BMC Next Generation Technology
General User Guide

Supporting

Version 12.1 of BMC Next Generation Technology Check for DB2 for z/OS
Version 12.1 of BMC Next Generation Technology Load for DB2 for z/OS
Version 12.1 of BMC Next Generation Technology LOBMaster for DB2 for z/OS
Version 12.1 of BMC Next Generation Technology Reorg for DB2 for z/OS
Version 12.1 of BMC Next Generation Technology Stats for DB2 for z/OS
Version 12.1 of BMC Next Generation Technology Unload for DB2 for z/OS
Version 12.1 of BMC Next Generation Technology Utility Manager for DB2 for z/OS

December 2016
Contacting BMC Software

Several methods are available for contacting BMC Software.

You can access the BMC Software website at http://www.bmc.com. From this website, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

United States and Canada

<table>
<thead>
<tr>
<th>Address</th>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC SOFTWARE INC</td>
<td>1 713 918 8800</td>
<td>1 713 918 8000</td>
</tr>
<tr>
<td>2103 CITYWEST BLVD HOUSTON TX 77042-2827 USA</td>
<td>or 1 800 841 2031</td>
<td></td>
</tr>
</tbody>
</table>

Outside United States and Canada

<table>
<thead>
<tr>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>+01 713 918 8800</td>
<td>+01 713 918 8000</td>
</tr>
</tbody>
</table>

© Copyright 2015-2016 BMC Software, Inc.

BMC, BMC Software, the BMC logo, the BMC Software logo, and other BMC marks are the exclusive properties of BMC Software, Inc. and are registered or may be registered in the U.S. and in other countries. BladeLogic and other BladeLogic marks are the exclusive properties of BladeLogic, Inc. and are registered or may be registered in the U.S. and in other countries. All other trademarks or registered trademarks are the property of their respective owners.

DB2, FlashCopy, IBM, IBM z, MVS, RACF, and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

The information included in this documentation is the proprietary and confidential information of BMC Software, Inc., its affiliates, or licensors. Your use of this information is subject to the terms and conditions of the applicable End User License agreement for the product and to the proprietary and restricted rights notices included in the product documentation.

Restricted rights legend

U.S. Government Restricted Rights to Computer Software. UNPUBLISHED—RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure of any data and computer software by the U.S. Government is subject to restrictions, as applicable, set forth in FAR Section 52.227-14, DFARS 252.227-7013, DFARS 252.227-7014, DFARS 252.227-7015, and DFARS 252.227-7025, as amended from time to time. Contractor/Manufacturer is BMC SOFTWARE INC, 2103 CITYWEST BLVD, HOUSTON TX 77042-2827, USA. Any contract notices should be sent to this address.
Customer support

Support website
You can obtain technical support from BMC 24 hours a day, 7 days a week at http://www.bmc.com/support. From this website, you can:

■ Read overviews about support services and programs that BMC offers
■ Find the most current information about BMC products
■ Search a database for problems similar to yours and possible solutions
■ Order or download product documentation
■ Download products and maintenance
■ Report a problem or ask a question
■ Subscribe to receive proactive e-mail alerts
■ Find worldwide BMC support center locations and contact information, including e-mail addresses, fax numbers, and telephone numbers

Support by telephone or e-mail
In the United States and Canada, if you need technical support and do not have access to the web, call 1 800 537 1813 or send an e-mail message to customer_support@bmc.com. (In the subject line, enter SupID:yourSupportContractID, such as SupID:12345). Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC
Have the following information available so that Customer Support can begin working on your issue immediately:

■ Product information
  — Product name
  — Product version (release number)
  — License number and password (trial or permanent)
■ Operating system and environment information
  — Machine type
  — Operating system type, version, and service pack or other maintenance level such as PUT or PTF
  — System hardware configuration
  — Serial numbers
  — Related software (database, application, and communication) including type, version, and service pack or maintenance level
■ Sequence of events leading to the problem
■ Commands and options that you used
■ Messages received (and the time and date that you received them)
  — Product error messages
  — Messages from the operating system
  — Messages from related software
License key and password information

If you have questions about your license key or password, contact Customer Support through one of the following methods:

- Send an e-mail message to customer_support@bmc.com. (In the Subject line, enter SupID:yourSupportContractID, such as SupID:12345.)
- In the United States and Canada, call 1 800 537 1813. Outside the United States and Canada, contact your local support center for assistance.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About this book</td>
<td>11</td>
</tr>
<tr>
<td>Related publications</td>
<td>11</td>
</tr>
<tr>
<td>Conventions</td>
<td>12</td>
</tr>
<tr>
<td>Syntax diagrams</td>
<td>12</td>
</tr>
<tr>
<td>Chapter 1 NGT utility products concepts</td>
<td>15</td>
</tr>
<tr>
<td>Wildcards</td>
<td>15</td>
</tr>
<tr>
<td>Utility Parameters</td>
<td>16</td>
</tr>
<tr>
<td>Journal</td>
<td>16</td>
</tr>
<tr>
<td>Servers</td>
<td>16</td>
</tr>
<tr>
<td>Utility concurrent access control</td>
<td>17</td>
</tr>
<tr>
<td>The matrix</td>
<td>17</td>
</tr>
<tr>
<td>BMCUTIL table</td>
<td>18</td>
</tr>
<tr>
<td>BMCSYNC table</td>
<td>20</td>
</tr>
<tr>
<td>Shared access levels of BMC utilities</td>
<td>24</td>
</tr>
<tr>
<td>Controlling contention</td>
<td>26</td>
</tr>
<tr>
<td>Tape commands</td>
<td>26</td>
</tr>
<tr>
<td>The NGTTAPE command</td>
<td>27</td>
</tr>
<tr>
<td>NGTTAPE global variables</td>
<td>31</td>
</tr>
<tr>
<td>Allocation strategies</td>
<td>32</td>
</tr>
<tr>
<td>VSAM data set allocation</td>
<td>33</td>
</tr>
<tr>
<td>Automation control points for allocation</td>
<td>38</td>
</tr>
<tr>
<td>Managing the model AMS define</td>
<td>39</td>
</tr>
<tr>
<td>Chapter 2 NGT SQL language</td>
<td>41</td>
</tr>
<tr>
<td>NGT SQL language elements</td>
<td>41</td>
</tr>
<tr>
<td>Features and functions of the NGT SQL language</td>
<td>45</td>
</tr>
<tr>
<td>Constants that NGT SQL supports</td>
<td>48</td>
</tr>
<tr>
<td>Special registers that the NGT SQL language supports</td>
<td>52</td>
</tr>
<tr>
<td>Functions that NGT SQL supports</td>
<td>53</td>
</tr>
<tr>
<td>BIGINT function</td>
<td>53</td>
</tr>
<tr>
<td>CHAR function</td>
<td>54</td>
</tr>
<tr>
<td>DATE function</td>
<td>58</td>
</tr>
<tr>
<td>DAY function</td>
<td>59</td>
</tr>
<tr>
<td>DAYS function</td>
<td>60</td>
</tr>
<tr>
<td>DAYOFMONTH function</td>
<td>61</td>
</tr>
</tbody>
</table>
Chapter 6  NGT parameters  145

Setting default parameters .................................................. 145
Syntax rules for parameters .................................................. 145
Overview of global utility parameters (UTLPARMS) ................. 146
Parameters grouped by function .......................................... 150
Global utility alphabetical parameter reference ........................ 155
+AGELIMIT ........................................................................... 155
+ALLERROR ........................................................................ 155
+CHECKDEFERRED .......................................................... 156
+CLEANUP .......................................................................... 157
+CONTROLPREFIX ................................................................ 158
+COPYPENDING .................................................................. 159
+DEADLINE ......................................................................... 160
+DELETEOUTPUT .................................................................. 161
+DELETEWORK ..................................................................... 162
+DIGITS ............................................................................... 162
+DRAIN1 ............................................................................... 164
+DSCRL ............................................................................... 165
+DSNUTILB .......................................................................... 166
+EDPROCSUP ....................................................................... 167
+FASTSWITCH ..................................................................... 167
+FLASHCOPY ......................................................................... 168
+HRECALL ............................................................................ 169
+IGNOBJSETPARTS ............................................................... 169
+JOURNAL ............................................................................. 170
+LARGEOBJSIZE ................................................................. 171
+LOGPDS .............................................................................. 172
+MAXCONNECT ..................................................................... 173
+MAXIDLE ............................................................................ 174
+MAXMEMORY ..................................................................... 175
+MAXSERVERS ..................................................................... 176
+MAXWAIT ........................................................................... 176
+MEMORYLIMIT .................................................................... 177
+MONITORRATE .................................................................... 178
+NGTAUTO1 .......................................................................... 179
+NGTAUTO2 .......................................................................... 179
+NGTCKPT ............................................................................ 180
+NGTTSVR ............................................................................ 181
+OBJSETSTOGROUP ............................................................ 182
Chapter 7     NGT utility reports    215

NGT utility age report ............................................................. 215
Processed objects report (OBJREPRT) ..................................... 217
Chapter 8  NGT work data set control  
Overview of the NGT work data sets ......................................................... 221  
Controlling when NGT work data sets are deleted ........................................ 222  
Usage notes for NGT work data sets ............................................................ 222  
Aging NGT work data sets ........................................................................... 223  
Permanent work data sets and the NGTWORK command ................................ 225  
  NGTWORK command syntax ................................................................. 226  
  workoptions block ................................................................................. 227  
  Example of NGTWORK JCL .................................................................. 230  
  Using NGTWORK with +WORKMODE ................................................. 230

Chapter 9  NGT Checkpoint data set maintenance  
Reorganizing the NGT Checkpoint repository ............................................... 231  
Backing up the NGT system data sets ......................................................... 232  
Restoring NGT System data sets ................................................................. 233

Chapter 10  NGTDISP  
NGTDISP input ............................................................................................ 235  
  NGTDISP DD statements ......................................................................... 235  
  SYSIN syntax diagrams ......................................................................... 237  
  Keywords ............................................................................................... 238  
NGTDISP report columns ........................................................................... 241  
  Display utility report columns ............................................................... 242  
  Matrix report columns .......................................................................... 243  
  Passwords report .................................................................................. 244  
  BMCUTIL report .................................................................................. 244  
  BMCSYNC report ................................................................................ 246  
Examples ................................................................................................... 247  
Relevant automation ................................................................................... 252  
  XSUTDISP .......................................................................................... 253

Appendix A  Using NGT1DBUT to copy NGT VSAM work files  
NGT1DBUT syntax ...................................................................................... 255  
NGT1DBUT usage notes .............................................................................. 256  
NGT1DBUT sample JCL .............................................................................. 257
About this book

This book contains detailed information about the associated product or products. This preface explains the special conventions that the book uses, and how to access related publications.

