MAP1MUPD ADDS MAPS TO BE MIGRATED.

MPUTMGEN GENERATES MAPS TO BE MIGRATED.

DDDLDDLLOD UPDATES LOAD MODULES TO BE MIGRATED.

BGENDGEN GENERATES ADSO DIALOGS TO BE MIGRATED.

BTATAGEN RUNS ADSOBTAT TO UPDATE THE ADS/A APPLICATION TABLE.

UCFDGMT (OPTIONAL) USES ‘UCFBATCH’ TO DRIVE DCMT TO VARY LOAD MODULES / MAPS TO NEW COPY.

NOTE: YOU MUST GENERATE ‘UCFBATCH’ MODULE AND SET ITS PROGRAM NAME HERE AS GENERATED IN YOUR SHOP.

****************************************************************************

MAP1MDEL EXEC PGM=NDVRBOOK,REGION=2048K
/SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno/
/SYSIPT DD DISP=SHR,DSN=user.ndvrdlvr.dsmdel//NDVRLST DD SYSOUT=* 
/NDVRERRR DD SYSOUT=* 
/SYSLST DD SYSOUT=*  
/SYSPCH DD DUMMY 
/SYSPHC DD DUMMY  
/SYSDMDS DD * 
/DMCL=dmcl-nameDICTNAME=dictionary-nameOther Optional SYSIDMS Parameters/*
/NDVRIPT DD *
SIGNON 
/ USER = youruserid /PASSWORD = yourpswd /DICTNAME userdict./ 
/ PROGRAM = RHDCMAP1. 
/ OPTION = MIGRATE.  
/*

UBSCUDDEL EXEC PGM=NDVRBOOK,REGION=2048K
/SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno/
/SYSIPT DD DISP=SHR,DSN=user.ndvrdlvr.dsdudel//NDVRLST DD SYSOUT=* 
/NDVRERRR DD SYSOUT=* 
/SYSLST DD SYSOUT=*  
/SYSPCH DD DUMMY 
/SYSIDMS DD * 
/DMCL=dmcl-nameDICTNAME=dictionary-nameOther Optional SYSIDMS Parameters/*
/NDVRIPT DD *
SIGNON 
/ USER = youruserid /PASSWORD = yourpswd /DICTNAME userdict./ 
/ PROGRAM = IDMSUBSC. 
/ OPTION = MIGRATE.  
/*

CHEMCDL EXEC PGM=NDVRBOOK,REGION=2048K
/SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno/
/SYSIPT DD DISP=SHR,DSN=user.ndvrdlvr.dscdel//NDVRLST DD SYSOUT=* 
/NDVRERRR DD SYSOUT=* 
/SYSLST DD SYSOUT=*  
/SYSPCH DD DUMMY 
/SYSIDMS DD * 
/DMCL=dmcl-nameDICTNAME=dictionary-nameOther Optional SYSIDMS Parameters/*
/NDVRIPT DD *
SIGNON 
/ USER = youruserid /PASSWORD = yourpswd /DICTNAME userdict./ 
/ PROGRAM = IDMSCHEM. 
/ OPTION = MIGRATE.  
/*

DDDLUDP EXEC PGM=NDVRBOOK,REGION=2048K
/SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno/
/SYSIPT DD DISP=SHR,DSN=user.ndvrdlvr.dsdupd//NDVRLST DD SYSOUT=*
CA IDMS - 19.0

/NDVRERR DD SYSOUT=*  
/SYSLST DD SYSOUT=*  
/SYSUDUMP DD DUMMY  
/SYSPCH DD DUMMY  
/SYSIDMS DD *  
DMC=dmcl-name  
DICTNAME=dictionary-name  
Other Optional SYSIDMS Parameters/*
/NDVRIPT DD *  
SIGNON  
USER = youruserid  
PASSWORD = yourpswd  
DICTNAME userdict.
PROGRAM = IDMSDDDL.
OPTION = MIGRATE.
/*  
/*  
CHEMCUPD EXEC PGM=NDVRBOOK,REGION=2048K  
/SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno/  
/SYSIPT DD DISP=SHR,DSN=user.ndvrdlvr.dscudp//NDVRLST DD SYSOUT=*  
/NDVRERR DD SYSOUT=*  
/SYSLST DD SYSOUT=*  
/SYSUDUMP DD DUMMY  
/SYSPCH DD DUMMY  
/SYSIDMS DD *  
DMC=dmcl-name  
DICTNAME=dictionary-name  
Other Optional SYSIDMS Parameters/*
/NDVRIPT DD *  
SIGNON  
USER = youruserid  
PASSWORD = yourpswd  
DICTNAME userdict.
PROGRAM = IDMSCHEM.
OPTION = MIGRATE.
/*  
/*  
UBSCUUPD EXEC PGM=NDVRBOOK,REGION=2048K  
/SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno/  
/SYSIPT DD DISP=SHR,DSN=user.ndvrdlvr.dsuupd//NDVRLST DD SYSOUT=*  
/NDVRERR DD SYSOUT=*  
/SYSLST DD SYSOUT=*  
/SYSUDUMP DD DUMMY  
/SYSPCH DD DUMMY  
/SYSIDMS DD *  
DMC=dmcl-name  
DICTNAME=dictionary-name  
Other Optional SYSIDMS Parameters/*
/NDVRIPT DD *  
SIGNON  
USER = youruserid  
PASSWORD = yourpswd  
DICTNAME userdict.
PROGRAM = IDMSSUBSC.
OPTION = MIGRATE.
/*  
/*  
MAPIMUPD EXEC PGM=NDVRBOOK,REGION=2048K  
/SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno/  
/SYSIPT DD DISP=SHR,DSN=user.ndvrdsel.dsmupd//NDVRLST DD SYSOUT=*  
/NDVRERR DD SYSOUT=*  
/SYSLST DD SYSOUT=*  
/SYSUDUMP DD DUMMY  
/SYSPCH DD DUMMY  
/SYSIDMS DD *  
DMC=dmcl-name  
DICTNAME=dictionary-name  
Other Optional SYSIDMS Parameters/*
/NDVRIPT DD *  
SIGNON  
USER = youruserid  
PASSWORD = yourpswd  
DICTNAME userdict.
PROGRAM = RHDCMAP1.
OPTION = MIGRATE.
/*  
/*  
MPUTMGEN EXEC PGM=NDVRBOOK,REGION=2048K  
/SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno/  
/SYSIPT DD DISP=SHR,DSN=user.ndvrdlvr.dsmgen//NDVRLST DD SYSOUT=*  
/NDVRERR DD SYSOUT=*  
/SYSLST DD SYSOUT=*  
/SYSUDUMP DD DUMMY  
/SYSPCH DD DUMMY  
/SYSIDMS DD *  
DMC=dmcl-name  
DICTNAME=dictionary-name  
Other Optional SYSIDMS Parameters/*
/NDVRIPT DD *
SIGNON

USER = youruserid  PASSWORD = yourpswd  DICTNAME userdict.

PROGRAM = RHDCMPUT.
OPTION = MIGRATE.

/*

//DDDDLLOD EXEC PGM=NDVBOOK,REGION=2048K
//SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno//
//SYSIPT DD DISP=SHR,DSN=user.ndvlvr.dsdlod//NDVRLST DD SYSOUT=* 
//NDVRRERR DD SYSOUT=* 
//SYSLIST DD SYSOUT=* 
//SYSUDUMP DD DUMMY 
//SYSPI CH DD DUMMY 
//SYSIDMS DD *

DMCL=dmcl-nameDICTNAME=di ctionary-nameOther Optional SYSIDMS Parameters/*
//NDVRPI RT DD *

SIGNON

USER = youruserid  PASSWORD = yourpswd  DICTNAME userdict.

PROGRAM = IDMSDDDL.
OPTION = MIGRATE.

/*

//BGENDGEN EXEC PGM=NDVBOOK,REGION=900K
//CDMSLIB DD DISP=SHR,DSN=usercv.loadlib// DD DISP=SHR,DSN=ndvrd.loadlib//
//SYSCTL DD DISP=SHR,DSN=ca.caiclid//SYSCL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno//SY SIPT DD DISP=SHR,DSN=user.ndvlvr.dsdlod//NDVRLST DD SYSOUT=* 
//NDVRRERR DD SYSOUT=* 
//SYSLIST DD SYSOUT=* 
//SYSUDUMP DD DUMMY 
//SYSPI CH DD DUMMY 
//SYSIDMS DD *

DMCL=dmcl-nameDICTNAME=di ctionary-nameOther Optional SYSIDMS Parameters/*
//NDVRPI RT DD *

SIGNON

USER = youruserid  PASSWORD = yourpswd  DICTNAME userdict.

PROGRAM = ADSOBCOM.
OPTION = MIGRATE.

/*

//BTATAGEN EXEC PGM=NDVBOOK,REGION=500K
//SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno//
//SYSIPT DD DISP=SHR,DSN=user.ndvlvr.dsdlod//NDVRLST DD SYSOUT=* 
//NDVRRERR DD SYSOUT=* 
//SYSLIST DD SYSOUT=* 
//SYSUDUMP DD DUMMY 
//SYSPI CH DD DUMMY 
//SYSIDMS DD *

DMCL=dmcl-nameDICTNAME=di ctionary-nameOther Optional SYSIDMS Parameters/*
//NDVRPI RT DD *

SIGNON

USER = youruserid  PASSWORD = yourpswd  DICTNAME userdict.

PROGRAM = ADSDOBAT.
OPTION = MIGRATE.

