northgate
INFORMATION SOLUTIONS

RealityX

Release 5.0

Differences Between Releases 4.1 and 5.0

Document No. UM70006653A
September 1997

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

This chapter describes the purpose and content of this manual, related manuals and any conventions used within the manual.

Purpose of this Manual

This manual summarises the differences seen by users upgrading from RealityX Release 4.1 to 5.0.

Contents

It is divided into only two chapters:

Chapter 1, About this Manual, describes the purpose and content of this manual, related manuals and any conventions used within the manual.

Chapter 2, Description of 4.1/5.0 Differences, describes the new features supported by RealityX 5.0.

Related Documents

RealityX 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)

SQL/ODBC for RealityX - Administrator's Guide

SQL/ODBC for RealityX - Developer's Guide

If you are upgrading from a pre-4.1 release of RealityX, the following manuals will be helpful:

RealityX Differences Between Releases 3.1 and 4.0 (*UM70006362A*)

RealityX Differences Between Releases 4.0 and 4.1 (*UM70006456A*)

If you are upgrading from a Series 18/19 (REALITY) system, the following manuals will also be helpful:

User's Guide to the REALITY Migration Utilities (*UM70006019A*)
(Part of Volume 1 of the REALITY 7.x Reference Set)

RealityX Differences Supplement (*UM70006051B*) which describes the 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**
Text	Characters or words in italics indicate parameters which must be supplied by the user. For example in **LIST** *file-name* the parameter *file-name* is italicized 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.
{ }	Braces enclose options and optional parameters in TCL commands. For example in **RESIZE-FILE** *file-name* {*index*}{*modulo*}{**(***options*} • *file-name* must be supplied, but the *index*, *modulo* and *options* parameters are all optional. • One or more single-letter options can be included, as defined for the command; these must be preceded by an open parenthesis, can be given in any order, and are not separated by spaces. Any number of options can be used except where specified in text.

[ ]	Square brackets enclose optional parameters in UNIX shell commands. For example: **sizemon** [*options*] [*file* [*file*...]] where *options* and one or more *file* names are both optional.
...	In syntax descriptions, indicates that the parameters preceding can be repeated as many times as necessary.
SMALL CAPITALS	Small capitals are used for the names of keys such as RETURN.
Enter	To enter means to type text then press RETURN. For instance, 'Enter the WHO command' means type **WHO**, then press RETURN. In general, the RETURN key (shown as ENTER or ↵ on some keyboards) must be used to complete all terminal input unless otherwise specified.
Press	Press single key or key combination, but do not press RETURN afterwards.

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, or email techpubs@northgate-is.com.

Chapter 2

Description of 4.1/5.0 Differences

This chapter describes the new features supported by RealityX 5.0, including:

- SQL/ODBC enhancements.
- File size monitoring - **sizemon** utility and SIZE-MONITOR verb.
- Dynamic resizing of files - **realresize** utility and RESIZE-FILE verb.
- Creation of index views using the MAKE-SPECIAL verb.
- Two new options for the ISTAT verb - U and J.
- T-DEVICE - A new TCL verb for redefining a tape unit.
- Password propagation in a FailSafe configuration.
- Item deadlock detection.
- Enhanced file error handling.
- Non-contiguous RealityX files in a Partition Database.
- Change to Rule 5. for Dynamic Array References.

‘SQL/ODBC for RealityX’ Enhancements

This section summarises the enhancements that are included in the SQL/ODBC interface for RealityX Release 5.0.

Note: ‘SQL/ODBC for RealityX’ is a purchasable option of RealityX.

For a detailed description of SQL/ODBC for RealityX, refer to the two manuals, *SQL/ODBC for RealityX - Administrator’s Guide* and *SQL/ODBC for RealityX - Developer’s Guide.*

Release 5.0 SQL/ODBC features include the following:

- SQL update capability
- Improved SQL catalog management
- SQL security enhancements
- SQL table creation
- ODBC 2 compliance
- Windows utility for RealityX SQL data source maintenance

SQL/ODBC Update Capability

The SQL/ODBC interface for RealityX 5.0 provides the capability to INSERT, UPDATE and DELETE rows in a RealityX data source from any ODBC 2 compliant PC application.

Many PC applications will not allow update capability unless indexes are supported. RealityX 5.0 therefore allows the creation, dropping and reporting of SQL indexes via the SQL/ODBC interface.

The SQLM utility is enhanced to enable the creation, deletion and display of indexes in the RealityX environment.