If applicable, the preface also summarizes the major changes included in the latest release of the product.

Related publications

From the BMC Support Central website, you can use the following methods to access related publications that support your product or solution:


- View Quick Course videos (short overviews of selected product concepts, tasks, or features), which are available from the following locations:
  - Documentation Center (primary center and secured center)
  - Support Central (at http://www.bmc.com/support/mainframe-demonstrations)
  - BMC Mainframe YouTube channel (https://www.youtube.com/user/BMCSoftwareMainframe)


Products with online interfaces also offer online Help via the F1 key or, for graphical user interfaces (GUIs), via a Help button.
Tip
If you prefer hardcopy documentation, you can order it from your BMC sales representative or from Support Central. Also, from Support Central you can subscribe to receive proactive e-mail alerts when BMC issues notices.

Conventions

This document uses the following special conventions:

- All syntax, operating system terms, and literal examples are presented in this typeface.

- Variable text in path names, system messages, or syntax is displayed in italic text: `testsyst(instance)(fileName)`

- Menu sequences use a symbol to convey the sequence. For example, `Actions => Create Test` instructs you to choose the `Create Test` command from the `Actions` menu.

Syntax diagrams

The following figure shows the standard format for syntax diagrams:
The following example illustrates the syntax for a hypothetical DELETE statement. Because the FROM keyword, alias variable, and WHERE clause are optional, they appear below the main command line. In contrast, the tableName variable appears on the command line because the table name is required. If the statement includes a WHERE clause, the clause must contain a search condition or a CURRENT OF clause. (The searchCondition variable appears on the main line for the WHERE clause, indicating that this choice is required.)

The following guidelines provide additional information about syntax diagrams:

- Read diagrams from left to right and from top to bottom.
- A recursive (left-pointing) arrow above a stack indicates that you may choose more than one item in the stack.
- An underlined item is a default option.
- If a diagram shows punctuation marks, parentheses, or similar symbols, you must enter them as part of the syntax.
- In general, IBM commands, keywords, clauses, and data types are displayed in uppercase letters. However, if an item can be shortened, the minimum required portion might be shown in uppercase letters, with the remainder in lowercase (for example, CANcel).
- The following conventions apply to variables in syntax diagrams:
  - Variables are typically displayed in lowercase letters and are always italicized.
  - If a variable is represented by two or more words, initial capitals distinguish the second and subsequent words (for example, databaseName).
NGT utility products concepts

This chapter describes common concepts for all NGT data base utilities. We strongly suggest you familiarize yourself with these concepts, as you will see them over and over again, and understanding them thoroughly will allow you to get the most out of NGT utility products.

**Note**

Enabled NFM (ENFM Mode) is not supported by *any* version of NGT utility products on *any* version of DB2.

### Wildcards

Many keywords in NGT utility products employ the use of DB2 SQL wildcards to reference a series of similarly named objects. DB2 SQL wildcards are used precisely as they are used in a DB2 WHERE clause after the LIKE keyword. Any combination of characters, % sign, or _ sign can be used in either name.

In the simplest case, no wildcards are used. The index creator and index name, or the data base name and the table space name are specified explicitly, without wildcards. To reorganize all the indexes with a specific creator ID specify creator.%.

When masked names are used, an SQL query will be constructed and issued to obtain the names of all indexes that match a specified naming mask.

A line will be printed in the SYSPRINT and SUMMARY sections of the job listing for each object found.

**Example**

REORG TABLESPACE APPL1.XXA%

Generates one REORG TABLESPACE statement for every table space in the DB2 Catalog in data base APPL1 and whose table space name begins with the characters XXA.
Utility Parameters

Each NGT utility product has a set of parameters used to control how it processes and uses resources. There are also global parameters that apply to all utilities, these are preceded by a + sign.

When NGT utility products are installed, the default for each of these parameters is configured, separately for each DB2 subsystem. Each utility job can then override these default parameters if desired for exceptional circumstances. These default parameters, and any override, are listed in each utility job output.

The configuration job to set the default parameters can be changed and run at any time to alter the default settings. For more information, see the section on INSCNFIG in the BMC Next Generation Technology Installation Guide.

The documentation for each utility has a chapter describing each of these parameters and how to override them in the utility JCL.

Journal

NGT introduces server technology where multiple jobs work on a table space or set of table spaces. Each job writes a summary of its output to its usual DDs, and writes the full detailed output to a VSAM NGT work data set which is called the Journal.

The primary use of the Journal is to enable analysis by BMC Customer Support when questions or issues arise. The various methods of getting the Journal level output to your job output, or to a data set for sending to support are discussed elsewhere in this guide.

Servers

NGT utility products can expand parallelism and multitasking by invoking NGT Servers. NGT Servers are jobs automatically initiated by a master utility job. When an NGT utility is submitted, parameters determine the maximum number of servers
that can be initiated by the submitted job. The master job creates a task list to identify the work required to complete the utility. The servers select work from this list.

Each server can invoke up to 15 subtasks (depending on the utility being processed) to perform its given work. For some utilities you can control the degree of parallelism over these subtasks.

For more information about servers, see “Servers” on page 129.

Utility concurrent access control

In order to prevent contention between data base objects and the multiple jobs that are running, NGT utility products have used an internal control called a matrix. NGT utility products now also use the BMCUTIL and BMCSYNC common tables for this purpose.

Note

The BMCUTIL and BMCSYNC tables do not replace the Matrix.

The matrix

The matrix ensures the integrity of data base objects in error. The matrix prevents objects, which are currently being processed or which are flagged by processing as being in error, from being used concurrently with other utility jobs.

Example

Suppose that in job LOAD22, table space DB1.GLEDGER is being loaded with NGT Load and experiences a problem. Assuming that +CLEANUP(YES) is not specified then the job LOAD22 is marked as in error and the object DB1.GLEDGER is held in the matrix for job LOAD22.

The next night, if a job REORG95 is run using NGT Reorg, processing will see table space DB1.GLEDGER in the matrix and alternate processing will occur. There are two basic choices for alternate processing:

- Processing may continue to wait on the object in the matrix. In this scenario, if the job LOAD22 is corrected that same night and run to a successful completion, then table space DB1.GLEDGER would be freed from the matrix, allowing the NGT Reorg job REORG95 to continue.

- Processing may skip the object in the matrix and continue with the next statement or object. In this scenario, table space DB1.GLEDGER would not be processed further until the original error condition was resolved.
BMCUTIL table

The BMCUTIL table contains information about utilities that are currently running or started.

Table 1 on page 18 describes the contents of the BMCUTIL table. The utilities use the table to control the use of utility IDs. Each BMC utility must have a unique ID for restart purposes. If you have more than one BMC utility installed, all of these utilities should share the same BMCUTIL table.

The utilities insert rows into the BMCUTIL table during the UTILINIT phase and update the table as the job status changes. The utilities delete rows from the BMCUTIL table during the UTILTERM phase.

Table 1: Contents of the BMCUTIL table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILID</td>
<td>CHAR(16)</td>
<td>Utility identifier</td>
</tr>
</tbody>
</table>
| STATUS      | CHAR(1)   | Execution status of the utility:  
  ■ A (active, not executing command)  
  ■ I (initializing)  
  ■ P (pausing or pause-stopped)  
  ■ S (stopped)  
  ■ T (terminating)  
  ■ X (executing command) |
| UTILNAME    | CHAR(8)   | Name of the executing utility:  
  ■ CHECK  
  ■ COPY  
  ■ STATS  
  ■ LOAD  
  ■ NGT Load  
  ■ RECOVER  
  ■ REORG  
  ■ UNLOAD |
| PHASE       | CHAR(8)   | Current phase of the utility  
 (NGT Copy does not use this column.  
 (NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload)  
 The value for this column is always UTILINIT. |
<p>| USERID      | CHAR(8)   | User ID executing the utility |
| SSID        | CHAR(4)   | DB2 subsystem where the utility is running |</p>
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| RESTART    | CHAR(1)     | Restart option:  
■ N (not restart)  
■ P (RESTART(PHASE))  
■ Y (RESTART)  
DASD MANAGER PLUS does not use this column.  
*(NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload)* The value for this column is always N. |
| NOTEID     | CHAR(8)     | TSO user ID to be notified  
DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column.                                                                                                           |
| DBNAME     | CHAR(8)     | *(NGT Recover and REORG PLUS)* Name of the database containing the table or index space for which the last checkpoint was taken  
This value can be blank.  
The other utilities do not use this column.                                                                                     |
| SPNAME     | CHAR(8)     | *(NGT Recover and REORG PLUS)* Name of the table or index space for which the last checkpoint was taken  
This value can be blank.  
The other utilities do not use this column.                                                                                     |
| SPSTATUS   | CHAR(5)     | *(REORG PLUS)* Space status before the utility stopped  
The other utilities do not use this column.                                                                                          |
| COMMANDNO  | SMALLINT    | *(NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload)* Always 1  
For all other utilities, this column is always 0.                                                                                   |
| COMMAND    | VARCHAR(256)| First 256 characters of the utility command text  
NGT Recover, DASD MANAGER PLUS, NGT Copy, NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column.                                                                            |
| STATE      | LONG VARCHAR| Utility state and sync information  
DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column.                                                                                                 |
| START_TIMESTAMP | TIMESTAMP | Starting timestamp of the utility  
NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column.                                                                                                                      |
BMCUTIL table maintenance

When a utility abends, unwanted rows might remain in the BMCUTIL and BMCSYNC tables.