/*

// 'UCFBATCH' MAY NOT BE THE PROGRAM-ID GENERATED IN YOUR SHOP.
// STEP IS OPTIONAL

/*

//UCFDCMT EXEC PGM=UCFBATCH,REGION=500K
//SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno//
//SYSIPT DD DISP=SHR,DSN=user.ndvlvr.dsdlod//NDVRLST DD SYSOUT=* 
//SYSUDUMP DD DUMMY 
//SYSPI CH DD DUMMY 
//SYSIDMS DD *

DMCL=dmcl-nameDICTNAME=di ctionary-nameOther Optional SYSIDMS Parameters/*
NDVRBOOK Migration JCL (Executable)

The following sample job illustrates the use of NDVRBOOK to create Migrate-in CLEs for executable entities migrated out with CA Endevor/DB (MIGRATE EXECUTABLE FORM to NDVRDLVR). It is contained in member SAMPMIGE on the CA Endevor/DB installation media JCL library:

Sample z/OS and OS/390 JCL

```plaintext
//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: SAMPMIGE
* PURPOSE: MIGRATE-IMPORT JOB FOR EXECUTABLE-ONLY STYLE MIGRATION.
* STEP: FUNCTION:
* ===== ===========
* DDRDLLOD IMPORTS LOAD MODULES FROM DELIVERY OUTPUT FILE
* BTATGEN RUNS ADSOBTAT TO UPDATE THE ADS/A APPLICATION TABLE.
* UCFDCMT (OPTIONAL) USES 'UCFBATCH' TO DRIVE DCMT TO VARY
* LOAD MODULES / MAPS TO NEW COPY.
* NOTE: YOU MUST GENERATE 'UCFBATCH' MODULE AND SET ITS
* PROGRAM NAME HERE AS GENERATED IN YOUR SHOP.
* */
********************************************************************************/
*-DDRDLLOD EXEC PGM=NDVRBOOK,REGION=1000K
*/SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno/
 NDVRLST DD SYSOUT=* 
 NDVRRERR DD SYSOUT=* 
 SYSLST DD SYSOUT=* 
 SYSUDUMP DD DUMMY 
 SYSIDMS DD * 
 DMCL=dmcl-nameDICTNAME=dictionary-name
Other Optional SYSIDMS Parameters/*
*/NDVRIPT DD *
SIGNON
 USER = youruserid PASSWORD = yourpswd DICTNAME userdict.
 PROGRAM = IDMSDDDL.
 OPTION = MIGRATE .
/*
*/SYSIPT DD DSN=user.ndvrdlvrs.dsdload,DISP=SHR
/*
*/BTATGEN EXEC PGM=NDVRBOOK,REGION=1000K
*/SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdsel.dseno/
 SYSIPT DD DISP=SHR,DSN=user.ndvrdlvrs.dsdagen//NDVRLST DD SYSOUT=* 
 NDVRRERR DD SYSOUT=* 
 SYSLST DD SYSOUT=* 
 SYSUDUMP DD DUMMY 
 SYSIDMS DD * 
 DMCL=dmcl-nameDICTNAME=dictionary-name
Other Optional SYSIDMS Parameters/*
*/NDVRIPT DD *
SIGNON
 USER = youruserid PASSWORD = yourpswd DICTNAME userdict.
 PROGRAM = ADSOBTAT.
 OPTION = MIGRATE .
/*
* 'UCFBATCH' MAY NOT BE THE PROGRAM-ID GENERATED IN YOUR SHOP.
* STEP IS OPTIONAL
*/
```
NDVRBOOK Generic Migration JCL (any program)

The following sample JCL illustrates the use of NDVRBOOK to create Migrate-in CLEs for any program executing on the target system. If you are using another vendor’s migrator, simply run the import step with the program name of the migrate in utility in the PROGRAM statement. Remember to include any DD statements required by the migration utility statement in this step as well:

Sample z/OS and OS/390 JCL

```jcl
// JOBNAME JOB YOUR.JOBCARD_INFORMATION//JOBLIB DD DISP=SHR,DSN=usercv.loadlib// DD DISP=SHR,DSN=ndvrb.loadlib// DD DISP=SHR,DSN=idms.loadlib// ********** **************** 10 *// ** JOB: SAMPBOOK ** ** PURPOSE: RUN ANY CA COMPILER WITH CA-ENDEVOR/DB USER/CCID SIGNON. ** ** ** STEP: FUNCTION: ** ===== =========== ** ** BOOKDDDL RUNS IDMSDDL UNDER CA-ENDEVOR/DB BOOK-END. ** ** (CHANGE PROGRAM SENTENCE TO RUN OTHER COMPILERS) ** ** ****** 10 ** //NDVRBOOK EXEC PGM=NDVRBOOK,REGION=1300K //SYSCTL DD DISP=SHR,DSN=idms.sysctl//NDVRLST DD SYSOUT=* //NDVRERR DD SYSOUT=* //SYSLST DD SYSOUT=* //SYSPCH DD DUMMY //SYSSIDM DD DUMMY //SYSPCH DD DUMMY //SYSSIDM DD * //DMCL=dmcl-nameDICTNAME=dictionary-nameOther Optional SYSIDMS Parameters/* //NDVRIPT DD * SIGNON USER = youruserid PASSWORD = yourpswd DICTNAME userdict. PROGRAM = IDMSDDL. /* //SYSIPT DD * SIGNON USER = idduserid PASSWORD = iddpswd DICTNAME userdict. **+ PUT YOUR IDMSDDL STATEMENTS HERE. **+ /*
```

NDVRDCF1 Target Confirmation

After a migration is executed under NDVRBOOK with OPTION=MIGRATE, Change Log Entries exist in the target CCDB that reflect the event. To create corresponding CLEs on the source system to reflect the fact that an entity was selected for migration and actually received on the target, a two-step procedure is invoked.
Contents

- NDVRDCF1 Command Syntax (see page 136)
- NDVRDCF1 Sample JCL (see page 137)
  - Sample z/OS and OS/390 JCL (see page 137)

Step 1 is the execution of NDVRDCF1 on the target system to extract an entity list. To perform this task, the NDVRENI file created by NDVRDSEL and previously consumed by NDVRBOOK is again consumed by NDVRDCF1. This is done to obtain the necessary control information to identify the appropriate CLEs involved. It is accomplished on the basis of the SOURCE SYSTEM/DICTIONARY and VERIFY DATE/TIME.

During Step 2, a new file is created DDNAME NDVRENO) which contains a confirmation file to be sent to NDVRDCF2 on the source system. The confirmation file will contain the control information from the NDVRENI file and a generated CONFIRM statement (See format below). The CONFIRM statement identifies the actual target system. One ENT statement will exist in the confirmation file for each entity received with OPTION=MIGRATE. The ENT statement will contain the DATE and TIME the entity was received.

Each ENT statement produced by NDVRDCF1 will be consumed by NDVRDCF2 on the source system and used to create a Migrate-out CLE (action code = C) in the CCDB. Each Migrate-out CLE will contain the actual received DATE/TIME and the TARGET SYSTEM/DICTIONARY names in its description. The CLE will appear in the source CCDB as of the VERIFY DATE/TIME of the entity after it is processed by NDVRDCF.

NDVRDCF1 Command Syntax

NDVRDCF1 accepts the following syntax:

```
SIGNON
  └─────┬─────────────────────────┬─┬──────────────────────────┬─────────────►◄
        └───────            └─┴─┐                              └─┘
    ┌───────┬─────────────────────────┐      ┌─┐
    │ DBName ──┬───dictname─┐      │ user-id─PASsword─password─
    └─DICtName └─┘      └─┘
    └───┬─┐      └───┬─┐
        └─┘      └─┘
    └───┬─┐      └─┘
        └─┘      └─┘
    ┌─┐      ┌─┐
    │ (ccid)─▼─(ccid)
    └─┘      └─┘
```

**SIGNON**

The SIGNON command identifies the user responsible for the migration and optional password and CCID list.

*Note: If no CCID list is specified, the default CCIDs for the user are assigned from the CCDB. The user must be authorized for MIGRATE=Y in the Security Class for this utility to operate.*
NDVRDCF1 Sample JCL

Use the following JCL to run NDVRDCF1. It is contained in member SAMPDCF1 on the CA Endevor/DB installation media JCL library:

Sample z/OS and OS/390 JCL

```jcl
//JOBNAME JOB YOUR.JOBCARD.INFORMATION//JOBLIB DD DISP=SHR,DSN=usercv.loadlib//
 DD DISP=SHR,DSN=ndvrb.loadlib// DD DISP=SHR,DSN=idms.loadlib/****

**********
ndvrdb.loadlib
idms.loadlib

***********************************************************

/*
//* JOB: SAMPDCF1
//* PURPOSE: POST-MIGRATION CONFIRMATION EXTRACT: RUN ON TARGET.
//* STEP: FUNCTION:
//* ===== =========
//* CONFIRM1 EXTRACT LIST OF MIGRATED ENTITIES RECEIVED AT TARGET 
//* SYSTEM TO LOG BACK ON SENDING SYSTEM.
//* */

/***********************************************************

//CONFIRM1 EXEC PGM=NDVRDCF1_REGION=1000K
//SYSCTL DD DSN=idms.sysctl,DISP=SHR
//SYSIDMS DD *
DMCL=dmcl-nameDICTNAME=dictionary-nameOther Optional SYSIDMS Parameters/*
//NDVRLST DD SYSOUT=* 
//NDVRDL DD SYSOUT=* 
//NDVRRR DD SYSOUT=* 
//NDVRENI DD DSN=user.ndvrdsel.dseno,DISP=SHR

/*
//* CONFIRM FILE TO PASS TO NDVRDCF2 ON THE SENDING SYSTEM:
//* */

//NDVRENO DD DSN=user.ndvrdcf1.dseno,DISP=(,CATLG,DELETE), 
// UNIT=disk,SPACE=(TRK,(15,5),RLSE),VOL=SER=volsr,
// DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//SYSUDUMP DD SYSOUT=* 
//NDVRIPT DD *
SIGNON 
      USER = youruserid PASSWORD = yourpswd DICTNAME userdict.
/*
```

NDVRDCF1 Outputs

Contents

- NDVRDCF1 Output File (ddname NDVRENO) (see page 138)
- NDVRDCF1 Control Report (ddname NDVRLST) (see page 138)
  - NDVRDCF1- Input Command Listing (see page 138)
  - NDVRDCF1- Input Entity List Header Report (see page 139)
  - NDVRDCF1- End-of-Job Statistics (see page 139)
  - NDVRDCF1 Detail Report (ddname NDVRDTL) (see page 140)
  - NDVRDCF1- Output Confirmation File Report (see page 140)

NDVRDCF1 produces two reports:
A control report (NDVRLST). This report echoes the user-specified syntax.

A detail report (NDVRDTL). This report itemizes all statements placed in the NDVRENO file.

**NDVRFDCF1 Output File (ddname NDVRENO)**

The output file from NDVRFDCF1 is source syntax as shown below.

```
CONFIRM SYSTEM = 'SYSTEM01' DBNAME = 'TGTNDVR'
    DATE = mm/dd/yy TIME = 09:14:34
    USER = 'EDB-SYSTEM-ADMINISTRATOR'
    CCID = ('EDB-SYSADMIN', ' ', ' ', ' ', ' ', ' ', ' ', ' ') 00000001
SOURCE SYSTEM = 'SYSTEM01' DBNAME = 'SRCNDVR'
    VERIFY DATE = mm/dd/yy TIME = 09:00:18
    TARGET SYSTEM = 'SYSTEM01' DBNAME = 'TGTNDVR'
    MODE = EXECUTE
    INPUT = DATABASE
    SIGNOUT TO CCID = 'EDB-SYSADMIN'
    *********** INCLUDE RULE NUMBER 0000
    INCLUDE ALL
    WHERE STATUS = 'MIGRATE-TEST'
    *********** INCLUDE RULE NUMBER 0000
    INCLUDE ALL
    FROM DATE = mm/dd/yy
    *********** EXCLUDE RULE NUMBER 0001
    EXCLUDE WHERE STATUS = 'NEVER-MIGRATE'
    EXPAND IDD CHANGE RELATIONSHIPS
    LIST FOLLOWS.
    ENT RECORD 'COVERAGE' 0100
    MIGRATED TO 'TGTNDVR SYSTEM01'
    DATE mm/dd/yy TIME = 06:38:36.
```

**NDVRFDCF1 Control Report (ddname NDVRLST)**

The control report produced by NDVRFDCF1 is comprised of three parts:

- An input command listing.
- An input entity header list.
- An End-of-Job statistics summary.

