AA–JD84E–TE

VAX Data Distributor

Release Notes

November 1990

This document contains the release notes for VAX Data Distributor Version 2.2.

Operating System:	VMS
Software Version:	VAX Data Distributor Version 2.2

digital equipment corporation
maynard, massachusetts

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Any software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license. No responsibility is assumed for the use or reliability of software or equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

© Digital Equipment Corporation 1988, 1989, 1990.

All rights reserved.
Printed in U.S.A.

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

The following are trademarks of Digital Equipment Corporation: ALL–IN–1, CDD/Plus, DEC, DEC/CMS, DECdecision, DECdtm, DECforms, DECintact, DEC/MMS, DECnet, DECtp, DECtrace, DECwindows, MicroVAX, ULTRIX, UNIBUS, VAX, VAX ACMS, VAX Ada, VAX BASIC, VAX C, VAX CDD, VAXcluster, VAX COBOL, VAX DATATRIEVE, VAX DBMS, VAXELN, VAX FMS, VAX FORTRAN, VAX Pascal, VAX RALLY, VAX Rdb/ELN, VAX Rdb/VMS, VAX RMS, VAX SPM, VAXstation, VAX TEAMDATA, VIDA, VMS, VT, and the DIGITAL Logo.

The following are third-party trademarks: IBM is a registered trademark and DB2 is a trademark of International Business Machines Corporation; IDMS/R is a trademark of Cullinet Software, Inc.; ORACLE is a trademark of the ORACLE Corporation.

This document is available in both printed and CDROM versions.

This document was prepared using VAX DOCUMENT, Version 1.2.

Contents

Preface vii

1 General Information

1.1 Installation Process 1–1
1.1.1 Requirements for Installing Data Distributor 1–1
1.1.2 Changing the Internal Structure of a Transfer Database 1–1
1.1.3 Changing the Internal Structure of a Source Database 1–2
1.1.4 Changing the Internal Structure of a Target Database 1–2
1.2 Where to Find Online Help and Error Message Documentation . . 1–2
1.2.1 Getting Help on Data Distributor Statements 1–2
1.2.2 Error Messages 1–2

2 New and Changed Features

2.1 Transfers to Existing Databases 2–1
2.2 New and Changed Table Definitions 2–1
2.2.1 Target Database Table Definitions 2–1
2.2.2 Transfer Database Table Definitions 2–2
2.3 Target Domain Description Changes 2–3
2.4 New and Changed Target Index Definitions 2–3
2.5 Rdb/VMS Version 4.0 Changes Affecting Data Distributor 2–4
2.5.1 Default Database and Table Protection 2–4
2.5.2 New SIGNED BYTE Datatype 2–4
2.6 Replication Transfers Performance Option 2–4
2.7 Limiting Concurrent Transfers on a CPU 2–6
2.8 DEC RdbAccess for VMS, ORACLE Server Software Support 2–6

3 Problems Corrected

3.1 Inconsistent Replicated Data When Triggers Are Involved 3–1
3.2 Retrying Lock Conflict and Deadlock Errors 3–1
3.3 Serial Execution of Reinitialized Replication Transfers 3–2
3.4 Bad SQL Transfer Definition from VIDA for DB2 3–2
3.5 Incorrect EPILOGUE Status Codes 3–3

4 Known Problems and Restrictions

4.1 Using LIB$DT_INPUT_FORMAT to Change Date Input Format 4–1
4.2 Restrictions for Extraction, Extraction Rollup, and Replication Transfers 4–2
4.2.1 Data Distributor Version 2.2 CREATE TRANSFER Statement Syntax Change and Existing Command Procedures 4–2
4.2.2 Transfer Schedule Defined for Time Before Transfer Execution 4–2
4.2.3 Transfer Completion Status After a Transfer Is Stopped 4–2
4.2.4 Data Definition Changes After a Transfer Is Defined 4–2
4.2.5 Information Retrieval from a Transfer Database 4–3
4.3 Restriction for Extraction and Extraction Rollup Transfers Only 4–4
4.3.1 SQL Does Not Translate Logical Names Referencing Source Databases 4–4
4.4 Restriction for Extraction Transfers Only 4–5
4.4.1 Views Containing Computed_By Columns Cannot Become Tables 4–5
4.5 Restrictions for Extraction Rollup Transfers Only 4–5
4.5.1 Column Names for Like Tables Must Be Identical 4–5
4.5.2 SQL Privileges Can Restrict Access to Source Tables 4–6
4.6 Restrictions for Replication Transfers Only 4–7
4.6.1 Restriction on Fields Named in an Rdb/VMS RDO RSE 4–7
4.6.2 Restriction on Columns Named in an SQL Select Expression 4–7
4.6.3 Deleting Transfer Definitions 4–7
4.6.4 Restriction on Two-Way Replication Transfers 4–8
4.7 Restrictions for Transferring Into an Existing Database 4–9
4.7.1 Transferring a Table to an Existing Database Containing the Same Table Name 4–9
4.7.2 No Extraction Rollup Into Existing Databases 4–10
4.7.3 No Union of Source and Target Data 4–10
4.7.4 No DEFINE TRANSFER Into an Existing Database 4–10
4.8 Cluster Restrictions 4–10
4.8.1 Load Balancing in a Cluster 4–11

Tables

2–1	Columns for the RDB$VINTAGE Table	2–2
2–2	Columns for the RDBVMS$TRANSFER_OWNER Table	2–2
2–3	New Columns for the DDAL$TRANSFERS Table	2–2
2–4	New Columns for the DDAL$TRANSFER_VIEWS Table	2–2
2–5	New Domain Descriptions	2–3
2–6	Indexes for the RDB$CHANGES Table	2–5
2–7	Equivalence Names for Index Logicals	2–5
3–1	Corrected Status Codes	3–3
4–1	Transfer Database Tables	4–4

Preface

This manual describes new and changed features, problems corrected in this release, and current restrictions. It also discusses general information about VAX Data Distributor Version 2.2 that can help you use the product. VAX Data Distributor software is referred to as Data Distributor throughout this document.

