

RealityX

Release 4.1

Differences Between Releases 4.0 and 4.1

All trademarks including but not limited to brand names, logos and product names referred to in this document are trademarks or registered trademarks of Northgate Information Solutions UK Limited (Northgate) or where appropriate a third party.

This document is protected by laws in England and other countries. Unauthorised use, transmission, reproduction, distribution or storage in any form or by any means in whole or in part is prohibited unless expressly authorised in writing by Northgate. In the event of any such violations or attempted violations of this notice, Northgate reserves all rights it has in contract and in law, including without limitation, the right to terminate the contract without notice.

© Copyright Northgate Information Solutions UK Limited, 1996.

Document No. UM70006456A
June 1996

Northgate Information Solutions UK Limited
Peoplebuilding 2
Peoplebuilding Estate
Maylands Avenue
Hemel Hempstead
Herts
HP2 4NW

Tel +44 (0)1442 232424

Fax +44 (0)1442 256454

www.northgate-is.com

Contents

Chapter 1 About this Manual

Purpose of this Manual	1-2
Contents	1-3
Related Documents	1-4
RealityX 4.0 Reference Manuals	1-4
Manuals for 7.0/Pre-7.0 Upgrade	1-4
Conventions	1-5
User Comments	1-7

Chapter 2 Security Enhancements

Password Integration	2-2
realusers	2-3
Restrictions on Access to the UNIX Shell	2-5
Security Mechanism for SYS, rdb or Unix Verbs	2-5
Security Files for SYS, rdb or Unix Verbs	2-5
Security Profile for SYS, rdb or Unix Verbs	2-6
Security Profile for DIRVIEW	2-6
Protection of Important Database Files and Transaction Logs	2-7
Important Database Files	2-7
Transaction Log Protection	2-7

Chapter 3 PLID to Port Mapping

Description of Mapping Mechanism	3-2
Devices File	3-3
PLID and Port Specifications	3-3
Keywords	3-4
Valid Keywords	3-4

Chapter 4 TANDEM Verb

Description of TANDEM Verb	4-2
----------------------------------	-----

Chapter 5 Multideck Save / Restore Utility - dbsave

Overview.....	5-2
Description of dbsave.....	5-3
Purpose	5-3
Syntax.....	5-3
Options	5-3
Restrictions.....	5-4
Files	5-4
Main Menu	5-4
Multideck Calibration	5-4
Calibrate System.....	5-4
Calibrate Account.....	5-5
Multideck System Save	5-5
Multideck System Restore.....	5-7
Configuration File.....	5-8
Example.....	5-9

Chapter 6 Structured Query Language/Open Database Connectivity

Introducing SQL/ODBC	6-2
----------------------------	-----

Chapter 7 Miscellaneous Differences

Tape Image Devices	7-2
Using an Ordinary UNIX File	7-2
Using a Named Pipe.....	7-2
MAKE-SPECIAL Command	7-4
SET-PRIORITY - Changing User Process Priority.....	7-9
T-STACKER - Managing Multi-Cassette FILE-SAVEs and RESTOREs.....	7-10
Messaging by PLID	7-12

Index

Chapter 1

About this Manual

This chapter describes the different sections of this manual and conventions used.

Purpose of this Manual

This manual is intended to be a resource for users upgrading from RealityX Release 4.0 to Release 4.1. It describes the new features supported by Release 4.1.

The following new features are described:

- **Security enhancements**, including, password integration, restrictions on access to the UNIX shell and protection of important database files and transaction logs.
- **PLID to port mapping**, a mechanism for allocating a consistent port number to user processes at the same terminal connection, based on the Physical Location Identifier (PLID).
- **TANDEM**, a TCL verb which enables a database user to monitor the terminal activity of another database user.
- **Multideck save/restore utility `dbsave`** which enables a database to be saved and restored more quickly by grouping the database into a number of equally sized sets of accounts and then saving (or restoring) the sets of accounts in parallel using multiple tape devices.
- **Structured Query Language/Open Database Connectivity (SQL/ODBC)** which enables users to access data on a RealityX database from an ODBC-compliant PC application using SQL commands.

Contents

Chapter 1, About this Manual, describes the different sections of the manual and any conventions used.

Chapter 2, Security Enhancements, describes the security enhancements supported by RealityX Release 4.1 which integrate RealityX and UNIX security, including: password integration, restrictions on Shell access and protection of important database files and transaction logs.

Chapter 3, PLID to Port Mapping, describes a mechanism for allocating a consistent port number to user processes at the same terminal connection, based on the Physical Location Identifier (PLID).

Chapter 4, TANDEM Verb, describes the specification and operation of the TCL verb TANDEM.

Chapter 5, Multideck Save/Restore Utility, describes the use of the **dbsave** utility to carry out Multideck Save and Restore procedures on a RealityX 4.1 database.

Chapter 6, Structured Query Language/Open Database Connectivity, introduces the new SQL/ODBC software developed for RealityX Release 4.1. For a detailed description of this feature, refer to the *SQL/ODBC for RealityX - Administrator's Guide*.

Chapter 7 Miscellaneous Differences, describes differences between 4.0 and 4.1, other than the features described in Chapters 2 to 6.

Related Documents

RealityX 4.0 Reference Manuals

Reality X Reference Manual Volume 1: General

RealityX Reference Manual Volume 2: Operation

RealityX Reference Manual Volume 3: Administration

ENGLISH Reference Manual (Vol. 4)

PROC Reference Manual (Vol. 5)

EDITOR Reference Manual (Vol. 6)

Screen Editor Reference Manual (Vol. 6)

DATA/BASIC Reference Manual (Vol. 7)

General Utilities and Printing (Vol. 8)

RealityX Differences Between Releases 3.1 and 4.0

Manuals for 7.0/Pre-7.0 Upgrade

If you are upgrading from a pre-7.0 REALITY, the following manuals may be helpful:

User's Guide to the REALITY Migration Utilities (UM30002112A)

RealityX Differences Supplement (Differences between REALITY 7.0 and RealityX 3.1)

Conventions

The following conventions are used in this manual:

Text	Bold text shown in this typeface is used to indicate input which must be typed at the terminal.
Text	Text shown in this typeface is used to show text that is output to the screen.
Bold text	Bold text in syntax descriptions represents characters typed exactly as shown. For example WHO
<i>Text</i>	Characters or words in italics indicate parameters which must be supplied by the user. For example in LIST <i>file-name</i> the parameter <i>file-name</i> is italicised to indicate that you must supply the name of the actual file defined on your system. Italic text is also used for titles of documents referred to by this document.
[<i>Optional param</i>]	Square brackets used in UNIX command syntax enclose options and optional parameters. For example in • Ls [options] [file--name] The list command can be used with or without options and with or without a file name. Without parameters it displays a simple list of the current directory.
SMALL CAPITALS	Small capitals are used for the names of keys such as RETURN.
CTRL+X	Two (or more) key names joined by a plus sign (+) indicate a combination of keys, where the first key(s) must be held down while the second (or last) is pressed. For example, CTRL+X indicates that the CTRL key must be held down while the X key is pressed.