Transaction Handling is supported for SQL/ODBC updates on a RealityX database.

Improved SQL Catalog Management

Release 5.0 contains the following improvements to the management of the SQL Catalog in a RealityX database:

- Handling of Q-pointers. Warning messages are given when they are incorrectly used.
- Synonym Table pointers.
- SQLM is enhanced to cover the improved catalog handling and can be driven from a select list. Also SQLM can be executed in batch mode, driven by a DATA/BASIC program.

SQL Security Enhancements

Table privilege grants can be specified for a security profile-id, as well as a user-id. When looking up privileges the SQL server first uses the user-id, as for Release 4.1. If no match is found, it then looks for a security profile-id. If still no match is found, it uses the PUBLIC user-id, as for Release 4.1.

SQLM is enhanced to allow security profile-ids to be entered and displayed.

SQL Table Creation

CREATE TABLE, DROP TABLE and ALTER TABLE statements can be executed by the SQL server on the host.

ODBC 2 Compliance

On Release 5.0 the SQL driver and server conform to a subset of ODBC 2.0 with Applications Programming Interface (API) Level 1.

Unsupported ODBC 2.0 Features

The following ODBC 2.0 features are **not** supported:

- Views
- Constraints
- Referential Integrity Checking
- Triggers
- Tables without primary keys and tables with primary keys which do not map easily to a RealityX item id.
- Stored procedures

Supported Data Types

As for Release 4.1, only the following data types are supported:

CHAR	INTEGER
VARCHAR	SMALLINT
LONGVARCHAR	DATE
DECIMAL	

SQL Maintenance

RealityX 5.0 allows you to set up and administer a RealityX SQL data source from TCL **and** from a PC Windows environment.

File Size Monitoring

The File Size Monitor Utility, supported by Release 5.0, scans the whole or part of a database and reports any badly sized files.

This utility can be executed at the UNIX shell or TCL prompt by the following commands:

- **sizemon**, at the UNIX shell.8
- SIZE-MONITOR, at TCL.

Sizemon Utility

sizemon [*options*] [*file* [*file*...]]

sizemon [*options*] [**-l**] *listfile*

Parameters

file	Specifies file to be scanned.
listfile	Specifies a UNIX text file containing a list of accounts and files to be scanned. See the topic *Rules for Setting Up a List of Files* below.

Options

-d *database*	Specifies a database, other than the default, to be scanned. Default is specified in REALDBASE.
-a *account*	Specifies logon account to be scanned. Default is specified in REALACC.
-s *size*	Reports any items greater than *size* blocks.
-o *limit*	Sets a lower limit on the average no. of overflow blocks before reporting.
-c	Calculates a new optimum modulo for any badly sized files.
-l	Specifies a UNIX file containing a list of accounts and files to be included, or excluded, from being monitored.
-v	Verbose mode.

Examples

`sizemon -d SYSTEM`
Scan the whole database (No inclusions or exclusions).

`sizemon -a account`
Scan all files in a specified account.

`sizemon` *`file1 file2`*
Scan specified files. Files must be specified in /*account*/*file* format.

`sizemon -a` *`account file1 file2`*
Scan specified files in a specified account. File names must **not** be in /*account*/*file* format.

`sizemon -l` *`listfile`*
Scan account(s) and file(s) listed in a UNIX text file *listfile*. The syntax of the list file is described below.

SIZE-MONITOR Verb

SIZE-MONITOR {**(***options*}
is valid when a select list is active. The list is expected to be of files in the current account.

SIZE-MONITOR *account* | *file* [*account* | *file* ...] {**(***options*}
Checks a simple list of files or accounts. These may use the */account* or */account/file* format. If *account* is SYSTEM then the whole database will be scanned.

SIZE-MONITOR *file item* **(L** {*options*}
Specifies an item containing a file list. See the topic *Rules for Setting Up a List of Files* below.

Options

C	Calculates a new optimum modulo for any badly sized files.
L	Specifies an item containing a file list. This option performs the same function as **sizemon -l**.
O	Sets a lower limit on the average number of overflow blocks before reporting. When using this option you are prompted for a numeric lower limit.
P	Sends report directly to a printer.

S	Reports any items greater than a specified number of blocks. When using this option you are prompted to specify the number of blocks.
V	Verbose mode.

Restrictions on SIZE-MONITOR

SYSMAN account only.