If you need to control expansion of these tables, use one of the following methods to delete rows:

- Use the TERM restart parameter on the EXEC statement to delete rows from the BMCUTIL and BMCSYNC tables. Do not delete any rows for instances of utilities that are awaiting restart or executing.

- Delete invalid rows in the BMCUTIL table. Do not delete any rows for instances of utilities that are awaiting restart or executing.

Then use the following SQL statement to delete rows from the BMCSYNC table:

```
DELETE
FROM creatorName.CMN_BMCSYNC
WHERE UTILID NOT IN
(SELECT UTILID FROM creatorName.CMN_BMCUTIL);
```

*Note*

The names of the BMCUTIL and BMCSYNC tables might have been changed at your site during installation.

BMCSYNC table

The BMCSYNC table contains information about the status of the objects that the currently executing utilities are accessing.

Table 2 on page 21 describes the contents of the BMCSYNC table. The BMCSYNC table synchronizes and controls access to DB2 spaces by concurrently executing BMC utility products. If you have more than one BMC utility installed, all of these utilities should share the same BMCSYNC table.

The following NGT utilities insert rows into BMCSYNC during the BEFOREACC phase and delete rows during the AFTERACC phase:

- NGT Check
- NGT Load
- NGT Reorg
- NGT Stats
- NGT Unload
All other utilities insert rows into BMCSYNC during the UTILINIT phase and delete rows during the UTILTERM phase; while the job executes, the utilities update the table when the status of the object changes.

### Table 2: Contents of the BMCSYNC table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| UTILID      | CHAR(16)  | Utility identifier
  *(NGT Recover)* This column is blank when a RECOVER UNLOADKEYS command creates the row and then a RECOVER BUILDINDEX command reads and deletes the row. |
| NAME1       | CHAR(8)   | Database name or creator name
  *(DASD MANAGER PLUS)* This value is the database name.
  *(CHECK PLUS, LOADPLUS, REORG PLUS, and UNLOAD PLUS)* If the value for NAME1 would exceed 8 bytes or the value for NAME2 would exceed 18 bytes, NAME1 contains the DBID for the object.
  *(NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload)* If the value for NAME1 would exceed 8 bytes, NAME1 contains the OBID in hexadecimal format. |
| NAME2       | CHAR(18)  | Space, table, or index name
  *(DASD MANAGER PLUS)* The BMCSTATS utility always inserts the space name (limited to a maximum of 8 characters).
  *(CHECK PLUS, LOADPLUS, REORG PLUS, and UNLOAD PLUS)* If the value for NAME1 would exceed 8 bytes or the value for NAME2 would exceed 18 bytes, NAME2 contains the table OBID or index ISOBID of the object in hexadecimal format.
  *(NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload)* If the value for NAME2 would exceed 18 bytes, NAME2 contains the OBID in hexadecimal format. |
| KIND        | CHAR(2)   | Type of object:
  ■ IP (index partition)
  ■ IX (index)
  ■ TB (table)
  ■ TP (table space partition)
  ■ TS (table space)
  ■ DD, DW, D1, D2 (dynamic work file allocation)
  ■ CI (copy information)
  ■ RD (restart data set block)
  ■ LK (limit key) |
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| PARTITION   | SMALLINT  | Physical partition number:  
|             |           | ■ Null or 0 for a single data set nonpartitioned space  
|             |           | ■ Data set number for a multi-data-set, nonpartitioned space  
|             |           | ■ Partition number for a partitioned space  
|             |           | *(all products except NGT Recover and RECOVERY MANAGER)* The value is null or 0 for any nonpartitioned space. |
| BMCID       | SMALLINT  | Internal identifier of the object  
|             |           | DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column. |
| UTILNAME    | CHAR(8)   | Name of the executing utility:  
|             |           | ■ CHECK  
|             |           | ■ CHECKIX  
|             |           | ■ COPY  
|             |           | ■ STATS  
|             |           | ■ LOAD  
|             |           | ■ NGTSTATS  
|             |           | ■ REBUILD  
|             |           | ■ RECOVER  
|             |           | ■ REORG  
|             |           | ■ UNLOAD |
| SHRLEVEL    | CHAR(1)   | Degree to which utilities can share this object:  
|             |           | ■ Blank means that no status is requested, and any other utility can obtain any status.  
|             |           | ■ S allows sharing among any number of SHRLEVEL S utilities.  
|             |           | ■ X indicates that exclusive control is required. No other utility can run with SHRLEVEL X.  
|             |           | *(For more information, see “Shared access levels of BMC utilities” on page 24.)* |
| STATUS      | CHAR(1)   | Status of the utility or object:  
|             |           | ■ Blank (indicates no processing has been done)  
|             |           | ■ C (for CHECK PLUS, indicates checked)  
|             |           | ■ L (for LOADPLUS, indicates loaded)  
|             |           | ■ U (for UNLOAD PLUS, indicates unloaded)  
|             |           | ■ R (for REORG PLUS, indicates reloaded)  
|             |           | DASD MANAGER PLUS does not use this column. |
| XCOUNT      | INTEGER   | Number of rows or keys processed in the current phase  
<p>|             |           | DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column. |</p>
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDNAME</td>
<td>CHAR(8)</td>
<td>Check, load, unload, or work ddname. DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column.</td>
</tr>
<tr>
<td>BLOCKS</td>
<td>INTEGER</td>
<td>Number of blocks for the check, load, unload, or work data set. DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column.</td>
</tr>
<tr>
<td>ORIG_STATUS</td>
<td>CHAR(8)</td>
<td>Encoded representation of the original DB2 status of the space. <em>(NGT Recover)</em> This column restores the DB2 status of a space after recovery, if necessary. DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column.</td>
</tr>
<tr>
<td>EXTRBA</td>
<td>CHAR(10)</td>
<td><em>(NGT Recover)</em> Log point at which this space was externalized. NGT Recover serialization logic uses this column. The other utilities do not use this column. <em>Note:</em> NGT Recover no longer uses EXTRBA.</td>
</tr>
<tr>
<td>STATE</td>
<td>LONG VARCHAR</td>
<td>Restart information for the space. For example, the STATE indicates the object state and sync information. DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column.</td>
</tr>
<tr>
<td>INSTANCE</td>
<td>SMALLINT</td>
<td><em>(RECOVERY MANAGER and NGT Recover)</em> Instance number of the current base objects (table and index). The default value is 1. The other utilities do not use this column.</td>
</tr>
</tbody>
</table>

**BMCSYNC table maintenance**

When a utility abends, unwanted rows might remain in the BMCUTIL and BMCSYNC tables.

If you need to control expansion of these tables, use one of the following methods to delete rows:

- Use the TERM restart parameter on the EXEC statement to delete rows from the BMCUTIL and BMCSYNC tables. Do not delete any rows for instances of utilities that are awaiting restart or executing.

- Delete invalid rows in the BMCUTIL table. Do not delete any rows for instances of utilities that are awaiting restart or executing.

Then use the following SQL statement to delete rows from the BMCSYNC table.

```
DELETE
FROM creatorName.CMN_BMCSYNC
```
WHERE UTILID NOT IN
(SELECT UTILID FROM creatorName.CMN_BMCUTIL);

*Note*
The names of the BMCUTIL and BMCSYNC tables might have been changed at your site during installation.

---

**Shared access levels of BMC utilities**

BMC utility jobs register DB2 objects in the BMCSYNC table.

The registering utility assigns a sharing level to each registered object. The sharing level controls access to that object from other BMC utilities. For partitioned DB2 spaces, registration is performed at the partition level.

*Note*
All BMC utility products use the BMCUTIL table to control the use of utility IDs, which identify executions of BMC utilities. Each BMC utility product must have a unique ID for restart purposes. This unique ID is stored in the BMCUTIL table. For more information about this table, see “BMCUTIL table” on page 18.

The BMCSYNC table allows multiple BMC utilities (or multiple instances of a utility) to operate concurrently on different partitions of a DB2 space if no nonpartitioning indexes are involved. In addition, some BMC utilities can operate concurrently on the same object or partition. For information about which products can operate concurrently, see the following table. For additional serialization and concurrency issues for each utility, see that utility’s reference manual.

The "Access level" column in the following table refers to the value of the SHRLEVEL column name in the BMCSYNC table (“BMCSYNC table” on page 20). The level can be one of the following values:

- **S** indicates shared access. Any other utility that registers with shared access (S) can run against the object.
- **X** indicates exclusive access. No other utility can run against the object.
- A blank value indicates that no status is requested and any other utility can run against the object.