Enter	<p>To enter means to type text then press RETURN. For instance, 'Enter the WHO command' means type WHO, then press RETURN.</p> <p>In general, the RETURN key (shown as ENTER or ↵ on some keyboards) must be used to complete all terminal input unless otherwise specified.</p>
Press	<p>Press single key or key combination, but do not press RETURN afterwards.</p>
X'nn'	<p>This denotes a hexadecimal value.</p>

User Comments

A Comment Sheet is included at the front of this manual. If you find any errors or have any suggestions for improvements in the manual please complete and return the form. If it has already been used then send your comments to the Technical Publications Manager at the address on the title page.

Chapter 2

Security Enhancements

This chapter describes the Release 4.1 security enhancements which integrate RealityX and UNIX security, closing a number of security loopholes. Enhancements include:

- Integration of RealityX user id and password
- Restrictions on the use of the SYS command
- Restrictions on shell access from the rdb debugger
- Restrictions on UNIX verb definitions
- Restrictions on the use of directory view
- Protection of important database files
- Transaction log protection

Password Integration

RealityX Release 4.1 gives you the option of integrating user-ids on a RealityX database with corresponding user-ids on the UNIX system. Each RealityX user-id and corresponding UNIX user-id then have the same password and any changes made in the RealityX database, using the SSM user maintenance screen or PASSWORD verb, are also be made in the UNIX environment in **/etc/passwd**. Conversely, any changes made in **/etc/passwd** are reflected in the RealityX USERS file.

Password integration is of most benefit when each user has a unique UNIX user-id and corresponding RealityX user-id. A user requiring access to two separate databases should be given two UNIX user-ids, so that the RealityX and UNIX passwords can be kept in step.

To configure a database with integrated RealityX/UNIX user-ids and passwords you must use the **realusers** utility at the UNIX shell. Refer to Chapter 4 for for a description of **realusers**. The procedure is as follows:

1. Enter **su** with appropriate password at the UNIX shell to become super-user.

2. Enter:

```
realusers -r database-name
```

to register the specified database to use UNIX passwords.

3. Enter:

```
realusers -u database-name
```

to create matching UNIX user-ids for any existing RealityX user-ids on the database.

realusers

Purpose	UNIX/RealityX password integration.
Syntax	realusers <i>options</i> [<i>database</i>]
Restrictions	Super-user (root) only. The specified database must be inactive.
Parameters	<i>database</i> The database to be registered to use UNIX passwords.
Options	-n No write. Just displays what it would otherwise do. -q Display the register status of the database. -r Register a database to use UNIX system passwords. -R De-register a database from using UNIX system passwords. -u Create/delete UNIX users so as to match those in USERS file.
Registering a Database	realusers with the -r option writes the database path into /etc/realdbases and puts the parameter SystemPassword=1 into the database's config file. It then connects to the database and replaces any existing RealityX password attribute in the USERS file with a null.
Updating UNIX Users	Once the database is registered to use UNIX passwords, realusers with the -u option is executed to bring user entries in /etc/passwd in-line with RealityX USERS file entries. It creates UNIX user ids to match RealityX user ids already on the database and removes any UNIX user entries for which there are no matching RealityX users.
Users Files	In order to create a new UNIX user id, realusers needs to know what to put in /etc/passwd . This is driven by the configuration file users which is looked for first in £REALDBPATH/configs , then in £REALROOT/files . See below for a sample default users file.

A **users** file is supplied with the following defaults set up:

```
# This file specifies how new users will be created
# by the realusers command
#
group other
home /home/realusers
shell /bin/sh
comment RealityX User
# Default password if none supplied. Need not be set.
# password welcome
# User id range start to end. End need not be set.
start 1000
#end 2000
```

Start and End User Id Numbers

The start and end numbers in a database's **users** file indicate the range of numbers allocated to user ids on the associated database. **realusers -u** when run on a particular database will only be able to remove user ids which have numbers within the range specified in the associated **users** file.

The minimum value for start is 500. If end is not set, then there is no upper limit. Existing user ids outside this range will not be modified.

reality -u and -U Options

The **reality -u** option which allows you to log to a database under a different user name and the **reality -U** option which suppresses the default user id and takes you to the 'Logon Please' are not available when user password integration is enabled. You must log on under your current user id.

Restrictions on Access to the UNIX Shell

RealityX provides a mechanism to restrict access to the UNIX shell by the following:

- **SYS** command which allows access to a UNIX shell, either to execute a 'single shot' command, or interactively until EOF.
- **rdb !** command which allows access to a UNIX shell to execute a single-shot command.
- UNIX verb definitions which can be created by RealityX users to access a UNIX shell to execute a 'single' shot command.
- Directory view which allows RealityX users to view and update a UNIX directory as if it were a RealityX file. This enables the majority of the UNIX file system to be accessed from RealityX.

On RealityX Release 4.0 and earlier, access to the UNIX shell using **SYS**, **rdb !**, a user-specified UNIX verb definition, or **DIR-VIEW**, is unrestricted. Operations are permitted based on UNIX security alone. This is potentially a security risk.

On RealityX Release 4.1 the use of **SYS**, **rdb !**, user-specified UNIX verb definitions or **DIR-VIEW** for non-privileged users may be restricted.