Rules for Setting Up a List of Files

The list file is a UNIX text file containing a list of accounts to be included or excluded from the scan. The following rules apply:

- If the file contains only exclusions, then all other accounts are included.
- If there are any explicitly included accounts, all other accounts are implicitly excluded.
- If there are any explicitly included files within an included account, then all other files in that account are implicitly excluded.
- If there is a mixture of included and excluded files, all other files in that account are implicitly excluded.

The rules for formatting the List File are as follows:

- Whitespace lines are ignored.
- A line starting with a hash character (#) is ignored.
- An account followed by a minus sign will be excluded.
- A list of files may follow the account. If the account is excluded, the file list is ignored.
- Each file appears on a separate line following a plus or a minus sign, indicating inclusion or exclusion, respectively.

Example List File

```
#Example list

ACCOUNTA - # Exclude this account.
ACCOUNTB   # Include all of this account.
ACCOUNTC   # Include only the files listed.
+ FILEC1
+ FILEC2
ACCOUNTD   # Include all files in account except those
- FILED1   # listed.
- FILED2
```

Dynamic Resizing of RealityX Files

The File Resize Utility, supported by Release 5.0, enables resizing of a RealityX file while maintaining access to it.

This utility can be executed at the UNIX shell or TCL prompt by the following commands:

- **realresize**, at the UNIX shell.
- RESIZE-FILE, at TCL.

Restrictions

- **The database must be a Partition Database.**
- Only data or index sections can be resized.
- Data sections cannot be accessed sequentially while a file is being resized. Any attempt to do so (e.g. by an ENGLISH query that is not using an index) will hang until the resize is complete. However, if the file has an index by item-id, the index will be used automatically while the resize is in progress, enabling the query to complete.
- Resizing should not be initiated while a long ENGLISH query is in progress, as this will cause the result of the query to be indeterminate.
- The modulo cannot be reduced.
- RESIZE-FILE can only be executed from the SYSMAN account.
- RESIZE-FILE can only be executed once per file per session of the database daemon.

Realresize Utility

realresize [*option*s] *file-name* [*modulo*]

Parameters

file-name	File to be resized. The account containing the file must be specified using the **-a** *account* option, unless it is the default account **$REALACC**, or the /*account*/*file* format is used.
modulo	New modulo for resized file/index.

Options

-d *database*	Specifies the database, other than the default, containing the file to be resized. The default database is defined in **$REALDBASE**.
-a *account*	Specifies the account, other than the default, containing the file to be resized. The default account is defined in **$REALACC**.
-i *index*	Specifies the index to be resized.
-b	Background mode (i.e. disconnect from terminal).
-r	Restart a resize which has been discontinued, for example, by a database shutting down. No modulo should be supplied.
-v	Verbose mode.

RESIZE-FILE Verb

RESIZE-FILE *file-name* {*index*}{*modulo*}{**(***options*}

Parameters

file-name	Name of file to be resized. For **realresize**, the account containing the file must also be specified using the **-a** *account* option, or by using the */account/file* format for *file-name*, or via $REALACC.
index	Index section to be resized.
modulo	New modulo for resized file/index.

Options

R	Restart a resize which has been discontinued, for example, by a database shutting down. No modulo should be supplied.
B	Background mode (i.e. disconnect from terminal).
V	Verbose mode.

Comments

RESIZE-FILE is not logged.

Creation of Index Views using the MAKE-SPECIAL Verb

MAKE-SPECIAL in Release 5.0 can be used to create a special view of an index by creating a read-only special RealityX file, the contents of which reflect the referenced index.

Syntax

MAKE-SPECIAL *file-name keyword* {*parameters*}

Syntax Elements

file-name is the name of the special view file to be created.

keyword specifies the type of special file to be created. Three types of keywords can be used, INDEX-RAW, INDEX-ITEM and INDEX-KEY. See below.

parameters Additional parameters required with INDEX-RAW, INDEX-ITEM and INDEX-KEY keywords are, as follows:

data-section index

where,

data-section specifies data section referenced by index. INDEX-RAW, INDEX-ITEM and INDEX-KEY views only.

index Specifies the index for which the special index view is to be created. INDEX-ITEM and INDEX-KEY views only.

Keywords

INDEX-RAW Specifies a view of the raw index nodes, giving simple read-only access to the items in the referenced index's native file.

Note: The INDEX-RAW view is for use by Northgate Support personnel only.

INDEX-ITEM Specifies a view of the index key values for the set of items in the *data-section* selected by the referenced *index*. Items in an INDEX-ITEM view file have the same item-ids as their associated data items, selected by the index, with associated key values in attribute one.

