AA-GK49E-TE

VAX CDD/Plus Release Notes

May 1989

This manual contains the release notes for VAX CDD/Plus Version 4.1. It includes the release notes for VAX CDD/Plus Version 4.0.

Operating Systems:	VMS Version 5.0
Software Version:	VAX CDD/Plus Version 4.1

digital equipment corporation
maynard, massachusetts

May 1989

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

ZK5256

This document was prepared using VAX DOCUMENT, Version 1.1

Contents

2 Notes About the Installation Procedure

3 Known Problems and Restrictions in CDD/Plus Utilities

4 Known Problems Using Rdb/VMS with CDD/Plus

5 Hints for Using CDD/Plus

6 Documentation Additions and Corrections

7 Guidelines for Submitting an SPR

Index

Preface

This manual provides information about VAX CDD/Plus Version 4.1 software, also referred to in this book as CDD/Plus. This manual replaces Version 4.0 of *VAX CDD/Plus Release Notes*.

Intended Audience

This manual contains information not included in other manuals in the documentation set. This information is intended for data administrators, system managers, programming supervisors, and programmers.

Operating System Information

Information about the versions of the operating system and related software that are compatible with this version of CDD/Plus is included in the CDD/Plus media kit, in either the *VAX CDD/Plus Installation Guide* or the Before You Install letter.

For information on the compatibility of other software products with this version of CDD/Plus, refer to the System Support Addendum (SSA) that comes with the Software Product Description (SPD). You can use the SPD/SSA to verify which versions of your operating system are compatible with this version of CDD/Plus.

Structure

This manual consists of seven chapters of release notes.

Chapter 1	Explains the new features and bug fixes in CDD/Plus Version 4.1.

Chapter 2	Contains hints and explanations about the CDD/Plus Version 4.1 installation procedure.
Chapter 3	Lists restrictions and known problems in the dictionary utilities.
Chapter 4	Contains release notes related to using Rdb/VMS with CDD/Plus.
Chapter 5	Presents some hints for using CDD/Plus more effectively.
Chapter 6	Includes corrections to the CDD/Plus Version 4.0 documentation.
Chapter 7	Explains what you need to include when you include a software problem.

Related Documents

The other manuals in the VAX CDD/Plus documentation set are:

- *VAX CDD/Plus User's Guide*

  Provides tutorial material for the CDO utility and describes how to coordinate definitions stored in CDO and DMU dictionaries.

- *VAX CDD/Plus Common Dictionary Operator Reference Manual*

  Provides reference material and syntax for all CDO commands.

- *VAX Common Data Dictionary Data Definition Language Reference Manual*

  Describes the VAX Common Data Dictionary Data Definition Language Utility (CDDL), which manipulates definitions in DMU dictionaries.

- *VAX Common Data Dictionary Utilities Reference Manual*

  Describes the Dictionary Management Utility (DMU) and the Dictionary Verify/Fix Utility (CDDV), utilities you use to manipulate DMU dictionaries.

The *VAX CDD/Plus Call Interface Manual* provides reference material for the system administrator on CDO dictionary architecture. This manual can be ordered separately.

Conventions

The special symbols used in this book are:

Symbol	Meaning
CTRL/x	This symbol tells you to press the CTRL (control) key and hold it down while pressing the specified letter key.
KPn	Key names that begin with KP indicate keys on the numeric keypad on the right side of the terminal keyboard.

Symbol	Meaning
SHIFT	The CDO editor uses the PF1 key as a shift key.
SHIFT-KPn	The hyphen in key names means that you press the two keys in the order listed.
RET	This symbol indicates the RETURN key.
BOLD	Bold lettering indicates the definition of a new term.
. . .	Vertical ellipsis in an example means that information not directly related to the example has been omitted.
$	The dollar sign is used to indicate the DCL prompt. This prompt may be different on your system.
Color	Color in examples shows user input.

References to Products

The VAX CDD/Plus documentation to which this document belongs often refers to other Digital products by their abbreviated names.

- VAX ACMS software is referred to as ACMS.
- VAX CDD software—released prior to VAX CDD/Plus—is referred to as CDD.
- VAX CDD/Plus software is referred to as CDD/Plus.
- VAX DATATRIEVE software is referred to as DATATRIEVE.
- VAX Rdb/VMS software is referred to as Rdb/VMS.
- VAX RMS software is referred to as RMS.
- VAX DBMS software is referred to as VAX DBMS.
- VAX TDMS software is referred to as TDMS.
- VAX COBOL software is referred to as VAX COBOL.
- VAX DECreporter software is referred to as DECreporter.
- VAX Language-Sensitive Editor software is referred to as LSE.
- VAX SQL software is referred to as SQL.

Technical Changes and New Features

CDD/Plus Version 4.1 provides bug fixes for CDD/Plus Version 4.0. It also provides the following new functions:

- Enhanced security.
- Protection for dictionary directories.
- Performance improvements.
- The CDO command ENTER to define additional directory entries for an existing dictionary definition. This command allows CDO access to the functions of the CDD$DEFINE_DIRECTORY_ENTRY routine.
- The CDO command REMOVE and the CDD$ERASE_DIRECTORY_ENTRY routine to remove directory entries from an existing dictionary definition.

CDD/Plus Version 4.0 Features

CDD/Plus Version 4.0 is an enhanced version of CDD Version 3.4. CDD/Plus supports dictionary definitions created by CDD Version 3.4 and earlier as well as by the new CDO utility and the CDD/Plus call interface. It provides the following major features:

- An Easy-to-Use Interface

  CDD/Plus provides a single user interface, known as CDO, where you can accomplish all data definition, administration, and management functions for CDO dictionaries. A menu-driven editor allows you to enter common field and record definitions easily. You can also read your DMU dictionary from CDO; CDD/Plus automatically translates the DMU definitions to a form CDO understands.

  CDD/Plus still provides the DMU, CDDL, and CDDV utilities to maintain definitions created by CDD Version 3.4 and earlier. You should use these utilities if you use VAX layered products that do not yet support CDO dictionary features or if you need to write to DMU dictionaries.

- Distributed Dictionary Implementation

  CDO dictionaries can be located on different devices on a single node, on different nodes on a VAXcluster, and on local or wide area networks. The CDO utility allows you to access all of these dictionaries, as well as your system DMU dictionary and subdictionaries, as one logical dictionary.

- Field-Level Data Descriptions

  Field definitions are the smallest fundamental data definitions in CDO dictionaries. You can include these field definitions in many different records and structures. These definitions can be shared by many other dictionary definitions, thereby reducing redundancy of data.

- Relationships

  CDD/Plus connects dictionary definitions automatically when you create definitions through the CDO utility. You can create relationships explicitly through the call interface. These relationships can connect different dictionaries on the same system or on different systems across a network, but cannot link definitions created with CDD Version 3.4 or earlier.

- Pieces Tracking

  CDD/Plus keeps track of all dictionary usage in CDO dictionaries. With CDO commands, you can locate the definitions that would be affected if a particular definition were to be changed.

  You control whether changes to CDO definitions take effect immediately or are incorporated into related definitions over time. If you want the change to take place at once, CDD/Plus automatically changes record definitions that include the field definition. If you want to phase the change in over time, create a new version of the definition. CDD/Plus attaches a notice about the change to CDO definitions but does not automatically change those definitions.

- Data Security and Integrity

  The CDD/Plus protection provisions are consistent with VMS and Rdb/VMS. To maintain integrity, CDD/Plus provides automatic journaling capabilities and commands you can use to verify the dictionary condition.

- Call Interface

  You can make direct calls from user programs to CDD/Plus routines that manipulate dictionaries with CDO format. The call interface is documented in the *VAX CDD/Plus Call Interface Manual*.

1

New Features and Bug Fixes in This Release

This chapter explains the enhancements, new features, and bug fixes in CDD/Plus Version 4.1.

1.1 Performance Enhancements

CDD/Plus Version 4.1 includes numerous improvements in the way memory is managed. A large Rdb/VMS INTEGRATE may take 60% fewer page faults, and other operations should show proportionate improvements.

If you do not see such improvements in INTEGRATE, you should file an SPR stating the problem. See Chapter 7 for the information you need to include with an SPR.

1.2 Security Enhancements

The following sections describe enhancements of security for CDD/Plus Version 4.1.

1.2.1 Restricted Access to Dictionary/Directory System Files

In versions of the CDD before Version 3.4, all metadata (except that created by the DMU CREATE/SUBDIRECTORY command) was kept in a single file identified by the logical name CDD$DICTIONARY. In order to allow users to define metadata in the dictionary hierarchy, this file allowed WORLD WRITE access—an alternative no longer acceptable in security-conscious times.

To enhance security, CDD/Plus Version 4.1 defines a new rights identifier, CDD$SYSTEM, in the VMS rights database. It then uses access control lists (ACLs) to assign access rights to this identifier while disallowing all other access. The privileged image grants and revokes the identifier, so only the dictionary code can access the files in the dictionary/directory system.

CDDV.EXE is also installed as a privileged image to allow it to access secure dictionary files.

You should not assign this identifier to any users. Do not delete it or you will not be able to access any dictionary files.

After you create a dictionary anchor directory, invoke the ACL editor to add an ACL to the directory as follows:

```
(IDENTIFIER=CDD$SYSTEM,ACCESS=READ+WRITE+EXECUTE+DELETE+CONTROL)
(IDENTIFIER=[*,*],ACCESS=READ)
```

You should also add UIC-based protection:

```
(RWED,,,)
```

You can add UIC-based protection either by using the /PROTECTION qualifier on the CREATE/DIRECTORY command or later by a separate SET PROTECTION command.

With these ACLs, only dictionary files can be created in a dictionary anchor directory. You can also modify the protection of any existing dictionary anchor directories to restrict access in this way.

Since CDD/Plus gives itself the identifier CDD$SYSTEM when it creates journal and directory files, you do not need to modify the ACLs of any other files in the anchor directory. The installation procedure will protect the compatibility dictionary and template.

The RDO utility can no longer directly access dictionary databases. You should still be able to use RMU/BACKUP and RMU/RESTORE if you invoke them from an account with system privileges.

VMS utilities can no longer directly access dictionary files unless you invoke them from an account with system privileges.

A local copy of a remote definition does not include the ACL from the remote node; the distributed copy is assigned the default protection. Protecting a local field will not cause it to be protected in a remote definition.

1.2.2 ACLs Supported for Directories

CDD/Plus Version 4.1 supports ACLs for directories. The following CDO commands now allow an option of DIRECTORY:

- DEFINE PROTECTION
- DELETE PROTECTION
- CHANGE PROTECTION
- SHOW PROTECTION
- SHOW PRIVILEGE

To allow directory ACLs to be created, changed, and read through the call interface, the directory information buffer has been modified to allow it to contain an ACL buffer. The format of the modified directory information buffer is shown in Section 6.4.3.

Directory protection works in a manner similar to VMS directory protection:

- SHOW privilege allows you to read the directory and its contents.
- CHANGE privilege allows you to define new objects and new versions, and to delete objects in the directory. You do not need CHANGE privilege to change existing objects in place.
- DELETE privilege allows you to delete the directory if it is empty.

1.3 LMF Support

CDD/Plus now supports DDSLA/LMF availability licensing in the VAX CDD/Plus kit installation.