Security Mechanism for **SYS**, **rdb** or UNIX Verbs

RealityX Release 4.1 allows the systems administrator to specify those **SYS** commands, **rdb !**, or UNIX verb definitions that are to be allowed. Those not explicitly allowed, are disallowed. This is configurable on a per database basis.

RealityX compares a **SYS** command, **rdb !**, or verb definition against lists of allowed commands/verbs held in UNIX files.

Security Files for **SYS**, **rdb** or UNIX Verbs

The files are:

- | | |
|--------------------|---|
| sys.needed | This lists the SYS commands that are essential for running RealityX. |
| sys.user | This lists the SYS commands that the user wishes to explicitly permit. |
| sys.profile | This lists the SYS commands that may be run by any user with the RealityX security profile ' <i>profile</i> '. |

The same files are used by **rdb** and UNIX verb definitions.

These files are each looked for first in **£REALDBPATH/configs** and, if not found, then in **£REALROOT/files**.

The syntax of the lists in each security file is simple. A shell command being run from RealityX must match one of the lines in the appropriate file. If the line ends in an ellipsis ("..."), the command must match only up to the ellipsis. Without the ellipsis, the command must match the line exactly. For example, **sys.needed** may have:

```
pdump -p £REALDBASE
tlrestore...
```

Security Profile for SYS, rdb or UNIX Verbs

A new attribute, *shell security level*, is defined in a user's security profile in the SECURITY file. This is configured using SSM. This can be:

- 0 Allow only essential shell commands.
- 1 Allow both essential and user-defined commands.
- 2 No restrictions.

The supplied SYSMAN security profile has a *shell security level* of 2. The supplied DEFAULT security profile also has a *shell security level* of 2.

Security Profile for DIR-VIEW

The DIR-VIEW verb is also enabled/disabled by a new attribute in the user's security profile. The supplied 'DEFAULT' profile and the SYSMAN profile both have DIR-VIEW enabled.

Protection of Important Database Files and Transaction Logs

Important Database Files

When a RealityX Release 4.1 database is created, update and retrieval locks are set automatically on a number of important files.

The files protected are:

```
SYSTEM
SYSMAN MD SECURITY* USERS* SYSPL
SYSFILES MD
SYSPROG MD
```

Note: Those marked with an asterisk (*) are read protected, as well.

A new PROC tool, INSTALL-DEFAULT-PROTECTION, is provided in SYSMAN to set these keys. The keys used are 'SYSREAD' and 'SYSUPDATE'.

Transaction Log Protection

RealityX Release 4.1 has tightened up the UNIX permissions on clean logs.

The default is to allow access to the clean logs only to the database owner. Therefore, the clean log directory permissions will be '711' and clean log permissions will be created '600'. Clean log permissions can be configured to allow group access '660'.

Chapter 3

PLID to Port Mapping

This chapter describes a mechanism for allocating a consistent port number to user processes at the same terminal connection, based on the Physical Location Identifier (PLID).

Description of Mapping Mechanism

PLID to Port Mapping enables fixed RealityX port numbers to be allocated to incoming terminal connections using their PLIDs (Physical Location Identifiers), so that whenever a user logs in at a particular terminal, the same port number is allocated each time. The allocated port number will be unique and consistent.

Before connecting to the database, RealityX requests its PLID from the UNIX Connect Session Manager, then performs a look-up in a 'PLID to Port' mapping file called **devices**. If the PLID matches a valid entry in the **devices** file, the corresponding port number is allocated to the terminal connection. Provided that the RealityX process is not a background server, this port number is then requested when making the connection to the RealityX database. If the PLID does not exist in the **devices** file, the next available virtual port number is allocated

The **devices** file is also used:

- To determine the PLID for background processes which are started on a particular port number (e.g. TIPHs and LOGONs).
- By TCL verbs, such as LOGON, and PH-START, which reference a port number. It is used to look up the PLID associated with the specified port number and determine the identity of the UNIX device file for that port number.

PLID to Port mapping is enabled by creating and configuring the **devices** file as described below.

The PLID to Port mapping facility is automatically enabled by the presence of the **devices** file in the RealityX database **configs** directory, or globally in **£REALROOT/files**. RealityX looks in the **configs** directory first and then in **£REALROOT/files**, using the first **devices** file it finds. If no **devices** file is found, or if the file contains no valid entries, RealityX will assign virtual port numbers to all connections.

Devices File

The **devices** file can contain two types of lines: Comment lines and Data lines.

Comment lines are blank or have a '#' as their first non-blank character. These lines are ignored. Data lines have the format: `PLIDSpec PortSpec [Keywords] [#Comment]`

Keywords and a trailing comment (following a '#' character) are optional.

Some sample **devices** file entries are shown below.

A template file is available in `£REALROOT/files` called **device.tmpl**.

Note: This example shows a wider variety of PLID types than would normally be found in a single system.

```
#
# Example devices file entries
#

UNIX-0-console          0      # Console
UNIX-0-tty[01-03]      1          # Moto V3 (Ports 1 to 3)
UNIX-0-tty[11-99]     11          # Moto V3 (Ports 11 to 99)

UNIX-0-tty0[00-3f]    100     HEX      # Moto V3 (Ports 100 to 163)
UNIX-0-tty1[00-3f]    164     HEX      # Moto V3 (Ports 164 to 227)

UNIX-0-term[0-199]    0      NOVIRT   # Moto V4 (Ports 0 to 199)

UNIX-0-rt00[01-20]    201     HEX      # ANNEX (Ports 201 TO 232)

inet-192.67.50.5-[0-31] 300          # Telnet (Ports 300 to 331)
TNET-00802D0045C6-[1-32] 351          # Telnet (Ports 351 to 382)
```

PLID and Port Specifications

There are two supported formats of Data line with differing PLID and port specifications.

1. Exact Match

e.g. `UNIX-0-console 0 # Console`

This form specifies a single PLID which maps to a single port number. The PLID must match exactly.

2. PLID Range

e.g. UNIX-0-tty[01-03] 1 # Moto V3 (Ports 1 to 3)

This form specifies a range of PLIDs from UNIX-0-tty01 to UNIX-0-tty03 which are mapped onto ports in the range 1 to 3. The port number specified is used for the first PLID in the range and subsequent PLIDs are assigned increasing port numbers according to their position in the range.