Intended Audience

This manual is intended for all Data Distributor users.

Product Compatibility

Information about the versions of the operating system and related software that are compatible with this version of Data Distributor is included in the *VAX Data Distributor Installation Guide*, which accompanies the Data Distributor media kit.

For information on the compatibility of other software products with this version of Data Distributor, 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 Data Distributor.

Structure

This manual contains four chapters:

Chapter 1	Provides general information about Data Distributor.
Chapter 2	Describes new and changed features in Data Distributor.
Chapter 3	Describes software problems that have been corrected in Data Distributor.
Chapter 4	Describes software restrictions in effect for Data Distributor.

Related Manuals

The other manuals in the Data Distributor documentation set are:

- *VAX Data Distributor Installation Guide*—Describes how to install Data Distributor.
- *VAX Data Distributor Handbook*—Provides information, guidelines, and examples for distributing all or portions of VAX Rdb/VMS, VIDA, or RdbAccess for ORACLE database products in a network.

For further information on topics covered in this document, refer to the following manuals:

- *VAX Rdb/VMS SQL Reference Manual*

  Describes the statements, the interactive, dynamic, and module language interfaces, and the syntax for SQL, the structured query language interface for Rdb/VMS. It provides full descriptions of Data Distributor statements used with interactive SQL.

- *VAX Rdb/VMS RDO and RMU Reference Manual*

  Provides full descriptions of Data Distributor statements used with the Relational Database Operator utility (RDO) of Rdb/VMS. The documentation in the reference manual is provided for your convenience. It documents the same statements found in Chapter 9 of the *VAX Data Distributor Handbook*.

- *VIDA for DB2 Overview*

  Describes how VIDA provides transparent access to DB2 data. The document contains an overview of the products that serve as interfaces for creating relational databases on a VAX system. Data Distributor lets you extract databases created by VIDA for DB2.

- *DEC RdbAccess for ORACLE on VMS Handbook*

  Describes how DEC RdbAccess for ORACLE on VMS software provides transparent access to data stored in ORACLE databases.

Conventions

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the RETURN key at the end of a line of input.

This manual uses the following conventions:

Color	In printed manuals, color in examples shows user input.
$	The dollar sign represents the DIGITAL Command Language prompt. This symbol indicates that the DCL interpreter is ready for input.

References to Products

The Data Distributor documentation to which this manual belongs often refers to products by their abbreviated names:

- VAX CDD/Plus software is referred to as CDD/Plus.
- VAX Rdb/VMS software is referred to as Rdb/VMS.
- VIDA with IDMS/R software is referred to as VIDA, unless differentiation from the VIDA for DB2 product is necessary.
- VIDA for DB2 software is referred to as VIDA, unless differentiation from the VIDA with IDMS/R product is necessary.
- DEC RdbAccess for ORACLE on VMS software is referred to as RdbAccess for ORACLE.

1
General Information

This chapter contains information about the Data Distributor installation process and other general topics.

1.1 Installation Process

This section contains information about the installation of Data Distributor Version 2.2.

1.1.1 Requirements for Installing Data Distributor

The following requirements must be met before Data Distributor Version 2.2 can be installed:

- VMS Version 5.3 or higher must be installed on your standalone system or on your VAXcluster.
- VAX Rdb/VMS Version 4.0 or higher must be installed on your system.
- Database conversion required by the VAX Rdb/VMS Version 4.0 installation must be performed. You can find instructions about database conversion in the *VAX Rdb/VMS Installation Guide* for Version 4.0.

1.1.2 Changing the Internal Structure of a Transfer Database

Customers who convert from Data Distributor Version 2.1 to 2.2 must perform a conversion procedure on their existing transfer databases. The conversion procedure adds new data definitions to the transfer database to support the Version 2.2 features. The DDAL$CONVERT_TR_DB.COM procedure has been modified to do the necessary conversion to Version 2.2.

1.1.3 Changing the Internal Structure of a Source Database

If you upgrade to Data Distributor Version 2.2 from an earlier release, there are no special Data Distributor conversion requirements for source databases. However, you may need to convert source databases to conform with the latest version of VAX Rdb/VMS. Refer to the *VAX Rdb/VMS Installation Guide* for conversion requirements.

1.1.4 Changing the Internal Structure of a Target Database

Data Distributor Version 2.2 has added columns to its RDB$VINTAGE table and has introduced a new table, RDBVMS$TRANSFER_OWNER. Databases that were the targets of transfers in earlier versions of Data Distributor need to be converted to the Version 2.2 format. You need not take any special action to perform this conversion. When you execute a transfer using Version 2.2, the copy process converts the target database automatically. However, you must convert existing target databases to the latest Rdb/VMS format before executing transfers using Data Distributor Version 2.2. For information on converting Rdb/VMS databases, see the *VAX Rdb/VMS Installation Guide*.

1.2 Where to Find Online Help and Error Message Documentation

This section describes how to access Data Distributor online help and error message documentation.