**NDVRFDCF1- Input Command Listing**

The Input Command Listing echoes the user-supplied syntax. All required control information for the execution of the extract comes from the NDVRENI file. The SIGNON is the only command displayed.
NDVRDCF1- Input Entity List Header Report

The Input Entity List Header Report displays the control information from the front of the NDVRENI file that was used to perform the extract. All Migrate-in CLEs containing the specified SOURCE SYSTEM, SOURCE DBNAME, and VERIFY DATE/TIME will be extracted into the confirmation file format described above. Since the control information reflects the NDVRDSEL run that last selected the entities, the appropriate data is extracted regardless of when the importation occurred. A unique date and time for importation will be placed on each record in the confirmation file.

NDVRDCF1- End-of-Job Statistics

The End-of-Job Statistics report summarizes the processing activity of NDVRDCF1. All entities in the CCDB are examined for CLEs relating to the migration.
### NDVRDCF1: I001 TARGET CONFIRMATION ENTITY TOTALS

<table>
<thead>
<tr>
<th>ENTITY TYPE</th>
<th>INSPECT</th>
<th>CONFIRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELEMENT</td>
<td>386</td>
<td>0</td>
</tr>
<tr>
<td>LOAD MODULE</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>MAP</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>MODULE</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RECORD</td>
<td>26</td>
<td>13</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>OTHER</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>435</td>
<td>21</td>
</tr>
</tbody>
</table>

### NDVRDCF1 Detail Report (ddname NDVRDTL)

The detail report produced by NDVRDCF1 itemizes confirmation sent to the NDVRENO file. Confirmation information is comprised of two parts. The first part represents the control information propagated from the NDVRENI file. The second part contains confirmation type ENT statements. These ENT statements produce DATEs and TIMEs that represent the time the run unit that updated the target dictionary actually began execution. This date and time will be placed in the descriptive portion of Migrate-out CLEs to be created when NDVRDCF2 (See below) reads the NDVRENO file on the source system.

### NDVRDCF1- Output Confirmation File Report

Each entity successfully received by the target system is displayed in collating sequence by name. After the NDVRENI file is reused by the next migration (and the control information is overlaid in the NDVRENI file), an entity list report on any migration can be obtained through the Target Migration Summary (See CA Endevor/DB for CA IDMS Using section).
NDVRDCF2 Source Confirmation

Contents

- NDVRDCF2 Command Syntax (see page 141)
- NDVRDCF2 Sample JCL (see page 142)
  - Sample z/OS and OS/390 JCL (see page 142)
- NDVRDCF2 Outputs (see page 143)
  - NDVRDCF2- Input Command Listing (see page 143)
  - NDVRDCF2- Entity File Listing (see page 143)
  - NDVRDCF2- End-of-Job Statistics (see page 144)

NDVRDCF2 processes the output file created by NDVRDCF1 as ddname NDVRENI to create Migrate-out (action code = C) CLEs on the source system. These CLEs serve as an essential audit trail to the target destination of entities. Any entity that has not been modified since it was last migrated from the source to the target will not be selected for subsequent migration to that target system by NDVRDSEL. Similarly, the NDVRDSEL will only examine CLEs up to the last Migrate-out CLE for the target system for integrity warnings.

Another function of NDVRDCF2 is to signin all entities signed out when NDVRDSEL originally ran. This will be done if the SIGNOUT TO command appears in the control information contained in NDVRENI.

Optionally, NDVRDCF2 can be used to backout previously created Migrate-out CLEs in the event that a migration is backed-off of the target system permanently. In this case, the appropriate CLEs are deleted. Backoff is accomplished by reading in the same NDVRENI file that was originally used to create the CLEs.

NDVRDCF2 Command Syntax

NDVRDCF2 accepts the following syntax:

```
************** EXCLUDE RULE NUMBER 0001
EXCLUDE WHERE STATUS = 'NEVER-MIGRATE'
EXPAND IDD CHANGE RELATIONSHIPS
LIST follows
ENT RECORD 'COVERAGE MIGRATED TO TGTNDVR SYSTEM81'
DATE mm/dd/yy TIME = 06:38:36.

NDVRDCF2 processes the output file created by NDVRDCF1 as ddname NDVRENI to create Migrate-out (action code = C) CLEs on the source system. These CLEs serve as an essential audit trail to the target destination of entities. Any entity that has not been modified since it was last migrated from the source to the target will not be selected for subsequent migration to that target system by NDVRDSEL. Similarly, the NDVRDSEL will only examine CLEs up to the last Migrate-out CLE for the target system for integrity warnings.

Another function of NDVRDCF2 is to signin all entities signed out when NDVRDSEL originally ran. This will be done if the SIGNOUT TO command appears in the control information contained in NDVRENI.

Optionally, NDVRDCF2 can be used to backout previously created Migrate-out CLEs in the event that a migration is backed-off of the target system permanently. In this case, the appropriate CLEs are deleted. Backoff is accomplished by reading in the same NDVRENI file that was originally used to create the CLEs.
```
**SIGNON**

The SIGNON command identifies the user responsible for the migration and optional password and CCID list. If no CCID list is specified, the default CCIDs for the user are assigned from the CCDB.

*Note:* For this utility to operate, the user must be authorized for MIGRATE=Y in the Security Class.

**MODE**

The MODE command instructs NDVRDCF2 to create or delete CLEs. In either case, the confirmation file produced by NDVRDCF1 must be supplied as ddname NDVRENI. MODE=BACKOFF must be coded to delete CLEs created by a prior MODE=EXECUTE.

*Note:* For this utility to operate, the user must be authorized for MIGRATE=Y in the Security Class.

### NDVRDCF2 Sample JCL

Use the following JCL to run NDVRDCF2. It is contained in member SAMPDCF2 on the CA Endevor/DB installation media JCL library:

**Sample z/OS and OS/390 JCL**

```plaintext
//JOBNAME JOB YOUR.JOBCARD.INFORMATION
//JOBLIB DD DISP=SHR,DSN=usercv.loadlib/
// DD DISP=SHR,DSN=ndvrdb.loadlib/
// DD DISP=SHR,DSN=idsm_loadlib/*
//** JOB: SAMPDCF2
//** PURPOSE: MARK ENTITIES ON SENDING SYSTEM 'CONFIRMED' AS MIGRATED
//** STEP: FUNCTION:
//** =======
//** CONFIRM2 UPDATE ENTITIES LISTED ON FILE FROM NDVRDCF1.
//******************************************************************************
//** CONFIRM2 EXEC PGM=NDVRDCF2,REGION=800K
//SYSCTL DD DISP=SHR,DSN=idsms.sysctl//NDVRENI DD DISP=SHR,DSN=user.ndvrdcf1.dseno/
//NDVRLST DD SYSDUMP=* 
//NDVRERR DD SYSOUT=* 
//SYSUDUMP DD DUMMY 
//SYSIDMS DD * 
//DMCL=dmcl-nameDICTNAME=dictionary-nameOther Optional SYSIDMS Parameters/*
//NDVRRIPT DD * 
SIGNON
    USER = youruserid PASSWORD = yourpswd     DICTNAME userdict.
    MODE = EXECUTE.
/**
```
NDVRDCF2 Outputs

NDVRDCF2 produces one control report (NDVRLST). The control report is composed of two parts as follows:

- An input command listing.
- An entity file listing.

The report formats are explained below in detail.

NDVRDCF2- Input Command Listing

The Input Command Listing displays the input syntax supplied in the NDVRIPT file.

```
volver                        CA, INC.
DATE      TIME      PAGE    C A - E N D E V O R
RELEASE  nn.n  mm/dd/yy  11:42:58  00001
/ D B
NDVRDCF2
INPUT COMMAND LISTING
SIGNON DBNAME SRCNDVR
USER EDB-SYSTEM-ADMINISTRATOR
CCID EDB-SYSADMIN.
MODE EXECUTE.
```

NDVRDCF2- Entity File Listing

The Entity File Listing displays the NDVRENI file that was processed by NDVRDCF2. All Migrate-out CLEs created on the source system will be inserted into the Change Log as of the date of selection by NDVRDSEL. This accurately reflects the time the entity was selected for migration. The descriptive portion of the Migrate-out CLEs created will contain the target system identifier and the date and time it was received at the target.

At this point in time, a permanent and complete audit trail exists on both the target and source systems. The descriptive portions of each system's CLEs point to each other.

```
volver                        CA, INC.
DATE      TIME      PAGE    C A - E N D E V O R
RELEASE  nn.n  mm/dd/yy  11:43:03  00002
/ D B
NDVRDCF2
ENTITY FILE LISTING
CONFIRM SYSTEM = 'SYSTEM81' DBNAME = 'TGTNDVR'
DATE = mm/dd/yy TIME = 10:27:25  00000001
USER = 'EDB-SYSTEM-ADMINISTRATOR'
CCID = ('EDB-SYSADMIN',)
SOURCE SYSTEM = 'SYSTEM81' DBNAME = 'SRCNDVR'
VERIFY DATE = mm/dd/yy TIME = 09:00:18  00000009
TARGET SYSTEM = 'SYSTEM81' DBNAME = 'TGTNDVR'
MODE = EXECUTE
INPUT = DATABASE
SIGNOUT TO CCID = 'EDB-SYSADMIN'
```

*********** INCLUDE RULE NUMBER 0000
NDVRDCF2- End-of-Job Statistics

The End-of-Job Statistics report summarizes the processing activity of NDVRDCF2.

<table>
<thead>
<tr>
<th>volser</th>
<th>DATE</th>
<th>TIME</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CA, INC.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RELEASE nn.n</td>
<td>CA - ENDEVOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/ DB mm/dd/yy</td>
<td>11:43:19</td>
<td>00004</td>
<td></td>
</tr>
</tbody>
</table>

END-OF-JOB STATISTICS

NDVRDCF2: I001 SOURCE CONFIRMATION ENTITY TOTALS

<table>
<thead>
<tr>
<th>ENTITY TYPE</th>
<th>CONFIRM</th>
<th>CONFIRM</th>
<th>CONFIRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD MODULE</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>MAP</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MODULE</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RECORD</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TOTAL--------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
</tbody>
</table>

The Source Code Comparator

CA Endevor/DB's Source Code Comparator is designed to identify changes to individual lines of code modified between one version of a system and another. Typically, the Comparator is employed as a troubleshooting aid, or as a means of identifying the changes that an installation has made to tailor a vendor-supplied application system. It is an invaluable tool when applying maintenance releases of vendor software or when combining parallel development versions. In these cases, the Promotion Support Facilities, in conjunction with the CCDB Change Log, identify and extract the affected entities, while the Comparator identifies the exact areas within those entities that have changed.

The Source Code Comparator can be executed in one of two modes:

- **Stand-alone mode**
  
  Any two sequential files with a record length of between 4 and 256 bytes (fixed or variable) can be compared with the stand-alone CA Endevor/DB utility NDVRCOMP.