1.4 ENTER and REMOVE Commands

The generic ENTER and REMOVE commands have been added to CDD/Plus Version 4.1.

ENTER creates additional directory names for a dictionary definition that already has one or more directory entries. The syntax of the ENTER command is explained in Section 6.2.2.

REMOVE removes additional directory names for a dictionary definition that has more than one directory entry. The syntax of the REMOVE command is explained in Section 6.2.3.

1.5 VERIFY Command Provides /COMPRESS Option

The VERIFY command now provides the /COMPRESS option to call RDB$CHANGE_DATABASE to shrink the snapshot file to its original size.

The /COMPRESS option is not a part of /ALL and can be used only by itself. If you specify both /ALL and /COMPRESS, /COMPRESS will not be used.

VERIFY/COMPRESS requires that you must be the only user of the database at the time you issue the command. If you do not have system privilege, CDD/Plus returns a no privilege error.

VERIFY/COMPRESS must also be the first command you enter after starting a CDO session. If it is not, you will get an error indicating there is a conflict with another user.

1.6 Using Remote Dictionaries

1.6.1 CDD$REMOTE Process Is Terminated

If a network error occurs during the remote access, the remote process will now be terminated correctly on the remote system based on system parameters for network processes. See the DECnet documentation for details.

1.6.2 Logical Names and Remote Access

Do not use a logical name such as CDD$COMPATIBILITY to reference a remote dictionary. The results are unpredictable. For example, the pathname NODE::CDD$COMPATIBILITY:FOO may not always return the results you expect. Use NODE::SYS$COMMON:[CDDPLUS]FOO instead.

1.6.3 Proxy Accounts

When you access a data definition that depends on a definition stored in a dictionary on a remote node, CDD/Plus creates a process on the remote node. If the account that is accessing the dictionary has proxy access to the remote system, that proxy account will be used; otherwise, the access takes place under control of the DECnet account.

Remote access works only with a proxy account or a DECnet account that has all the process quotas listed in the *VAX CDD/Plus Installation Guide*. The default DECnet account does *not* have enough process quotas for remote dictionary operations.

1.7 CDD/Plus Enables ASTs

CDD/Plus Version 4.1 requires that ASTs are enabled. If CDD/Plus detects that ASTs are disabled, it will enable them and then restore the previous state before returning to the user.

1.8 Bug Fixes

This section describes bug fixes in CDD/Plus Version 4.1.

1.8.1 Corruption Bug Fixed

In several cases of corrupt dictionaries, a SHOW RECORD command returned an "object not found" error. This bug has been fixed.

If you created an object with a zero-length unstructured attribute, and used that object in another object through a grouping relationship, the second object became unreadable. Most commonly this happened when you used the EDIT FIELD command, created a description but didn't edit it (pressed RET twice), and exited.

If you then created a record using the field and entered a CDO SHOW RECORD command, you would see this error.

The field in question will show a 'value is unprintable' error on a CDO SHOW FIELD for the field description.

Zero-length unstructured attributes are now treated as missing and you are not allowed to create this kind of field.

CDD/Plus will correctly handle existing objects with this problem, so new records created using an existing field will work correctly.

You can read existing corrupt records, though reading such records will be more inefficient than reading a good record. You can also delete existing records, and copy them.

You cannot change a field that is used by a corrupted record. If you try this, a BADSUBOBJ error will be signaled with the name of the corrupted record. To correct the problem, copy the record to a new location, delete the original, and try the change again on the copy.

If the field was owned by multiple records, you will have to repeat this procedure for each record that owns the field.

1.8.2 MISSING Clause in VALID IF Works Correctly

Formerly, a field definition such as the following generated incorrect internal representations of the expression:

```
DEFINE FIELD Y VALID IF Y NOT MISSING.
```

An attempt to show field Y resulted in an invalid metadata buffer error.

This problem has been fixed.

1.8.3 CDO Reporting of Syntax Errors Fixed

Formerly, if you made a syntax error in a CDO command and used a tab character in that command, CDO did not reproduce the line correctly to indicate the error. CDO now reproduces the command line and puts the pointer in the right place.

1.8.4 Errors in CONVERT Command Fixed

Formerly, the scale of missing and initial values was not converted correctly. Now it is.

DMU records containing structures are now handled correctly.

1.8.5 Infinite Loop on Syntax Error Fixed

Formerly, CDO would loop endlessly if you typed the following syntactically incorrect CDO command:

```
CDO> DEFINE FIELD FOO DESCRIPTION="text"...
```

This bug has been fixed.

1.8.6 Bugs Fixed and Restrictions Lifted in DEFINE RMS_DATABASE Command

DEFINE RMS_DATABASE can now use records with VALID IF and MISSING values.

DEFINE RMS_DATABASE can now use keys that are not strings and represents their data type correctly.

DEFINE RMS_DATABASE can now use index keys that are group items.

DEFINE RMS_DATABASE now works correctly when the record's directory name is different from its processing name.

DEFINE RMS_DATABASE allows you to define RMS databases that use records that contain variants. CDD/Plus will determine the correct record size. However, indices cannot be part of the variant structure.

1.8.7 Text Data in SHOW GENERIC Prints Correctly

GENERIC attributes that have a text subtype are now printed correctly by the SHOW GENERIC command.

1.8.8 Directory Corruption Fixed

Sometimes when changes to a large directory were rolled back, the directory was left in an inconsistent state. This problem has been fixed.

1.8.9 Comparison in Within Query Fixed

In programs that invoke the CDD/Plus call interface directly, in a dictionary within query passed to CDD$FETCH_START, queries that specified either a greater than or a less than comparison with attributes whose values were dates or numeric in value were not compared correctly. An example of such a query is:

```
begin
dictionary_within_query V1.0
    within_exp
    query
    and
        eql
            attribute CDD$PROTOCOL_TAG
            literal LONGWORD  0  "2818549"
        and
            geq
                attribute CDD$CREATED_TIME
                literal DATE  "01-MAY-1988 00:00:00.00"
            lss
                attribute CDD$CREATED_TIME
                literal DATE  "01-JUN-1988 00:00:00.00"
    end
eoc
```

This within query requests all entities from the stream that are fields and that were created during the month of May.

This problem did not occur in CDO because CDO does not generate this kind of within buffer.

1.8.10 CDD$_NODICT Message Text Changed

The text of the CDD$_NODICT message has changed to:

```
%CDD-F-NODICT, CDD$DICTIONARY not defined as a system executive mode logical
```

If the logical name CDD$DICTIONARY is not defined as a system executive mode logical name, you will receive this message and you will not be able to access the dictionary. You should never see this message unless you have redefined system logical names.

1.8.11 CDD Version 3.4 Allows System Logical Names in Subdictionary Specifications

In previous versions of CDD, within a DMU subdictionary file specification, you could use only logical names from the LNM$SYSTEM_TABLE table. With the release of CDD/Plus Version 4.0, you can use a logical name from any system logical name table defined in LNM$SYSTEM.

2

Notes About the Installation Procedure

2.1 System and Process Quotas

Before attempting to install CDD/Plus, you should make sure your system has the quotas shown in the *VAX CDD/Plus Installation Guide*. If any quotas are too low, the IVP or your applications may fail with unexplained errors.

After installing CDD/Plus, make sure each process that will be accessing the dictionary has the process quotas shown in *VAX CDD/Plus Installation Guide*. Low quotas can cause applications to run poorly, hang, or fail with unexplained errors. For large applications, the page file quota may need to be made even larger than the minimum shown.

2.2 Upgrading Protocols

CDD/Plus Version 4.1 introduces new protocols that an upgrade process adds to existing dictionaries.

This upgrade process takes place in two separate steps:

1 The IVP procedure automatically upgrades the protocols in the CDD$COMPATIBILITY dictionary.

2 At your convenience, you can upgrade the protocols in the other dictionaries on your system using: CDO> CONVERT/DICTIONARY [anchor-name]. However, until you upgrade the protocols in your dictionaries, you *cannot* use any CDO commands except for VERIFY.

When you use the CONVERT/DICTIONARY command, CDO asks you if you are satisfied with the backup of your dictionary. If you have *not* backed up your dictionary at this time, you should answer NO and proceed to back up your dictionary. If you are satisfied with the backup of your dictionary, you should answer YES.

After CDO successfully upgrades the protocols in your dictionary, it responds with the following success message:

```
%CDO-I-UPGRADE_SUCCEED, dictionary successfully upgrade to new protocols
CDO>
```

If the protocols in a dictionary need to be upgraded, all attempts to use this dictionary (except for the VERIFY command) return the following error message asking you to upgrade your dictionary:

```
-CDD-F-NO_AUTOMATIC_UP, upgrade protocols using CONVERT/DICTIONARY
CDO>
```

If you issue the VERIFY command and then the CONVERT/DICTIONARY command in the same CDO session, CDO *does not* upgrade the protocols in your dictionary, even though it displays the success message.

You can use the VMS BACKUP Utility to back up your dictionary only if *no one else* is using the dictionary. The following command backs up a dictionary to a file:

```
$ BACKUP [anchor.dict1 ]*.* dict_feb1.bck /SAVE_SET
```

The following example shows how you might attempt to upgrade the protocols in your dictionary:

1. Enter CDO.
2. Start to convert your dictionary.
3. If you do *not* have a backup of the dictionary, answer NO at the prompt from the CONVERT command.
4. Exit from CDO and back up the dictionary, using the VMS BACKUP Utility.
5. Enter CDO again and upgrade your dictionary using the CONVERT/DICTIONARY command.

```
$ DICTIONARY OPERATOR

Welcome To CDO V1.1
The CDD/Plus V4.1 User Interface
Type HELP for help

CDO> CONVERT/DICTIONARY [-.DICT1]
are you satisfied with the backup of your dictionary, proceed? [Y/N] (N) N

CDO>EXIT

$ DICTIONARY OPERATOR

Welcome To CDO V1.1
The CDD/Plus V4.1 User Interface
Type HELP for help
```

```
CDO> CONVERT/DICTIONARY [-.DICT1]
are you satisfied with the backup of your dictionary, proceed? [Y/N] (N)Y
%CDO-I-UPGRADE_SUCCEED, dictionary successfully upgrade to new protocols
CDO>
```

In the following example, a user accesses a dictionary that needs to have its protocols upgraded. The DEFINE FIELD command returns an error message stating that the protocols need to be upgraded. Only the VERIFY command does not return an error message, before you upgrade the protocols.

```
$ DICTIONARY OPERATOR

Welcome to CDO V1.1
The CDD/Plus V4.1 User Interface
Type HELP for help

CDO> SET DEF [-.DICT1]
CDO> DEFINE FIELD X.
%CDO-E-ERRDEFINE, error defining object
-CDD-F-NO_AUTOMATIC_UP, upgrade protocols using CONVERT/DICTIONARY
CDO>
```

2.3 Concealed Device Name Errors

If the SYSTEM account UAF entry does not define a default device, only a directory (for example, SYS$LOGIN = "[SYSMGR]"), the CDD/Plus installation fails when it executes the command DEFINE DICTIONARY CDD$TEMPLATE just prior to starting the IVP. The error displayed is:

```
%RDMS-F-NOTSYSCONCEAL, non system concealed device name in file name
```

Rdb/VMS uses the SYS$LOGIN logical name when creating a recovery-unit journal in the context of the CDO command. Rdb/VMS tries to translate SYS$LOGIN_DEVICE and gets an error when the definition does not include a device name.