1.2.1 Getting Help on Data Distributor Statements

Data Distributor provides an online version of these release notes, which are installed on your system in the SYS$HELP directory. The file is called DDAL022.RELEASE_NOTES. In Rdb/VMS Version 4.0, the SQL help topic Data_Distributor Release_Notes includes the location of the online release notes file.

Help information on Data Distributor commands is part of SQL HELP for Rdb/VMS. After you type HELP at the SQL> prompt, specify a Data Distributor statement by entering DATA_DISTRIBUTOR and the statement name. For example, to find help on the CREATE TRANSFER statement, type HELP DATA_DISTRIBUTOR CREATE_TRANSFER at the SQL> prompt. For a complete list of Data Distributor help topics, type HELP DATA_DISTRIBUTOR.

1.2.2 Error Messages

Data Distributor error message information is contained in a file that you can view on line by using the DCL TYPE command or print by using the DCL PRINT command. The DDAL$MSG.DOC error message file is located in the SYS$HELP directory.

2

New and Changed Features

This chapter describes new and changed features for VAX Data Distributor Version 2.2.

2.1 Transfers to Existing Databases

A common aspect of all Data Distributor transfers prior to Version 2.2 is that the target database is created and maintained by Data Distributor. With Version 2.2, you now have a choice: you can direct Data Distributor to create a new database as before, or you can specify a database that already exists. This feature is documented in Chapter 8 of the *VAX Data Distributor Handbook*.

2.2 New and Changed Table Definitions

In order to manage transfers into existing databases, the tables maintained by Data Distributor in target and in transfer databases have been augmented. This section outlines those changes. For a complete description of the augmented tables, refer to Appendix A in the *VAX Data Distributor Handbook*.

2.2.1 Target Database Table Definitions

The RDB$VINTAGE table in target databases has been changed to include new columns. In addition, for transfers to existing databases, a new table, RDBVMS$TRANSFER_OWNER, has been added.

Table 2–1 lists the columns found in the RDB$VINTAGE table, shows the datatype, and indicates whether the column is new for Data Distributor Version 2.2.

Table 2–1 Columns for the RDB$VINTAGE Table

Name	Datatype	New for V2.2?
RDB$VINTAGE_TSER	INTEGER	No
RDB$VINTAGE_TIMESTAMP	DATE	No
RDBVMS$VINTAGE_TRANSFER_NODE	CHAR(20)	Yes
RDBVMS$VINTAGE_TRANSFER_ADDR	VARCHAR(402)	Yes
RDBVMS$VINTAGE_TRANSFER_NAME	CHAR(31)	Yes
RDBVMS$VINTAGE_TRANSFER_ID	SMALLINT	Yes
RDBVMS$VINTAGE_TRANSFER_BUSY	CHAR(1)	Yes

The RDBVMS$TRANSFER_OWNER table keeps track of which transfers own which objects in the target database. Table 2–2 lists the columns found in the RDBVMS$TRANSFER_OWNER table, shows the datatype, and indicates whether the column is new for Data Distributor Version 2.2.

Table 2–2 Columns for the RDBVMS$TRANSFER_OWNER Table

Name	Datatype	New for V2.2?
RDBVMS$TRANSFER_OWNER_ID	SMALLINT	Yes
RDBVMS$TRANSFER_OBJECT_TYPE	SMALLINT	Yes
RDBVMS$TRANSFER_OBJECT_NAME	CHAR(31)	Yes

2.2.2 Transfer Database Table Definitions

Data Distributor's transfer database includes columns added to existing tables. Tables 2–3 and 2–4 show only the new columns, not the entire table definitions.

Table 2–3 New Columns for the DDAL$TRANSFERS Table

Name	Datatype	New for V2.2?
DDAL$TRANSFER_TO_EXISTING	CHAR(1)	Yes

Table 2–4 New Columns for the DDAL$TRANSFER_VIEWS Table

Name	Datatype	New for V2.2?
DDAL$TARGET_VIEW_NAME	CHAR(31)	Yes

2.3 Target Domain Description Changes

The description for three domains has changed for Data Distributor Version 2.2. Table 2–5 lists the three domains, the table each is found in, and the new description.

Table 2–5 New Domain Descriptions

Domain	Table	Description
RDB$VINTAGE_TIMESTAMP	RDB$VINTAGE	Vintage timestamp
RDB$VINTAGE_TSER	RDB$VINTAGE	Vintage transaction serialization number
DDAL$DBKEY	user[1]	Database key of a source database row

[1]The copy process adds the DDAL$DBKEY column to each user table created in the target database for a replication transfer. Database key values contained in this column provide Data Distributor with information about the corresponding row in the source database.

2.4 New and Changed Target Index Definitions

Multiple rows can now appear in the RDB$VINTAGE table in the target database. This occurs when several transfers target the same existing database.

The column RDBVMS$VINTAGE_TRANSFER_ID in the RDB$VINTAGE table contains a unique numeric value that identifies each transfer. To ensure that each transfer has a unique value, there is a sorted index on the RDB$VINTAGE table, based on the column RDBVMS$VINTAGE_ TRANSFER_ID, that does not allow duplicates. The name of this index is RDBVMS$VINTAGE_INDEX1.

For replication transfers, Data Distributor also creates an index in the target database for each table copied from a source database. Each copied table contains an additional column, DDAL$DBKEY, which stores the database key of the row as it exists in the source database. The name of the index depends on whether the transfer is to a new or existing database.