- **Migration mode**
  
  In migration mode, the standard output files from a NDVRDLVR run (See Section 8) is compared with the target dictionary. The CA Endevor/DB utility program NDVRDCMP:
Automatically parses the input files and identifies the entities contained therein;

- Dynamically invokes IDD to extract the corresponding entities from the target dictionary;
- Internally invokes the source statement comparator for each entity;
- Index and summarize the results by entity name and type in a control report.

Facilities also exist to limit the compare to entities of specific types in a single execution. For example, only ELEMENTs and RECORDs might be singled out for comparison even though the input files contain entities of all types. Sample JCL for the execution of the compare in migration mode is contained on the CA Endevor/DB installation media.

Running the Comparator in Stand-alone Mode

Contents

- JCL (see page 146)
  - Sample z/OS and OS/390 JCL (see page 146)
  - SAMPLE z/OS and OS/390 JCL (see page 147)
  - z/OS and OS/390 JCL (see page 148)
  - NDVRRJCL Command Syntax (see page 148)
    - Syntax Rules (see page 148)
  - NDVRRJCL Inputs (see page 149)
  - NDVRRJCL Outputs (see page 150)
  - Return Codes (see page 150)
- Running NDVRCOMP (see page 151)
  - Control Card Specifications (see page 151)
    - Syntax (see page 151)
  - Input Parameters (see page 152)
  - Sample Outputs (see page 153)
  - Return Codes (see page 156)

CA Endevor/DB provides a utility, NDVRCOMP, which compares the contents of two PDS members and/or sequential files, and reports the differences between them.

NDVRCOMP accepts as input two files, which can be either PDS members or sequential files. The files can be fixed or variable length, but cannot exceed an LRECL of 256 (260 for variable-length files).

NDVRCOMP reports the differences between the two files. The first file, NDVRIN1, is assumed to be the "base" file. The second file, NDVRIN2, is assumed to be the "changed" file. NDVRCOMP reports the differences in file-2 as compared to file-1.

The files are compared line-by-line, based on the contents of particular (contiguous) characters. The range of characters included in the compare is defined in terms of a from and thru column. For example, you might want to compare two files based on the contents of positions 1-5 only.
The output from NDVRCOMP can be formatted either for file browse (without ASA characters and headings) or for hardcopy printout (including ASA characters and headers).

**JCL**

Use the JCL below to run NDVRCOMP. It is contained in member SAMPCOMP on the CA Endevor/DB installation media JCL library:

**Sample z/OS and OS/390 JCL**

```plaintext
//JOBNAME JOB YOUR.JOBCARD.INFORMATION//JOBLIB DD DISP=SHR,DSN=ndvrdb.loadlib/*
/***/
//* JOB: SAMPCOMP */
//* PURPOSE: COMPARE THE CONTENTS OF TWO SOURCE-CODE TYPE FILES */
//* (NDVRIN1 AND NDVRIN2) AND PRODUCE A FILE DESCRIBING */
//* THEIR DIFFERENCES. */
//* YOU TELL NDVRCOMP WHAT TO DO BY SPECIFYING A 'COMPARE' COMMAND */
//* THE SYNTAX IS AS FOLLOWS: */
//* CO**
//* COLUMN N TO M RECORD TYPE = FIXED/VARIABLE LENGTH = NNNN */
//* PAD = BLANK/NUL/FF OUTPUT = CHANGES/HISTORY/NEW */
//* FORMAT = FILE/DISPLAY SIZE = NNNNN TITLE = 'YOUR TITLE' */
//* . */
//* ALL CLAUSES ARE OPTIONAL. THE DEFAULT VALUES ARE AS FOLLOWS: */
//* COMPARE COLUMN = 1 TO 72 RECORD TYPE = FIXED LENGTH = 80 */
//* PAD = BLANK OUTPUT=CHANGES FORMAT = DISPLAY SIZE = 10000 */
//* TITLE = ' '. */
//* WHERE: */
//* COLUMN THE START AND END COLUMNS TO INSPECT */
//* RECORD THE RECORD FORMAT AND (MAXIMUM) RECORD LENGTH */
//* PAD THE CHARACTER TO USE IN EXTENDING VARIABLE- */
//* LENGTH RECORDS BEFORE COMPARING THEM */
//* OUTPUT CONTENT OF OUTPUT FILE: */
//* CHANGES SHOW THE INSERTIONS AND DELETIONS */
//* HISTORY SHOW THE INSERTS AND DELETES IN THE CONTEXT */
//* OF THE NDVRIN1 SOURCE */
//* NEW SHOW THE INSERTS IN THE CONTEXT OF THE */
//* NDVRIN2 SOURCE */
//* FORMAT WHERE THE OUTPUT IS TO BE WRITTEN */
//* FILE WRITE TO NDVRPCH FILE */
//* DISPLAY WRITE TO NDVRLST FILE */
//* SIZE THE ESTIMATED COUNT OF THE NUMBER OF RECORDS */
//* IN NDVRIN1 PLUS THE NUMBER OF RECORDS IN */
//* NDVRIN2 (OVER ESTIMATE IF YOU DON'T KNOW) */
//* TITLE A TITLE FOR THE TOP OF EACH PAGE OF OUTPUT */
//* WHEN FORMAT=DISPLAY IS SPECIFIED */
//* RESTRICTION: LRECL FOR INPUT FILES (NDVRIN1 AND NDVRIN2) MAY */
//* NOT EXCEED 256 */
//***/
//*** EXEC PGM=NDVRCOMP,REGION=400K */
//NDVRIN1 DD DISP=SHR,DSN=original.source.dataset.or.member//NDVRIN2 DD DISP=SHR, */
DSN=changed.source.dataset.or.member//NDVRPCH DD DSNS=user.changes.dataset,DISP=(NEW, */
CATLG,DELETE), */
// UNIT=disk, VOL=SER=volser, SPACE=(TRK,(5,5), RLSE), */
```
// DCB=(RECFM=FB,LRECL=88,BLKSIZE=3168)
//SORTWK01 DD UNIT=disk,SPACE=(CYL,(2,1))
//SORTWK02 DD UNIT=disk,SPACE=(CYL,(2,1))
//SORTWK03 DD UNIT=disk,SPACE=(CYL,(2,1))
//SORTWK04 DD UNIT=disk,SPACE=(CYL,(2,1))
//NDVRSLST DD SYSOUT=*  
//NDVRERR DD SYSOUT=*  
//SYSOUT DD DUMMY
//SYSUDUMP DD DUMMY
//NDVRRIPT DD *

COMPARE COLUMN = 1 TO 72 RECORD TYPE = FIXED LENGTH = 80
   PAD = BLANK OUTPUT=CHANGES FORMAT = DISPLAY SIZE = 10000
   TITLE = '.
/*
SAMPLE z/OS and OS/390 JCL

//JOBNAME JOB YOUR.JOBCARD.INFORMATION/JOBLIB DD DISP=SHR,DSN=usercv.loadlib//
   DD DISP=SHR,DSN=ndvrdlvr.loadlib//
   DD DISP=SHR,DSN=idsms.loadlib/**

// JOB:   SAMPDCMP
// PURPOSE:  PRODUCES SOURCE-COMPARISON REPORT FROM NDVRDLVR OUTPUT
// STEP:  FUNCTION:
// ==========

// COMPARE PROGRAM NDVRDCMP DRIVES IDMSDDDL, RHDCMPUT, ETC. TO
// COMPARE TARGET SOURCE WITH NDVRDLVR MIGRATION EXPORT
// FILE CONTENTS.

//COMPARE EXEC PGM=NDVRDCMP,REGION=1008K
//SYSCTL DD DISP=SHR,DSN=idsms.sysctl//SYSIPT DD DSN=&.SYSIPT.,SPACE=(TRK,(1,1)),
   DISP=(NEW,DELETE),
   // UNIT=disk,DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPCH DD DSN=&.SYSPCH.,SPACE=(CYL,(2,1)),DISP=(NEW,DELETE),
// NDVRIN1 DD DSN=&.NDVRIN1.,SPACE=(CYL,(2,1)),DISP=(NEW,DELETE),
   // UNIT=disk,DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//NDVRIN2 DD DSN=&.NDVRIN2.,SPACE=(CYL,(2,1)),DISP=(NEW,DELETE),
   // UNIT=disk,DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SORTIN DD DSN=&.SORTIN.,SPACE=(CYL,(1,1)),DISP=(NEW,DELETE),
   // UNIT=disk,DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SORTOUT DD DSN=&.SORTOUT.,SPACE=(CYL,(1,1)),DISP=(NEW,DELETE),
   // UNIT=disk,DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SORTWK01 DD SPACE=(CYL,(4,4)),UNIT=disk
//SORTWK02 DD SPACE=(CYL,(4,4)),UNIT=disk
//SORTWK03 DD SPACE=(CYL,(4,4)),UNIT=disk
//SORTWK04 DD SPACE=(CYL,(4,4)),UNIT=disk

// THE FOLLOWING FILE CONCATENATES THE "SOURCE CODE" EXPORT *
// FILES PRODUCED BY NDVRDLVR AT THE SOURCE SYSTEM/DICTIONARY. *
//NDVRSRC DD DISP=SHR,DSN=user.ndvrdlvr.dsdupd//
   DD DISP=SHR,DSN=user.
   ndvrdlvr.dscupd//
   DD DISP=SHR,DSN=user.ndvrdlvr.dsuupd//
   DD DISP=SHR,DSN=user.
   ndvrdlvr.dsmupd//NDVRLST DD SYSOUT=*  
//NDVRERR DD SYSOUT=*  
//SYSLST DD DUMMY
//SYSOUT DD DUMMY
//SYSUDUMP DD DUMMY
//SYSIDMS DD *
//DMCL=dmlc-nameDICTNAME=dictionary-nameOther Optional SYSIDMS Parameters/*
//NDVRRIPT DD *
SIGNON
   USER = youruserid PASSWORD = yourpswd DICTNAME userdict.
   REPORT CHANGES AND DETAILS.
COMPARE TYPES = (PROCESS, PROGRAM, ELEMENT, RECORD, SCHEMA, SUBSCHEMA, QFILE, TABLE, MODULE, MESSAGE, MAP).

/*

z/OS and OS/390 JCL

 JOBNAME JOB YOUR.JOBCARD.INFORMATION
 JOBLIB DD DISP=SHR,DSN=ndvrdb.loadlib

***************************************************************************

* JOB: SAMPRJCL
* PURPOSE: CONVERT JCL TO USE NDVRBOOK.
* STEP: FUNCTION:
*** ***

* GORJCL CONVERTS JCL TO EXECUTE NDVRBOOK WITH INSTRUCTIONS AS
* TO THE PROGRAM TO EXECUTE AND THE USER TO ASSIGN CHANGE
* LOG ENTRIES TO.

***************************************************************************

* GORJCL EXEC PGM=NDVRRJCL,REGION=640K
* NDVRJCLI DD DISP=SHR,DSN=your.input.jcl.dataset
* NDVRJCLO DD DISP=OLD,DSN=your.output.jcl.dataset
* NDVRJLIST DD SYSOUT=* 
* NDVRERRR DD SYSOUT=* 
* SYSUDUMP DD DUMMY
* NDVRIPT DD DATA,DLM='##'

CONVERT JOBSTREAM
 USER = youruserid PASSWORD = yourpswd
 DICTNAME userdict.