In this format the range specified in brackets must be a numeric range with no gaps and must be at the right hand end of the PLID specification. The numbers specified may be in decimal (default), or hexadecimal (indicated by the HEX keyword), and may be of fixed or variable length as appropriate to the PLID format.

Keywords

The optional Keywords field may contain one or more keywords (separated by spaces) which have two main uses.

1. To describe the format of the final part of the PLID and control the conversion between PLID format and relative port number. These keywords are named according to the format they describe.
2. To specify attributes associated with the PLID. They are named according to the attribute they specify.

Valid Keywords

The current list of valid keywords is:

DEC	Specifies that the PLID range is given in decimal numbers. This is the default and need not be specified explicitly.
HEX	Specifies that the PLID range is given in hexadecimal numbers.
UHEX	Specifies that the PLID range is given in upper-case hexadecimal numbers.
NOVIRT	Specifies that a connection should fail rather than assign a Virtual Port number to this PLID.

If no keyword is specified to define the algorithm for converting between PLIDs and relative port numbers, the final part of the PLID is assumed to be a decimal number.

Chapter 4

TANDEM Verb

This chapter describes the specification and operation of the TANDEM command at TCL.

Description of TANDEM Verb

Purpose	Connects the TANDEM user's port to a specified target port so that target port activity is displayed at the TANDEM user's terminal for monitoring purposes. Also, in feed mode, it allows the TANDEM user's port to pass input to the target user process.
Syntax	TANDEM <i>target-port</i>
Syntax Elements	<i>target-port</i> is the number of the port to be monitored.
Restrictions	Both users must be connected to the same RealityX database. The target port must be enabled for tandem operation by entering ALLOW-TANDEM (E at the target terminal. Refer to the topic <i>ALLOW-TANDEM</i> earlier in this chapter.
Restrictions with Field Read	Also, the following restrictions apply when TANDEM is used in conjunction with Field Read. <ol style="list-style-type: none">1. User input to the target port is not displayed on the TANDEM user's terminal.2. To minimise differences between the target and TANDEM user's terminal displays, the TANDEM user's port must be set up in exactly the same terminal mode as the target port before commencing tandem operation. For example, if Field Read operation is disabled on the target port, it must be disabled on the TANDEM user's port. If the Field Read terminal executive (T.E.) is loaded and Field Read is enabled on the target port, then the T.E. must be loaded and Field Read enabled on the TANDEM user's port.3. You cannot enter the Tandem commands using the menu prompt, described below under the topic <i>Tandem Commands</i>. You must enter the complete command sequence in one go. For example, to select feed mode you must enter CTRL+_f RETURN.
Tandem Modes	Tandem operation is enabled at the target port by ALLOW-TANDEM (E and can be disabled again using ALLOW-TANDEM (D). TANDEM connects the user's port to the specified target port and operates with the target port in one of two modes: <ul style="list-style-type: none">• View-only mode, where all activity on the target port is also displayed on the TANDEM user's terminal.• Feed mode, where input to the TANDEM user's port is passed to the target user process, except for the break sequences described below. All target input/output is displayed on both the target and TANDEM users' terminals.

Tandem Commands

To change a mode or quit Tandem operation altogether, you must break the current mode by entering `CTRL+KEY`, where *KEY* is the under-score character (`_`) by default, or some other character selected at the menu prompt shown below. Pressing `CTRL+KEY` displays the menu prompt:

```
q:quit f:feed v:view-only b<key>:break-key :
```

Enter:

- | | |
|-------------|---------------------------|
| f | To select feed mode. |
| v | To select view-only mode. |
| bKEY | To change the break key. |
| q | To quit TANDEM operation. |

When tandem mode is requested, it becomes effective at the start of a read or a write operation to the port. Any active read or write is allowed to complete before Tandem mode is started.

If after entering TANDEM, no activity occurs on the target port after a predetermined time, TANDEM times out, displaying the message 'port isn't responding'. This message will also be displayed if tandem operation is disabled on the Target port.

Example

```
:TANDEM 401
```

Allows you to monitor on your current terminal the screen activity on the terminal connected to port 401.

Chapter 5

Multideck Save / Restore Utility - **dbsave**

This chapter describes the use of the **dbsave** utility to carry out Multideck Save and Restore procedures on a RealityX Release 4.1 database.

Overview

The Multideck Save/Restore utility, executed by entering **dbsave** at the UNIX shell, increases the speed of a database file-save/restore by providing the capability to group the database into a number of logical groups of accounts, based on the size of accounts and the length of time it takes to save each account. It then saves (and restores) the groups in parallel using multiple tape devices. The grouping attempts to make each set of accounts save in about the same time.

Calculation of equal size groups of accounts is referred to as 'Calibration'. This must be carried out first, using Option 3 on the Multideck Save/Restore Menu, before you can execute a multideck save. Re-calibration of the groups can be performed as frequently as required. The number of groups must be equal to the number of decks available to carry out the Save or Restore which is configurable.

Once the groups have been established, a Save is initiated using the pre-configured set of tape decks. This starts a number of RealityX Save processes (one Save process per group). Each Save process saves the set of accounts defined in that group. The Save processes run in parallel. As all the sets of accounts are about the same size, the Save of each group should complete in about the same time.

A RealityX database saved using this technique can be restored in a similar way by using the Multideck Save/Restore utility to perform parallel ACCOUNT-RESTORE's. Multideck Restores can only be performed using the same number of decks as the corresponding Save used.

The Save tapes do not have to be loaded on exactly the same decks to restore, for example, if the tape is saved on deck 2, it is valid to restore it on deck 4, assuming both decks 2 and 4 are configured for Multideck Save/Restore use. Before the restore starts, all tape labels are read to check that the tapes from a complete save set are loaded. The labels written by the Save processes are of the format:

```
MULTI-DECK SAVE USING LISTn hh:mm:ss dd mmm yyyy
```

This technique is not suitable for saving or restoring databases in which the accounts cannot be easily divided up into equal sets, for example, where there is only one application which is wholly contained in one account.

Multideck Save/Restore will operate with File Store Databases or Partition Databases. It uses RealityX save/restore techniques and cannot be used with other back-up schemes such as, **cpio** and **dd**.

Description of dbsave

Purpose To perform a Multideck Save or Restore of a RealityX database.