**Table 3: Shared access levels of BMC utilities**

<table>
<thead>
<tr>
<th>Product</th>
<th>Access level</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK PLUS</td>
<td>S</td>
<td>None</td>
</tr>
<tr>
<td>Product</td>
<td>Access level</td>
<td>Additional information</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| DASD MANAGER PLUS (BMCSTATS) | S            | ■ If BMCSTATS is processing multiple objects and encounters an object that is held by another utility, the BMCSTATS job issues a warning. The warning identifies the object and the utility that is using it. BMCSTATS continues processing the next object.  
■ If BMCSTATS is processing an object and another utility requires exclusive control of that object, the other utility stops execution at initialization time. |
| LOADPLUS                    | X            | If you specify PART, LOADPLUS registers only the specified partitions with exclusive access (X). If no nonpartitioned indexes exist on the table space, you can run other utilities on different partitions while running this job. |
| NGT Check                   | S            | None                                                                                                                                                   |
| NGT Copy                    | S or blank   | If you specify COPY IMAGECOPY, NGT Copy registers the object with no access status (blank). Otherwise, NGT Copy registers the object with shared access (S). |
| NGT Load                    | X            | If you are loading specific partitions, NGT Load registers only the specified partitions with exclusive access (X).                                        |
| NGT Recover                 | X, S, or blank| NGT Recover registers an object with shared access (S) under the following conditions:  
■ The table space for an index is registered with shared access if the index is being rebuilt and its table space is not recovered in the same job.  
■ A table space partition is registered with shared access if the keys for that partition are unloaded with a RECOVER UNLOADKEYS operation.  
NGT Recover registers an object with no access status (blank) if you specify the following commands or options:  
■ The ACCUM command  
■ OUTCOPY ONLY  
■ INDEP OUTSPACE  
NGT Recover registers the object with exclusive access (X) in all other cases. |
| NGT Reorg                   | X            | If you are loading specific partitions, NGT Reorg registers only the specified partitions with exclusive access (X).                                        |
### Product Access level Additional information

<table>
<thead>
<tr>
<th>Product</th>
<th>Access level</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>NGT Stats</td>
<td>S</td>
<td>None</td>
</tr>
<tr>
<td>NGT Unload</td>
<td>S</td>
<td>None</td>
</tr>
<tr>
<td>RECOVERY MANAGER</td>
<td>S</td>
<td>None</td>
</tr>
<tr>
<td>REORG PLUS</td>
<td>X</td>
<td>If you specify PART, REORG PLUS registers only the specified partitions with exclusive access (X). If no nonpartitioned indexes exist on the table space, you can run other utilities on different partitions while running this job.</td>
</tr>
<tr>
<td>UNLOAD PLUS</td>
<td>S</td>
<td>None</td>
</tr>
</tbody>
</table>

---

**WARNING**  
Do not run an IBM utility, command, or SQL statement that attempts to manipulate the structure, data, or status of an object that a BMC utility is currently processing. For example, commands and SQL statements such as `STOP`, `START`, `EXCHANGE`, and `ALTER` will produce unpredictable results.

---

### Controlling contention

The contents of the matrix and BMC common tables can be manipulated using either of the following facilities:

- You can use the NGTDISP command as a SYSIN statement to display reports containing the contents of the Matrix, BMCUTIL, or BMCSYNC tables. For more information, see “NGTDISP” on page 235.

- Automation Control Point XSUTMTRX  
  Automation control points are REXX programs used to automate specific NGT processing. One such program is called XSUTMTRX and may be used to automatically determine how processing should proceed when an object in the matrix is encountered. For more information on the use of XSUTMTRX and automation in general, see the *BMC Next Generation Technology Automation Reference Manual*.

---

### Tape commands

The NGTTAPE statement lets you specify all tape usage information in the SYSIN of the BMC Next Generation Technology Reorg for DB2 for z/OS or BMC Next Generation Technology Copy for DB2 for z/OS products. This command supplies the copy destinations and tape details to the tape servers. Copies can go to multiple...
destinations - local primary, local backup, remote primary, and remote backup - and each can use different devices or multiple drives with varying RETPD, EXPDT.

**The NGTTAPE command**

The NGTTAPE command passes tape unit allocation data to the utility.

*Note*  
The NGTTAPE command runs at parser time in the master job, therefore BMC recommends that you insert NGTTAPE as the *first statement* in the SYSIN.

The NGTTAPE statement sets variables that can be used by any of the NGT automation control points. The automation logic required to read these variables and put the corresponding tape DDs in each tape server job is minimal and straightforward. During a new installation the automation control points needed for this are included.

Contact BMC Customer Support for assistance to integrate this function into an existing automation scheme.

**Figure 1: NGTTAPE command syntax**

```
NGTTAPE (ICBackupSpec)  
  LP  
  RP  
  LB  
  RB  
  UR (ICBackupSpec) — DDNAME(DDname) — ID(id)  
  MAXTAPESERVERS (n)  
```
NGTTAPE Keywords

NGTTAPE

*(required)* Indicates start of instruction

LP / RP / LB / RB / UR

*(required)* Indicates for which tape unit this information will apply. At least one tape unit must be specified.
- LP - Local primary copy
- RP - Remote primary copy
- LB - Local backup copy
- RB - Remote backup copy
- UR - User DD (See DDNAME on page 28). UR requires the DDNAME and ID keywords.

ICBackupSpec block

*(required)* Specification for tape unit. Must be enclosed in parentheses. See ICBackupSpec block keywords on page 30.

DDNAME(DDname)

DDNAME is required to name a user DD, other than one of the four copy destinations, for example Discard or Syspunch DDs.
Note

The ID keyword is also required.

Use a UR to provide the DSN, see following example.

Example

```plaintext
NGT_TAPE
UR(
    UNITNAME(CART)
    RETPD(30)
    MAXVOLS(5)
    STORCLAS(NGTSTOR)
)
DDNAME(TAPETEST)ID(UP)
```

UR rules are:

- You cannot use the DDNCOUNT keyword for this type of NGTTAPE statement. The default for this keyword is always be treated as 1.

- The DDNAME and ID keywords must be specified somewhere after the ICBBackupSpec block for the UR. This keyword is invalid for all other types (LP / RP / LB / RB).

- You can define up to 36 different user defined DDs with NGTTAPE UR statements within a single SYSIN stream. (The 37th will result in an error.)

**ID(id)**

If you are defining a user DD, you must use the ID keyword together with the DDNAME keyword. The system uses ID to distinguish between multiple user defined DDs.

`id` must be in the format Ux, where x can be any alphanumerical character.

**MAXTAPESERVERS(n)**

(optional) Specifies the maximum number of tape servers to be used for the selected tape unit. The value must be an integral from 1 through 32.

If the value was not specified during installation, MAXTAPESERVERS sets the number of tape servers in `+MAXSERVERS(D,T)`. If it was specified during installation value, then that value is used. Whichever value is used applies to `+SVRTAPE` for only this run.
ICBackupSpec block keywords

Note
The ICBackupSpec block keywords must be enclosed by parentheses.

UNITNAME(name)
(required) Specifies the tape unit name.

DDNCOUNT(n)
The number of DD names. A value from 1 to 10, default is 1.

MAXVOLS(n)
The maximum number of volumes that may be used by tape processing. A value from 1 to 255.

RETPD(nnm)
The retention period for any created tape files. A value from 1 to 9999

EXPDT(date)
The expiration date for any created tape files. A date in the format yyddd or yyyyddd (Julian date formats only). Examples: 09002 for 1/2/09, 2010032 for 2/1/2010

STORCLAS(name)
The storage class for any created tape files.

USER1(string), USER2(string), USER3(string)
Specifies additional attributes for an image copy data set (such as DCB information, management class, etc.).

Example
NGTTAPE LP(
    UNITNAME(CART)
    USER1(MGMTCLAS=ACLAS)
)
COPY TABLESPACE DB.TS

This creates an LP1 DD in the form: //LP1 DD
UNIT=(CART,.DEFER),VOL=(,,255),MGMTCLAS=ACLAS

Multiple JCL options may also be specified in each user keyword.
**Example**

For LBI RB copies specify:

```plaintext
NGTTAPE RB(
    UNITNAME(CTAPE)
    USER1(BLKSIZE=61400)
)
COPY TABLESPACE DB.TS
```

This creates an RP1 DD in the form: //RB1 DD
UNIT=(CTAPE,DEFER),VOL=(,,255),BLKSIZE=61400

**NGTTAPE global variables**

The NGTTAPE command generates global variables that are fetched by using the NGT_gread() function.

The following variables are generated:

- TAPE_LP_UNITNAME
- TAPE_LP_DDNCOUNT
- TAPE_LP_MAXVOLS
- TAPE_LP_RETPD
- TAPE_LP_EXPDT
- TAPE_LP_STORCLAS
- TAPE_LB_UNITNAME
- TAPE_LB_DDNCOUNT
- TAPE_LB_MAXVOLS
- TAPE_LB_RETPD
- TAPE_LB_EXPDT
- TAPE_LB_STORCLAS
- TAPE_RP_UNITNAME
- TAPE_RP_DDNCOUNT
- TAPE_RP_MAXVOLS
- TAPE_RP_RETPD
- TAPE_RP_EXPDT
- TAPE_RP_STORCLAS
- TAPE_RB_UNITNAME
- TAPE_RB_DDNCOUNT
- TAPE_RB_MAXVOLS
- TAPE_RB_RETPD
- TAPE_RB_EXPDT
The following global variables are included for UR (x is a single character from A to Z or from 0 to 9):

- TAPE_Ux_DDNAME
- TAPE_Ux_UNITNAME
- TAPE_Ux_MAXVOLS
- TAPE_Ux_RETPD
- TAPE_Ux_EXPDT
- TAPE_Ux_STORCLAS
- TAPE_Ux_USER1
- TAPE_Ux_USER2
- TAPE_Ux_USER3

**Allocation strategies**

An allocation strategy is the implementation of your requirements for data set allocation. This includes data set placement, allocation values (Primary quantity and Secondary quantity), data set reuse, and updating the DB2 Catalog values.

An allocation strategy:

- Determines how a utility is to allocate and use the table spaces and indexes that are being processed.

- Includes where and how big the data sets should be defined, and whether the existing data set should be written over or deleted and redefined.
Note
Regardless of whether a data set is user-managed or DB2-managed (STOGROUP defined), NGT utilities delete and redefine the VSAM data set. Therefore, it is vital to review this information.

It is possible to make the NGT utility products define data sets in precisely the way DB2 does, but that is not always the best way. The NGT utility products offer allocation features that enhance the process. For example, unused space in a table space can be reclaimed as a normal part of REORG. A PQTY value that will not fit on a volume and would cause a DB2 utility to fail can be dynamically reduced to fit.

Note
This section uses the terms priqty and secqty to refer to the VSAM allocation values for the data set. When referring specifically to the DB2 catalog values for these terms, PQTY and SQTY are used since these are the DB2 Catalog column names.