If this error occurs, change the logical name to include a device name and retry the installation.

2.4 New Text in Installation Procedure

The following text (marked with bars) is additional information that will appear as part of the output of the installation procedure. The installation does not ask any questions about the rights identifier.

```
.
.
.
 ************************************************************

 The CDD/Plus V4.1 Installation Verification Procedure
 (IVP) has been provided and can be run after the installation
 is complete.  It is invoked as follows:
```

```
$  @SYS$COMMON:[SYSTEST.CDD]CDDIVP
*************************************************************
*************************************************************

CDD$SYSTEM has been added to the rights database as a
non-resource, non-dynamic identifier.  All dictionary files
are protected by this identifier.

*************************************************************

.
.
.
```

There may be other changes to the installation procedure text; check the *VAX CDD/Plus Installation Guide* for the text of the current installation procedure.

2.5 Temporary Files Created in Right Location

The installation procedure for CDD/Plus Version 4.0 did not execute correctly if the installer set the process default to the directory SYS$SYSROOT:[SYSTEST.CDD] when using a system configured with a common system disk. The problem occurred when the IVP tried to create a temporary file in the directory SYS$SPECIFIC:[SYSTEST.CDD], which does not exist.

The installation has been changed to make sure the temporary file is created in SYS$COMMON.

2.6 Installing Known Images with SYSPRV

If you use the INSTALL utility to install a known image with the qualifier /PRIVILEGED=(SYSPRV), you must install the associated message file with the qualifier /SHARED. Otherwise, the image cannot access its own message file. For example, if you install DMU with SYSPRV, you should install DMUEXC.EXE, DMU's message file, with the qualifier /SHARED.

2.7 Installation Failures on Old DATATRIEVE Sites

The installation procedure now checks for and deletes old Rdb/VMS dummy files installed by DATATRIEVE Version 3.1 and earlier. Installations should no longer fail with the "Image Ident mismatch" error.

2.8 CDD/Plus with TEAMDATA

If you use TEAMDATA with CDD/Plus, you should run the CDD/Plus startup procedure again after each time you run the TEAMDATA shutdown procedure. The TEAMDATA shutdown procedure shuts down a remote support file, RPC$SHARE.EXE, that CDD/Plus requires to run.

2.9 Installing Compilers with CDD/Plus

When you install a VAX language, such as VAX C V.24-026, on a VMS V5 system, you must have the following files installed and started on the system:

RPC$SHARE.EXE
CDDSHR.EXE
RDBSHR.EXE

If the files listed are not installed and started on the system, running a language compiler causes an "image file not found" error and the installation verification procedure for the compiler fails.

You can start these files by running the CDD startup procedure:

```
$ @SYS$STARTUP:CDDSTRTUP
```

2.10 DESKTOP-VMS Restriction

CDD/Plus Version 4.0 supports DESKTOP-VMS, with the following restriction. The logical names CDD$COMPATIBILITY and CDD$TEMPLATE, which define the location of the CDD/Plus compatibility and template dictionaries, cannot be defined to reside on the logical device SYS$COMMON. Therefore, you cannot choose the default (press RET) for either of the following questions, during the installation of VAX CDD/Plus.

```
* Enter the CDD/Plus root dictionary file's device and directory:
[SYS$COMMON:[CDDPLUS]]

* Enter the CDD/Plus template dictionary file's device and directory:
[SYS$COMMON:[CDD$TEMPLATE]]
```

Instead, respond to these questions with SYS$SYSDEVICE:[CDDPLUS] and SYS$SYSDEVICE:[CDD$TEMPLATE], respectively, or another device and directory of your choice. The following examples illustrate valid answers to these two questions:

```
* Enter the CDD/Plus root dictionary file's device and directory:
[SYS$COMMON:[CDDPLUS]] SYS$SYSDEVICE:[CDDPLUS] RET

* Enter the CDD/Plus template dictionary file's device and directory:
[SYS$COMMON:[CDD$TEMPLATE]] SYS$SYSDEVICE:[CDD$TEMPLATE] RET
```

Or:

```
* Enter the CDD/Plus root dictionary file's device and directory:
[SYS$COMMON:[CDDPLUS]] DUA1:[CDDPLUS] RET

* Enter the CDD/Plus template dictionary file's device and directory:
[SYS$COMMON:[CDD$TEMPLATE]] DUA1:[CDD$TEMPLATE] RET
```

3

Known Problems and Restrictions in CDD/Plus Utilities

This chapter describes restrictions and known problems in the CDD/Plus utilities, including CDO, DMU, CDDL, and the call interface.

3.1 Known Problems and Restrictions in CDO

This section describes known problems and restrictions in the CDO utility, including the call interface.

3.1.1 ENTER Command Does Not Accept FROM GENERIC Clause

The ENTER command returns a syntax error if you enter a command in the form ENTER GENERIC xxx FROM GENERIC yyy. The second GENERIC clause will not be accepted; only RECORD or DATABASE are valid.

3.1.2 ENTER Restriction with RMS Databases

The ENTER FROM DATABASE clause of the ENTER command works only for Rdb/VMS databases. It does not work for RMS databases.

3.1.3 Restriction Verifying a Dictionary Without Privilege

If you use the VERIFY command to verify a dictionary, and you do not have privilege to all dictionary directories, you will receive a NOPRIV error and will not be able to verify those areas of the dictionary to which you do not have access.

Verifying the dictionary requires read access; VERIFY/FIX may also require write access.

3.1.4 Bit-Aligned Fields Aligned on Byte Boundary

If you use the CDO command DEFINE RECORD to specify BIT alignment on a CDO field, the field is aligned on the next byte boundary rather than the next bit boundary.

3.1.5 VAX PASCAL Version 3.4 Required for CDO Support

To include CDO definitions in VAX PASCAL programs, you must have VAX PASCAL Version 3.4 or higher.

3.1.6 DATATRIEVE SHOW Does Not Display Source of CDO Record Definitions

The DATATRIEVE command SHOW displays only record definitions created through the DMU call interface or converted by CDD/Plus Version 4.1 or later. It cannot display the source of a record definition created by CDD/Plus Version 4.0. If you try to display a record definition created by CDD/Plus Version 4.0, you will receive a "Source text for record-name not found in dictionary" error message from DATATRIEVE.

To display the CDO record definition, use the DATATRIEVE command EXTRACT record-name or convert the record using CDD/Plus Version 4.1 or later.

3.1.7 Restrictions for CDO Command CONVERT

The following sections explain restrictions for the CDO command CONVERT.

3.1.7.1 Change of Processing Name on CONVERT If you convert a DMU-format record definition that has a different directory name from its processing name, then the converted record in CDO format will not have its processing name changed.

For example, the following CDDL statement shows a record, YACHTS, that defines a record whose processing name is BOAT:

```
DEFINE RECORD YACHTS.
        01 BOAT
    .
    .
    .
```

If you then convert the record definition using the CDO command CONVERT YACHTS YACHTS_NEW, the directory entry for the new record is YACHTS_NEW, but the processing name is still BOAT:

```
DEFINE RECORD YACHTS_NEW
        01 BOAT
```

In contrast, the following CDDL statement shows a record whose directory name and processing name are the same:

```
DEFINE RECORD YACHTS.
        01 YACHTS
    .
    .
    .
```

3.1.7.2 Converting Fields Defined by a DATATRIEVE COMPUTED BY Expression in a DMU Record Definition When you convert a record definition from DMU format to CDO format, any field that is defined by a DATATRIEVE COMPUTED BY expression is not included in the converted CDO record definition.

3.1.7.3 ACLs for DMU Definitions Are Not Converted When you convert a record definition from DMU format to CDO format, the CDO record receives a default access control list.

3.1.7.4 Converting DMU Records Containing OCCURS...DEPENDING Clause When you convert a DMU record definition that contains an OCCURS...DEPENDING ON clause, the name of the DMU record definition that contains the item referred to in the OCCURS...DEPENDING ON clause is not changed. If you change the name of the DMU record definition that contains the OCCURS...DEPENDING ON clause, the new CDO record cannot be used with DATATRIEVE. This can be remedied by changing the CDO record yourself to have the correct reference.

3.1.7.5 Some DMU Field Attribute Clauses Not Converted Not all DMU field attribute clauses are supported in CDO record definitions:

- A CONDITION NAME clause that contains the EXTERNAL NAME clause is not converted, and no informational message is issued. However, CONDITION NAME clauses that do not contain the EXTERNAL NAME clause are converted.
- The VALID FOR DATATRIEVE IF clause is not converted, although the field definition containing the clause is converted. You receive an informational message telling you that "some DTR attributes could not be converted."

3.1.7.6 Alignment Differences in DMU Record Definitions Containing the ALIGNED Clause

You can convert a DMU record definition that contains an ALIGNED clause; however, in the converted record definition, the alignment of the field definition may be different from the alignment of the field definition within the DMU record definition. Because DMU stores bit offsets instead of alignment, a field definition in a converted CDO record definition will be aligned on a larger boundary if the field happens to fall on the larger boundary.

For example, if the DMU record definition has a field definition that is aligned on a word boundary, the field definition may be aligned on a longword boundary in the CDO record definition if the field falls on a longword boundary.

3.1.7.7 Directories Not Automatically Converted

When you try to convert a DMU-format definition to CDO format, and the DMU-format definition is in a directory that exists only in DMU, the CONVERT command will not convert the directory automatically and will return an error.

The workaround is to create the CDO directory by hand using the DEFINE command before you attempt to convert the other definitions.

The following example illustrates the problem:

```
Welcome to CDO V1.1
The CDD/Plus V4.1 User Interface
Type HELP for help
CDO> set default cdd$top.corporate
CDO> dir
 Directory NAD$DISK:[DICTIONARY]CORPORATE
ADDRESS_RECORD;1                                    RECORD
ALLTYPE_LIST;1                                      RECORD
ARRAY;1                                             RECORD
CREW_LIST;1                                         RECORD
EMPLOYEE_LIST;1                                     RECORD
STOCK_RECORD;1                                      RECORD
TEAM_LIST;1                                         RECORD
VARIANT;1                                           RECORD
VIRTUAL;1                                           RECORD

CDO> convert team_list converted.*
%CDO-E-ERRCOPY, error copying object
-CDD-E-DNF, directory NAD$DISK:[DICTIONARY]CORPORATE not found
```

3.1.8 Record Access Denied on Field Protection Violation

If a record contains a field that you do not have access to, CDD/Plus denies you access to the entire record, rather than returning "no privilege" messages for the individual fields that you do not have access to. For example:

```
CDO> DEFINE RECORD EMPLOYEE.
cont> EMPLOYEE_NAME.
cont> SALARY_CLASS1.
cont> BADGE_NUMBER.
cont> END.

CDO> CHANGE PROTECTION FOR
cont> FIELD SALARY_CLASS1
cont> POSITION 1 ACCESS NOSHOW.

CDO> SHOW RECORD EMPLOYEE
%CDO-E-ERRSHOW, error displaying object
-CDD-E-NOREAD, no privilege to read dev:[directory]NFB$EMBED_NOPRIV.
SALARY_CLASS1;2
```

3.1.9 Specifying Full Path Names to Segments in DEFINE RMS_DATABASE Command

The segment you specify in the SEGMENT clause of the DEFINE RMS_DATABASE command must be a field within the record specified by the RECORD clause. You must specify the full path to the field in the record.

