NPS Reality v15.2
Online Documentation (Revision 3)

This Documentation Note summarises the changed topics in the Revision 3 release of the NPS Reality v15.2 Online Documentation, made in response to enhancements, bug fixes and Documentation Comments from our customers. Minor typographical corrections and indexing changes are omitted.

For more detailed information, refer to the Online Documentation itself.

Security System Maintenance (SSM) Option 2 - Define User Profiles

Added a note to **Option 17 - Remote System** to explain that if a network password is prefixed with a plus + character it is automatically encrypted (minus the + character) when stored.

realusers (Host administration)

Updated to explain that before running **realusers** a user ID `realusers` should be created on the system. This user must be given a home directory, typically **/home/realusers** and this path must be set in the **users** file.

Printing to Reality Printers from the Host

Updated to say that once the operation to add a printer has completed successfully a printer interface script of the same name as the UNIX printer will have been created in **/etc/lp/interfaces**. The **lp** command can be used with the **-o** switch to pass an optional parameter in argument 5 ($5), this can be passed on to the Reality spooler as SP-ASSIGN options by appending **-o$5** to the REALPRINT command line.

DESPOOLER.CONTROL File Maintenance

Updated to note that when using **Option 2 - Device Name** of DESPOOLER.CONTROL File Maintenance to identify the name of a private printer on a UNIX host, any optional parameters that you specify will be passed to the interface script as arguments $5 and up.

SQL reserved words

Added a reminder that SQL reserved keywords, as well as not being available for use as table or column names, cannot be used as user IDs when granting permissions to Reality SQL tables.

FIND statement (DataBasic)

Updated the description of **FIND** syntax elements to explain when *valVar* and *subvalVar* may be automatically set to 0.

FIND *value* **IN** *dynArray*{,*occurrence*} **SETTING** *attrVar*{,*valVar*{,*subvalVar*}} [**THEN** *statement(s)* | **ELSE** *statement(s)*]

value	An expression that evaluates to the string or value being searched for.
dynArray	The name of the dynamic array in which to search.

occurrence	The number of the occurrence of the element being searched for in *dynArray*. If *occurrence* is not specified, it defaults to 1.
attrVar	A variable that is set to the attribute position where the element is found.
valVar	A variable that is set to the value position where the element is found; if the element is found in an attribute that has no values, *valVar* and *subvalVar* will both be set to 0.
subvalVar	A variable that is set to the subvalue position where the element is found; if the element is found in an attribute or value that has no subvalues, *subvalVar* will be set to 0.
statement(s)	Either a THEN or ELSE clause (or both). A statement must be included. The THEN clause is executed if the FIND is successful; otherwise the ELSE clause is executed.

Also added two more examples.

STOP statement (DataBasic)

Corrected the table in Example 2.

P and D commands (DataBasic debugger)

Updated the description of the **P** debugger command to explain that the initial `On/Off` status is that set by the **P** TCL command; this status is restored on exiting the DataBasic program.

An updated Version 2 of the **D** command now reports the following additional information:

- Whether the trace is suppressed (**T** command).
- Whether child and parent contexts are active (**DCC** and **DPC** commands).
- Whether output is going to the printer (**LP** command).
- Whether terminal output is suppressed (**P** command).

Also, the information has been reordered and made easier to comprehend.

Bitwise logical functions (DataBasic)

Added descriptions of the following DataBasic bitwise logical functions:

BITNOT(*expression***)**

BITAND(*expression1*,*expression2***)**

BITNAND(*expression1*,*expression2***)**

BITOR(*expression1*,*expression2***)**

BITNOR(*expression1*,*expression2***)**

BITXOR(*expression1*,*expression2***)**

BITXNOR(*expression1*,*expression2***)**

Here each *expression* is a valid DataBasic expression or any string, substring or value; may be expressed as a variable name, a value, or a string enclosed in quote marks that evaluates to a decimal number (of which only the integer part is used).

The operation of these functions is summarised below.

Expression	Bit string
Expression A	0011
Expression B	0101
Function	**Result**
BITNOT(A)	1100
BITNOT(B)	1010
BITAND(A,B)	0001
BITNAND(A,B)	1110
BITOR(A,B)	0111
BITNOR(A,B)	1000
BITXOR(A,B)	0110
BITXNOR(A,B)	1001

MAKE-SPECIAL command

Updated the **MAKE-SPECIAL** command with two new keywords:

GFB	A special view of a table of allocated global file blocks (GFBs) each of which identifies the associated file and the number of ports that have that file open (may currently be zero). No additional parameters required.
LFB	A special view of a table of allocated local file blocks (LFBs). No additional parameters required.

tlmenu command

Added two new alternate forms of the **tlmenu** command which together allow you to record commands against one database and play them back against another:

tlmenu record *script-file* {*database1*}

tlmenu runs as normal against *database1*, but in addition all user input is recorded in the *script-file*.

tlmenu {*database2*} *script-file*

The sequence of commands recorded in the *script-file* is repeated for *database2*.

Any **tlmenu** command can be recorded except the dbsave backup method. Script files are also used by the **tlmulti** command.

tlmulti command

Described the new **tlmulti** command which allows **tlmenu** to be successively run on multiple databases.

tlmulti requires a list of database names to use. By default, it looks for a **$REALROOT/files/tlmulti_dbases** file, containing one database per line, where REALROOT is the base root for the system.; alternatively, you can specify another list file that contains the same information. The databases can be in different instances and **tlmulti** will handle the switch from one instance to another automatically.

Dynamic arrays

Corrected the description of rule 9 for dynamic array references to say that if the dynamic array is null, the first negative value is replaced by plus one, and Rule 10 is applied.