Allocation can be managed at one of the following levels:

- “VSAM data set allocation” on page 33
- “Automation control points for allocation” on page 38

VSAM data set allocation

The +ALLO and +VOLUMES parameters control VSAM data set allocation. They are utility-specific because there are separate parameters of each for NGT Reorg and NGT Load utilities.

Note
If SMS is used for DB2 data sets (+VOLUMES(*)), then the most important aspects of an allocation strategy are already defined to SMS. You only have to define the +ALLO parameter to select the (PRIQTY and SECQTY) allocation values.

Model AMS DEFINE

The values determined from the +ALLO and +VOLUMES parameters are used to build an IDCAMS DEFINE statement. These values are substituted in a Model DEFINE statement that can be modified to conform to your installation requirements. The example of a DEFINE model below is modified to support SMS where +VOLUMES(*) is specified.
**Note**

The variable &VOLUMES is included in the MANAGEMENTCLASS keyword. At run time, this variable is substituted with the value specified in the +VOLUMES parameter.

**Example**

For example, to use a management class named DB201, you would specify:

```
+VOLUMES(DB201)
```

```
DEFINE CLUSTER -
  (NAME('&DSNAMEC.') -
   LINEAR  REUSE SPEED &ERASE.ERASE -
   MANAGEMENTCLASS( -
      &VOLUMES )
   &UNIT(&PRI &SEC)   SHAREOPTIONS(3 3)) -
DATA(NAME('&DSNAMED.'))
```

You can also effectively disable the +VOLUMES parameter by changing the model DEFINE statement. If you remove the &VOLUMES variable from the model statement, then any +VOLUMES parameter specification will be ignored.

For more details about how to customize the model AMS DEFINE statement for your installation or for a specific DB2 subsystem, see “Managing the model AMS define” on page 39.

**STOGROUPS**

When a DB2 space is DB2-managed (defined in a DB2 Storage Group):

1 A NGT utility will dynamically add volumes when the data set needs to be extended.

2 If an NGT utility, while writing to a DB2 space, determines that the volume is full, the DB2 Catalog definition for the Storage Group will be accessed to find another volume.

3 The volume from the STOGROUP with the largest contiguous free space will be dynamically added to the list of volumes for the space using the IDCAMS ALTER ADDVOLUMES command.

4 Then the I/O will be retried, allowing the data set to extend to the new volume.

You can specify a default list during installation. Specifying +VOLUMES at execution time overrides the default. If you do not specify a volume override list, the new data set is defined on the same list of volumes used for the original data set, which is obtained from a LISTCAT of the original data set.
You can modify this model DEFINE statement during installation, for example, to support the use of SMS. If the variable &VOLUMES is used in a DEFINE keyword other than VOLUMES(&VOLUMES), for example MANAGEMENTCLASS(&VOLUMES), the value specified on the +VOLUMES parameter is substituted for it.

A disadvantage of specifying multiple volumes for the DEFINE is that all the volumes after the first become VSAM candidate volumes, which are used when the data set is extended past the first volume. However, since the candidate volume number is set at the time of the original DEFINE, there is no guarantee that those volumes will have space when the time comes to extend the data set.

**+ALLO**

The +ALLO parameter determines the data set PQTY and SEQTY value. It controls the value that the NGT Utility uses for a new table space allocation. Use this parameter in RRGPARMS to control table space allocations and in RRGPARMX to control index data set allocations.

**USED**

Specifies that the NGT Utility should use the space amount from the existing data set as the primary quantity for the new data set. This value is obtained from the high-used RBA in the VSAM catalog. This is the default value.

**ALLO**

Specifies that the NGT Utility should use the currently allocated space amount from the existing data set as the primary quantity for the new data set. The value is obtained from the high-allocated RBA in the VSAM catalog.

**DB2**

Specifies that the NGT Utility should use primary quantity and secondary quantity from the DB2 catalog as the values for the new data set. If the object is not DB2-managed or if the PQTY value is less than the high-used RBA for the data set, the NGT Utility uses the high-allocated value.
KEEPDB2

Specifies that the NGT Utility should use values from the (PQTY) DB2 catalog for the primary quantity of the new data set. This is different from +ALLO (DB2) in that +ALLO(DB2) is a conditional switch, whereas the KEEPDB2 parameter forces the NGT Utility to use the DB2 values.

If the data set is user-managed, the NGT Utility uses the values in use for primary quantity as it is defined to AMS for the new data set.

If a value of -1 is set for any object in the catalog for either PRIQTY or SECQTY, the NGT Utility correctly keeps the DB2 primary and secondary allocations.

MXIG

This parameter has a similar meaning to the MVS JCL SPACE parameter. MXIG requests that space allocated to the data set be the smaller of the PQTY needed or the space available on the volume. This parameter affects only primary space allocation. Secondary extents should be available when you use this parameter.

REUSE

Specifies that the NGT Utility should write the newly loaded data set over the original data set. +ALLO(REUSE) can be honored only if the VSAM cluster has the REUSE attribute and the existing size is at least large enough for the data set to be loaded.

Example

In the following example, the NGT Utility issues the currently allocated space to determine the primary allocation for the new data set.

+ALLO(ALLO)

+VOLUMES

The +VOLUMES parameter specifies a list of volumes to be used in the DEFINE statement for each new data set. You can abbreviate this parameter as +VOLS.

volserList

A list of volume serials.
+VOLUMES(*) uses SMS for DB2 data sets.

You may specify up to nine volsers or you can use the conditional form of the parameter. When the first value specified in the +VOLUMES parameter is /n (n is a number from 1 to 9), then n volumes are chosen from the list based on the space available on the volume.

For example, +VOLUMES(/1,AAAAAA,BBBB,CCCCC,DDDDD) will cause the utility to find a volume in the set of four volumes specified that contains contiguous space for the PRIQTY value of the dataset. The volume with the largest contiguous free space will be used.

The list of volsers is prefixed with /1 so that only the one volume in the list with the largest contiguous free space sufficient for the PRIQTY requirements of the data set will be used for the new data set allocation. The volume selection is done immediately before the DEFINE is issued, so this is a good method for balancing the space on a set of volumes.

You can specify a default list during installation. Specifying +VOLUMES at execution time overrides the default. If you do not specify a volume override list, the new data set is defined on the same list of volumes used for the original data set, which is obtained from a LISTCAT of the original data set.

A disadvantage of specifying multiple volumes for the DEFINE is that all the volumes after the first become VSAM candidate volumes that will be used when the dataset must be extended past the first volume. However, since the candidate volume number is set at the time of the original DEFINE, there is no guarantee that those volumes will have space when the time comes to extend the data set.

You can modify this model DEFINE statement during installation, for example, to support the use of SMS. If the variable &VOLUMES is used in a DEFINE keyword other than VOLUMES(&VOLUMES), for example MANAGEMENTCLASS(&VOLUMES), the value specified on the +VOLUMES parameter is substituted for it.

---

**Note**

You can use '*' for a single volume request and '*,' for a multi-volume request. Use +VOL(*,*,*) for three volumes.

To allocate volumes as they are needed, specify +VOL(*) and specify VOLUMES="*" in XSUTXTN to add volumes when extend is necessary.

XSUTALLO overrides this parameter.
**Example**

In the following example, +VOLUMES substitutes "*" for the &VOLUMES variable in the DEFINE statement.

```
+VOLUMES(*)
```

**Conditional volumes**

A conditional form of the +VOLUMES parameter specifies that a subset of the volumes are to be chosen from the list based on the space available on the volume just before allocation occurs.

When the first value specified in the +VOLUMES parameter is "/n" (n being a number from 1 to 8), then n volumes are chosen from the list based on the space available on the volume. n is the number of volumes to include in the final list.

In the following example, +VOLUMES requests a check of the space available on the four volumes specified to find one volume that contains contiguous space for the PRIQTY value of the data set.

```
+VOLUMES(/1,AAAA,BBBB,CCCC,DDDD)
```

The volume with the largest contiguous free space will be used. Only the one volume will be used for the data set because "/1" was specified.

To include multiple volumes in the final allocation list, specify "/n", where n is more than 1. Additional volumes will be specified as candidate volumes starting at the beginning of the list, not including the one picked for the primary volume, until the count is reached.

In the following example, volume CCCC has the largest contiguous free space of any of the named five volumes.

```
+VOLUMES(/4,AAAA,BBBB,CCCC,DDDD,EEEE)
```

In this case, the volumes list passes as follows:

```
(CCCC,AAAA,BBBB,DDDD)
```

**Automation control points for allocation**

The main automation control point for allocation is XSUTALLO. The NGTAUTO DD statement must be included in the utility job to name the library that contains these execs. For more information regarding automation control points, see the *BMC Next Generation Technology Automation Reference Manual*.
At the simplest, the XSUTALLO automation control point can return a list of volumes for every space to be defined. However, any level of complexity is permitted. The automation control point could read a sequential file or access a DB2 table to obtain a list of volumes for a particular space.

The XSUTALLO automation control point is passed many variables, some of them identifying the particular space, others specifying the allocation space requirements. The output from the automation control point is the volume(s), if needed, to use for the DEFINE and the primary and secondary quantity.

If you use DB2-managed data sets and want the utility to allocate data sets only on the volumes included in the storage group assigned to the table space or index, then use the Dynamic Allocation automation control point XSUTALLO. The list of volumes in the Storage Group defined for the object being allocated is passed to the automation control point. The automation control point can modify the volumes list or arrange the list in order of descending space. An internal function, _lspace() is provided to extract the space on a particular volume.

The automation control point is invoked even if the object is not DB2-managed. An internal variable indicates how the object is managed. Sample automation control points are distributed in the CNTL library.

You can use a user-defined automation control point to update the DB2 catalog. Call BMC Customer Support for examples and considerations.

Managing the model AMS define

NGT utility products include a model AMS DEFINE statement available in CNTL(JCLRXMDL) that you can use to allocate DB2 VSAM data sets. Copy and edit this member to suit your needs and define it to the NGT Checkpoint.