++ PUT YOUR SEARCH FOR COMMANDS HERE. ++
##

NDVRRJCL Command Syntax

Through the use of input commands the CA Endevor/DB user, password, and dictionary are specified. In addition, blocks of JCL statements can be replaced.

►►────── CONVert JOBstream ───────────────────────────────────────────►

USEr is user-id ─ PASsword is password ─

DICtname is dictname .

DBName

SEARCH FOR search-for-jcl-statements

REPLACE WITH replace-with-jcl-statements

SEARCH END .

Syntax Rules
CONVERT JOBSTREAM

The CONVERT JOBSTREAM command is used to identify which CA Endevor/DB user to assign dictionary updates. When the converted JCL is executed, the CA Endevor/DB user name and password are passed to CA Endevor/DB's Security System/Change Monitor prior to invoking the utility that performs dictionary updates. The dictname specifies the dbname of the dictionary to which the updates are to be performed.

NDVRRJCL can convert any type of JCL. For example, you can run NDVRRJCL in CMS and convert a z/OS or OS/390 jobstream. NDVRRJCL automatically recognizes the job language.

The user, password, and dbname(dictname) clauses are used to build the SIGNON command so that NDVROBOOK can perform the CA Endevor/DB signon (when the converted JCL is executed). Depending on your CA Endevor/DB signon requirements, all or none of these clauses may be required. In addition, if the USER clause is not specified, the PASSWORD clause is not used to build the subsequent CA Endevor/DB SIGNON command.

SEARCH FOR REPLACE WITH

The SEARCH FOR command is used to search for blocks of JCL statements within the input JCL and to replace those blocks with the "REPLACE WITH" JCL statements in the output JCL. The command must be written as follows:

- The SEARCH FOR clauses must begin in card column 1 and be exact, up to and including card column 72.
- The REPLACE WITH clauses must begin in card column 1 and include up to and including card column 72.
- The "REPLACE WITH" and "SEARCH END" clauses must begin in card column 1 and cannot be abbreviated.
- This command must be entered exactly in the sequence as indicated in the diagram above.
- Replacement of partial statements is not supported.

SEARCH END

The SEARCH END clause terminates the SEARCH FOR/REPLACE WITH blocks.

NDVRRJCL Inputs

The NDVRRJCL JCL input file (NDVRJCLI) contains the JCL jobstream(s) which are to be converted. The following is an example of a jobstream contained in that file.

```
//SAMPDDL JOB (?????????, 'IDD UPDATE'
//DDDL EXEC PGM=IDMSDDDL,REGION=1024K
//STEPLIB DD DISP=SHR,DSN=SYSTEM.LOAD
// DD DISP=SHR,DSN=EDB.LOAD
// DD DISP=SHR,DSN=IDMS.LOAD
//SYSLST DD SYSOUT=*.
//SYSPRINT DD SYSOUT=*.
//SORTMSG DD SYSOUT=*.
//SYSDUMP DD SYSOUT=*.
//SYSCTL DD DISP=SHR,DSN=SYSTEM.SYSCTL
//SYSPCH DD SYSOUT=*.
//SYSIDMS DD *
//DMCL=CVDMCL/*
//SYSIPT DD *
//SIGNON USER DBADMIN PASSWORD DBADMIN DICTNAME SRCNDVR.
```
The NDVRRJCL input commands file (NDVRIPT) contains the input command syntax NDVRRJCL uses to convert the JCL. The following are the input commands contained in that file.

CONVERT JOBSTREAM USER IS EDBADMIN PASSWORD IS EDBADMIN DICTNAME IS DEMO.
SEARCH FOR
//STEPLIB DD DSN=TEST.LOADLIB,DISP=SHR
// DD DSN=DEVEL.LOADLIB,DISP=SHR
// DD DSN=RELEASE.LOADLIB,DISP=SHR
REPLACE WITH
//STEPLIB DD DSN=TEST.LOADLIB,DISP=SHR
// DD DSN=DEVEL.LOADLIB,DISP=SHR
// DD DSN=RELEASE.LOADLIB,DISP=SHR
// DD DSN=ENDEVOR.LOADLIB,DISP=SHR
SEARCH END.

NDVRRJCL Outputs

The NDVRRJCL output file (NDVRJCLO) will contain the converted JCL jobstream(s) which have been converted using the input commands against the input file.

The NDVRRJCL output listing file (NDVRLST) will contain a summary of the input commands read and all error and informational messages issued.

The following is an output file contained in the NDVRJCLO file.

//SAMPDDDL JOB (?????????),'IDD UPDATE'
//DDDLD EXEC PGM=NDVRBOOK,REGION=1024K
//STEPLIB DD DISP=SHR,DSN=SYSTEM81.LOADLIB
// DD DISP=SHR,DSN=DIST.CAABF0.LOADLIB
// DD DISP=SHR,DSN=DIST.CAGJE0.LOADLIB
//SYSLST DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SORTMSG DD SYSOUT=* 
//SYSUDUMP DD SYSOUT=* 
//SYSCTL DD DISP=SHR,DSN=SYSTEM.SYSCTL
//SYSPCH DD SYSOUT=* 
//SYSIDMS DD *
DMCL=CVDMCL
//
//SYSIPT DD *
SIGNON USER DBADMIN PASSWORD DBADMIN DICTNAME SRCNDVR.
...
//NDVRLLST DD SYSOUT=* 
//NDVRIPT DD *
SIGNON USER NAME IS "EDBADMIN"
 PASSWORD IS "EDBADMIN"
 DICTNAME IS "SRCNDVR"
.
PROGRAM IS IDMSDDDL.
/*

Return Codes

NDVRRJCL will return a 0 if it is successful, or an 8 if an error is encountered. If an error is encountered, the output NDVRLST file will contain message(s) stating the reason for the failure. The following is an example of an output listing contained in the NDVRLST file.
Running NDVRCOMP

You can use either the NDVRIPT file (control card specifications) or input parms to supply information to run NDVRCOMP. Each of the methods is described in the following sections.

Control Card Specifications

Control cards are supplied on file NDVRIPT and are free-form. The control cards consist of the word COMPARE followed by optional clauses and ending with a period (.). This method provides the greatest degree of flexibility and control.

Syntax

```
COMpare COLumn is start-column to end-column
RECORD TYPE is FIXED| VARIABLE LENgth is length
PAD BLAnk 'pad-char' OUTput is CHAnges| HIStory| NEW
FORmat is DISplay FILE| IEBupdte| PDM SIZe is file-size
TITle is 'user-title'
```

A description of the syntax clauses follows:

- **COLUMN**
  Specifies the columns to compare from and thru. If omitted, columns 1-72 are used.

- **OUTPUT**
  Specifies which records are to be output. The default is CHANGES only. The options are: HISTORY (shows the existing member together with both inserts and deletes), and NEW (shows the new member, highlighting inserts only).

- **FORMAT**
  Specifies the output format. The default is DISPLAY that writes the output to file NDVRLST. The syntax listing is produced first, followed by the original output file in report format - with carriage control and page headings. The FILE option (i.e., BROWSE) writes the data to DDname NDVRPCH without page headings.
- **PAD**
  PAD is applicable to variable length records only. The default is BLANK (which pads short records with blanks up to the compare length). The options are NULL (which pads with binary zeros) or "x" (which pads with the specified single character).

- **SIZE**
  Specifies a file-size estimate for the sort. By default, this option is ignored.

- **TITLE**
  Appears before the first data line as a line of asterisks, followed by the title string and another line of asterisks. Useful if FORMAT=FILE.

### Input Parameters

Specify PARM values as described below. Separate the values using a single comma, leaving no spaces between the values.

- **output-format**
  Two-character code that indicates the type of comparison information you want reported (character 1) and the format of the output file (character 2). The default is CD.

Specify the first character (type of information you want) as follows:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Print only the changes between the two files; that is, those lines inserted from file 2 and those lines deleted from file 1. A modification to a line of code displays as an insert of the modified line(s) immediately followed by a delete of the old line(s).</td>
</tr>
<tr>
<td>H</td>
<td>Print a history of both files including: Inserts. Any lines that were in file-2 but are not in file-1, highlighting those lines with %INSERT to the far left: % allows you to scan for changes easily; INSERT indicates that the line was new in file-2. Deletes. Any lines that were in file-1 but are not in file-2, highlighting those lines with %DELETE to the far left: % allows you to scan for changes easily; DELETE indicates that the line is not in file-2. Equals. Any lines that were equal in file-1 and in file-2. These lines display with blanks in the far left.</td>
</tr>
<tr>
<td>B</td>
<td>Print (browse) the contents of file-2, highlighting only the statements inserted relative to file-1 with %INSERT in the far left.</td>
</tr>
</tbody>
</table>

Specify the second character (output format) as follows:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>The output file is in browse format and does not have any ASA characters or headers. The output is written to DDname NDVRPCH.</td>
</tr>
<tr>
<td>D</td>
<td>The output file is formatted for print, and includes ASA characters and headers. The output is written to DDname NDVRLST.</td>
</tr>
</tbody>
</table>
• **from**
  Starting character for the compare. NDVRCOMP begins its search at this position, within both files. The default is 1.

• **thru**
  Ending character for the compare. NDVRCOMP ends its search with this position, within both files. For variable-length records, if the record in one file is longer than that in the other, and the **thru** character extends beyond the end of the record, NDVRCOMP pads according to the **pad-char** specification before performing the compare.
  The default **thru** specification is 72.

• **rec-count**
  Largest number of records in either file. The default is 10000. Estimate high when specifying this value.

• **pad-char**
  Pad character used for variable-length records, as described for the **thru** parameter above.
  Specify this as follows. (The default is BLANK.)