INDEX-KEY Specifies a view of the item-ids associated with each key value in a referenced index. Items in the INDEX-KEY view file have numeric item-ids which reference the 1st to the **n**th unique keys in the index.

Each item comprises two attributes. The first containing the key value for the associated unique key item id and the second containing a multi-valued list of the data item-ids associated with the key value in attribute one.

Example

Create an index GUESTS-LAST-NAME for the GUESTS data section which has index entries ordered according to the key value LAST-NAME. For example:

```
:DEFINE-INDEX GUESTS BY LAST-NAME

TO:GUESTS-LAST-NAME

[1281] Index definition 'GUESTS-LAST-NAME' created.

:CREATE-INDEX GUESTS GUESTS-LAST-NAME

[1280] Index 'GUESTS-LAST-NAME' created.
```

Then, using MAKE-SPECIAL, create two views of the index GUESTS-LAST-NAME. For example, GSN-ITEM-VIEW and GSN-KEY-VIEW, as follows:

```
:MAKE-SPECIAL GSN-ITEM-VIEW INDEX-ITEM GUESTS GUESTS-LAST-NAME

[417] File 'GSN-ITEM-VIEW' created.

D code =D, modulo = 1, separ = 1

[417] File 'GSN-ITEM-VIEW' created.

D code =DY, modulo = 0, separ = 0

:MAKE-SPECIAL GSN-KEY-VIEW INDEX-KEY GUESTS GUESTS-LAST-NAME

[417] File 'GSN-KEY-VIEW' created.

D code =D, modulo = 1, separ = 1

[417] File 'GSN-KEY-VIEW' created.
```

```
D code =DY, modulo = 0, separ = 0
```

Now look at the items in the two special view files using the CT verb. For example:

Index Item View

```
:CT GSN-ITEM-VIEW *

    100                             412
001 Anderson                    001 Lewis

    122                             140
001 Anderson                    001 Lynch

    444                             142
001 Curtis                      001 Madison

    535                             234
001 Evans                       001 McSweeney

    289                             309
001 Fennelly                    001 Mendell

    365                             222
001 Ferguson                    001 O'Brien

    411                             401
001 Gallagher                   001 Palmer

    143                             318
001 Hennessey                   001 Petrillo

    194                             428
001 Hynes                       001 Postma

    144                             117
001 Irving                      001 Rizzo

    147                             119
001 Janson                      001 Scott

    478                             354
001 Kolman                      001 Taylor

    211                             355
001 Lewis                       001 Taylor
```

Index Key View

```
:CT GSN-KEY-VIEW *

    1                               12
001 Anderson                    001 Lewis
002 100]122                     002 211]412

    2                               13
001 Curtis                      001 Lynch
002 444                         002 140

    3                               14
001 Evans                       001 Madison
002 535                         002 142

    4                               15
001 Fennelly                    001 McSweeney
002 289                         002 234

    5                               16
001 Ferguson                    001 Mendell
002 365                         002 309

    6                               17
001 Gallagher                   001 O'Brien
002 411                         002 222

    7                               18
001 Hennessey                   001 Palmer
002 143                         002 401

    8                               19
001 Hynes                       001 Petrillo
002 194                         002 318

    9                               20
001 Irving                      001 Postma
002 144                         002 428

    10                              21
001 Janson                      001 Rizzo
002 147                         002 117

    11                              22
001 Kolman                      001 Scott
002 478                         002 119
```

ISTAT Verb - U and J Options

Two new options are available with the ISTAT verb in release 5.0:

Note: A full description of the ISTAT command is given in *RealityX Reference Manual Volume 1*.

U	Utility option. Displays the hashing statistics of a file in an alternative way, as shown in Example 2.
J	Displays the hashing statistics of a index in a similar form to the U option.

Both the U and J options allow you to interactively change the report displayed by entering commands at the `Command:` prompt. Commands are available for generating different hashing statistics for different moduli and for changing the number of rows in the report. See the topic *Commands* below.

Alternative Report using U Option

The alternative report, displayed by ISTAT with the U option, is shown in Example 2. The report displayed for an index using the J option is similar. It comprises four columns each with *n* rows. where *n* can be specified at the Command: prompt using the command **n***n*. The default number of rows , is 0 to 11 (See Example 2). The columns are:

`N`	which lists the value of N for each row.
`Number of groups with N items.`	which lists the number of groups in the file comprising exactly ‘N’ items.
`Number of groups with N used IG frames.`	which lists the number of groups in the file which use exactly ’N’ in-group frames.
`Number of groups with N %  usage of IG frames`	which lists the number of groups in the file that use ‘N’ per cent of in-group frame space. The bottom line represents the number of groups with from N up to 100% of in-group frame spaces.