If you have specific requirements for the DEFINE statement or a need to allocate in a unit other than KB, this can easily be changed at the global level or for a specific DB2 subsystem.

**NGTAMS DEFINE**

The following statement can be run in the SYSIN DD statement of the standard NGT utility products JCL:

```bash
NGTAMS  DEFINE  DDNAME(ddname) GLOBAL
```
The NGTAMS DEFINE statement reads the DB2 VSAM Define model statement from the specified ddname and saves it to the NGT Checkpoint for use. If GLOBAL is specified this model will be used for all DB2 subsystems using this checkpoint data set. Without GLOBAL the model override is only for the DB2 subsystem specified in the NGT utility products job running this NGTAMS statement.

**NGTAMS DELETE**

The following statement can be run in the SYSIN DD statement of standard NGT utility products JCL:

```
NGTAMS — DELETE
```

```
GLOBAL
```

The NGTAMS DELETE statement removes any DB2 VSAM Define model statement override from the NGT Checkpoint. The GLOBAL keyword is used to specify which model to delete, the global model or the model specified for this DB2 subsystem.

**NGTAMS DISPLAY**

The following statement can be run in the SYSIN of standard NGT utility products JCL:

```
NGTAMS — DISPLAY
```

```
GLOBAL
```

The NGTAMS DISPLAY statement reports the DB2 VSAM Define model statement defined to the checkpoint. The GLOBAL keyword is used to specify which model to display, the global model or the model specified for this DB2 subsystem.

**Note**

Any DB2 VSAM Define model override can also be displayed using the Display AMS option on the CDBI menu.
NGT SQL language

NGT utilities provide a SQL-like language known as the NGT SQL Language. The language provides constructs and built-in functions similar to the DB2 SQL language.

NGT utility products use the NGT SQL language in the following contexts:

- WHERE clause
- Select list
- Index on expression (IOE)
- Check constraints
- NGT Utility manager
- ORDER by
- VALUE clause of the NGT LOAD utility

NGT SQL language elements

The basic elements of the NGT SQL language are expressions and predicates. When an expression is evaluated, it normally returns a value of a known data type defined in the language. You can use expressions to form predicates, or to manipulate data elements. When a predicate is evaluated, it normally results in a true/false condition. You can use predicates to define search conditions.

An expression in its basic form can be one of the following items:

- Function
- Case expression
- Constant
- Column name
- Special register
- Labeled duration
An expression can also include other expressions. The following diagram shows the syntax of an expression.

Figure 3: NGT SQL syntax for expressions

Figure 4: NGT SQL syntax for expressions (labeled-duration syntax)

Figure 5: NGT SQL syntax for expressions (CASE statement)
Figure 6: searched WHEN clause

```
WHEN searchCondition resultExpression
    NULL
```
Features and functions of the NGT SQL language

The following table lists the features and functions that NGT SQL supports.

<table>
<thead>
<tr>
<th>Feature/Function</th>
<th>Supported by NGT SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHERE clause elements</td>
<td></td>
</tr>
<tr>
<td>Column-to-column compares</td>
<td>Yes</td>
</tr>
<tr>
<td>Constant-to-Constant compares</td>
<td>No</td>
</tr>
<tr>
<td>Constant-to-Column compares</td>
<td>Yes</td>
</tr>
<tr>
<td>Host variables</td>
<td>No</td>
</tr>
<tr>
<td>WHERE clause keywords</td>
<td></td>
</tr>
<tr>
<td>BETWEEN, IN, LIKE keywords</td>
<td>Yes</td>
</tr>
<tr>
<td>NOT BETWEEN, NOT IN, NOT LIKE keywords</td>
<td>Yes</td>
</tr>
<tr>
<td>IS NULL, IS NOT NULL keywords</td>
<td>Yes</td>
</tr>
<tr>
<td>CONCAT keyword</td>
<td>Yes</td>
</tr>
<tr>
<td>Data manipulation</td>
<td></td>
</tr>
<tr>
<td>Concatenation</td>
<td>Yes</td>
</tr>
<tr>
<td>CONSTANT data formats</td>
<td></td>
</tr>
<tr>
<td>CHAR (character string), HEX (hexadecimal string), GRAPHIC (graphic string)</td>
<td>Yes</td>
</tr>
<tr>
<td>Feature/Function</td>
<td>Supported by NGT SQL</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>INTEGER, DECIMAL, FLOAT (floating point), DECFLOAT (decimal floating point)</td>
<td>Yes</td>
</tr>
<tr>
<td>Mixed data</td>
<td>Yes</td>
</tr>
<tr>
<td>Date formats: EUR/ISO/JIS/USA</td>
<td>Yes</td>
</tr>
<tr>
<td>Time formats: EUR/ISO/JIS/USA</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Date/Time elements</strong></td>
<td></td>
</tr>
<tr>
<td>Date/time durations</td>
<td>Yes</td>
</tr>
<tr>
<td>Day of week</td>
<td>Yes</td>
</tr>
<tr>
<td>Day of month</td>
<td>Yes</td>
</tr>
<tr>
<td>Day of year</td>
<td>Yes</td>
</tr>
<tr>
<td>Timestamp duration</td>
<td>Yes</td>
</tr>
<tr>
<td>Labeled durations</td>
<td>Yes</td>
</tr>
<tr>
<td>Week</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Arithmetic</strong></td>
<td></td>
</tr>
<tr>
<td>Date/time arithmetic</td>
<td>Yes</td>
</tr>
<tr>
<td>Decimal arithmetic</td>
<td>Yes</td>
</tr>
<tr>
<td>Float/decimal float arithmetic</td>
<td>Yes</td>
</tr>
<tr>
<td>Integer arithmetic</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Functions</strong></td>
<td></td>
</tr>
<tr>
<td>DATE(), TIME(), TIMESTAMP()</td>
<td>Yes</td>
</tr>
<tr>
<td>YEAR(), MONTH(), SECOND(), MICROSECOND()</td>
<td>Yes</td>
</tr>
<tr>
<td>DAYOFWEEK(), DAYOFMONTH(), DAYOFYEAR()</td>
<td>Yes</td>
</tr>
<tr>
<td>INTEGER(), DECIMAL(), FLOAT(), REAL(), SMALLINT()</td>
<td>Yes</td>
</tr>
<tr>
<td>BIGINT(), DECFLOAT()</td>
<td>Yes</td>
</tr>
<tr>
<td>VARCHAR()</td>
<td>Yes</td>
</tr>
<tr>
<td>VALUE()</td>
<td>Yes</td>
</tr>
<tr>
<td>LENGTH()</td>
<td>Yes</td>
</tr>
<tr>
<td>SOUNDEX()</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBSTR(), STRIP()</td>
<td>Yes</td>
</tr>
<tr>
<td>HEX(), CHAR()</td>
<td>Yes</td>
</tr>
<tr>
<td><em>(CHAR(numeric argument) is not supported for Index on Expression)</em></td>
<td></td>
</tr>
<tr>
<td>UPPER(), LOWER()</td>
<td>Yes</td>
</tr>
<tr>
<td>Feature/Function</td>
<td>Supported by NGT SQL</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>UCASE(), LCASE()</td>
<td>Yes</td>
</tr>
<tr>
<td>AVG(), MIN(), MAX(), SUM(), COUNT()</td>
<td>No</td>
</tr>
<tr>
<td>DIGITS()</td>
<td>Yes</td>
</tr>
<tr>
<td>VARGRAPHIC()</td>
<td>No</td>
</tr>
<tr>
<td>REPEAT()</td>
<td>Yes</td>
</tr>
<tr>
<td>LOCATE()</td>
<td>Yes</td>
</tr>
<tr>
<td>TRANSLATE()</td>
<td>Yes</td>
</tr>
<tr>
<td>(not supported for Index on Expression)</td>
<td></td>
</tr>
<tr>
<td>IFNULL()</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Operators</strong></td>
<td></td>
</tr>
<tr>
<td>AND, OR</td>
<td>Yes</td>
</tr>
<tr>
<td>NOT</td>
<td>No</td>
</tr>
<tr>
<td>+, -, *, /</td>
<td>Yes</td>
</tr>
<tr>
<td>Remainder divide (/)</td>
<td>Yes</td>
</tr>
<tr>
<td>Concatenation (</td>
<td></td>
</tr>
<tr>
<td><strong>Special registers</strong></td>
<td></td>
</tr>
<tr>
<td>CURRENT DATE</td>
<td>Yes</td>
</tr>
<tr>
<td>CURRENT TIME</td>
<td>Yes</td>
</tr>
<tr>
<td>CURRENT TIMESTAMP WITHOUT TIMEZONE</td>
<td>Yes</td>
</tr>
<tr>
<td>CURRENT TIMEZONE, CURRENT TIMESTAMP WITH TIMEZONE</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Other clauses</strong></td>
<td></td>
</tr>
<tr>
<td>GROUP BY, HAVING</td>
<td>No</td>
</tr>
<tr>
<td>JOINS, UNIONS</td>
<td>No</td>
</tr>
<tr>
<td>Correlated queries</td>
<td>No</td>
</tr>
<tr>
<td><strong>Miscellaneous</strong></td>
<td></td>
</tr>
<tr>
<td>Aliases</td>
<td>Yes</td>
</tr>
<tr>
<td>Data conversion for numeric comparisons</td>
<td>Yes</td>
</tr>
<tr>
<td>Feature/Function</td>
<td>Supported by NGT SQL</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Qualified column names</td>
<td>No</td>
</tr>
<tr>
<td>Synonyms</td>
<td>Yes</td>
</tr>
<tr>
<td>Views</td>
<td>Yes</td>
</tr>
<tr>
<td>Number of predicates supported</td>
<td>700</td>
</tr>
<tr>
<td>External file support for IN predicate</td>
<td>Yes</td>
</tr>
<tr>
<td>Subselect in an IN predicate</td>
<td>Yes</td>
</tr>
<tr>
<td>FIELDPROC in a WHERE clause</td>
<td>No</td>
</tr>
</tbody>
</table>

**Constants that NGT SQL supports**

This topic lists and describes the constants that NGT SQL language supports.