<table>
<thead>
<tr>
<th>Code Pad with:</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLA</td>
</tr>
<tr>
<td>NK</td>
</tr>
<tr>
<td>NUL</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td><strong>nnn</strong></td>
</tr>
</tbody>
</table>

**Sample Outputs**

The following report is returned when you specify **output-format code CD** (Changes Report). It shows only the changes between the two files: that is, those lines that are in file-2 but not in file-1 (marked with %INSERT), or those lines that are missing from file-2 that were in file-1 (marked with %DELETE).

```
NDVRCOMP          FILE COMPARE UTILITY
COMPARE COLUMN 1 TO 72
RECORD TYPE FIXED LENGTH 80
PAD = BLANK
OUTPUT = CHANGES
FORMAT = DISPLAY
SIZE = 250

INSERT        ADD MODULE NAME IDMS-STATUS VERSION 2 LANGUAGE IS COBOL 000001
DELETE        ADD MODULE NAME IDMS-STATUS VERSION 1 LANGUAGE IS COBOL 000001
INSERT        IF DB-STATUS-OK GO TO IDMS-STATUS-EXIT.
DELETE        IF DB-STATUS-OK GO TO ISABEX.
INSERT        IF ERROR-STATUS = '0295' OR '0895' OR '0995' OR '1295' 000009
INSERT        DISPLAY 'CA-ENDEVOR
```
The following report is returned when you specify output-format code HD (History Report). It lists the contents of file-2, highlighting inserts from file-2 and deletes from file-1.
ABORT.
%INSERT IF ERROR-STATUS = '0295' OR '0895' OR '0995' OR '1295'
%INSERT /DB AUTHORIZATION ERROR'
% INSERT UPON CONSOLE
1
%INSERT GO TO IDMS-STATUS-EXIT.
% INSERT DISPLAY
3
%INSERT 'PROGRAM ' PROGRAM-NAME ' ABORTING WITH '
%INSERT 'ERROR STATUS ' ERROR-STATUS
%INSERT , ' NOTIFY DATABASE ADMINISTRATION GROUP'
% INSERT UPON CONSOLE.
7
%DELETE DISPLAY '**************************'
9
%DELETE ' ABORTING - ' PROGRAM-NAME
%DELETE ' , ' ERROR-STATUS
%DELETE ' , ' ERROR-RECORD
%DELETE **** RECOVER IDMS ****
% DELETE UPON CONSOLE.
4
%DISPLAY 'PROGRAM NAME ------ ' PROGRAM-NAME.
%DISPLAY 'ERROR STATUS ------ ' ERROR-STATUS.
%DISPLAY 'ERROR RECORD ------ ' ERROR-RECORD.
%DISPLAY 'ERROR SET --------- ' ERROR-SET.
%DISPLAY 'ERROR AREA -------- ' ERROR-AREA.
%DISPLAY 'LAST GOOD RECORD -- ' RECORD-NAME.
%DISPLAY 'LAST GOOD AREA ---- ' AREA-NAME.
%DISPLAY 'DML SEQUENCE ------ ' DML-SEQUENCE.
%ROLLBACK.
26
%CALL 'ABORT'.
27
%INSERT IDMS-STATUS-EXIT.
% INSERT EXIT.
9
% DELETE ISABEX. EXIT.
5
%MSEND.
30
****** RECORDS: FILE 1 = 00026 FILE 2 = 00030 INSERTS = 00013 DELETES = 00009 *****
Return Codes

The COND CODE values below can be returned by NDVRCOMP. Code 3007 is the expected result. Other values might be returned, indicating a problem with the sort. If this happens, rerun the job to obtain the sort messages, specifying //SYSOUT DD SYSOUT=*.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>The input files are identical for the columns compared. No reports were produced.</td>
</tr>
<tr>
<td>3001</td>
<td>An input or output file could not be opened. Ensure that the DD statements are correct for all files and try again.</td>
</tr>
<tr>
<td>3002</td>
<td>The number of records in one or both of the input files exceeds the maximum count specified by the rec-count parameter. Increase the count and try again. It is better to estimate high rather than low.</td>
</tr>
<tr>
<td>3003</td>
<td>The LRECL for an input file exceeded 256 (260 for variable-length). You cannot use this file as input.</td>
</tr>
<tr>
<td>3005</td>
<td>The record format for an input file is Undefined. The record must specify either Fixed or Variable.</td>
</tr>
<tr>
<td>3006</td>
<td>An input parameter is missing or invalid (e.g., thru &gt; from). Check your syntax with the parameter descriptions above, correct the problem, and resubmit the job.</td>
</tr>
<tr>
<td>3007</td>
<td>NDVRCOMP completed its compare successfully, and found differences between the files. This is the standard return code.</td>
</tr>
</tbody>
</table>

Running the Comparator in Migration Mode

To run a compare in migration mode, the utility program NDVRDCMP is executed. NDVRDCMP accepts as input the standard source files extracted by the NDVRDLVR program (See Section 8 for details on NDVRDLVR).

The contents of the incoming files are compared with the contents of the target dictionary on an entity-by-entity, line-by-line basis. The output reports of changes is identical to the stand-alone utility display. The target dictionary is considered file-1, and the input files produced by NDVRDLVR comprise file-2. When an entity exists in the incoming file, but does not exist in the target system, no compare is attempted. However, the summary report following the execution will reflect this condition. Similarly, when the incoming entity and the entity at the target system are identical, no report is produced.

NDVRDCMP Command Syntax

Contents

- JCL (see page 161)
Through the use of an input command file it is possible to limit a compare run to specific entity types and/or vary the output report format. NDVRDCMP will parse the input files for only the type(s) specified.

**SIGNON**

The SIGNON command is fed directly to the IDMSDDDL and other CA utility programs, and is used to identify the dictionary, CA IDMS user and password which will be used to extract entity definitions. Note that this is CA IDMS SIGNON and it specifies a CA IDMS user, not a CA Endevor/DB user. The SIGNON commands processed by all other CA Endevor/DB utilities are CA Endevor/DB SIGNON commands - this implementation is unique to the NDVRDCMP program.

**REPORT**

The REPORT command is optionally used to vary the printed results in the NDVRLST file.

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGES</td>
<td>Only the inserted and deleted lines will display (same as stand-alone option C as described above). This is the default setting.</td>
</tr>
<tr>
<td>ONLY</td>
<td>The contents of the target dictionary for each entity are displayed with the inserted and deleted lines in context (same as the stand-alone option H as described above).</td>
</tr>
</tbody>
</table>

**COMPARE**

The COMPARE command is optionally used to limit the compare to specified entity types.

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPES =</td>
<td>All the entities contained in the input files under dd statement NDVRSRC are compared to ALL the target dictionary. This is the default setting.</td>
</tr>
</tbody>
</table>
Option | Meaning
---|---
TYPES = *(type, ...)* | A single entity type or list of entity types can also be specified. The allowable types are:
  - SCHEMA
  - SUBSCHEMA
  - PROGRAM
  - PROCESS
  - FILE
  - ELEMENT
  - QFILE
  - TABLE
  - MODULE
  - MESSAGE
  - RECORD
  - MAP
For example, when COMPARE TYPES = *(MAP, SUBSCHEMA)*. is coded, only the Maps and Subschema from the input files will be compared to the target dictionary.

**SET TARGET IDMSDDDL SESSION OPTIONS**

⚠️ **Note:** To accurately compare the differences between the source and target dictionaries, this set of options must be identical to the 'SET SOURCE IDMSDDDL SESSION OPTIONS' specified for NDVRDLVR in Section 8.

This command allows changes to the default session options used by CA Endevor/DB for the target dictionary.

CA Endevor/DB sets the following target IDMSDDDL session options:

```
SET SESSIONS OPTIONS
QUOTE IS ' DEFAULT IS ON INPUT 1 THRU 72 OUTPUT 80.
```

These options can be overridden or added to by specifying the SET TARGET IDMSDDDL SESSION OPTIONS command. For example, the command:

```
SET TARGET IDMSDDDL SESSION OPTIONS
REGISTRATION NO OVERRIDE.
```

will cause a second SET SESSION OPTIONS command to be used, which will turn off entity occurrence security.

Refer to the [CA IDMS IDD DDDL Reference section](#) for a full discussion of IDD session options.

**SET TARGET IDMSDDDL RECORD OPTIONS**

⚠️ **Note:** To accurately compare the differences between the source and target dictionaries, this set of options must be identical to the 'SET SOURCE IDMSDDDL RECORD OPTIONS' specified for NDVRDLVR in Section 8.

This command allows changes to the default punch record options specified by CA Endevor/DB when punching record definitions from the target dictionary. The following options are set by default when punching record definitions from the target dictionary.
PUNCH RECORD record-name VERSION version-nr ALSO WITH FILES ELEMENTS SUBORDINATE ELEMENTS.

Note that the 'ALSO WITH' indicates that these options are to be added to the target IDD session options. The punch record options can be overridden by specifying the SET SOURCE IDMSDDDL RECORD OPTIONS command. If you specify any options here, you must specify them all. The option text specified must begin with the 'ALSO WITH' or 'WITHOUT' or 'WITH' text. For example, the command:

SET TARGET IDMSDDDL RECORD OPTIONS ALSO WITH OLQ HEADERS.

includes OLQ headers in the punched record output. With this command, the following punch record command is issued:

PUNCH RECORD record-name VERSION version-nr ALSO WITH OLQ HEADERS.

Refer to the CA IDMS IDD DDDL Reference section for a full discussion of DDDL PUNCH RECORD options.

- **SET TARGET IDMSDDDL ELEMENT OPTIONS**

  **Note:** To accurately compare the differences between the source and target dictionaries, this set of options must be identical to the 'SET SOURCE IDMSDDDL ELEMENT OPTIONS' specified for NDVRDLVR in Section 8.

  This command allows changes to the default options specified by CA Endevor/DB when punching element definitions from the target dictionary. The following options are set by default when punching element definitions from the target dictionary:

  Punch element element-name version version-nr.

  These punch element options can be overridden by specifying the SET TARGET IDMSDDDL ELEMENT OPTIONS command. If you specify any options here, you must specify them all. The options text specified must begin with the 'ALSO WITH' or 'WITH' or 'WITHOUT' text. For example, the command:

  SET TARGET IDMSDDDL ELEMENT OPTIONS ALSO WITH subordinate elements SYNONYMS ATTRIBUTES.

  will include subordinate elements, element synonyms, and attributes in the punched element output. With this command, the following punch element command is issued:

  PUNCH ELEMENT element-name VERSION version-nr ALSO WITH SUBORDINATE ELEMENTS SYNONYMS ATTRIBUTES.

  Refer to the CA IDMS IDD DDDL Reference section for a full discussion of DDDL PUNCH ELEMENT options.

- **SET TARGET SCHEMA OPTIONS**
Note: To accurately compare the differences between the source and target dictionaries, this set of options must be identical to the 'SET SOURCE IDMSDDDL SCHEMA OPTIONS' specified for NDVRDLVR in Section8.

This command allows changes to the default options specified by CA Endevor/DB when punching schema definitions from the target dictionary.
The following options are set by default when punching schema definitions from the target dictionary:

PUNCH SCHEMA schema-id VERSION version-nr.

These punch schema options can be overridden by specifying the SET TARGET SCHEMA OPTIONS command. If you specify any options here, you must specify them all. The options text specified must begin with 'ALSO WITH' or 'WITHOUT' or 'WITH' text. For example, the command:

SET TARGET SCHEMA OPTIONS ALSO WITH HISTORY.

includes all history information for the schema.
With this command, the following punch schema command is issued:

PUNCH SCHEMA schema-id VERSION version-nr ALSO WITH HISTORY.

Refer to the CA IDMS Database Administering section for a full discussion of PUNCH SCHEMA options.

• SET TARGET SUBSCHEMA OPTIONS
To accurately compare the differences between the source and target dictionaries, this set of options must be identical to the 'SET SOURCE IDMSDDDL SUBSCHEMA OPTIONS' specified for NDVRDLVR in Section8.
This command allows changes to the default options specified by CA Endevor/DB when punching subschema definitions from the target dictionary.
The following options are set by default when punching subschema definitions from the target dictionary:

PUNCH SUBSCHEMA subschema-id.