For example, the following command defines a record named EMPLOYEES. The second command defines an RMS_DATABASE named EMPLOYEE_ STORAGE. Note that the SEGMENT clause in the second command includes the full path to the LAST_NAME field.

```
CDO> DEFINE RECORD EMPLOYEES.
cont>   EMPLOYEE_NAME STRUCTURE.
cont>      LAST_NAME.
cont>      FIRST_NAME.
cont>   END EMPLOYEE_NAME STRUCTURE.
cont>   ADDRESS.
cont> END.

CDO> DEFINE RMS_DATABASE EMPLOYEE_STORAGE.
cont>   RECORD EMPLOYEES.
cont>   FILE_DEFINITION
cont>        ORGANIZATION INDEXED.
cont>   KEYS.
cont>   KEY 0
cont>        SEGMENT LAST_NAME IN EMPLOYEE_NAME IN EMPLOYEES.
cont>   END KEYS.
cont> END.
```

3.2 CDO Editor Restrictions

The following sections explain CDO editor restrictions.

3.2.1 CDO Editor Does Not Display Entities Without Processing Names

The CDO editor lists entities that have the same directory name and processing name. If an entity has no processing name, the CDO editor does not list that entity in any of its menus. If the processing name and the directory name differ, the CDO editor displays the entity by its processing name, and when you exit the editor, CDO creates a new entity with the same directory name as the processing name.

3.2.2 CDO Editor Does Not Read the Missing Value Attribute Correctly

The EDIT FIELD command does not read missing values correctly, although it does store them correctly.

As a workaround, you can use the DEFINE FIELD or CHANGE FIELD commands to define or change field definitions that have a missing value.

3.2.3 CDO Editor Does Not Handle Expressions or Arrays

The CDO editor does not handle arrays or any attributes that have expression buffers as values. It will not recreate these attributes if you edit a definition that contains one of these attributes.

3.3 Known Problem in the CDD/Plus Call Interface

The following section explains a known problem in the CDD/Plus call interface.

3.3.1 CDD$FETCH_NEXT Returns an Extra Dot in Path Names of DMU Directories

Directory information buffers returned by CDD$FETCH_NEXT will contain an extra dot in the path name if the name returned is a DMU directory under CDD$TOP.

For example, the following buffer contains an extra dot in the path name between [CDDPLUS] and CDD$EXAMPLES:

```
13 directory_info_dsc V2.0
14    directory_name_list
45       directory_name "SYS$COMMON:[CDDPLUS].CDD$EXAMPLES"
46    end
51    type 2818049
56        size 0
58    dictionary_type "109"
59 eoc
```

3.4 CDDV Is a Privileged Image

CDDV.EXE is now an installed privileged image. This is done because CDDV needs to gain access to secure dictionary files.

3.5 Known Problems and Restrictions in DMU and CDDL

The following sections describe restrictions and problems in the DMU and CDDL utilities.

3.5.1 Wildcard Characters Not Allowed in Logical Names

You cannot use a logical name for a DMU pathname if that logical name includes wildcard characters anywhere in the pathname, including the version. If the logical name includes a wildcard character, you will receive a "node not found" error.

3.5.2 CDDL Supports the VAX Language-Sensitive Editor (LSE)

Version 4.0 of CDDL supports the VAX Language-Sensitive Editor (LSE). If the VAX Language-Sensitive Editor is installed on your system, you can use it to help write, compile, and debug CDDL definitions. The CDDL Language-Sensitive Editor provides templates and menus to walk you through CDDL options and syntax. It is especially useful for users unfamiliar with CDDL.

To invoke the Language-Sensitive Editor, type LSEDIT at the DCL prompt. The following command, for example, creates a file ADDRESS.CDDL and displays a CDDL record definition template to guide you through the process of describing a CDDL record:

```
$ LSEDIT ADDRESS.CDDL
```

When LSE compiles your source definition, it expects a file type of .CDDL. The CDDL compiler now recognizes both file types .CDDL and .DDL.

The /DIAGNOSTICS qualifier with the CDDL command creates a diagnostics file that lists errors occurring during compilation. /DIAGNOSTICS is designed for use from the LSE environment. /DIAGNOSTICS lists errors in a file that has the default name of your definition file and the extension .DIA. The diagnostic file is reserved for use by Digital. LSE uses the diagnostic file to display diagnostic messages and to position the cursor on the line and column where a source error exists.

You cannot use /DIAGNOSTICS with CDDL/RECOMPILE.

For complete information on using LSE, see the *VAX Language-Sensitive Editor User's Guide*.

3.5.3 Moving Subdictionary Files

When you move a DMU subdictionary file from one system to another, make sure that the DMU subdictionary's path name on the new system is the same as it was on the old system. Because both DATATRIEVE and VAX DBMS use full path names by default to locate data definitions, changing the path name of a DMU subdictionary will produce errors when DATATRIEVE or VAX DBMS tries to access definitions in that subdictionary.

Currently, CDD/Plus does not check to ensure that DMU subdictionary path names remain unchanged.

3.5.4 Avoid the CDDL ALIGNED Clause in Template Records

You should not use the ALIGNED clause in template records. When CDDL stores the template record, the position of an aligned field is fixed within the record and is not changed when the record is copied into another record definition. Therefore, the newly created field may not align properly in the new record definition.

DMU records created with the ALIGNED clause using previous versions of CDDL may not have aligned fields properly. CDD Version 3.1 corrected this alignment problem. However, if you recompile the records using the ALIGNED clause, data already stored will no longer match the recompiled data definition.

3.5.5 Copying and Renaming a Dictionary Object Protected by a Password

In some cases, you are granted UPDATE and CONTROL privileges for a DMU object only through a password. In this situation, you cannot copy a version of another object with the same name as the protected object to the directory containing the protected object. This is because the DMU command COPY provides no way to specify the protected object's password.

In a case where you are granted the UPDATE privilege for an object only through a password, you cannot rename another object as a new version of the protected object. The DMU command RENAME provides no way to specify the protected object's password.

3.5.6 Using /STAGE with the DMU RESTORE Command

If you specify the /STAGE qualifier with the DMU command RESTORE, all changes remain in virtual memory until the restoration is completed, which assures that either all or none of the changes are made. Therefore, the size of the portion you back up is limited by the amount of virtual memory. By contrast, /NOSTAGE (the default) frees virtual memory in stages, as each directory is restored.

The DMU commands BACKUP and RESTORE should be used to back up only portions of the DMU dictionary hierarchy. DMU BACKUP will back up only DMU definitions, not CDO definitions. Use the DCL command BACKUP to back up an entire DMU dictionary file or the entire CDD/Plus dictionary system.

3.5.7 Using a Wild Card with the DELETE/SUBDICTIONARY Command

When you use a wild card character with the DELETE/SUBDICTIONARY command, you can delete only 255 or fewer DMU subdictionaries.

3.5.8 DMU DELETE * Aborts When a CDO Definition is Encountered

If you type "DELETE *" at the DMU> prompt, DMU can delete only those definitions stored in DMU dictionaries. If it encounters a CDO format definition while interpreting the wild card path name, DMU aborts.

3.5.9 DMU LIST Does Not Alphabetize Contents of Rdb/VMS Database

When you list the contents of an Rdb/VMS database with the DMU command LIST, the field and relation definitions are not sorted alphabetically.

4

Known Problems Using Rdb/VMS with CDD/Plus

The following sections explain known problems using Rdb/VMS with CDD/Plus.

4.1 Rolling Back Transactions to Avoid Inconsistent Metadata

When you invoke a database with the RDO statement INVOKE...PATHNAME and an error occurs in either CDO or RDO, you *must* roll back the current transaction. If the current transaction is not rolled back, the dictionary and the database may be inconsistent.

4.2 DEFINE RELATION FROM PATHNAME Cannot Rename Record

You cannot rename a record definition when you copy it from the dictionary into an Rdb/VMS database using the RDO statement DEFINE RELATION FROM PATHNAME.

The workaround is to define a CDO record with the name you want for the relation in the Rdb/VMS database, then copy the new CDO record definition into the database.

4.3 Avoid Copying Both Field Definition and Fields Based on That Definition

If you have a CDO field definition that is based on another CDO field definition, do not copy both fields into an Rdb/VMS database. Include only the field definition that is based on the original definition. RDO refuses to define a relation from a CDD/Plus record that includes one or more fields based on other fields in the same record. Such a definition is rejected with the following error message:

```
%CDD-E-BOGLOBAL, global field based on global field in same database,
```

The workaround is to avoid defining CDD/Plus records that include fields based on other fields in the same record; the fields and record should be defined like this:

```
CDO> DEFINE FIELD X DATATYPE .....
CDO> DEFINE FIELD XP BASED ON X.
CDO> DEFINE FIELD Y  BASED ON X.
CDO> DEFINE RECORD R.
CDO>   XP.
CDO>   Y.
CDO> END.
```

This will cause XP and Y to be defined as global fields in the Rdb/VMS database; field X will not appear in the data base.

4.4 Assigning Segmented String Attributes to Avoid New Versions

When you use CDO to define a field with a data type of SEGMENTED STRING and plan to use the definition in an Rdb/VMS database, assign the field a SEGMENT_LENGTH and SEGMENT_TYPE value. Otherwise, Rdb/VMS will assign the default SEGMENT_LENGTH and SEGMENT_TYPE values when you copy the field into the database. Later, when you use the RDO statement INTEGRATE to copy your database definitions into the dictionary, new versions of these segmented string fields containing the default Rdb/VMS SEGMENT_LENGTH and SEGMENT_TYPE values will be created in the dictionary.

4.5 Restriction on Asterisk (*) Wild Card Character in SHOW Command

You can use the asterisk (*) wild card character for the field name or record name when you display fields or records in an Rdb/VMS database definition. For example:

```
CDO> SHOW FIELD * FROM DATABASE PARTS

CDO> SHOW RECORD * FROM DATABASE PARTS
```

However, the asterisk wild card character must replace the entire field name or record name. You cannot specify part of the name and use the asterisk to identify the rest. The following commands receive error messages:

```
CDO> SHOW FIELD A* FROM DATABASE PARTS
%CDO-E-ERRSHOW, error displaying object
-CDO-E-NOTFOUND, entity A* not found in dictionary

CDO> SHOW RECORD EMP_* FROM DATABASE PARTS
%CDO-E-ERRSHOW, error displaying object
-CDO-E-NOTFOUND, entity EMP_* not found in dictionary
```

4.6 Displaying Indices or Constraints

You can use the CDO command SHOW GENERIC to display indices or constraints in an Rdb/VMS database definition. The syntax for the command is:

```
SHOW GENERIC protocol-name entity-name FROM DATABASE name
```

For example, you can view the index RDB$REL_REL_ID_NDX using this command:

```
CDO> SHOW GENERIC CDD$INDEX RDB$REL_REL_ID_NDX
cont> FROM DATABASE MY_RDB_DB
Definition of RDB$REL_REL_ID_NDX    (Type : CDD$INDEX)
|    CDD$UNIQUE_INDEX                 1
|    Contains CDD$INDEX_SEGMENT
|    |    *** name is unspecified ***      (Type : CDD$DATA_VALUE)
CDO>
```

5

Hints for Using CDD/Plus

This chapter contains hints for using CDD/Plus and for diagnosing some common problems.