---

**Note**

All constants have the attribute NOT NULL.

---

**Integer constants**

An *integer constant* specifies an integer as a signed number with a maximum of 19 digits and without a decimal point.

The data type of an integer constant is:

- A large integer if the value is within the range of a large integer
- A big integer if the value is outside the range of a large integer, but within the range of a big integer
- A long decfloat constant if the value is defined outside the range of big integer values

**Floating-point constants**

A *floating-point constant* specifies a double-precision floating-point number as two numbers separated by an E:

- The first number can include a sign and a decimal point.
- The second number can include a sign but not a decimal point.

The constant's value is the product of the first number and the power of 10 specified by the second number. If the value is outside the range of floating-point numbers,
the utility converts the value to a long decimal-floating point constant, as explained in Decimal floating-point constants on page 49.

**Decimal constants**

A *decimal constant* is a signed or unsigned number of no more than 31 digits that includes a decimal point.

The precision is the total number of digits, including those (if any) to the right of the decimal point. The total includes all leading and trailing zeros. The scale is the number of digits to the right of the decimal point, including trailing zeros.

**Decimal floating-point constants**

A *decimal floating-point constant* specifies a decimal floating-point number as two numbers separated by an E:

- The first number can include a sign and a decimal point.
- The second number can include a sign but not a decimal point.

The value of the constant is the product of the first number and the power of 10 specified by the second number. The value must be within the range of DECFLOAT(34).

Basically, a constant is a decimal floating-point constant only if one of the following conditions exists:

- The value is specified as two numbers separated by E, and the value is outside the range of a floating-point constant.
- The value is specified as a number that does not contain E, and the number has more than 31 digits.

In addition to numeric constants, you can use the following special values to specify decimal floating-point special values:

- INF or INFINITY — represents infinity
- NAN — represents quiet not-a-number
- SNAN — represents signaling not-a-number

You can precede the special values by an operational sign (+ or -).

When you use special values in a predicate, the following order of precedence applies:

\[-\text{NAN} < -\text{SNAN} < -\text{INFINITY} < -0 < 0 < \text{INFINITY} < \text{SNAN} < \text{NAN}\]
Character string constants

A character string constant specifies a fixed-length character string that does not exceed 255 characters.

The string is a sequence of characters enclosed by a pair of apostrophes ('). The string length does not include the delimiters. If the string itself contains a string delimiter, then it should be duplicated. Blanks within the string are preserved.

Hexadecimal constant

A hexadecimal constant is a string representation of hexadecimal digits in the form of \( X'\text{HH...HH}' \).

The number of hexadecimal digits must not exceed 510. A hexadecimal digit can be a numeric digit, any letter from A through F, or a combination. Each pair of hexadecimal digits represents a character.

Graphic string constant

A graphic string constant specifies a fixed-length graphic string that does not exceed 128 characters. The string is in the following format:

\[ G'<gg...gg>' \]

\( gg \) is a graphic character (double byte) enclosed by a shift-in '<' and a shift out '>' characters.

DATE string constant

A DATE string constant contains a value that conforms to one of the valid formats for string representations of dates, subject to the following rules:

- Leading blanks are not allowed.
- You can omit leading zeros from the month and day elements of the date. An implicit specification of 0 is assumed for any omitted digit.
- You must include leading zeros for the year element of the date.
- You can include trailing blanks.

TIME string constant

A TIME string constant must contain a value that conforms to one of the valid formats for string representations of times, subject to the following rules:

- Leading blanks are not allowed.
You can omit the seconds element of the time.
You can include trailing blanks.
If the USA format is not used and the minutes and seconds are all zeros, the hour can be 24. If the format is USA, \textit{mm} may be omitted but \textit{hh} must be specified.

\textbf{TIMESTAMP} string constant or \textbf{TIMESTAMP WITH TIMEZONE} constant

A string representation of a timestamp is a character string whose value conforms to a date string constant concatenated with a time string constant and separated by a hyphen (-).

A \textbf{TIMESTAMP} string constant must contain a value that conforms to a date string constant concatenated with a time string constant, and separated by a hyphen. It is subject to the following rules:

- The string is at least 16 characters.
- Imbedded blanks are not allowed.
- Trailing blanks are allowed.
- \textit{Optionally} the constant can be concatenated with a maximum of 12 digits, representing fractions of a second, called \textit{precision}.

Also, you can optionally append a time zone to the string according to the following syntax:

- \texttt{+HH:MM}
- \texttt{-HH:MM}

The variables are defined as follows:

- You can specify \texttt{HH} as a single \texttt{H}.
- You can specify \texttt{MM} as a single \texttt{M}. If you include \texttt{MM} or \texttt{M}, it must follow the colon; if you omit \texttt{MM} or \texttt{M}, omit the colon.

\textbf{Example}

A timestamp string of precision 6 with a time zone:
\texttt{2010-12-31-12.00.50.123456-6:00}
Special registers that the NGT SQL language supports

This topic lists and describes the special registers that NGT SQL language supports.

**CURRENT DATE**

The CURRENT DATE special register specifies a date that is based on a reading of the extended time-of-day (TOD) clock when the SQL statement is executed for a particular row of a table. The TOD clock value is then converted to local time, using the current time zone defined to the IBM z/OS system.

If, *in a single statement*, you use CURRENT DATE more than once or use it with CURRENT TIME or CURRENT TIMESTAMP, all corresponding values are based on a single clock reading.

Specifying CURRENT_DATE is equivalent to specifying CURRENT DATE.

**CURRENT TIME**

The CURRENT TIME special register specifies a time that is based on a reading of the extended time-of-day (TOD) clock when the SQL statement is executed for a particular row of a table. The TOD clock value is then converted to local time, using the current time zone defined to the IBM z/OS system.

If, *in a single statement*, you use CURRENT DATE more than once or use it with CURRENT TIME or CURRENT TIMESTAMP, all corresponding values are based on a single clock reading.

Specifying CURRENT_TIME is equivalent to specifying CURRENT TIME.

**CURRENT TIMESTAMP(n)**

The CURRENT TIMESTAMP special register specifies a timestamp that is based on a reading of the extended time-of-day (TOD) clock when the SQL statement is executed for a particular row of a table. The TOD clock value is then converted to local time, using the current time defined to the IBM z/OS system.

Specifying CURRENT_TIMESTAMP is equivalent to specifying CURRENT TIMESTAMP.

Specifying (n) generates a timestamp with n precision. Valid values for n are 0 through 12. The default is 6.
CURRENT TIMEZONE

The CURRENT TIME ZONE special register specifies a value that contains the difference between UTC and local time. The data type is DECIMAL(6,0).

The difference between UTC and local time is represented by a time duration.

Subtracting CURRENT TIME ZONE from a local time converts the local time to UTC.

CURRENT TIMESTAMP(n) WITH TIME ZONE

The CURRENT TIMESTAMP WITH TIME ZONE special register specifies a timestamp that is based on a reading of the extended time-of-day (TOD) clock when the SQL statement is executed for a particular row of a table. The TOD clock value is then converted to local time using the implicit time zone defined in DSNHDECP; if DSNHDECP does not include a time-zone definition, the utility uses the current time zone defined to the IBM z/OS system.

Specifying (n) generates a timestamp with n precision. Valid values for n are 0 through 12. The default is 6.

Note

CURRENT TIMESTAMP might be referenced in a timestamp with time zone context (for example, when compared with a timestamp with a time zone column). In that case, the utility bases the implicit time zone for CURRENT TIMESTAMP on the implicit time zone system parameter, which could be a different value from the CURRENT TIME ZONE special register.

Functions that NGT SQL supports

This topic lists and describes the functions that NGT SQL language supports.

BIGINT function

The BIGINT function returns a very large integer representation of a number.
**numericExpression**

The `numericExpression` is converted to a big integer data type from one of the following numerical data types:
- SMALLINT
- INTEGER
- BIGINT
- FLOAT
- REAL
- DECIMAL
- DECFLOAT

The numerical expression is not rounded before conversion and any fractional part is truncated. If the result exceeds the allowable values for a BIGINT, an overflow condition occurs. The valid range is -9223372036854775808 through 9223372036854775807.

**stringExpression**

The `stringExpression` can be in any external numerical format. It is first converted to its corresponding format, and then converted to BIGINT. If the CCSID of the string is not in the EBCDIC encoding scheme, the CCSID is cast to EBCDIC CCSID(37) before conversion takes place.

---

**Note**

If a constant, then the argument must adhere to integer value rules. Specifying BIGINT(1.5) causes an error since 1.5 is not an integer.

If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.

---

**CHAR function**

The CHAR function returns a fixed-length character string representation of one of the following values:
- An integer number if the first argument is a small, large, or big integer
- A decimal number if the first argument is a decimal number
- A floating-point number if the first argument is a single or double precision floating-point number
- A decimal floating-point number if the first argument is a decimal floating-point number
- A character string value if the first argument is any type of string
- A date/time value if the first argument is a date, time, timestamp, or timestamp with time zone
The arguments are defined as follows:

**integerExpression**

`integerExpression` returns a value that is a built-in integer data type (SMALLINT, INTEGER, or BIGINT).

The result is the fixed-length character string representation of the argument in the form of an integer constant. The result consists of `n` characters that are the significant digits that represent the value of the argument. If the argument is negative, the result has a preceding minus sign.

The result is left justified. The result length must be 6 for a small integer, 11 for a large integer, and 20 for a big integer. Shorter results are padded on the right with blanks to achieve the required length.

A positive value always includes one leading blank.