Syntax

```
dbsave [-d database] [ -C | -c ]
dbsave [-d database] [ -R | -r ]
dbsave [-d database] [ -S | -s ] [ -t time ]
dbsave [-d database] [ -a account ]
dbsave [-d database] [ -v ]
```

Options Options available are as follows:

Note: If you specify no options, other than the database name, a menu is displayed, as described under the topic *Main Menu*.

- a** *account* Calibrate named account.
- c** Calibrate database. Executes the procedure described under the topic *Calibrate System*.
- C** Calibrate (roughly) database.
- d** *database* Database to be saved or restored. If a database name is not specified, £REALDBASE is used (if set). If REALDBASE is not defined anywhere, the database name is prompted for.
- r** Restore database, suppressing file resizing. Executes the procedure described under the topic *Multideck System Restore*.
- R** Restore database and resize files. Executes the procedure described under the topic *Multideck System Restore*.
- s** Save database and verify save. Executes the procedure described under the topic *Multideck System Save*.
- S** Save database, suppressing verify save. Executes the procedure described under the topic *Multideck System Save*.
- t** *time* Time to start save. If a time is not specified, the save is started immediately
- v** Verify save.

Restrictions

`dbsave` assumes that the specified tape devices all have the same capacity and does not support multi-reel operation.

Files

`dbsave` resides in `REALROOT/bin`.

Main Menu

Entering `dbsave` at the shell prompt without options, other than specifying the database name, displays a menu similar to the following:

```
Multideck Save and Restore Menu
```

1. Multideck System Save
2. Multideck System Restore
3. Multideck Calibration

```
Enter Option (1-3):
```

Before performing a Multideck System Save you must calibrate the database using Option 3 Multideck Calibration. If you do not do this, the Save will fail with the message:

```
***FAILED***No account calibration statistics available
```

Multideck Calibration

On selecting option 3 from the main menu, the following calibration sub-menu is displayed:

```
Multideck Calibration
```

1. Calibrate system
2. Calibrate account

```
Enter Option (1-2):
```

Calibrate System

This procedure is run by selecting option 1 on the Calibration sub-menu or by entering `dbsave -c` or `-C` at the UNIX shell to generate file statistics for the whole database.

Note: This procedure executed from the main menu, or using `dbsave` with the `-c` option, carries out a full calibration. When executed using `dbsave` with the `-C` option, it carries out a rough calibration only, calculating the approximate size of accounts by totalling their file modulus. Rough calibration is therefore much quicker than the full calibration of a database. Where a full calibration may take several hours, the rough calibration may only take a few minutes.

The following procedure is carried out:

1. It selects each account in turn and generates file statistics for it.
2. It processes the statistics just generated and writes the results into a data section of STAT-FILE called CALIBRATION, using the account name as the item-id.

Each account item comprises four attributes, as follows:

```
001 Time taken to save account
002 Total size of account (including in group and indirect)
003 In group size
004 Indirect size
```

Calibrate Account

This procedure is run by selecting option 2 on the Calibration sub-menu, or by entering **dbsave -a account** at the UNIX shell to re-calibrate a single account, for example, if a new account has been restored, or if an existing account has changed considerably.

Multideck System Save

On selecting option 1 on the main menu, or entering **dbsave** with the **-s** or **-S** options at the UNIX shell, the utility interrogates the Multideck configuration file (described later in this chapter) to obtain details about which tape devices are to be used, and then displays a pre-test screen similar to the following.

```
Multideck System Save
```

```
Using deck selection:
```

```
TAPE x
TAPE y
TAPE z
```

```
Pre-test will attach, rewind, write then rewind decks.
Load each deck with write enabled medium
Then press 'y' when ready to continue:
```

On entering 'Y', the pre-tests are started, and once successfully completed, the appropriate number of account lists are created. The lists are written to a data section of STAT-FILE called CALIBRATIONLISTS. The lists are called LIST1, LIST2 etc.

All default SYSTEM accounts are automatically put into LIST1, including SYSTEM, SYSMAN, SYSPROG, SYSFILES, DENAT, ENGLISH-TUTORIAL, HOTEL, UPGRADE.ACCOUNT and SQLDEMO.

If the Multideck System Save is executed from the main menu, you are prompted with the option to sleep a specified time before commencing the Saves, as follows:

```
Enter time to start the Saves (hh:mm) or return to start now :
```

Entering a time displays the message:

```
Sleeping until hh:mm hours...
```

If the Save is executed using the **-S** or **-s** option, no prompt is given. Any sleep time must be specified using the **-t** option, otherwise the Save is started immediately.

The Multideck System Saves are then started and constantly monitored until all of them have finished. The monitoring is displayed in a format similar to the following:

Multideck System Save

```
Starting Save using list 3
Starting Save using list 4
Starting Save using list 6
Sleeping for 5 seconds
Monitoring 3 Save Processes
Save Process using deck 4 complete with no errors detected
Save Process using deck 6 complete with no errors detected
Save Process using deck 3 complete with no errors detected
System Save COMPLETE
```

When all Save processes have completed successfully, a Multideck verify-save is started and monitored, except when the **-S** option is executed, in which case the verify save is suppressed and the procedure terminated.

Monitoring is displayed in a format similar to the following:

```
Starting verify using list 3
Starting verify using list 4
Starting verify using list 6
Sleeping for 5 seconds
Monitoring 3 verify processes
Verify process using deck 4 complete with no errors detected.
Verify process using deck 6 complete with no errors detected.
Verify process using deck 3 complete with no errors detected.
system verify COMPLETE
```

If any errors occurred, they will also be displayed. For example:

```
Error detected on Save process using deck 4
Errors detected
System Save INCOMPLETE
```

The error messages displayed should give an indication as to why the process has failed, but for further details of errors encountered, you must look in the file `/tmp/db_ERRx`, where x is the number of the tape deck which failed.

Multideck System Restore

On selecting option 2 on the main menu, or entering **dbsave** with the **-r** or **-R** options at the UNIX shell, the utility interrogates the Multideck configuration file (described later in this chapter) to obtain details about which tape devices are to be used and then displays a pre-test screen similar to the following:

```
Multideck System Restore
```

```
Using a deck selection:
```

```
TAPE x
TAPE y
TAPE z
```

```
Pre-test will attach, rewind, read then rewind decks.
Load each deck with write protected medium
Then press 'Y' when ready to continue:
```

On entering 'Y', the pre-tests are started.