- If the copied table is created by a transfer to a new database (Data Distributor creates the target database), the index name takes the form DDAL$DBKEY_INDEXn, for example, DDAL$DBKEY_INDEX1, DDAL$DBKEY_INDEX2, . . .
- If the copied table is created by a transfer to an existing database, the index name also includes the unique transfer ID. For example, an index for a replication transfer into an existing database where the transfer has an ID value of 3 may be named DDAL$DBKEY_INDEX1_3.

2.5 Rdb/VMS Version 4.0 Changes Affecting Data Distributor

Apart from specific support for Data Distributor Version 2.2 features, there are a few changes and additions in Rdb/VMS Version 4.0 that affect Data Distributor.

2.5.1 Default Database and Table Protection

Rdb/VMS Version 4.0 has changed the default database and table protection when new databases are created. One change also affects the protection given to databases and tables created by Data Distributor transfers.

Data Distributor does not copy source database protection attributes to the target databases. Instead, it lets Rdb/VMS apply whatever are its defaults. In Rdb/VMS systems prior to Version 4.0, public accounts ([*,*]) were by default given virtually unlimited access to the database. If the target database is on a system running such a version of Rdb/VMS, that behavior will continue to be in effect. If the target database is on a system running Rdb/VMS 4.0, the public by default gets no access privileges to the database or to tables within the database.

Note *If you are a Data Distributor customer who has used Version 2.1 or earlier, beware of the effect of this protection change. If your target database users or applications rely on the default access protection of Rdb/VMS, you will have to take steps to deal with the change. You can use epilogue files to change database protection after a transfer has taken place.*

2.5.2 New SIGNED BYTE Datatype

Rdb/VMS Version 4.0 has added a new datatype called, SIGNED BYTE. In VAX SQL, it is called a TINYINT. Data Distributor supports this new datatype.

Data Distributor supports the transfer of columns having this datatype to systems that support it. When creating transfers between two systems running different versions of Rdb/VMS, be aware that one system might not support the SIGNED BYTE datatype. If you attempt such a transfer, the transfer definition is accepted but the transfer fails upon execution. You can determine the reason for the failure by examining the error message in the copy process log file associated with the transfer.

2.6 Replication Transfers Performance Option

Data Distributor Version 2.2 includes an option for adding indexes to Data Distributor's RDB$CHANGES table in a replicated source database. Certain classes of applications can realize overall performance benefits from this option. Enabling and disabling of this option will be done by logical name definition. This section presents a brief overview of the indexing option. For a full description, refer to Chapter 6, section 6.14, of the *VAX Data Distributor Handbook*.

You have the option of automatically adding two sorted indexes for the RDB$CHANGES table in a replicated source database. See Table 2–6.

Table 2–6 Indexes for the RDB$CHANGES Table

Index	based on column
RDBVMS$CHANGES_INDEX1	RDB$TRANSACTION_TSER RDB$TRANSACTION_TID RDB$TRANSACTION_SEQUENCE
RDBVMS$CHANGES_INDEX2	RDB$TRANSACTION_TID RDB$TRANSACTION_SEQUENCE

You can enable or disable indexing by defining one of the following logical names.

DDAL$CHANGES_INDEX_transfer_name
DDAL$CHANGES_INDEX

Table 2–7 shows the equivalence names that you can use for these logical names.

Table 2–7 Equivalence Names for Index Logicals

String	Meaning
CREATE_ON_UPDATE	Create indexes during the replication update cycle, if necessary.
DROP	Drop indexes during replication initialization, reinitialization, update or deletion.
NONE	No indexing desired. Queries will assume indexes don't exist.
USE	Create indexes during replication initialization or reinitialization, if necessary. During replication update, assume RDB$CHANGES is indexed.

2.7 Limiting Concurrent Transfers on a CPU

Data Distributor has a built-in limit to the number of concurrent transfers it will attempt to execute. That number is 20. This limit is enforced by Data Distributor's transfer monitor since it is responsible for starting up the copy processes. This limit does not guarantee that 20 transfers will run concurrently. System resources may lower the limit to less than 20.

A system manager can now limit the number of concurrent transfers that can be executed on a given CPU by defining a system-wide logical name, DDAL$MAX_COPY_PROCESS.

For a full description of this feature, refer to Chapter 6, section 6.15, of the *VAX Data Distributor Handbook*.

2.8 DEC RdbAccess for VMS, ORACLE Server Software Support

Data Distributor Version 2.2 now supports ORACLE databases as source databases for extraction and extraction rollup transfers. Data Distributor uses RdbAccess for ORACLE as its interface to ORACLE.

3

Problems Corrected

The following notes describe problems in Data Distributor that have been corrected in the current release.

3.1 Inconsistent Replicated Data When Triggers Are Involved

A problem exists in earlier versions of Data Distributor involving the replication of tables on which triggers are defined. It is possible to create triggers that cause a row to be modified more than once. For example, the insertion of a row could activate a trigger that then alters a column within that same row.

Data Distributor did not correctly handle this type of self-modifying behavior. The result was that the row transferred to the target database would be missing some of the changes that had occurred. This problem is corrected in the current version of Data Distributor.

3.2 Retrying Lock Conflict and Deadlock Errors

In prior versions of Data Distributor, a transfer would be suspended with a severe error if a copy process received either of the following two status codes:

RDB$_LOCK_CONFLICT
RDB$_DEADLOCK

Retrying the transfer required manual intervention even if you had an appropriately-defined schedule for the transfer. This problem occurred in all transfer types: extraction, extraction rollup, or replication.

Transfers that receive the RDB$_LOCK_CONFLICT or RDB$_DEADLOCK status codes are now automatically retried if retry is defined in the transfer schedule.