5.1 Compatibility of CDD/Plus Version 4.1 and Version 4.0

If you have two physical dictionaries on different nodes and upgrade node A to run Version 4.1 but not node B, then:

- Using definitions on a Version 4.0 node within a structure defined on node A using Version 4.1 will not trigger a protocol upgrade. Upgrading of protocols is done on the local node.
- The two dictionaries should work together because CDD/Plus reads each object relative to the protocol on its system. You should be able to use definitions on either node in creating new structures on the other node.
- You can perform operations that use existing structures. However, you won't get new features, like directory protection checking.

5.2 Assigning Resource Identifiers to Prevent Disk Quota Errors

A problem can occur if there are quotas set on the disk where the compatibility dictionary or any CDO dictionary resides when CDD/Plus users do not have

disk quotas enabled for them on the disk. When the users without disk quotas try to define something in the compatibility dictionary, they receive the following message:

```
CDO> DEFINE FIELD LAST_NAME
cont> DATATYPE IS TEXT
cont> SIZE IS 20.
%CDO-E-ERRDEFINE, An error occurred during the DEFINE command.
%CDD-F-NOJNLCRE, CDD/Plus was unable to create the journal file in
the given anchor
-RMS-E-OCRE, ACP FILE CREATE FAILED
```

To permit CDD/Plus users to access their dictionaries, the system manager should assign resource identifiers to control access to the CDD/Plus compatibility dictionary's VMS directory. Follow these steps:

1 Define a rights identifier called CDD_USER.

```
UAF> ADD/ID CDD_USER/ATTR=RESOURCE
```

2 Grant the CDD_USER identifier whatever quota seems reasonable on the compatibility dictionary's disk. It should be generous, so that no problems with running out of journal file space will occur.

3 Create the CDD/Plus compatibility dictionary's anchor with the CDD_USER identifier as the owner. For example, if your compatibility dictionary's anchor is SYS$SYSROOT:[000000]CDDPLUS.DIR, execute the following command:

```
$ SET FILE/OWNER=CDD_USER SYS$SYSROOT:[000000]CDDPLUS.DIR
```

4 Grant the CDD_USER identifier with the resource attribute to all users of the dictionary.

```
UAF> GRANT/ID CDD_USER/ATTR=RESOURCE user-name
```

This procedure means that all the space allocated to the dictionary files in the compatibility dictionary is owned by the CDD_USER identifier. Therefore, no individual user needs quotas on the system disk. The only people who can use the dictionary are users with the CDD_USER identifier and users with quotas explicitly set for them on the compatibility dictionary's disk.

For more information about setting disk quotas, see the *Guide to VAX/VMS System Security*.

5.3 Avoiding Locking Problems

Locking problems mayu occur when multiple database users write to a dictionary. For example, when a user updates metadata using RDO, the database indices that the user accesses are locked against other users.

Locking problems may also occur when users access multiple Rdb/VMS databases maintained in the same dictionary location. To avoid these locking problems:

- Place databases in separate dictionaries whenever possible
- In RDO, use the COMMIT or ROLLBACK statement as often as possible
- Perform large updates, such as with the INTEGRATE and RESTORE statements, as after-work batch jobs

5.4 Using RMU/BACKUP and RMU/RESTORE on CDO Dictionaries

If you used RMU/BACKUP and then RMU/RESTORE on a dictionary, you can restore your dictionary directory using VERIFY/REBUILD_DIRECTORY even if the dictionary directory files no longer exist in the anchor directory. Note that RMU/BACKUP only backs up the CDO part of the dictionary, not the DMU part of the dictionary.

The security enhancements made to CDD/Plus Version 4.1 mean that RMU must be invoked from an account with system privileges. Otherwise, RMU will not be allowed to access the dictionary files.

If users are constantly accessing the dictionary and you cannot back up the dictionary using the VAX/VMS BACKUP utility, you can use the RMU/BACKUP command. You can also use RMU/BACKUP to take advantage of RMU backup compression. The following example backs up the CDD$DATABASE in the anchor directory DIC$ANCHOR_DIR to a file called DICTIONARY:

```
$ RMU/BACKUP DIC$ANCHOR_DIR:CDD$DATABASE DICTIONARY
```

Make sure the directory where you want to restore your dictionary database is empty, then restore the dictionary database:

```
$ RMU/RESTORE/NOCDD DICTIONARY
```

After restoring the dictionary, you must use VERIFY/REBUILD_DIRECTORY in CDO to recreate the dictionary's directory system. For example:

```
$ DICTIONARY OPERATOR VERIFY/REBUILD_DIRECTORY DIC$ANCHOR_DIR
```

If you changed the location of the dictionary, during the backup you must also use VERIFY/LOCATION:

```
$ DICTIONARY OPERATOR VERIFY/LOCATION DIC$ANCHOR_DIR
```

6

Documentation Additions and Corrections

This chapter describes documentation errors and provides additional corrected text for new functions.

6.1 Corrections to the Online HELP

6.1.1 Documentation of Error Messages

The CDO and CDD/Plus error messages are included in the online help.

In addition, the CDO error messages are included in the file SYS$HELP:CDDPLUS_MSG.DOC. The CDD/Plus error messages can be found in the file SYS$HELP:CDO_MSG.DOC.

6.2 Additions to the *VAX CDD/Plus Common Dictionary Operator Reference Manual*

The following sections describe additions to the *VAX CDD/Plus Common Dictionary Operator Reference Manual.*

6.2.1 DIRECTORY Option on Protection Commands

The following commands now allow an option of DIRECTORY:

SHOW PROTECTION FOR DIRECTORY ...
DEFINE PROTECTION FOR DIRECTORY...
CHANGE PROTECTION FOR DIRECTORY...
DELETE PROTECTION FOR DIRECTORY...
SHOW PRIVILEGE FOR DIRECTORY...

6.2.2 ENTER Command

The CDO ENTER command creates a directory entry for GENERIC entities and specifies the directory name to be created. You can also use the ENTER command to create additional directory names for objects that already have a directory entry.

The directory name can identify a definition in a dictionary on another node; if it does, a local copy will be maintained.

Syntax

The syntax of the ENTER command is:

```
ENTER { FIELD                 |
        RECORD                |    name1 { from-clause |
        GENERIC protocol_name }          for-clause   }

from-clause ::= FROM {   RECORD             |
                       DATABASE             |
                       GENERIC protocol_name } name2
for-clause  ::= FOR name3
```

Parameters

protocol_name

The name of the protocol on which the entity is based.

name1

The processing name of the entity you are entering in the directory.

name2

The name of the entity that owns name1.

name3

The name to be created in the directory.

protocol-name

The name of the protocol on which the entity is based.

Usage Notes

The name-clause is required. Either the from-clause or for-clause must be used to form a legal command but both cannot be used in the same command.

The from-clause creates a directory name for an element that is a member of a relationship, for example, a record in a field.

The for-clause creates a directory name for an element that already has a directory name, for example, giving a record a new name on a remote node.

The from-clause behaves as it did for CDD/Plus Version 4.0; the for-clause enters name1 (a directory name) in the directory as an alternate name for name3 (a directory name).

6.2.3 REMOVE Command

The CDO REMOVE command removes a directory name from the directory.

Syntax

The syntax of the REMOVE command is:

```
REMOVE { FIELD | RECORD | GENERIC protocol_name }
       directory_name [ ,directory_name ]...
```

Parameters

protocol-name

The name of the protocol on which the entity is based.

directory_name

The name of the entity you want to remove from the directory.

Usage Notes

If the name specified by directory_name is the only directory entry for the entity, and the entity does not have any relationships to other existing objects, the directory entry will not be removed and an error will be issued.

6.2.4 Enhanced Syntax for Expressions

Expressions have been enhanced to allow specification of array subscripts in a field segment and more complex comparisons of strings.

You can request case-sensitive comparisons of alphanumeric strings:

```
[CASE_SENSITIVE] relational_operator
```

The relational_operator parameter can be one of the following:

=
>\
GE
<
LE
<>
MATCHING
BETWEEN
STARTING WITH
CONTAINING

The following options can be used where MISSING is valid in expressions:

```
field_expression ALPHABETIC
field_expression LOWERCASE_ALPHABETIC
field_expression UPPERCASE_ALPHABETIC
field_expression EMPTY_FIELD
field_expression FULL_FIELD
field_expression NUMERIC
```

6.2.5 Enhancements to Edit Strings

The following new field attributes can be used in the CDO DEFINE, CHANGE and EDIT FIELD commands wherever a field attribute is valid:

```
[language] EDIT_STRING [IS] edit_string
language ::= { COBOL | DTR | PLI | RPG | [no]FORMS }

[language] INPUT_EDIT_STRING [IS] edit_string
language ::= { [no]FORMS }

          The CDD$INPUT_EDIT_STRING attribute is created if no
          language is specified. The  CDD$INPUT_EDIT_STRING_FORMS
          attribute is created if FORMS is specified.

DISPLAY_SCALE [IS] n

          n is a signed integer and indicates the number of places
          to shift the decimal point when the field is displayed. Negative
          n indicates a shift of n places to the left, and positive n
          indicates a shift of n places to the right.

DECIMAL_POINT IS quoted-string

          quoted-string specifies the character(s) to be displayed
          as the decimal point.

CURRENCY_SIGN IS quoted-string

          quoted-string specifies the character(s) to be displayed
          as the currency sign.

JUSTIFIED { RIGHT | LEFT | CENTER | DECIMAL }

HELP_TEXT IS quoted-string

          quoted-string specifies the help text to be associated
          with the field.

DATATYPE [IS] { ALPHABETIC | TEXT | VARYING_STRING }
                [SIZE IS] numeric-literal [CHARACTERS]
              { LOWERCASE | UPPERCASE | CASE_INSENSITIVE }

          LOWERCASE, UPPERCASE and CASE_INSENSITIVE specify the
          alphabetic case to be stored in the field.  It can only be
          associated with the ALPHABETIC, TEXT, and VARYING STRING
          data types.

INPUT_VALUE { REQUIRED | OPTIONAL }
```

6.3 Corrections to the *VAX CDD/Plus User's Guide*

6.3.1 VALID IF Does Not Have an Underscore

On page 4–5, Table 4–1 lists the VALID_IF attribute. This should read VALID IF (no underscore).

The error occurs again on page 2–8, Table 2–2.

6.3.2 DEFINE DATABASE Examples Show Wrong Syntax

In Section 7.2.3 on defining a new database, the following examples are given:

```
RDO> DEFINE DATABASE DEPT1
cont>  IN 'CDD$TOP.PERSONNEL'.
RDO> DEFINE DATABASE DEPT2
cont>  IN 'CDD$TOP.PERSONNEL'.
```

The commands should read as follows:

```
RDO> DEFINE DATABASE DEPT1
cont>  IN 'CDD$TOP.PERSONNEL.DEPT1'.
RDO> DEFINE DATABASE DEPT2
cont>  IN 'CDD$TOP.PERSONNEL.DEPT2'.
```

6.3.3 Copying Database Definitions

In Section 2.4 of the *VAX CDD/Plus User's Guide*, the word "currently" should be removed from the second sentence after the Note. The sentence should read:

You cannot use CONVERT to copy database definitions.