If the Restore is executed from the main menu, you are then asked about file sizing with the prompt:

```
Do you want to resize files during restores (y/n) ? :
```

If the Restore is executed using the **-r** or **-R** option, this prompt is not displayed. Instead, file sizing will be run automatically if you use the **-R** option and will not be carried out if you use the **-r** option.

Entering 'y' will cause all the Restore processes to run with resizing of files, if selected, and where specified by attribute 13 of a file definition item. This will result in a slower restore time. Restore processes are started and constantly monitored until all of them have finished.

Monitoring will be displayed in a format similar to the following:

Multideck System Restore

```
Starting Restore using list 3
Starting Restore using list 4
Starting Restore using list 6
Sleeping for 5 seconds
Monitoring 3 Restore processes
Restore Process using deck 4 complete with no errors detected
Restore Process using deck 6 complete with no errors detected
Restore Process using deck 3 complete with no errors detected
System Restore COMPLETE
```

If any errors occurred, they will also be displayed. For example:

```
Error detected on Restore process using deck 4
Errors detected
System Restore INCOMPLETE
```

If this occurs, retry Option 2. If it still fails, you will need to contact your Support representative.

The error messages displayed should give an indication as to why the process failed, but for further details of any errors encountered you must look in the file `/tmp/db_ERRx`, where x is the number of the tape deck which failed.

Configuration File

The Multideck Save/Restore utility uses a tape deck configuration file located in the database **configs** directory, called **dbsave_config**. The format of this file is simple and consists of one line of entries, where each entry specifies the number of a RealityX tape device to be used for the Multideck Save/Restore. Each entry must be configured as a tape device in the normal way in the database **config** file.

The **TapeDevSizen** parameter should be defined in the database **config** file for each tape device used for the Multideck Save. **TapeDevSizen** defines the maximum amount of data in Megabytes that can be stored by the tape device and is used by the calibration process when allocating a set of accounts to that device. This prevents the save from exceeding the tape size.

Example

The following example of configuration file format defines that tape devices 3, 4 and 6 are to be used:

```
3 4 6
```

This example would generate three groups of accounts to be saved. If the **dbsave_config** file is missing, the utility will fail, displaying the error message:

```
Cannot read MultiDeck configuration file, unable to continue
```

Chapter 6

Structured Query Language/Open Database Connectivity

This chapter introduces the new SQL/ODBC feature for RealityX available on Release 4.1.

Introducing SQL/ODBC

SQL (Structured Query Language) is designed for retrieving and manipulating data in a relational database.

Open Database Connectivity (ODBC) is a Microsoft standard application programming interface which allows any ODBC-compliant application to access data in any DBMS, using SQL as the standard means of formulating queries. An ODBC-compliant application can therefore extract data from different databases, independent of the type of DBMS from which they derive their data.

RealityX Release 4.1 provides an SQL engine which interprets SQL queries, enabling an SQL user to access data in a RealityX database. Queries may originate from the TCL interface (SQL verb) or from an Open Database Connectivity (ODBC) compliant server.

An ODBC SQL server is provided on the RealityX host which communicates with a ODBC/SQL client (RealSQL driver) resident on a PC. The client server interface uses DDA communication across a LAN. The ODBC compliance enables Microsoft ODBC PC applications to access the RealityX database using SQL commands. The ODBC interface conforms to the 'Core Set' of ODBC functions.

An SQLDEMO account is provided on a Release 4.1 database. To see the range of SQL queries that can be performed on a RealityX database, logon to the SQLDEMO account and run the program SQLDEMO.

Before SQL can be used, an SQL environment must be set up for each file on the RealityX database. This is carried out using the SQL Maintenance (SQLM) utility.

The SQLM utility is used to create SQL definitions for existing files, update current table and column definitions and display table and column definitions. SQLM accepts a selected list of specified file or table names. If no file or table names are passed, it will prompt the user when an SQLM menu option is selected.

For a more detailed introduction to this feature, refer to the *SQL/ODBC for RealityX - Administrator's Guide*, supplied with the RealityX SQL/ODBC product.

Chapter 7

Miscellaneous Differences

In addition to the major features described in Chapters 2 to 7, Release 4.1 supports a number of other features which are described in this chapter. They include:

- Tape image devices
- MAKE-SPECIAL command
- SET-PRIORITY command
- T-STACKER command
- Messaging by PLID

Tape Image Devices

A tape image device is an ordinary UNIX file or a named pipe which is used to emulate a RealityX tape device.

Using an Ordinary UNIX File

For example, to configure a tape image device using the UNIX file **tape-image** in your present working directory, proceed as follows:

1. Create the UNIX file.

```
touch tape-image
```

2. Configure a tape device in the database configuration file (**dbase/configs/config**) to be a tape image device. For example

```
Tape3=/user1/daveh/tape-image  
TapeDevType=9
```

Note: A tape image device is Tape Device Type 9.

3. Attach to RealityX tape device 3

```
T-ATT 3
```

You can now use the complete set of TCL tape commands to perform tape operations on the tape image device as through it is a normal tape unit.

Using a Named Pipe

For example, to configure a tape image device using the a named pipe **pip** in your present working directory, proceed as follows:

1. Create the pipe.

```
/etc/mknod pip p
```

2. Configure a tape device in the database configuration file (**dbase/configs/config**) to be a tape image device. For example

```
Tape3=/user1/daveh/pip  
TapeDevType=9
```

Note: A tape image device is Tape Device Type 9.

3. Attach to RealityX tape device 3

T-ATT 3

You can now use the complete set of TCL tape commands to perform tape operations on the tape image device as through it is a normal tape unit.