3.3 Serial Execution of Reinitialized Replication Transfers

The initial execution of a replication transfer copies source database tables to a target database. Subsequent executions of this replication transfer copy only the source table updates to the target tables. In certain circumstances you might want to reinitialize a replication transfer to recreate the source tables in the target in their entirety instead of only copying the updates.

Data Distributor now permits parallel execution of multiple reinitialized transfers that use the same source database. In prior versions of Data Distributor, such transfers executed one after the other because they performed the same logic as initial replication transfers.

Initial replication transfers (replication transfers that are executed for the first time) must store a portion of the transfer definition in special system tables in a source database. This must be done with exclusive access to the database. Therefore several initial replication transfers using the same source database will continue, as in the past, to execute serially.

With the release of Data Distributor Version 2.2, when a transfer is reinitialized and rerun, Data Distributor checks to see if the transfer is already stored in the source database and if that definition is unchanged. If so, then exclusive access to that database is not needed.

When several reinitialized transfers using the same database are run concurrently, it is possible for them to receive lock conflict errors unless the database has already been manually opened from all nodes that access it. Opening a database manually is done using the RMU/OPEN command of Rdb/VMS.

3.4 Bad SQL Transfer Definition from VIDA for DB2

In Data Distributor Version 2.1, you could not use VAX SQL to successfully create and execute extraction transfers from a VIDA for DB2 database. Data Distributor *appeared* to accept the transfer definition. However, examination of the transfer definition using the SQL SHOW TRANSFER DEFINITION FOR transfer-name statement would have revealed a blank on the ''From'' line where ''/TYPE=VIDA2... (VIDA access control string)'' should have appeared.

Execution of this transfer would fail. A copy process log file showing the error may have look like the following (note that the name of the source database is missing):

```
15:07:35  %DDAL-I-ATTACHSDB, attaching to source database
----- 11-JUL-1990 15:07:35.99 ----- Error           -----------
%RDB-E-BAD_DB_FORMAT,
-RMS-F-SYN, file specification syntax error
```

You could successfully define and execute this type of transfer using the RDO interface.

This problem has been corrected in VAX SQL Version 4.0, which coincides with Data Distributor Version 2.2.

3.5 Incorrect EPILOGUE Status Codes

In Version 2.1 of the *Data Distributor Handbook*, prologue and epilogue failure codes are listed with the wrong values in Appendix B, Table B-3. These codes also are incorrectly defined in the Data Distributor procedure SYS$MANAGER:DDAL$EPILOGUE.COM. Consequently, prologue and epilogue failures were not correctly reported in the transfer status.

This has been corrected for Data Distributor Version 2.2, in both the *Handbook* and the command procedure. The correct values are also shown in Table 3–1.

Table 3–1 Corrected Status Codes

Incorrect Code	Correct Code	Explanation
0126C423	0126C44B	Prologue failed, copy aborted, but retrying
0126C42B	0126C453	Prologue failed, copy aborted, and not retrying
0126C433	0126C45B	Epilogue failed but retrying
0126C43B	0126C463	Epilogue failed and not retrying

4

Known Problems and Restrictions

This chapter documents restrictions currently in effect for Data Distributor and one VMS restriction that affects both interactive SQL and the RDO utility of Rdb/VMS.

4.1 Using LIB$DT_INPUT_FORMAT to Change Date Input Format

In the discussion of the CREATE SCHEDULE statement in Chapter 9 of the *VAX Data Distributor Handbook*, the Usage Notes describe how to change the default date and time format (established before VMS Version 5.0) to a date and time format that complies with the LIB$ international date and time formatting routines of VMS Version 5.0 and higher. (Refer to the Usage Notes for the DEFINE TRANSFER statement in Chapter 10 for information about changing the date and time format in RDO.)

A problem exists with the LIB$CONVERT_DATE_STRING function of the VMS Version 5.3 Run-Time Library. Rdb/VMS uses the function in precompiled programs, SQL module language, and the RDO and SQL interactive interfaces to convert dates to internal format. When the logical name LIB$DT_INPUT_FORMAT is used to change the date input format, the run-time library sometimes causes an access violation that probably prevents the precompiler or module language compiler from continuing, but does not cause any loss of data from interactive SQL. The access violation is shown in the following example:

```
%SQL-F-DATCONERR, Data conversion error
-SYSTEM-F-ACCVIO, access violation,  . . .
```

4.2 Restrictions for Extraction, Extraction Rollup, and Replication Transfers

This section documents the restrictions that apply equally well to extraction, extraction rollup, and replication transfers.

4.2.1 Data Distributor Version 2.2 CREATE TRANSFER Statement Syntax Change and Existing Command Procedures

The TO clause in the CREATE TRANSFER statement now requires that you specify whether the transfer is to a new or existing target database. CREATE TRANSFER syntax prior to Version 2.2 required the keyword TO with the target-file-spec parameter. Version 2.2 now requires the keywords TO NEW FILENAME with the target-file-spec parameter. Therefore, you must edit any existing command procedures that specify the keyword TO by adding the additional keywords NEW FILENAME.

This restriction applies only to the SQL CREATE TRANSFER statement, **not** to the RDO DEFINE TRANSFER statement. The syntax for the DEFINE TRANSFER statement has not changed for Data Distributor Version 2.2.

4.2.2 Transfer Schedule Defined for Time Before Transfer Execution

When you define a one-time-only transfer schedule and specify a transfer time that has already passed, issuing a START TRANSFER statement changes the transfer status from suspended to scheduled. However, when you issue a SHOW TRANSFER STATUS statement, the ''next transfer to be executed'' phrase is not included. To avoid this problem and cause the transfer to execute, use the START TRANSFER NOW statement.