These punch subschema options can be overridden by specifying the SET TARGET SUBSCHEMA OPTIONS command. If you specify any options here, you must specify them all. The options text specified must begin with the 'ALSO WITH' or 'WITHOUT' or 'WITH' text. For example, the command:

SET TARGET SUBSCHEMA OPTIONS ALSO WITH HISTORY.

will include the date and time the subschema was created or last modified.
With this command, the following punch subschema command is issued:

PUNCH SUBSCHEMA subschema-id ALSO WITH HISTORY.

Refer to the CA IDMS Database Administering section for a full discussion of PUNCH SUBSCHEMA options.
JCL

Use the JCL below to run NDVRDCMP. It is contained in member SAMPDCMP on the CA Endevor/DB installation media JCL library:

NDVRDCMP Inputs

The NDVRDCMP input file (NDVRSRC) may either be constructed using NDVRDLVR or built manually.

- If you wish to use the NDVRDLVR files, simply run NDVRDLVR and then specify the NDVRDUPD, NDVRCUPD, NDVRUUPD and NDVRMUPD files as the input to NDVRDCMP.

- If you wish to construct the NDVRSRC file for NDVRDCMP manually, you must use one or more of the following CA IDMS utilities:
  - RHDCMPUT. Used to produce MAP and PANEL source.
  - IDMSCHEM. Used to produce Schema source.
  - IDMSUBSC. Used to produce Subschema source.
  - IDMSDDDL. Used to produce source for all other entity types.

When you run these CA IDMS compilers, you must instruct them to output entity source via PUNCH commands (PROCESS TERSE in the case of RHDCMPUT). Note that NDVRDCMP depends critically on the exact PUNCH command options used by IDMSDDDL, IDMSCHEM and IDMSUBSC.

For RHDCMPUT, use the following commands:

```plaintext
PROCESS=TERSE}mapname VERSION=nnnn
```

For IDMSCHEM, use the following commands:

```plaintext
SET OPTIONS QUOTE IS 'OUTPUT 80 DISPLAY AS SYNTAX VERB ADD.
PUNCH SCHEMA schema VERSION nnnn.
```

For IDMSUBSC, use the following commands:

```plaintext
SET OPTIONS QUOTE IS 'OUTPUT 80 DISPLAY AS SYNTAX VERB ADD.
PUNCH SUBSCHEMA subschema SCHEMA schema VERSION nnnn.
```

For IDMSDDDL, use the following commands:

```plaintext
SET OPTIONS QUOTE IS 'OUTPUT 80 DISPLAY AS SYNTAX VERB REPLACE WITH ALL.
PUNCH PROGRAM program VERSION nnnn ALSO WITH ENTRY POINTS TASKS.
PUNCH TABLE table VERSION nnnn.
PUNCH QFILE qfile VERSION nnnn.
PUNCH MESSAGE message.
PUNCH PROCESS process VERSION nnnn.
```
NDVRDCMP Outputs

Contents

- NDVRDCMP - Input Command Listing (see page 162)
- NDVRDCMP - Entity Comparison Listing (see page 162)
- NDVRDCMP - Index to Entity Listing (see page 163)
- NDVRDCMP - Processing Summary (see page 164)

NDVRDCMP produces a 4-part output report in ddname NDVRLST. The output report comprises:

- An Input Command Listing
- An Entity Comparison Listing
- An Entity Comparison Index
- A Processing Summary

NDVRDCMP - Input Command Listing

The Input Command Listing echoes the user-supplied syntax contained in the NDVRIPT file.

```
volser PAGE DATE TIME
IME nn.n C A - E N D E V O R / D B
34:35 00001 CA, INC. mm/dd/yy 15:

INPUT COMMAND LISTING MIGRATION COMPARISON PROCESSOR
SIGNON USER DBADMIN PASSWORD ??????? DBNAME TGTNDVR.
REPORT CHANGES ONLY.
COMPARE TYPES = (SCHEMA, SUBSCHEMA, PROGRAM, PROCESS, QFILE, TABLE,
MODULE, MESSAGE, FILE, MAP).
```

NDVRDCMP - Entity Comparison Listing

The Entity Comparison Listing displays the results of each unequal entity comparison requested. A display of changes only, or of the entire target dictionary entity with inserted and deleted statements embedded within will be produced. Output formats are varied with the REPORT statement. In this listing the target dictionary is considered file-1 and the incoming source statements comprise file-2. Modified statements show as inserts followed by deletes.

```
volser PAGE DATE TIME
IME nn.n CA, INC. mm/dd/yy 15:
```

NDVRDCMP - Index to Entity Listing

The Index to Entity Listing displays the result of each comparison in alphabetical sequence by entity name. If there is a difference between the source and target entities, a page number will be found in the right-hand column. Detailed comparison results will be found on the page number indicated in the Entity Comparison Listing as shown above. When an entity appears more than once in the input entity file produced by NDVRDLVR (as when an element is contained in more than one migrating record), it may appear more than once in the comparison listing.
TABLE
ADSCSELB
VER 100 SOURCE IS DIFFERENT THAN TARGET

PROGRAM
AUTOCLAIR
VER 100 SOURCE AND TARGET IDENTICAL (NOT LISTED)

MODULE
AUTOUSER-FLD-
HELP
VER 100 SOURCE AND TARGET IDENTICAL (NOT LISTED)

MODULE
AUTOUSER-MAP-
HELP
VER 100 SOURCE AND TARGET IDENTICAL (NOT LISTED)

FILE
CUSTOMER-
VER 1 SOURCE AND TARGET IDENTICAL (NOT LISTED)

MESSAGE
DC681086
VER 1 SOURCE AND TARGET IDENTICAL (NOT LISTED)

PROGRAM
PRANDEM1
VER 1 SOURCE AND TARGET IDENTICAL (NOT LISTED)

FILE
RPTFILE
VER 1 SOURCE AND TARGET IDENTICAL (NOT LISTED)

TABLE
TESTCODE
VER 1 SOURCE IS DIFFERENT THAN TARGET

TABLE
TESTEDIT
VER 1 SOURCE IS DIFFERENT THAN TARGET

MAP
EMPMAPP1
VER 1 NOT AT TARGET, NEW ENTITY COMING FROM SOURCE (NOT LISTED)

MAP
EMPMAPP2
VER 1 NOT AT TARGET, NEW ENTITY COMING FROM SOURCE (NOT LISTED)

SCHEMA
EMPSCHM
VER 100 SOURCE AND TARGET IDENTICAL

SUBSCHEMA
EMPS01
EMPSCHM
VER 100 SOURCE AND TARGET IDENTICAL

MAP
EMPMAP01
VER 2 NOT AT TARGET, NEW ENTITY COMING FROM SOURCE (NOT LISTED)

MAP
EMPMAP02
VER 1 NOT AT TARGET, NEW ENTITY COMING FROM SOURCE (NOT LISTED)

NDVRDCMP - Processing Summary

The Processing Summary displays end-of-job statistics that reflect the results of the comparison run.

<table>
<thead>
<tr>
<th>ENTITY TYPE</th>
<th>PROCESSED</th>
<th>NEW FROM SOURCE</th>
<th>CHANGED AT SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILES</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MAPS</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>MESSAGES</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MODULES</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PROGRAMS</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SCHEMAS</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUBSCHEMAS</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TABLES</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>TOTAL</td>
<td>20</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
The JCL Converter

CA Endevor/DB's JCL Converter converts JCL that executes dictionary update utilities to use CA Endevor/DB's NDVRBOOK program. NDVRBOOK assigns the dictionary updates to a particular user or CCID. Typically, the JCL Converter is employed as a conversion aid to tailor vendor-supplied sets of JCL or old jobstreams used at a shop prior to the installation of CA Endevor/DB. It is a valuable tool when applying maintenance releases of vendor software.

The JCL Converter can convert JCL to execute against a monitored dictionary. It will dynamically determine the type of JCL that it is converting and build the appropriate control statements for that environment. Additionally, it will accept input "SEARCH FOR/REPLACE WITH" commands to replace blocks of JCL statements. Typically, these commands are used to replace JOBLIB or STEPLIB statements to include CA Endevor/DB's load libraries. These "SEARCH FOR/REPLACE WITH" commands can, however, be used against any blocks of statements.

Why JCL Needs To Be Converted

When a dictionary is monitored by CA Endevor/DB, dictionary updates are identified and assigned to users and/or CCIDs. When executing batch dictionary updates, CA Endevor/DB's NDVRBOOK program identifies the CA Endevor/DB user and/or CCIDs to validate and assign the updates. Instead of specifying the dictionary update utility on the EXEC statement in your JCL, specify NDVRBOOK. An input NDVRIPT file contains control statements for NDVRBOOK, specifying the particular user to whom the updates are to be credited and the dictionary update utility to invoke. When executed, NDVRBOOK passes this user identification to CA Endevor/DB's Security System/Change Monitor and then loads and passes control to the dictionary update utility. The subsequent dictionary updates are assigned to the user and/or CCIDs specified to NDVRBOOK.

Use the JCL, following, to run NDVRRJCL. It is contained in member SAMPRJCL on the CA Endevor/DB installation media JCL library:

```plaintext
//JOBNAME    JOB  YOUR.JOBCARD.INFORMATION
//JOBLIB      DD  DISP=SHR,DSN=ndvrdb.loadlib
//*
//*****************************************************************************
//*
//*    JOB:            SAMPRJCL
//*
//*    PURPOSE:    CONVERT JCL TO USE NDVRBOOK.
//*
//*    STEP:          FUNCTION:
//*    =====         =========
//*    GORJCL        CONVERTS JCL TO EXECUTE NDVRBOOK WITH INSTRUCTIONS
//*    TO THE PROGRAM TO EXECUTE AND THE USER TO ASSIGN LOG ENTRIES TO.
//*   *****************************************************************************
//*    GORJCL      EXEC  PGM=NDVRRJCL,REGION=640K
//NDVRJCLI  DD  DISP=SHR,DSN=your.input.jcl.dataset
//NDVRJCL0  DD  DISP=OLD,DSN=your.output.jcl.dataset
//NDVRLST    DD  SYSOUT=* 
//NDVRERR    DD  SYSOUT=* 
```
Security Menu Mask Definitions

Security Class definitions allow the tailoring of menu screens. The following table defines the menu items that correspond to the rows and columns on the Security Class Detail screen. An N in all subfunctions for a category causes the suppression of that category on the main menu screen. Conversely, a Y next to any subfunction causes that category to appear on the main menu screen.