MAKE-SPECIAL Command

Purpose	Creates a RealityX file which provides a special view of a UNIX environment.																
Command Class	Cataloged DATA/BASIC program.																
Syntax	MAKE-SPECIAL <i>file-name</i> <i>keyword</i> { <i>UNIX_file</i> { <i>delimiter</i> }}																
Syntax Elements	<table><tr><td><i>file-name</i></td><td>is the name of the special view file to be created.</td></tr><tr><td><i>keyword</i></td><td>specifies the type of special view to be created. The following keywords can be used: <table><tr><td>ENV</td><td>specifies a special view of the current UNIX environment.</td></tr><tr><td>FILE</td><td>specifies a special view of a UNIX text file. This keyword requires the additional <i>parameter</i> <i>UNIX_file</i>, and optionally a <i>delimiter</i>.</td></tr><tr><td>ILOCKS</td><td>specifies a special view of the item lock table.</td></tr><tr><td>NULL</td><td>specifies a null device file.</td></tr></table></td></tr><tr><td><i>UNIX_file_path</i></td><td>is the full path-name of the UNIX file for which you wish to create the special view.</td></tr><tr><td><i>delimiter</i></td><td>specified the character(s) to be converted to a value mark. If <i>delimiter</i> is not specified, no value marks are inserted in the file view item.</td></tr></table>	<i>file-name</i>	is the name of the special view file to be created.	<i>keyword</i>	specifies the type of special view to be created. The following keywords can be used: <table><tr><td>ENV</td><td>specifies a special view of the current UNIX environment.</td></tr><tr><td>FILE</td><td>specifies a special view of a UNIX text file. This keyword requires the additional <i>parameter</i> <i>UNIX_file</i>, and optionally a <i>delimiter</i>.</td></tr><tr><td>ILOCKS</td><td>specifies a special view of the item lock table.</td></tr><tr><td>NULL</td><td>specifies a null device file.</td></tr></table>	ENV	specifies a special view of the current UNIX environment.	FILE	specifies a special view of a UNIX text file. This keyword requires the additional <i>parameter</i> <i>UNIX_file</i> , and optionally a <i>delimiter</i> .	ILOCKS	specifies a special view of the item lock table.	NULL	specifies a null device file.	<i>UNIX_file_path</i>	is the full path-name of the UNIX file for which you wish to create the special view.	<i>delimiter</i>	specified the character(s) to be converted to a value mark. If <i>delimiter</i> is not specified, no value marks are inserted in the file view item.
<i>file-name</i>	is the name of the special view file to be created.																
<i>keyword</i>	specifies the type of special view to be created. The following keywords can be used: <table><tr><td>ENV</td><td>specifies a special view of the current UNIX environment.</td></tr><tr><td>FILE</td><td>specifies a special view of a UNIX text file. This keyword requires the additional <i>parameter</i> <i>UNIX_file</i>, and optionally a <i>delimiter</i>.</td></tr><tr><td>ILOCKS</td><td>specifies a special view of the item lock table.</td></tr><tr><td>NULL</td><td>specifies a null device file.</td></tr></table>	ENV	specifies a special view of the current UNIX environment.	FILE	specifies a special view of a UNIX text file. This keyword requires the additional <i>parameter</i> <i>UNIX_file</i> , and optionally a <i>delimiter</i> .	ILOCKS	specifies a special view of the item lock table.	NULL	specifies a null device file.								
ENV	specifies a special view of the current UNIX environment.																
FILE	specifies a special view of a UNIX text file. This keyword requires the additional <i>parameter</i> <i>UNIX_file</i> , and optionally a <i>delimiter</i> .																
ILOCKS	specifies a special view of the item lock table.																
NULL	specifies a null device file.																
<i>UNIX_file_path</i>	is the full path-name of the UNIX file for which you wish to create the special view.																
<i>delimiter</i>	specified the character(s) to be converted to a value mark. If <i>delimiter</i> is not specified, no value marks are inserted in the file view item.																
Restrictions	SYSMAN account only.																
General Comments	<p>A special view file is virtually indistinguishable from an ordinary RealityX database file. This enables the referenced UNIX environment to be accessed and manipulated by a database user with standard TCL commands, as well as other processors, such as ENGLISH, DATA/BASIC, PROC, etc.</p> <p>More than one special file can be created and can co-exist for the same UNIX file, each constituting a synonym file name for the same special file view.</p>																

File View

Lines of text within the referenced UNIX file appear as RealityX items in the special RealityX file and each new line of text in a UNIX text file appears as an attribute line within the corresponding RealityX item.

Even if a directory file name does not exist, MAKE-SPECIAL will still create a D-pointer in the MD and a special file dictionary. However, any attempt to view such a directory will return the error message:

```
'directory' IS NOT A FILE NAME
```

A special file view contains RealityX items for lines of regular UNIX text files. Non-regular files such as, sub-directories, pipes, devices and so on are not visible as items.

Special file view is primarily for viewing text files. An attempt to create a special view on to a binary file and manipulate binary data will have an undefined effect. Also a special file view is a read only file. Any attempt to write to it will result in an error message.

Environment View

Each environment variable appears as an item in the special RealityX file. The item id is the variable name and the value of the variable is in attribute one. You can change your current UNIX environment by updating items in this file.

Item Lock List View

Each item lock appears as an item in the special RealityX file.

An item lock view file is read only. Any attempt to write to it will result in an error message.

When you create the special view file, a set of attribute definition items are copied into the special file dictionary which are used by the ENGLISH processor to display a report of item locks with the following headings:

Value	File	Item	Port	Lock	Level	Waiting	Waiting
	Hash	Hash		Count			Level

Their meanings are as follows:

Value	Item lock number
File Hash	File number part of hash
Item Hash	Item number part of hash
Port	Port holding lock

Lock Count Number of times port holds lock

Level Context, or perform, level

Waiting Multivalued list of ports waiting for this lock

Waiting Level Context level of waiting port(s)

Example 1

Create a special view of the file **/user1/admin/addresses** which contains data in the format:

```
John Smith:24, Brandles Close:Letchworth:Herts:SG6/6QW
Alan Brown: 67, South Avenue:Littleover:Derby:DE36GH
Alison Macdonald:23 Castle Road:Southampton:Hants:SO2 5FE
```

Enter:

```
MAKE-SPECIAL ADDRESSES FILE /user1/admin/addresses :
```

The following system message is returned:

```
[417] FILE 'ADDRESSES' created.
D code =D, modulo =1, separ = 1
```

```
[417] FILE 'ADDRESSES' created.
D code =DX, modulo =0, separ = 0
```