4.2.3 Transfer Completion Status After a Transfer Is Stopped

When you issue a STOP TRANSFER statement for a transfer in the active state, the state changes to suspended. However, the last transfer completion status is not updated and does not reflect the incomplete transfer.

4.2.4 Data Definition Changes After a Transfer Is Defined

In an extraction transfer, you can change table definitions in the source database or definitions of fields on which the transferred tables are based. However, those changes are not reflected in the transfer definition.

In replication transfers, you can make such changes until the initial transfer begins. Thereafter, any attempts to change or delete definitions for tables or columns in the replication transfer are not allowed.

In both extraction and replication transfers, if you make changes that cause the source tables or columns to be incompatible with the transfer definitions, Data Distributor detects these differences at the time of transfer execution. The transfer fails, the transfer status reflects the failure, and the transfer is suspended.

However, with extractions or initial replications, the copy process does not detect changes to the data type of a field. For example, you can change a field from the data type TEXT to the data type LONGWORD without causing a failure.

4.2.5 Information Retrieval from a Transfer Database

Within Data Distributor, the transfer monitor creates a transfer database. Only one of the tables in this database has an associated index. When many transfers are defined and when many tables with many columns are included in the transfer definitions, a significant portion of the time it takes for a transfer to execute can be attributed to the time it takes to look up a transfer definition.

In the current version of Data Distributor, you can determine the length of time spent during transfer execution to read the transfer definition from the transfer database. Data Distributor records a message in the copy process log file for each transfer that notes the time the copy process begins reading the transfer definition. The next message records the time the copy process attaches to the source database. The difference between the two times is the total time spent reading the transfer definition.

If you find that reading the transfer definition takes too long, you can add indexes to the transfer database tables to improve reading speed. This enhancement may be done automatically in a future release of Data Distributor.

The decision to index a table depends on whether the number of data rows in the table has become or is expected to become large. You should optimize access to the transfer database for data retrieval because data storage occurs less frequently. The Data Distributor tables in a transfer database are shown in Table 4–1. The first two or three of these tables frequently contain the most data rows. Therefore, you may decide to index the first three tables only. Digital recommends that you name any index you add by using the table name and appending the suffix ‘‘_INDEX’’. Simple sorted indexes are sufficient.

Table 4–1 Transfer Database Tables

Table Name	Possible Index Based on Column	Duplicates Allowed?
DDAL$TRANSFER_FIELDS	DDAL$TRANSFER_NAME	Yes
	DDAL$RELATION_NAME	Yes
DDAL$TRANSFER_RELATIONS	DDAL$TRANSFER_NAME	No
DDAL$TRANSFER_VIEWS	DDAL$TRANSFER_NAME	No
DDAL$INVOKE_TIME	DDAL$TRANSFER_NAME	No
DDAL$ROLLUP_DATABASES	DDAL$TRANSFER_NAME	No
DDAL$TRANSFERS_STATUS	DDAL$TRANSFER_NAME	No
DDAL$TRANSFER_SCHEDULE	DDAL$TRANSFER_NAME	No

There are two tables not shown in Table 4–1. DDAL$TRANSFERS is the one table that is automatically indexed by Data Distributor. The second table is DDAL$TRANSFER_DATABASE, which contains a single row and needs no index.

4.3 Restriction for Extraction and Extraction Rollup Transfers Only

This section describes a restriction that applies to extraction and extraction rollup transfers only.

4.3.1 SQL Does Not Translate Logical Names Referencing Source Databases

When you create an extraction or extraction rollup transfer and you want to use a source database that requires the /TYPE access string, you must first enter a DECLARE SCHEMA statement for that database. One method of referencing the source database is to define a logical name, as shown in the following example:

```
$ DEFINE SOURCE_DB "/TYPE=VIDA2/DATABASE=SHIPPING . . .  "
$ SQL
SQL> DECLARE SCHEMA FILENAME SOURCE_DB;
SQL> CREATE TRANSFER FROM_SHIPPING TYPE IS EXTRACTION  . . .
```

However, SQL does not translate the logical name into the access string when the source schema is declared by referring to a logical name. Using the previous example, examination of the transfer definition will show ‘‘From . . . SOURCE_DB’’, instead of ‘‘From . . . /TYPE=VIDA2/DATABASE=SHIPPING . . . ’’.

The transfer will fail to execute properly unless the SOURCE_DB logical name is defined when the copy process runs. A copy process runs as a detached process using the same VMS account that you use to create the transfer. Therefore, the SOURCE_DB logical name will be properly translated if you define it in your LOGIN.COM file. The logical name can also be defined in your group or system logical name tables. The logical name ''SOURCE_DB'' is used here as an example; you can use any name you want.

4.4 Restriction for Extraction Transfers Only

This section describes a restriction that applies to extraction transfers only.

4.4.1 Views Containing Computed_By Columns Cannot Become Tables

In extraction transfers, you can specify a view name in the MOVE TABLES clause of the CREATE TRANSFER statement. The view in the source database becomes a real table, with data of its own, in the target database.

However, if a view contains computed_by columns, Data Distributor cannot materialize the view as a table. Execution of the transfer can fail with the message ''%RDMS-I-BUGCHKDMP, generating bugcheck dump file'' and the bugcheck dump showing an exception at RDMS$$DSDI_BAD_CODE.

4.5 Restrictions for Extraction Rollup Transfers Only

This section describes restrictions that applies to extraction rollup transfers only.