Mask Values

<table>
<thead>
<tr>
<th>Main Menu Category</th>
<th>Subfunction</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Signin/Signout</td>
<td>Browse entities signed out.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Signout entities</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Signin entities</td>
</tr>
<tr>
<td>2</td>
<td>Authorization</td>
<td>Browse preauthorizations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Add preauthorizations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delete preauthorizations</td>
</tr>
<tr>
<td>3</td>
<td>Lock/Unlock</td>
<td>Browse locked users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lock users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unlock users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Browse locked CCIDs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lock CCIDs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unlock CCIDs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Browse locked dictionaries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lock dictionaries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unlock dictionaries</td>
</tr>
<tr>
<td>4</td>
<td>Entity</td>
<td>Browse entity descriptors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Add a new entity descriptor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modify entity descriptors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delete entity descriptors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Browse entity change history</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Browse entity status history</td>
</tr>
<tr>
<td>5</td>
<td>CCID</td>
<td>Browse CCID descriptors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Add a CCID descriptor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modify CCID descriptors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delete CCID descriptors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Browse CCID/change associations</td>
</tr>
</tbody>
</table>

/*
Security Menu Mask Definitions

/*

15-Jan-2018  166/171
<table>
<thead>
<tr>
<th>Main Menu Category</th>
<th>Subfunction</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Add CCID/change associations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify CCID/change associations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete CCID/change associations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse entity status for CCID</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Status</td>
<td>Browse status descriptors</td>
</tr>
<tr>
<td></td>
<td>Add a status descriptor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify status descriptors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete status descriptors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse status/entity associations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add a status/entity association</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify status/entity associations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete status/entity associations</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>User</td>
<td>Browse user descriptors</td>
</tr>
<tr>
<td></td>
<td>Add a user descriptor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify user descriptors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete user descriptors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse user/change associations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add a user/change association</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify user/change associations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete user/change associations</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Dictionary</td>
<td>Browse dictionary descriptors</td>
</tr>
<tr>
<td></td>
<td>Modify dictionary descriptors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete dictionary descriptors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse change log entries</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify change log entries</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete change log entries</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Management Group</td>
<td>Browse management groups</td>
</tr>
<tr>
<td></td>
<td>Add a management group</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify management groups</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete management groups</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse MGRP/CCID associations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add a MGRP/CCID association</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify MGRP/CCID associations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete MGRP/CCID associations</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Control</td>
<td>Browse CCDB descriptor records</td>
</tr>
<tr>
<td></td>
<td>Modify CCDB descriptor records</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse security descriptors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add a security descriptor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify security descriptors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete security descriptors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse monitor dict. stat blocks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify monitor dict. stat blocks</td>
<td></td>
</tr>
</tbody>
</table>
Online/Batch Control Flags

The Y/N flags on the Security Class Detail screen (described in Section 4) control a user's ability to use Batch front end commands and command options. PUNCH mode is the CA Endevor/DB Batch equivalent of Browse actions in the CA Endevor/DB Online front end.

The full breakdown of MENU flags and the Online and Batch functions that they control is as follows:

<table>
<thead>
<tr>
<th>Online Function</th>
<th>Batch Function</th>
<th>Option Command</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Menu</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signin/Signout Functions</td>
<td>Browse Entities Signed Out</td>
<td>1</td>
<td>Signin and Signout</td>
</tr>
<tr>
<td></td>
<td>Signout Entities</td>
<td>2</td>
<td>Signout</td>
</tr>
<tr>
<td></td>
<td>Signin Entities</td>
<td>3</td>
<td>Signin</td>
</tr>
<tr>
<td><strong>Preauthor-ization Functions</strong></td>
<td>Browse Preauthor-izations</td>
<td>1</td>
<td>Add Preauthor-ization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Modify Preauthor-ization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Delete Preauthor-ization</td>
</tr>
<tr>
<td></td>
<td>Add Preauthor-ization</td>
<td>2</td>
<td>Add Preauthor-ization</td>
</tr>
<tr>
<td></td>
<td>Delete Preauthor-izations</td>
<td>3</td>
<td>Delete Preauthor-ization</td>
</tr>
<tr>
<td></td>
<td>Modify Preauthor-izations</td>
<td>4</td>
<td>Modify Preauthor-ization</td>
</tr>
<tr>
<td><strong>Lock Functions</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse Locked Users</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lock Users</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unlock Users</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse Locked CCIDs</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lock CCIDs</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unlock CCIDs</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse Dictionaries</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lock Dictionaries</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unlock Dictionaries</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td><strong>Entity Functions</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse Entity Descriptors</td>
<td>1</td>
<td>Add Entity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Modify Entity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Delete Entity</td>
</tr>
<tr>
<td></td>
<td>Add New Entity Descriptor</td>
<td>2</td>
<td>Add Entity</td>
</tr>
<tr>
<td>Main Menu</td>
<td>Subfunction Menu</td>
<td>Option Command</td>
<td>Mode</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------------------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Modify Entity Descriptors</td>
<td>3</td>
<td>Modify Entity</td>
<td>Process</td>
</tr>
<tr>
<td>Delete Entity Descriptors</td>
<td>4</td>
<td>Delete Entity</td>
<td>Process</td>
</tr>
<tr>
<td>Browse Entity Change History</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browse Entity Status History</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCID Processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCID Processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCID Processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browse CCID Descriptors</td>
<td>1</td>
<td>Add CCID</td>
<td>Punch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modify CCID</td>
<td>Punch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delete CCID</td>
<td>Punch</td>
</tr>
<tr>
<td>Add CCID Descriptor</td>
<td>2</td>
<td>Add CCID</td>
<td>Process</td>
</tr>
<tr>
<td>Modify CCID Descriptors</td>
<td>3</td>
<td>Modify CCID</td>
<td>Process</td>
</tr>
<tr>
<td>Delete CCID Descriptors</td>
<td>4</td>
<td>Delete CCID</td>
<td>Process</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browse CCID/ Change Associations</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add CCID/ Change Association</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modify CCID/ Change Association</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete CCID/ Change Association</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status Processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status Processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status Processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browse Status Descriptors</td>
<td>1</td>
<td>Add Status</td>
<td>Punch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modify Status</td>
<td>Punch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delete Status</td>
<td>Punch</td>
</tr>
<tr>
<td>Add Status Descriptor</td>
<td>2</td>
<td>Add Status</td>
<td>Process</td>
</tr>
<tr>
<td>Modify Status Descriptors</td>
<td>3</td>
<td>Modify Status</td>
<td>Process</td>
</tr>
<tr>
<td>Delete Status Descriptors</td>
<td>4</td>
<td>Delete Status</td>
<td>Process</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browse Status/ Entity Associations</td>
<td>5</td>
<td>Add Entity/ STATUS clause</td>
<td>Punch</td>
</tr>
<tr>
<td>Add Status/ Entity Associations</td>
<td>6</td>
<td>Add Entity/ STATUS clause</td>
<td>Process</td>
</tr>
<tr>
<td>Modify Status/ Entity Associations</td>
<td>7</td>
<td>Modify Entity/ STATUS clause</td>
<td>Process</td>
</tr>
<tr>
<td>Delete Status/ Entity Associations</td>
<td>8</td>
<td>Delete Entity/ STATUS clause</td>
<td>Process</td>
</tr>
</tbody>
</table>

15-Jan-2018  169/171
<table>
<thead>
<tr>
<th>Main Menu</th>
<th>Subfunction Menu</th>
<th>Option Command</th>
<th>Command</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Processing</td>
<td>Browse User Descriptions</td>
<td>1</td>
<td>Add User</td>
<td>Punch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Modify User</td>
<td>Punch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Delete User</td>
<td>Punch</td>
</tr>
<tr>
<td>Add User Description</td>
<td></td>
<td>2</td>
<td>Add User</td>
<td>Process</td>
</tr>
<tr>
<td>Modify User Descriptions</td>
<td></td>
<td>3</td>
<td>Modify User</td>
<td>Process</td>
</tr>
<tr>
<td>Delete User Descriptions</td>
<td></td>
<td>4</td>
<td>Delete User</td>
<td>Process</td>
</tr>
<tr>
<td>Browse User/ Change Associations</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add User/ Change Association</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modify User/ Change Associations</td>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete User/ Change Associations</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dictionary Processing</td>
<td>Browse Dictionary Descriptors</td>
<td>1</td>
<td>Modify Dictionary</td>
<td>Punch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>Modify Dictionary</td>
<td>Process</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>Delete Dictionary Descriptors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse Change Log Entries</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify Change Log Entries</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete Change Log Entries</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management Group Processing</td>
<td>Browse Management Groups</td>
<td>1</td>
<td>Add Management Group</td>
<td>Punch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>Modify Management Group</td>
<td>Punch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>Delete Management Group</td>
<td>Punch</td>
</tr>
<tr>
<td>Add Management Group</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modify Management Groups</td>
<td></td>
<td>5</td>
<td>Modify Management Groups</td>
<td>Process</td>
</tr>
<tr>
<td>Delete Management Groups</td>
<td></td>
<td>6</td>
<td>Delete Management Groups</td>
<td>Process</td>
</tr>
<tr>
<td>Browse Mgrp/ CCID Associations</td>
<td></td>
<td>5</td>
<td>Add Mgmt Grp/ CCID clause</td>
<td>Punch</td>
</tr>
<tr>
<td>Add Mgrp/ CCID Association</td>
<td></td>
<td>6</td>
<td>Add Mgmt Grp/ CCID clause</td>
<td>Process</td>
</tr>
<tr>
<td>Main Menu</td>
<td>Subfunction Menu</td>
<td>Option Command</td>
<td>Mode</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------</td>
<td>----------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify Mgmt Grp/ CCID clause</td>
<td>Proces</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify Mgrp/ CCID Associations</td>
<td>7</td>
<td>Modify Mgmt Grp/ CCID clause</td>
<td>Proces</td>
</tr>
<tr>
<td></td>
<td>Delete Mgrp/ CCID Associations</td>
<td>8</td>
<td>Modify Mgmt Grp/ CCID clause</td>
<td>Proces</td>
</tr>
<tr>
<td>Control Functions</td>
<td>Browse CCDB Descriptor Record</td>
<td>1</td>
<td>- - - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify CCDB Descriptor Record</td>
<td>2</td>
<td>- - - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browse Security Descriptors</td>
<td>3</td>
<td>Add Security Class</td>
<td>Punch</td>
</tr>
<tr>
<td></td>
<td>Modify Security Class</td>
<td>Punch</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete Security Class</td>
<td>Punch</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add Security Descriptor</td>
<td>4</td>
<td>Add Security Class</td>
<td>Proces</td>
</tr>
<tr>
<td></td>
<td>Modify Security Descriptors</td>
<td>5</td>
<td>Modify Security Class</td>
<td>Proces</td>
</tr>
<tr>
<td></td>
<td>Delete Security Descriptors</td>
<td>6</td>
<td>Delete Security Class</td>
<td>Proces</td>
</tr>
<tr>
<td></td>
<td>Browse Monitor Dict Stat Blocks</td>
<td>7</td>
<td>- - - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modify Monitor Dict Stat Blocks</td>
<td>8</td>
<td>- - - -</td>
<td></td>
</tr>
<tr>
<td>CA Endevor/DB</td>
<td>Signon and Return to CA IDMS</td>
<td>1</td>
<td>- - - -</td>
<td></td>
</tr>
<tr>
<td>Signon Functions</td>
<td>Signon and Go To Function Menu</td>
<td>2</td>
<td>Signon</td>
<td></td>
</tr>
</tbody>
</table>