6.3.4 Reference to Nonexistent Compiler

Page 7–24 in the chapter on using VAX Rdb/VMS with VAX CDD/Plus refers to a precompiler called RDML/ADA, which does not exist.

6.3.5 INTEGRATE Statement Implies START_TRANSACTION

The first line, as follows, should be removed from both examples on page 7–26:

```
RDO> START_TRANSACTION READ_WRITE
```

The last line in the CAUTION box and the last line on page 7–26 should also be removed. The INTEGRATE statement implies both an INVOKE DATABASE and a START_TRANSACTION statement; the INTEGRATE FROM will fail if preceded by a START_TRANSACTION statement.

6.3.6 Using VAX SQL

The following information for VAX SQL users should be added to Chapter 7, Section 7.3.5, of the *VAX CDD/Plus User's Guide*. The following text should be inserted after the paragraph that ends, "For more information on RDML, see the *RDML Reference Manual* and the *VAX Rdb/VMS Guide to Programming*":

With VAX SQL, users can define, update, and query relational databases. VAX SQL provides the following environments for issuing SQL statements:

- An interactive SQL utility
- A precompiler that lets users embed SQL statements in Ada, FORTRAN, PL/I, COBOL, or C programs
- SQL module language modules containing SQL statements that any language can call
- A dynamic SQL interface that processes SQL statements generated when the program runs

VAX SQL uses a single precompiler, SQL$PRE, to preprocess programs written in Ada, C, COBOL, FORTRAN, and PL/I. The precompiler lets you embed DML and DDL statements in programs that will access Rdb/VMS databases.

You can define symbols to invoke the precompiler for your host language:

```
$ SADA :== $SYS$SYSTEM:SQL$PRE/ADA
$ SCC  :== $SYS$SYSTEM:SQL$PRE/CC
$ SCOB :== $SYS$SYSTEM:SQL$PRE/COBOL
$ SFOR :== $SYS$SYSTEM:SQL$PRE/FORTRAN
$ SPLI :== $SYS$SYSTEM:SQL$PRE/PLI
```

You can invoke the precompiler in one of two ways:

- Enter the input file name on the same line as the invoke command.
- Invoke the precompiler without an input file name and wait for the INPUT prompt to enter the input file name.

For example:

```
$ SADA PROGRAMC
```

6.4 Corrections and Additions to the *VAX CDD/Plus Call Interface Manual*

The following sections describe corrections and additions to the *VAX CDD/Plus Call Interface Manual*.

6.4.1 Directory Name Required When Requesting Information from CDD$FETCH_START

When you call the CDD/Plus entry point CDD$FETCH_START and pass a directory information buffer as a template for output information, the buffer must contain a directory name tag. The syntax for the directory information buffer shows the directory name tag to be optional, but in this situation, the tag is required. Specify a length of zero:

```
13 directory_info_dsc V2.0
14    directory_name_list
45       directory_name 0
46    end
59 eoc
```

6.4.2 Comment Line Missing in RETRIEVE_RECORD.PAS

The example program in Section 3.4 is missing a line in the comment block at the top of page 3–28. This comment line described the EOC tag for the directory name buffer embedded in the metadata buffer for the CDD$GET_ELEMENT call.

The code itself is correct; only the comment was wrong. The comment block in question should look like this:

```
    .
    .
    .
{ We are including a directory buffer because we want this information     }
{ returned; therefore, we are not including a name in the template buffer. }
{ However, the buffer syntax does not give us the option of including the  }
{ CDD$K_DIRECTORY_NAME tag without including a name so we specify a name   }
{ with a length of zero on the input buffer.                               }
{                                .                                         }
{                                       CDD$K_DIRECTORY_NAME        | byte  }
{                                           length                  | word  }
{                               CDD$K_END                           | byte  }
{ Missing line here with        CDD$K_EOC                           | byte  }
{                               CDD$K_ATTRIBUTE_LIST                | byte  }
{                                   CDD$K_ATTRIBUTE                 | byte  }
{                                       CDD$K_ATT_PROCESSING_NAME   | longword }
{                               CDD$K_END                           | byte  }
{                       CDD$K_END                                   | byte  }
{               CDD$K_EOC                                           | byte  }
    .
    .
    .
```

6.4.3 Directory Information Buffer Accepts ACL Buffer

The directory information buffer is a text descriptor containing the name of the directory node whose ACL is to be modified. It can now also optionally contain an ACL buffer defining the ACL to be associated with the new directory node.

The directory information buffer has the following format; a vertical bar character in the left margin marks the new syntax. See the *VAX CDD/Plus Call Interface Manual* for the complete description of this descriptor and the ACL buffer.

```
<directory_info> ::= <block_header>
                     [: <directory_name_list>
                        <ent_type>
                        <size>
                        <protocol_name>
                        <dictionary_type>
                        <physical_access>
 |                      <directory_protection> :]
                        <block_terminator>

<directory_protection> ::=
               { CDD$K_PROTECTION <longword_length> <acl_buf_dsc> |
                 CDD$K_NOPROTECTION }
```

The CDD$K_PROTECTION clause defines directory protection on a call to CDD$DEFINE_DIRECTORY and changes the directory protection on a call to CDD$CHANGE_DIRECTORY.

CDD$K_PROTECTION 0 requests that directory protection information be returned from a call to CDD$FETCH_START.

CDD$K_NOPROTECTION deletes directory protection on a call to CDD$CHANGE_DIRECTORY. In a buffer returned from CDD$FETCH_START, it means that no protection exists for the entity.

The CDD$CHANGE_DIRECTORY routine is new for CDD/Plus Version 4.1. Its syntax and parameters are explained in Section 6.4.7.

6.4.4 Modified Expression Buffer

The expression buffer has been enhanced to allow specification of array subscripts in a field segment and more complex comparisons of strings. The modified buffer has a major version number of 1 and a minor version number of 1.

The buffer has the following format. Only items changed for CDD/Plus Version 4.1 are explained here; the rest of the buffer is explained in the *VAX CDD/Plus Call Interface Manual*. The new syntax is marked by a vertical bar character at the left margin. (Vertical bars on the right are part of the syntax notation, which is fully explained in the *VAX CDD/Plus Call Interface Manual*.)

```
<value_expression> ::=  <arithmetic_expression>  |
                        <dbkey_expression>       |
|                       <field_expression>       |
                        <from_expression>        |
                        <function_expression>    |
                        <literal_expression>     |
                        <statistical_expression> |
                        <string_expression>      |
                        <via_expression>         |
                        <aggregate_function>     |
                        <group_value>            |
                        <via_table_expression>   |
|                       <variable_expression>

<field_expression> ::=
     CDD$K_EXP_ELEMENT_NAME [<context_variable>] <field_name> |
     CDD$K_EXP_ELEMENT_ID [<context_variable>] <field_id>

  <context_variable> ::=
     CDD$K_EXP_CONTEXT <length> DEC Multinational Character Set string
  <field_name>       ::=
     CDD$K_EXP_FIELD <field_segment> [<field_segment>]...
| <field_segment>    ::=
|    CDD$K_EXP_FIELD_SEGMENT <length> DEC Multinational Character Set string
|        [ <array_specification> ]
|
| <array_specification> ::=
|    CDD$K_EXP_ARRAY_SUBSCRIPT_LIST
|        { CDD$K_EXP_ARRAY_SUBSCRIPT unsigned word } ...

<variable_expression> ::=
     CDD$K_EXP_VARIABLE <variable_id>

  <variable_id>         ::= unsigned word

<expression_buffer> ::=  <value_expression>             |
|                        <boolean_expression>           |
                         <RSE>                          |
                         <conditional_value_expression> |
                         <table>

| <boolean_expression> ::=
|    [ <case_sensitive> ] <relational_expression> |
|    <literal_expression>    |
|    <logical_expression>    |
|    <rse_expression>

| <case_sensitive> ::=
|    CDD$K_EXP_CASE_SENSITIVE
```

```
<relational_expression> ::=
    CDD$K_EXP_COT      <value_expression> <value_expression> |
    CDD$K_EXP_EQL      <value_expression> <value_expression> |
    CDD$K_EXP_GTR      <value_expression> <value_expression> |
    CDD$K_EXP_GEQ      <value_expression> <value_expression> |
    CDD$K_EXP_INT      <field_expression> <table_name>       |
    CDD$K_EXP_LSS      <value_expression> <value_expression> |
    CDD$K_EXP_LEQ      <value_expression> <value_expression> |
    CDD$K_EXP_MATCHES  <value_expression> <value_expression> |
    CDD$K_EXP_NEQ      <value_expression> <value_expression> |
    CDD$K_EXP_BETWEEN  <value_expression> <value_expression>
                                          <value_expression>      |
    CDD$K_EXP_STW      <value_expression> <value_expression>

| <logical_expression> ::=
|    CDD$K_EXP_ALPHABETIC <field_expression>                      |
|    CDD$K_EXP_ALPHABETIC_LOWER <field_expression>                |
|    CDD$K_EXP_ALPHABETIC_UPPER <field_expression>                |
|    CDD$K_EXP_AND <boolean_expression> <boolean_expression>      |
|    CDD$K_EXP_EMPTY_FIELD <field_expression>                     |
|    CDD$K_EXP_FULL_FIELD <field_expression>                      |
|    CDD$K_EXP_MIS <value_expression>                             |
|    CDD$K_EXP_NOT <boolean_expression>                           |
|    CDD$K_EXP_NUMERIC <field_expression>                         |
|    CDD$K_EXP_OR  <boolean_expression> <boolean_expression>      |
|    CDD$K_EXP_XOR <value_expression> <value_expression>
```

The syntax elements valid in an expression buffer are explained in the order in which they appear in the syntax diagram. Only items new for CDD/Plus Version 4.1 are explained.

- CDD$K_EXP_ARRAY_SUBSCRIPT_LIST

  This tag is used to signify that the subsequent information will be a list of array subscripts for the field whose description it follows.

- CDD$K_EXP_ARRAY_SUBSCRIPT

  This tag specifies that the word value following it defines one subscript for an array. One of these tags is used to define the subscript for each dimension of an array.

- CDD$K_EXP_CASE_SENSITIVE

  This tag is used to determine whether relational comparisons are to be case-sensitive or case-insensitive. If the tag is present, the comparisons are case-sensitive.

- CDD$K_EXP_ALPHABETIC

  This tag is used in a logical expression to determine if a given field's value is entirely alphabetic.

- CDD$K_EXP_ALPHABETIC_LOWER

This tag is used in a logical expression to determine if a given field's value is entirely lower case alphabetic.

- CDD$K_EXP_ALPHABETIC_UPPER

  This tag is used in a logical expression to determine if a given field's value is entirely uppercase alphabetic.

- CDD$K_EXP_EMPTY_FIELD

  This tag is used in a logical expression to determine if a given field's value has not been entered.