Commands

When using the J and U options, the following commands can be entered at the `Command:` prompt:

i	Search for the optimum modulo to display hashing statistics with no overflow frames by trying moduli in increments of ‘1’, until the optimum is found, then display.
m*n*	Use modulo *n* to generate hashing statistics. 0 restores original.
n*n*	Enter new maximum value *n* of row number (N) to generate hashing statistics.
p	Search for optimum modulo to display hashing statistics with no overflow frames by trying prime numbers only, until optimum is found, then display.
q	Quit.

Comments

ISTAT can be used to determine the group that a particular item resides in and the distribution of data within a file. This information is useful when you are trying resize files. 'IND.' in the report stands for 'indirect', meaning out-of-group.

Example 1

```
ISTAT ACCPAY (H

FILE: ACCPAY MOD: 19 SEP:                          11:50:16  25 May 1995
  BYTES ITMS
    1281  17 *>>>>>>>>>>>>>>>>>
    1865  20 *>>>>>>>>>>>>>>>>>>>>
     505  12 *>>>>>>>>>>>>
    1353  22 *>>>>>>>>>>>>>>>>>>>>>>
     417  13 *>>>>>>>>>>>>>
    1361  15 *>>>>>>>>>>>>>>>
    1353  13 *>>>>>>>>>>>>>

Item count=         112, byte count=    8128, avg. bytes/item=     72.5
Avg.  items/group=  6.5, std. deviation= 3.8, avg. bytes/group=  1161.1
Frames in use=       12, sec. frames=      1, empty groups=           0
ind. items=          12, ind. frames=     14
```

Example 2

```
:ISTAT ERRMSG (U

File='ERRMSG' Modulus=119 Frame-size=1008

    |       Number of groups with :-
 N  | N items  N used IG frames  N % usage of IG frames

 0  -     0            0          0        (No empty groups)
 1  -     0          113          0
 2  -     0            6          0        (6 groups have one overflow
block)
 3  -     0            0          0
 4  -     0            0          0
 5  -     4            0          0
 6  -     4            0          0
 7  -     7            0          0
 8  -    16            0          0        (16 groups have 8 items)
 9  -    18            0          0
10  -    12            0          0
11+ -    58            0        119        (58 groups have 11 items, or
                                           more.  119 groups have 11%
                                           usage, or more)

Total items=  1265, Total bytes=   86597, Total frames=   141
IG bytes=    76924, Overflow frames   6, Wasted bytes=  49076(38%)
OG items=          10, OG frames=      16, Groups used=    119

Command:
```

T-DEVICE - TCL Verb for Redefining a Tape Unit

Purpose

Redefines the tape device definition, temporarily, for a specified tape unit.

Syntax

T-DEVICE {*tape_unit* {*device_path* {*device_type*}}}

Syntax Elements

tape_unit Specifies the tape unit for which the tape device is to be redefined.

If *tape_unit* is not specified, all tape units configured on the database are reset to their default tape definitions, as specified in the database configuration file.

device-path Specifies the new device to be defined for the specified tape unit.

If *device-path* is not specified, the specified tape unit is reset to its default device setting, specified in the database configuration file.

device_type Specifies the new device type for the specified tape unit. The following numbers can be assigned:

1 = 1/2 inch tape
2 = 8mm Exabyte tape
3 = 1/4 inch (QIC) cartridge
5 = 4mm (DAT) cartridge
9 = tape image

If *device_type* is unspecified, but *tape_unit* and *device_path* are specified, *device_type* defaults to 9 (tape image).

Comments

Ensure that the specified tape unit is not currently assigned, otherwise T-DEVICE will fail.

After T-DEVICE has been executed, re-assign the specified tape unit to your terminal, using the T-ATT or ASSIGN verb, to enable the tape unit to be accessed.

Note that T-DEVICE does not validate the specified device path. Validation is performed by the T-ATT or ASSIGN verb when you re-assign the tape unit.

After running T-DEVICE, the T-STATUS verb reports the status of the new tape device, showing whether or not it is assigned.

T-DEVICE without any specified parameters resets all tape definitions to the default settings defined in the database configuration file.