Now if you enter:

```
CT ADDRESSES *
```

you will see the following items:

```
1
001 John Smith
    24, Brandles Close
    Letchworth
    Herts
    SG6/9QW
```

```
2
001 Alan Brown
002 67, South Avenue
003 Littleover
004 Derby
005 DE3 9GH
```

```
3
001 Alison Macdonald
002 23 Castle Road
003 Southampton
004 Hants
005 SO2 7FE
```

Example 2

Create a special view of your UNIX environment.

```
REALROOT=/realman/4.1A
SHELL=/bin/ksh
PWD=/user1/admin
REALACC=SYSMAN
etc.
```

Enter

```
MAKE-SPECIAL ENVIEW ENV
```

The following system message is returned:

```
[417] FILE 'ENVIEW' created.
D code =D, modulo =1, separ = 1

[417] FILE 'ENVIEW' created.
D code =DX, modulo =0, separ = 0
```

Now if you enter:

```
CT ENVIEW *
```

the following items will be displayed:

```
REALROOT
001 /realman/4.1A
```

```

        SHELL
001 /bin/ksh

        PWD
001 /user1/admin

        REALACC
001 SYSMAN

etc.

```

Example 3

Create a special view of the item lock table. For example, enter:

```
MAKE-SPECIAL ILOCKTAB ILOCKS
```

then, enter:

```
LIST ILOCKTAB
```

This will display a report similar to the following

Value	File Hash	Item Hash	Port	Lock Count	Level	Waiting	Waiting Level
03100058	0310	0058	405	1	0	403 404 402	0

Example 4

To create a null device file called NULL-FILE, enter:

```
MAKE-SPECIAL NULL-FILE NULL
```

SET-PRIORITY - Changing User Process Priority

Purpose

Used to change the priority of the local **reality** user process so as to increase or decrease its response time.

It was initially developed to enable the priority of background processes to be lowered.

Note: Information about this TCL verb was not available when the *RealityX 4.1 User's Reference Manuals* were printed. This information will be incorporated in *RealityX Reference Manual Volume 3* when it is next updated.

CAUTION

Improper use of SET-PRIORITY may degrade system response time.

Syntax

SET-PRIORITY {*n*}

Syntax Element

n specifies the required priority level:

0	Low
1	Default
2	High

If *n* is omitted, the current priority number is displayed, but not changed.

Restrictions

SET-PRIORITY is only available on UNIX System V Release 4 systems.

It can only be used to modify the priority of the local **reality** process. It cannot be used to modify another process.

Only the UNIX superuser (**root**) operating with SYS2 privileges on the database can increase the priority of its process to 2 (high). However, an ordinary user can reduce the priority of their own process and re-instate it back to its original default.

T-STACKER - Managing Multi-Cassette FILE-SAVEs and RESTOREs

Purpose

The T-STACKER command enables multi-reel FILE-SAVE/RESTORE operations to be carried out on a multiple cassette tape stacker/autoloader unit without needing to type 'C' to continue at the end of each tape.

T-STACKER therefore enables automatic sequential access to some or all tapes in the multiple cassette unit until the end of the last tape, as defined by T-STACKER, is reached. Executing T-STACKER with the *slots* parameter greater than zero declares that the tape unit is to be operated in auto-sequential mode.

This command is for use with Sun Desktop multiple tape drives. However, it may be suitable for use with other multi-cassette stacker/autoloader units supported by Northgate in the future.

Note: Information about this TCL verb was not available when the the *RealityX User's Reference Manuals* were printed. This information will be incorporated in *RealityX Reference Manual Volume 2* when it is next updated.

Syntax

T-STACKER *unit slots*

where,

unit specifies the tape unit number.

slots specifies the number of slots in the tape stacker that are occupied. A slot value of 0 (zero) returns the tape unit to normal operation.

CAUTION

If there are less tape cartridges than are specified by the *slots* parameter, T-STACKER will wrap-around and over-write the first tape and any subsequent tapes until the specified number of slots have been written to .

Ensure that all tapes are write-enabled and the file-save will fit on the specified number of tapes, otherwise the complete save will fail and T-STACKER will have to be re-run.

Note: The tape stacker is reset to start at the first tape by re-installing the tape carrier.

Example

```
T-STACKER 3 4
```

Specifies that tape device number 3 is a Stacker or Autoloader drive with four slots loaded with tapes. Four tapes can then be accessed and unloaded.

After each tape is filled by the FILE-SAVE, the message

```
MOUNT REEL# N ON DRIVE 2  
TYPE 'C' TO CONTINUE
```

is displayed. However the software continues on automatically, until the fourth and last tape is accessed. After the last tape is unloaded, further access to the unit is denied until T-STACKER is executed again.

Messaging by PLID

An enhancement to the MESSAGE (MSG) verb is supported by Release 4.1 which allows you to send a message by specifying the Physical Location Identifier (PLID) of the user(s) to receive the message.

The syntax is as follows:

MSG %*PLID*{*PLID*}... *text*

where *PLID* is a full or partial PLID. For example:

UNIX-12345-pts28

or

UNIX-12345

or

UNIX

B

Backup
multideck 5-3

C

Configuration files
users 2-3

D

dbsave utility 5-3
dbsave_config file 5-8
devices file
keywords 3-4
main description 3-3
PLID specification 3-3
port specifications 3-3
reverse lookup 3-4
Directory view 2-5

F

File protection
important system files 2-7

K

Keywords in **devices** file 3-4

M

MAKE-SPECIAL command 7-4
MSG command
PLID variant 7-12
Multideck save/restore
overview 5-2
Multideck save/restore utility – *see dbsave utility*

P

Password
integration 2-2, 2-3
PLID
Port mapping 3-2
port specification 3-3
Port
specification 3-3

R

rdb ! command 2-5
realusers 2-2, 2-3

S

SECURITY file protection 2-7
SET-PRIORITY command 7-9
SQL 6-2
Structured query language – *see SQL*
SYS command 2-5
SYSFILES MD protection 2-7
SYSMAN MD protection 2-7
SYSPL file protection 2-7
SYSTEM file protection 2-7

T

TANDEM command 4-2
Tape image device 7-2
Transaction log protection 2-7
T-STACKER command 7-10

U

USERS file protection 2-7