- CDD$K_EXP_FULL_FIELD

  This tag is used in a logical expression to determine if a given field's value fills the field's output size.

- CDD$K_EXP_NUMERIC

  This tag is used in a logical expression to determine if a given field's value is numeric.

6.4.5 Modified Edit String Buffer

The block header of the edit string buffer has a type of CDD$K_EXPRESSION_BUF_DSC. The major version number is 1 and the minor version number is 1. The tags are each one word in length. Data type values (DSC$K_DTYPE_x) are stored as a word, not a byte. Revisions are marked by a vertical bar on the left. (Vertical bars on the right are part of the syntax notation, which is fully explained in the *VAX CDD/Plus Call Interface Manual.*)

Format:

```
<expression> ::=
    <block_header>
        <expression_buff>
    <block_terminator>
<expression_buff> ::=
    <value_expr>              |
    <boolean_expr>            |
    <RSE>      |
    <conditional_value_expr> |
    <table>

<table> ::=
    CDD$K_EXP_TABLE
        { <value_expr> | <boolean_expr> } ...
    CDD$K_EXP_END
```

```
<value_expression> ::=
    <arithmetic_expression>  |
    <dbkey_expression>       |
    <field_expression>       |
    <from_expression>        |
    <function_expression>    |
    <literal_expression>     |
    <statistical_expression> |
    <string_expression>      |
    <via_expression>         |
    <aggregate_function>     |
    <group_value>            |
    <via_table_expression>   |
    <variable_expression>

<arithmetic_expression> ::=
    CDD$K_EXP_ADD <value_expression> <value_expression> |
    CDD$K_EXP_ASL <value_expression> <value_expression> |
    CDD$K_EXP_ASR <value_expression> <value_expression> |
    CDD$K_EXP_ONES_CMP <value_expression>               |
    CDD$K_EXP_DIV <value_expression> <value_expression> |
    CDD$K_EXP_MUL <value_expression> <value_expression> |
    CDD$K_EXP_NEG <value_expression>                    |
    CDD$K_EXP_SUB <value_expression> <value_expression>

<dbkey_expression> ::=
    CDD$K_EXP_DBKEY <length> <context_variable>

<length> ::= unsigned word

<field_expression> ::=
    CDD$K_EXP_ELEMENT_NAME [<context_variable>] <field_name> |
    CDD$K_EXP_ELEMENT_ID [<context_variable>] <field_id>

<context_variable> ::=
    CDD$K_EXP_CONTEXT <length> DEC MCS string
<field_name> ::=
    CDD$K_EXP_FIELD <field_segment> [<field_segment>]...

| <field_segment> ::=
|    CDD$K_EXP_FIELD_SEGMENT <length> DEC MCS string
|        [ <array_specification> ]
|
| <array_specification> ::=
|    CDD$K_EXP_ARRAY_SUBSCRIPT_LIST
|        { CDD$K_EXP_ARRAY_SUBSCRIPT unsigned word } ...

<field_id> ::= unsigned word
<from_expression> ::=
    CDD$K_EXP_FROM <RSE> <value_expression>
```

```
<function_expression> ::=
    CDD$K_EXP_ABS <value_expression>                        |
    CDD$K_EXP_EXP <value_expression> <value_expression>     |
    CDD$K_EXP_FAC <value_expression>                        |
    CDD$K_EXP_MOD <value_expression> <value_expression>     |
    CDD$K_EXP_RND <value_expression>                        |
    CDD$K_EXP_SGN <value_expression>                        |
    CDD$K_EXP_SQRT <value_expression>                       |
    <user_function_call>
<user_function_call> ::=
    CDD$K_EXP_FUNCTION [<file_name>]
         <function_name>
         <parameter_list>
         CDD$K_EXP_END

<file_name> ::=
    CDD$K_EXP_FUNCTION_FILE <length> DEC MCS string
<function_name> ::=
    CDD$K_EXP_FUNCTION_NAME <length> DEC MCS string

<parameter_list> ::= <value_expression> [...]

<literal_expression> ::= CDD$K_EXP_LITERAL
    { DSC$K_DTYPE_B <scale> <fixed_point_value>     |
      DSC$K_DTYPE_BU <scale> <fixed_point_value>    |
      DSC$K_DTYPE_W <scale> <fixed_point_value>     |
      DSC$K_DTYPE_WU <scale> <fixed_point_value>    |
      DSC$K_DTYPE_L <scale> <fixed_point_value>     |
      DSC$K_DTYPE_LU <scale> <fixed_point_value>    |
      DSC$K_DTYPE_Q <scale> <fixed_point_value>     |
      DSC$K_DTYPE_QU <scale> <fixed_point_value>    |
      DSC$K_DTYPE_O <scale> <fixed_point_value>     |
      DSC$K_DTYPE_OU <scale> <fixed_point_value>    |
      DSC$K_DTYPE_F <floating_point_value>          |
      DSC$K_DTYPE_D <floating_point_value>          |
      DSC$K_DTYPE_G <floating_point_value>          |
      DSC$K_DTYPE_H <floating_point_value>          |
      DSC$K_DTYPE_FC <floating_point_value>         |
      DSC$K_DTYPE_DC <floating_point_value>         |
      DSC$K_DTYPE_GC <floating_point_value>         |
      DSC$K_DTYPE_HC <floating_point_value>         |
      DSC$K_DTYPE_NU <word_length> <scale> <numeric_byte_string>  |
      DSC$K_DTYPE_NL <word_length> <scale> <numeric_byte_string>  |
      DSC$K_DTYPE_NLO <word_length> <scale> <numeric_byte_string> |
      DSC$K_DTYPE_NR <word_length> <scale> <numeric_byte_string>  |
      DSC$K_DTYPE_NRO <word_length> <scale> <numeric_byte_string> |
      DSC$K_DTYPE_NZ <word_length> <scale> <numeric_byte_string>  |
      DSC$K_DTYPE_P <word_length> <scale> <numeric_byte_string>   |
      DSC$K_DTYPE_T <word_length> <byte_string>     |
      DSC$K_DTYPE_VT <word_length> <byte_string>    |
      DSC$K_DTYPE_V <longword_length> <byte_string>|
      DSC$K_DTYPE_VU <word_length> <byte_string>    |
      DSC$K_DTYPE_ADT <binary_date_time>            |
      DSC$K_DTYPE_Z <word_length> <byte_string>     |
      CDD$K_DTYPE_ALPHABETIC <word_length> <byte_string>          |
      CDD$K_DTYPE_POINTER <4_byte_string>           |
```

```
      CDD$K_DTYPE_SEG_STRING <longword_length> <byte_string> }

<scale> ::= unsigned byte

<statistical_expression> ::=
    CDD$K_EXP_AVG <value_expression>  [<RSE>]       |
    CDD$K_EXP_COUNT <RSE>                           |
    CDD$K_EXP_MAX <value_expression>  [<RSE>]       |
    CDD$K_EXP_MIN <value_expression>  [<RSE>]       |
    CDD$K_EXP_SDV <value_expression>          <RSE> |
    CDD$K_EXP_TTL <value_expression>          <RSE> |
    CDD$K_EXP_RCT                                   |
    CDD$K_EXP_RTT <value_expression>

<string_expression> ::=
    CDD$K_EXP_AS2  <field_expression>                     |
    CDD$K_EXP_ASK  <field_expression>                     |
    CDD$K_EXP_CO2 <value_expression> <value_expression>  |
    CDD$K_EXP_CO3 <value_expression> <value_expression>  |
    CDD$K_EXP_CON <value_expression> <value_expression>  |
    CDD$K_FORMAT <field_expression> <edit_string>        |
    <substring_expression>
<substring_expression> ::=
    CDD$K_EXP_SUBSTRING <source_string> <start_position>
        <substring_length>

<source_string> ::= <value_expression>
<start_position> ::= <value_expression>

<substring_length> ::= <value_expression>
<edit_string> ::= (see CDD$K_EDIT_STRING_DSC buffer)

<via_expression> ::=
    CDD$K_EXP_VIA <RSE> <value_expression> <value_expression>
<aggregate_function> ::=
    <agg_count_fn> | <agg_stat_fn>

<agg_count_fn> ::=
    CDD$K_EXP_AGG_COUNT [<filter>] CDD$K_EXP_END
<agg_stat_fn> ::=
    <agg_stat> <value_expression> [<filter>] CDD$K_EXP_END

<agg_stat> ::=
    CDD$K_EXP_AGG_AVERAGE |
    CDD$K_EXP_AGG_MAX     |
    CDD$K_EXP_AGG_MIN     |
    CDD$K_EXP_AGG_TOTAL

<filter> ::= <project> | <select>
<project> ::=
    CDD$K_EXP_PROJECT <count> <value_expression>...

<select> ::=
    CDD$K_EXP_BOOLEAN <boolean_expression>
<group_value> ::=
    CDD$K_EXP_GROUP_VALUE <group_value_id>
```

```
<via_table_expression> ::=
    CDD$K_EXP_VTB <value_expression> <table_name>
<variable_expression> ::=
    CDD$K_EXP_VARIABLE <variable_id>

<variable_id> ::= unsigned word
| <boolean_expression> ::=
|    [ <case_sensitive> ] <relational_expression> |
|    <literal_expression>                          |
|    <logical_expression>                          |
|    <rse_expression>
|
| <case_sensitive> ::=
|    CDD$K_EXP_CASE_SENSITIVE

<relational_expression> ::=
    CDD$K_EXP_COT     <value_expression> <value_expression> |
    CDD$K_EXP_EQL     <value_expression> <value_expression> |
    CDD$K_EXP_GTR     <value_expression> <value_expression> |
    CDD$K_EXP_GEQ     <value_expression> <value_expression> |
    CDD$K_EXP_INT     <field_expression> <table_name>       |
    CDD$K_EXP_LSS     <value_expression> <value_expression> |
    CDD$K_EXP_LEQ     <value_expression> <value_expression> |
    CDD$K_EXP_MATCHES <value_expression> <value_expression> |
    CDD$K_EXP_NEQ     <value_expression> <value_expression> |
    CDD$K_EXP_BETWEEN <value_expression> <value_expression>
       <value_expression>                                  |
    CDD$K_EXP_STW     <value_expression> <value_expression>

| <logical_expression> ::=
|    CDD$K_EXP_ALPHABETIC <field_expression>                  |
|    CDD$K_EXP_ALPHABETIC_LOWER <field_expression>           |
|    CDD$K_EXP_ALPHABETIC_UPPER <field_expression>           |
|    CDD$K_EXP_AND <boolean_expression> <boolean_expression> |
|    CDD$K_EXP_EMPTY_FIELD <field_expression>                |
|    CDD$K_EXP_FULL_FIELD <field_expression>                 |
|    CDD$K_EXP_MIS <value_expression>                        |
|    CDD$K_EXP_NOT <boolean_expression>                      |
|    CDD$K_EXP_NUMERIC <field_expression>                    |
|    CDD$K_EXP_OR  <boolean_expression> <boolean_expression> |
|    CDD$K_EXP_XOR <value_expression> <value_expression>
<rse_expression> ::=
    CDD$K_EXP_ANY <RSE> | CDD$K_EXP_UNQ <RSE>

<RSE> ::=
    CDD$K_EXP_RSE   <relation_count>
        { <context_variable> <RSE_source> }...
        [ <rse_clause>... ]
        CDD$K_EXP_END
```

```
<RSE_source> ::=
    CDD$K_EXP_RELATION <RDB relation name>    |
    CDD$K_EXP_RELATION_ID <RDB relation_id>   |
    CDD$K_EXP_DOMAIN <DATATRIEVE domain name> |
    CDD$K_EXP_COLLECTION <name>               |
    CDD$K_EXP_LIST <name>                     |
    CDD$K_EXP_RECORD <name>                   |
    CDD$K_EXP_MERGE <merge_list>              |
    CDD$K_EXP_AGGREGATE <aggregate_exp>

<merge_list> ::= <count> <merge_item>...
<merge_item> ::= <RSE> <map>

<aggregate_exp> ::= <RSE> [<grouping>] <map>
<grouping>      ::= CDD$K_EXP_GROUP_BY <count> <grouping_exp>...

<grouping_exp> ::=
    CDD$K_EXP_GROUP_VALUE <group_value_id> <value_expression>

<map> ::=
    CDD$K_EXP_MAP <count> {<logical_field> <value_expression>}...

<logical_field>  ::= CDD$K_EXP_MAP_FIELD <length> DEC MCS string
<group_value_id> ::= unsigned byte
<count>          ::= unsigned byte
<relation_count> ::= unsigned byte
<name>           ::=  <length> DEC MCS string
<relation_id>    ::=  unsigned byte
<context_variable> ::=  <length> DEC MCS string

<rse_clause>     ::=
    CDD$K_EXP_ALL                                          |
    CDD$K_EXP_FIRST <field_expression>                     |
    CDD$K_EXP_BOOLEAN <boolean_expression>                 |
    CDD$K_EXP_REDUCE <reduction_count> <field_expression>... |
    CDD$K_EXP_SORT <item_count> <sort_item>...

<reduction_count> ::= unsigned byte
<item_count>      ::= unsigned byte
<sort_item>       ::=
     [ CDD$K_EXP_ASCENDING  | CDD$K_EXP_DESCENDING ]
        <field_expression>

<conditional_value_expression> ::=
    CDD$K_EXP_COND  <if-expression>
         <then-expression>
         [<else-expression>]
<if-expression>   ::= CDD$K_EXP_IF <boolean_expression>
<then-expression> ::= CDD$K_EXP_THEN <value_expression>
<else-expression> ::= CDD$K_EXP_ELSE <value_expression>
```