T-DEVICE only affects the current port. The tape device definition set by T-DEVICE is lost when the current port is logged off.

Examples

Example 1

```
:T-DEVICE 2 /dev/rmt/ctapen 3
```

Redefines tape unit 2 as a quarter inch cartridge device **/dev/rmt/ctapen**.

Example 2

```
:T-DEVICE 3 /user1/daveh/tape3
```

Redefines tape unit 3 as a tape image device **/user1/daveh/tape3**.

Example 3

```
:T-DEVICE 1
```

Resets tape unit 1 to its default tape definition specified in the database configuration file.

Example 4

```
:T-DEVICE
```

Resets all tape units to their default tape definitions specified in the database configuration file.

Miscellaneous Enhancements

Password Propagation in a FailSafe Configuration

UNIX/RealityX password integration is enhanced on Release 5.0 so as to enable updated passwords on a FailSafe primary system to be logged and propagated to the secondary. This feature can be suppressed by creating the entry "`SuppressPasswordLogging=1`" in the database **config** file.

Item Deadlock Detection

An Item Deadlock, also known as a Deadly Embrace, occurs when two processes are competing for items locked by the other. For example, suppose Process A asserts a lock on Item B and then attempts to assert a lock on Item A. Simultaneously, Process B asserts a lock on Item A and then attempts to assert a lock on Item B. Process A and Process B each are asserting an item lock on an item the other process also requires. The result is a stalemate in which neither process is able to gain access to all the items it requires.

RealityX provides the facility to detect an item deadlock. After detecting a deadlock it waits for a configurable length of time, then sends a warning message to both the terminal screen and the daemon log informing the user that a deadlock is in progress and giving the names of the account and file involved. The daemon log output will be of the form:

```
Feb 12 15:56:35 #27043 port402 WARNING: Item deadlock:
ACC1:, AFILE (DATA) 'ITEMA' held by 401
Feb 12 15:56:35 #27042 port401 WARNING: Item deadlock:
ACC1:, AFILE (DATA) 'ITEMB' held by 402
```

File Error Handling

File error reporting is improved so that the error description which appears on the user's screen and in the daemon log contains the RealityX file name. Both system I/O failure and GFE messages have been enhanced. The GFE error message is of the form:

`File error` *error number*`: Group Format Error`
`{ Block | Group } block of` *file account* [*level* [*name*]]
(raw file description)

For example:

```
File error 2033: Group Format Error
First item: expected offset 10 got 42
Group 7 of file ACCOUNTA FILEA
(ACCOUNTA/3FILEA)
```

Non-Contiguous RealityX Files in a Partition Database

A RealityX file with modulo M, on a Partition Database, requires a contiguous segment of M free blocks. RealityX 5.0 supports a mechanism which allows large files with non-contiguous in-group space.

Non-contiguous space is handled using a scatter map containing a list of addresses which locate the various segments of the file. Currently, one block of 120 addresses is supported, hence a file is limited to 120 non-contiguous segments. Attribute 2 of a non-contiguous file definition item (D-pointer) contains values which identify the location of the scatter map.

A contiguous file definition item (D-pointer) contains the following values in attribute 2:

base, modulo, re-useid

where,

base	is the base block address of the file.
modulo	is the modulo of the file
re-useid	is the re-use identifier for the file.

A non-contiguous file definition item (D-pointer) contains the following values in attribute 2:

base, modulo, re-useid, scatter

where,

base	is the base block address of the scatter map.
modulo	is the modulo of the file.
re-useid	is the re-use identifier for the file.
scatter	is the number of blocks in the file's scatter map. Currently this can only be '1'.

Change to Rule 5. for Dynamic Array References

Rule 5 for RealityX release 4.1 on Page 2-53 of the *DATA/BASIC Reference Manual*:

5. Negative values return a null value.

changes to:

5. Negative value# & subval# values return a null value.
   Negative attr# values return the requested attribute counting backwards from the end of the array. For instance, -1 is the last attribute, -2 is the second to last attribute and so on. Rule 6 still holds.

RealityX Spooler Access from UNIX

This is a reminder of a feature which is available with earlier releases of RealityX.

The UNIX-Connect remote printing facility enables you to direct printer output from UNIX to a RealityX formqueue using the UNIX **lp** command. Refer to the *UNIX-Connect User Guide* for details on how to do this, and to the *UNIX-Connect Administrator's Guide* for the procedure to set up this facility.

Index