4.5.1 Column Names for Like Tables Must Be Identical

A restriction that applies to extraction rollup transfers (created using the interactive SQL) states that column names of like tables must be identical.

SQL lets you create an extraction rollup transfer of like tables that differ only in the column names. For example, the following fragment shows a transfer definition of this type:

```
MOVE TABLES
    SELECT PLAYER_ID FROM DB1.PLAYERS
        UNION ALL
    SELECT PLAYER_CODE FROM DB2.PLAYERS
        INTO ALL_PLAYERS
```

The PLAYERS table in the two source schemas are identical:

- They have the same number of columns.
- The columns are defined in the same order.

- The data definitions for each column (data type, length, scale) are the same.

They differ only in the two column names they each use, namely PLAYER_ID and PLAYER_CODE.

If you create such a transfer, the definition is accepted by SQL and stored in the transfer database. When a Data Distributor copy process executes the transfer, the transfer fails. The copy process log then logs the following error messages:

- DDAL-E-RDBERR, RDB-E-OBSOLETE_METADATA
- RDMS-F-BAD_SYM

The messages indicate that the PLAYER_CODE column name caused the error.

Extraction rollup transfers must use like column names for transfers to succeed.

4.5.2 SQL Privileges Can Restrict Access to Source Tables

Users with only the SQL SELECT privilege for source databases cannot execute extraction rollup transfers. The following example illustrates this restriction. You want to define an extraction rollup transfer on two source databases, DB1 and DB2. Assume that you have the SELECT privilege on database DB2 and assume that you have the SELECT privilege for all the tables in DB2. If you enter the following transfer definition, you will receive the error message shown at the end of the example.

```
SQL> DECLARE DB1 SCHEMA FILENAME PERS1;
SQL> DECLARE DB2 SCHEMA FILENAME PERS2;
SQL> CREATE TRANSFER ERP_1 TYPE IS EXTRACTION ROLLUP
cont>   MOVE TABLES
cont>      SELECT FIRST.*
cont>      FROM DB1.EMPLOYEES FIRST WHERE FIRST.STATE = "NH"
cont>   UNION ALL
cont>      SELECT SECOND.*
cont>      FROM DB2.EMPLOYEES SECOND WHERE SECOND.STATE = "MA"
cont>   INTO NEW_MA_NH_EMPS
cont>   TO NEW FILENAME DISK1:[DIR1]PERS3
cont>   LOG FILE IS DISK1:[DIR1]PERS3.LOG
cont>   COMMENT IS "Pull test all local"
cont> ;

%RDB-E-NO_PRIV, privilege denied by database facility
```

This is an SQL enforced restriction that is present in Rdb/VMS V4.0. Use the RDO interface to work around this problem.

4.6 Restrictions for Replication Transfers Only

This section contains restrictions for replication transfers only.

4.6.1 Restriction on Fields Named in an Rdb/VMS RDO RSE

All fields that you name in the record selection expression (RSE) of the DEFINE TRANSFER statement must also be included in the SELECT FIELDS clause. If you do not name the fields explicitly in the SELECT FIELDS select-field-name statement or implicitly in the select-field-name clause, Data Distributor returns an error message when the transfer executes. The message indicates that the field name is not found in the symbol table.

4.6.2 Restriction on Columns Named in an SQL Select Expression

All columns that you name in a select expression of the CREATE TRANSFER statement must also be included in the SELECT column-name clause. If you do not name the columns explicitly in the SELECT column-name statement or implicitly in the SELECT ALL clause, Data Distributor returns an error message when the transfer executes. The message indicates that the column name is not found in the symbol table.

4.6.3 Deleting Transfer Definitions

When a replication transfer definition is created, Data Distributor stores that definition in the transfer database. When the transfer runs for the first time, Data Distributor stores that definition in the source database as well. In Data Distributor Version 1.0, a problem arose when you attempted to delete a replication transfer while the RDO utility was attached to the source database.

To delete a replication transfer definition, you first had to start a transaction. Before you could start a transaction, you had to attach to a database. However, if you attached to the source database that contained the transfer definition, the copy process, which needed exclusive access to the source database containing the definition, could not delete the transfer definition. To avoid this problem, before you deleted a replication transfer, you had to attach to a database other than the source database for the replication transfer definition you intended to delete.

For Data Distributor Version 1.1, Version 2.0, and Version 2.1, because the copy process still needs to have exclusive access to the source database, you must detach from the source database before you issue the DROP TRANSFER statement for a replication transfer. But, for Data Distributor Version 1.1, Version 2.0, and Version 2.1, you need not attach to another database to delete a replication transfer.

If the source database is inaccessible or no longer exists and you need to delete the transfer, you can work around the problem as follows:

1. Create a new database with the same name and fields or relations as the inaccessible source database.
2. Locate this new database at the address where the old one was.
3. Reinitialize and then start the transfer.

These steps will enter the transfer definition in the source database.

After the definition has been entered into the RDB$TRANSFER_RELATIONS relation in the source database, stop the transfer. Then issue a DROP TRANSFER statement in SQL or a DELETE TRANSFER statement in RDO to remove the unwanted transfer definition.