The syntax elements valid in an edit string buffer are explained in the order in which they appear in the syntax diagram. Only items new for CDD/Plus Version 4.1 are explained. Refer to the *VAX CDD/Plus Call Interface Manual* for explanations of the other items.

- CDD$K_EXP_ARRAY_SUBSCRIPT_LIST

This tag signifies that the subsequent information will be a list of array subscripts for the field whose description it follows.

- CDD$K_EXP_ARRAY_SUBSCRIPT

  This tag specifies that the word value following it defines one subscript for an array. One of these tags is used to define the subscript for each dimension of an array.

- CDD$K_EXP_CASE_SENSITIVE

  This tag is used to determine whether or not relational comparisons are to be done case-sensitive or case-insensitive. If the tag is present, the comparisons are case-sensitive.

- CDD$K_EXP_ALPHABETIC

  This tag is used in a logical expression to determine if a given field's value is entirely alphabetic.

- CDD$K_EXP_ALPHABETIC_LOWER

  This tag is used in a logical expression to determine if a given field's value is entirely lowercase alphabetic.

- CDD$K_EXP_ALPHABETIC_UPPER

  This tag is used in a logical expression to determine if a given field's value is entirely uppercase alphabetic.

- CDD$K_EXP_EMPTY_FIELD

  This tag is used in a logical expression to determine if a given field's value has not been entered.

- CDD$K_EXP_FULL_FIELD

  This tag is used in a logical expression to determine if a given field's value fills the field's output size.

- CDD$K_EXP_NUMERIC

  This tag is used in a logical expression to determine if a given field's value is numeric.

CDD$GET_ELEMENT

6.4.6 New Parameter to CDD$GET_ELEMENT

A new optional parameter has been added to the CDD$GET_ELEMENT call to return the full length of the output metadata buffer. Through this parameter, layered products will be able to determine the real length when a static descriptor is passed and, in the case of an overflow, the size of the remaining portion of the buffer.

Format

CDD$GET_ELEMENT exception_vector, element_handle, metadata_buf_in_dsc, metadata_buf_out_dsc, metadata_buf_out_size

New Parameter

metadata_buf_out_size

VMS usage:	longword_unsigned
type:	longword (unsigned)
access:	modify
mechanism:	by reference

The address of a longword to be set to the length of the entire output metadata buffer.

Usage Notes

If the entire buffer can be copied into the buffer described by the metadata_buf_out_dsc parameter, this parameter's value will be the length of the portion of the buffer that was used.

If the entire buffer cannot fit in the output metadata buffer, the size will reflect the length of the entire buffer. The metadata_buf_out_dsc parameter's length will be the length of the portion of the buffer that has been returned. Subsequent calls to CDD$GET_ELEMENT can be made to get the remainder of the buffer.

If the metadata_buf_out_size parameter is specified on a subsequent call, the value will be the length of the entire buffer minus the length of any portions that were returned in previous calls to CDD$GET_ELEMENT. The length of the portion of the buffer returned in a given call will always be included in this length.

CDD$CHANGE_DIRECTORY

6.4.7 CDD$CHANGE_DIRECTORY Routine Added

Change the ACL for a node in a directory hierarchy.

Format

CDD$CHANGE_DIRECTORY exception_vector, session_handle, directory_info_buffer

Returns

VMS usage:	condition_value
type:	longword (unsigned)
access:	write only
mechanism:	by value

Longword condition value. All CDD/Plus callable routines return (by immediate value) a condition value in R0. Condition values that can be returned by this service include:

SS$_NORMAL	Operation completed successfully.

Arguments

exception_vector

VMS usage:	message_vector
type:	array of 20 longwords
access:	write only
mechanism:	by reference

An array of 20 longwords conforming to the VMS exception vector format.

session_handle

VMS usage:	quadword_unsigned
type:	quadword (unsigned)
access:	read only
mechanism:	by reference

Unsigned quadword uniquely identifying the dictionary session for which directory ACLs are modified. This value is assigned by CDD$START_SESSION.

directory_info_buffer

VMS usage:	unstructured
type:	text
access:	read only
mechanism:	by descriptor (static or dynamic)

A text descriptor containing the name of the directory node whose ACL is to be modified and the ACL buffer defining the ACL to be associated with the directory node. See Section 6.4.3 for the description of this buffer.

CDD$DELETE_DIRECTORY

6.4.8 CDD$DELETE_DIRECTORY Routine Missing from the *VAX CDD/Plus Call Interface Manual*

The description of the CDD$DELETE_DIRECTORY entry point was accidentally omitted from the *VAX CDD/Plus Call Interface Manual.* The following information should appear in Chapter 4, following page 4–28.

The CDD$DELETE_DIRECTORY routine deletes a directory node. If any objects exist in the directory, an error is returned.

Format

CDD$DELETE_DIRECTORY exception_vector, session_handle, directory_info_buffer

Returns

VMS usage:	condition value
type:	longword (unsigned)
access:	write only
mechanism:	by value

Longword condition value. All CDD/Plus callable routines return (by immediate value) a condition value in R0. Condition values that can be returned by this service include:

SS$_NORMAL	Operation completed successfully.

Arguments

exception_vector

VMS usage:	message_vector
type:	array of 20 longwords
access:	write only
mechanism:	by reference

An array of 20 longwords conforming to the VMS exception vector format.

session_handle

VMS usage:	quadword_unsigned
type:	quadword (unsigned)
access:	read only
mechanism:	by reference

An unsigned quadword that uniquely identifies a dictionary session. This value is assigned by CDD$START_SESSION.

directory_info_buffer

VMS usage:	unstructured
type:	text
access:	read only
mechanism:	by descriptor (static or dynamic)

A text descriptor containing the name of the directory path to delete. The format of this descriptor can be found in Section 5.3 in *VAX CDD/Plus Call Interface Manual.*

Usage Notes

If there are any entries in the named directory node, an error is returned.

CDD$ERASE_DIRECTORY_ENTRY

6.4.9 CDD$ERASE_DIRECTORY_ENTRY Routine

The CDD$ERASE_DIRECTORY_ENTRY routine has been added to the call interface. This routine removes a name from a directory, but does not delete the associated dictionary element.

Format

CDD$ERASE_DIRECTORY_ENTRY exception_vector, element_handle, directory_info_buffer

Returns

VMS usage:	condition_value
type:	longword (unsigned)
access:	write only
mechanism:	by value

Longword condition value. All CDD/Plus callable routines return (by immediate value) a condition value in R0. Condition values that can be returned by this service include:

SS$_NORMAL	Operation completed successfully.

Arguments

exception_vector

VMS usage:	message_vector
type:	array of 20 longwords
access:	write only
mechanism:	by reference

An array of 20 longwords conforming to the VMS exception vector format.

element_handle

VMS usage:	quadword_unsigned
type:	quadword (unsigned)
access:	read only
mechanism:	by reference

An unsigned quadword identifying the dictionary element for which CDD/Plus deletes the directory entry.

directory_info_buffer

VMS usage:	unstructured
type:	text
access:	read only
mechanism:	by descriptor (static or dynamic)

A buffer identifying the directory entry CDD/Plus deletes. For a complete description of directory information buffers, see the *VAX CDD/Plus Call Interface Manual.*

Usage Notes

Because the CDD/Plus dictionary and directory systems are separate, you can delete directory information without deleting metadata. This call deletes no metadata.

If you try to delete the only directory entry for a particular dictionary element, and that element is not the member of some relationship, CDD/Plus does not delete the directory entry and returns an error to your program. Your dictionary remains unchanged.

7

Guidelines for Submitting an SPR

If you find a CDD/Plus software error, please submit a Software Performance Report (SPR) to Digital SPR Administration if you are eligible for that service. Your SPR should include the following information:

- A statement of the problem.
- A VMS BACKUP of the dictionaries in which the error occurred.
- A hard copy of all error messages, including any traceback information for bugchecks.
- The exact sequence of commands issued that caused the problem.

If you want to submit machine-readable information, please include all necessary files to compile and link your program. You should also include a command procedure (or instructions) on how to link the program.

Index

D

E

F

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-DEC-DEMO (800-332-3366) using a 1200- or 2400-baud modem. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local DIGITAL subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local DIGITAL subsidiary or approved distributor
Internal[1]	————	SDC Order Processing - WMO/E15 *or* Software Distribution Center Digital Equipment Corporation Westminster, Massachusetts 01473

[1]For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

VAX CDD/Plus Release Notes
AA–GK49E–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

What I like best about this manual is _____

_____

What I like least about this manual is _____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

Do Not Tear - Fold Here and Tape

digital™

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

Do Not Tear - Fold Here

Reader's Comments

VAX CDD/Plus Release Notes
AA–GK49E–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

What I like best about this manual is _____

_____

What I like least about this manual is _____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

- Do Not Tear - Fold Here and Tape -----

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

— Do Not Tear - Fold Here -----