4.6.4 Restriction on Two-Way Replication Transfers

Two-way transfers enable two databases to exchange data. A transfer definition created on each database uses the other database as a target. If the transfers defined on each database are the replication type, in certain circumstances the copy process will issue an error message and fail to execute the second transfer. Consider the following example. You have two databases, PERS1 and PERS2, neither of which has been created or accessed by a transfer. You define a replication transfer, XFER1, from PERS1 into PERS2 and execute it. Database PERS2 now contains the table RDB$VINTAGE.

```
SQL> DECLARE SCHEMA FILE PERS1;

SQL> CREATE TRANSFER XFER1 TYPE IS REPLICATION
cont>  MOVE TABLES *
cont>  TO EXISTING FILENAME DISK2:[DIR2]PERS2 WITH NO CHECKING
cont>  LOG FILE IS DISK1:[DIR1]XFER1.LOG
cont> ;
SQL> START TRANSFER XFER1 NOW;
SQL> FINISH;
```

Next, you define a second transfer, XFER2, from PERS2 into PERS1.

```
SQL> DECLARE SCHEMA FILENAME PERS2;

SQL> CREATE TRANSFER XFER2 TYPE IS REPLICATION
cont>  MOVE TABLES *
cont>  TO EXISTING FILENAME DISK1:[DIR1]PERS1 WITH NO CHECKING
cont>  LOG FILE IS DISK2:[DIR2]XFER2.LOG
cont> ;

SQL> FINISH;
```

This causes the copy process to attempt to create the RDB$VINTAGE table twice, which generates the following error message:

```
----- 17-SEP-1990 15:13:42.47 ----- Error -------------------------
%DDAL-E-MISGBLFLD, local field RDB$VINTAGE_TSER not matched to a global field
```

You can avoid this problem using one of the following methods:

- Specify all the tables you want moved and avoid using the wildcard character (*) in the MOVE TABLES clause. This is documented in Chapters 9 and 10 of the *VAX Data Distributor Handbook*.
- If you know that you want to define a two-way replication transfer, define both transfers at the same time. The names of the tables to be moved in each transfer are stored immediately, and because the RDB$VINTAGE table has not yet been created, it is not stored as part of either transfer definition.

4.7 Restrictions for Transferring Into an Existing Database

This section documents the restrictions that apply to transfers that use an existing database as a target.

4.7.1 Transferring a Table to an Existing Database Containing the Same Table Name

If you define a transfer to an existing database and specify a table name in the transfer definition that already exists in the target database, SQL checks to determine if the table was created by the current transfer.

- If the transfer owns the table, then the transfer definition is valid.
- If the transfer does not own the table, SQL issues an error message and the transfer fails. You must use the INTO keyword with the table-name parameter to rename the duplicated table.

However, in SQL Version 4.0, SQL looks for the source table name in the target database instead of the table name specified in the INTO table-name clause. Because the source table name does exist but does not belong to the current transfer, SQL issues an error message declaring that the transfer does not own the table.

The following example illustrates this situation. Assume that EMP_ COLLEGES in the target database PERS_1 is not owned by transfer XYZ.

```
SQL> DECLARE SCHEMA FOR FILENAME "PERSONNEL";
SQL> CREATE TRANSFER XYZ TYPE IS REPLICATION
cont>  MOVE TABLES
cont>    SELECT * FROM COLLEGES INTO EMP_COLLEGES
cont>  TO EXISTING FILENAME DISK1:[DIR1]PERS_1
cont>  LOG FILE IS DISK1:[DIR1]XYZ.LOG
cont>  ;

%SQL-F-TRANOTOWNER, This transfer is not the owner of table
                    EMP_COLLEGES
```

You can resolve this problem by using the WITH NO CHECKING qualifier with the CREATE TRANSFER statement, as shown in the following example:

```
SQL> DECLARE SCHEMA FOR FILENAME "PERSONNEL";
SQL> CREATE TRANSFER XYZ TYPE IS REPLICATION
cont>  MOVE TABLES
cont>    SELECT * FROM COLLEGES INTO EMP_COLLEGES
cont>  TO EXISTING FILENAME DISK1:[DIR1]PERS_1 WITH NO CHECKING
cont>  LOG FILE IS DISK1:[DIR1]XYZ.LOG
cont>  ;
```

4.7.2 No Extraction Rollup Into Existing Databases

You cannot create an extraction rollup transfer into an existing database. If you need to merge disparate tables into one physical database, you can do so by defining several transfers into an existing database. If you also require the union of data from several like tables, which currently exist in separate databases, you can do the same and then use a view to accomplish the union.

4.7.3 No Union of Source and Target Data

If the source and target databases have tables of identical definition, it might seem natural to transfer data from the source table into the target table. This is not permitted in Data Distributor. Instead you must transfer the source table into the target database under a new name, thereby keeping the two sets of data separated.

4.7.4 No DEFINE TRANSFER Into an Existing Database

DEFINE TRANSFER is the RDO equivalent of SQL's CREATE TRANSFER statement. The DEFINE TRANSFER statement has not been changed to support transfers to existing databases.

4.8 Cluster Restrictions

This section documents the restrictions that apply to cluster installations of Data Distributor.

4.8.1 Load Balancing in a Cluster

When there are multiple transfers defined in a cluster transfer database, each node's transfer monitor is supposed to participate in the scheduling activity for transfers. However, sometimes one node does all the work.

For example, assume that three transfers have been defined, but none has a schedule definition. Transfer monitors are running on Nodes A, B, and C, and each is hibernating. If, on Node A, you define schedules for the three transfers, only the transfer monitor on Node A becomes aware of the new schedules. Therefore, all transfer operations are done by Node A.

If, on Node B, you issue a START TRANSFER statement from SQL or RDO, the transfer monitor on Node B subsequently participates in the scheduling of existing transfers. Similarly, you can issue a START TRANSFER statement from Node C to further balance the work load.