**decimalExpression**

`decimalExpression` returns a value that is a built-in decimal data type. The result is the fixed-length character string representation of the argument. The result includes a decimal point and up to `p` digits; `p` is the precision of the `decimalExpression` with the preceding minus sign if the argument is negative. If the number of bytes in the result is less than the defined length of the result, the result is padded on the right with blanks. The decimal point character is a period (.) or a comma (,) based on DSNHDECP specifications.

**floatingExpression**

`floatingExpression` returns a value that is a built-in floating-point data type (DOUBLE or REAL).
The result is the fixed-length character string representation of the argument in the form of a floating-point constant. The length of the result is 15 bytes for a short floating point and 23 for a long floating point in the following format:

+/-.mE+-nn

/ denotes an OR condition, m represents mantissa, and n represents an exponent.

**decfloatExpression**

*decfloatExpression* returns a value that is a built-in decimal floating data type (long or short).

The result is the fixed-length character string representation of the argument in the form of a floating-point constant. The length of the result is 34 bytes for a short decfloat and 42 for a long decfloat in the form of: +/-mE+-nnn and 42 bytes for a long floating point number in the following form:

+/-.mE+-nnnn

/ denotes an OR condition, m represents mantissa, and n represents an exponent.

If the DECFLOAT value is one of the special values INFINITY, SNAN, or NAN, the strings INFINITY, SNAN, or NAN, respectively, are returned. If the special value is negative, a minus sign is the first character in the returned string.

**stringExpression, length**

*stringExpression* must be an expression that returns a value of a built-in character string whose length equals the value of the length argument.

The length value must be an integer constant not exceeding 255.

**datetimeExpression, format**

**datetimeExpression, string**

*datetimeExpression* must be an expression that is one of the following built-in data types:

- DATE
- TIME
- TIMESTAMP
- TIMESTOMP WITH TIMEZONE

The expression is converted to an external format using the format template, if applicable.
The format must be defined as follows:

<table>
<thead>
<tr>
<th>Expression type</th>
<th>Format type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>ISO, USA ,EUR, JIS, LOCAL</td>
</tr>
<tr>
<td></td>
<td>If the format is not specified, it defaults to ISO.</td>
</tr>
<tr>
<td></td>
<td>Additional non-conventional formats can be specified in the form of a string of date format elements. Valid values are:</td>
</tr>
<tr>
<td></td>
<td>■ MM for month</td>
</tr>
<tr>
<td></td>
<td>■ DD for day</td>
</tr>
<tr>
<td></td>
<td>■ YY or YYYY for year</td>
</tr>
<tr>
<td></td>
<td>You can specify these elements in any order and can delimit them with any character, number, or series of characters or numbers. Although the process does not require a delimiter, it treats any other characters as constants in their specified position.</td>
</tr>
<tr>
<td>TIME</td>
<td>ISO, USA ,EUR, JIS, LOCAL</td>
</tr>
<tr>
<td></td>
<td>If the format is not specified, it defaults to ISO.</td>
</tr>
<tr>
<td></td>
<td>Additional non-conventional formats can be specified in the form of a string of time format elements. Valid values are:</td>
</tr>
<tr>
<td></td>
<td>■ HH for hour</td>
</tr>
<tr>
<td></td>
<td>■ MM for minute</td>
</tr>
<tr>
<td></td>
<td>■ SS for second</td>
</tr>
<tr>
<td></td>
<td>■ XM for AM or PM</td>
</tr>
<tr>
<td></td>
<td>You can specify these elements in any order and can delimit them with any character, number, or series of characters or numbers. Although the process does not require a delimiter, it treats any other characters as constants in their specified position.</td>
</tr>
<tr>
<td>TIMESTAMP WITHOUT TIMEZONE</td>
<td>The result is the character string representation of the timestamp. If $datetimeExpression$ is a TIMESTAMP(0) value, the length of the result is 19. If $datetimeExpression$ is a TIMESTAMP ( integer ) value, the length of the result is 20+ integer . Otherwise, the length of the result is 26.</td>
</tr>
<tr>
<td></td>
<td>The second argument of $string$ may be specified with timestamp(6) as a formatting template that has date/time/microseconds elements. Valid values are:</td>
</tr>
<tr>
<td></td>
<td>■ MM for month</td>
</tr>
<tr>
<td></td>
<td>■ DD for day</td>
</tr>
<tr>
<td></td>
<td>■ YY or YYYY for year</td>
</tr>
<tr>
<td></td>
<td>■ HH for hour</td>
</tr>
<tr>
<td></td>
<td>■ MM for minute</td>
</tr>
<tr>
<td></td>
<td>■ SS for second</td>
</tr>
<tr>
<td></td>
<td>■ XM for AM or PM</td>
</tr>
<tr>
<td></td>
<td>■ NN for hundredths of a second</td>
</tr>
<tr>
<td></td>
<td>■ NNNN for milli-second</td>
</tr>
<tr>
<td></td>
<td>■ NNNNNNNN for a microsecond.</td>
</tr>
<tr>
<td>Expression type</td>
<td>Format type</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TIMEZONE WITH TIMEZONE</td>
<td>The result is the character string representation of the timestamp with time zone, formatted as <em>yyyy-mm-ddhh.mm.ss.nnnnnn±th:tm</em> with the appropriate number of <em>n</em> characters for the precision of the timestamp. If <code>datetimeExpression</code> is a <code>TIMESTAMP (0)</code> with timezone, the length of the result is 25. If <code>datetimeExpression</code> is a <code>TIMESTAMP (integer)</code> with timezone, the length of the result is 26+ <code>integer</code>. The second argument must not be specified.</td>
</tr>
</tbody>
</table>

**Note**

If the argument can be null, the result can be null; if the argument’s value is null, the result is the null value.

---

**DATE function**

The `DATE` function returns an internal representation of one of the following date data types:
- DATE
- TIMESTAMP
- TIMESTAMP WITH TIMEZONE
- CHAR
- INTEGER

```sql
DATE(
    stringExpression
)
```

**stringExpression**

`stringExpression` must be a valid string representation of a date or timestamp with or without a timezone. If the encoding of the character string is not EBCDIC, the character string is cast to an EBCDIC CCSID(37) before conversion takes place.

**integerExpression**

`integerExpression` must be a positive integer not exceeding 3652059. This value represents the number of days since Jan 1,0001 plus 1.
**dateExpression**

The result is the date returned by the expression.

**timestampExpression**

The result is the date part of the timestamp.

In the case of a timestamp with a time zone, the expression is cast to `TIMESTAMP` without timezone, and then the date portion is extracted.

**Example**

```
Date('1991-12-31')
Date(COL01+100 days)                  Where COL01 is either a date or timestamp with/out time zone
Date('1991'||'-'||SUBSTR(COL02,1,2)) Where COL02 is a var/character field having a valid day number.
Date(22)                             returns 0001-01-22
```

**Note**

If the argument can be null, the result can be null; if the argument’s value is null, the result is the null value.

---

**DAY function**

The `DAY` function returns an integer data type.

```
DAY(expression)
```

The `expression` must return one of the following built-in data types:

- DATE
- TIMESTAMP
- `TIMESTAMP WITH TIMEZONE`
- Date duration
- Timestamp duration
A CHAR/VARCHAR argument is not accepted unless converted by a DATE, TIMESTAMP, or TIMESTAMP_TZ function.

A date duration occurs as the result of subtracting two dates; a timestamp duration occurs as the result of subtracting two timestamps with precision=6. The result is the day portion of the derived date or date/timestamp duration.

If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.

---

**Example**

```sql
DAY (DATE ('1991-12-31'))
```

Will result in an integer of value = 31

It is invalid to specify `DAY ('1991-12-31')` since the argument is of type CHAR.

---

**Example**

```sql
DAY (DATE ('1991-12-31') - DATE ('1995-12-31'))
```

Will result in an integer of value = 0

---

**DAYS function**

The DAYS function returns an integer data type.

```sql
DAYS(expression)
```

The *expression* must return one of the following built-in data types:

- DATE
- TIMESTAMP
- TIMESTAMP WITH TIMEZONE
- CHAR

The result is 1 more than the number of days from January 1, 0001 to the date represented by the argument after conversion to a date data type.

---

**Note**

If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.
DAYOFMONTH function

The DAYOFMONTH function returns the day part of its argument. The function is similar to the DAY function, except DAYOFMONTH does not support a date or timestamp duration as an argument.

```
DAYOFMONTH( expression )
```

The `expression` must return one of the following built-in data types:

- DATE
- TIMESTAMP
- TIMESTAMP WITH TIMEZONE
- Valid string representation of a date or timestamp with or without a time zone

**Note**
The `expression` cannot be specified as a constant.

If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.

**Example**

```
UNLOAD FORMAT UNLOAD
SELECT CUSTOMER_NUM, DAYOFMONTH(BILLING_DATE)
FROM NWREGION.ACCTS_RECEIVABLE
REORG TABLESPACE NWREGION.ARTABLES
DISCARD FROM TABLE NWREGION.ACCTS_RECEIVABLE
WHERE (DAYOFMONTH(BILLING_DATE) = 15)
```

**Note**
Parentheses are not required when specifying a WHERE clause condition.

DAYOFWEEK function

The DAYOFWEEK function returns an integer in a range of 1 through 7 that represents the day of the week, where 1 is Sunday and 7 is Saturday.

```
DAYOFWEEK( expression )
```

The `expression` must return one of the following built-in data types:

- DATE
- TIMESTAMP
- TIMESTAMP WITH TIMEZONE
Valid string representation of a date or timestamp with or without a time zone

**Note**

The *expression* cannot be specified as a constant.

If the argument can be null, the result can be null; if the argument’s value is null, the result is the null value.

### Example

```sql
UNLOAD FORMAT CSV
SELECT EMPLOYEE_NUM, SALARY, DAYOFWEEK(PAYDATE)
FROM SWREGION.ACCTS_PAYABLE
REORG TABLESPACE SWREGION.APTABLES
DISCARD FROM TABLE SWREGION.ACCTS_PAYABLE
WHERE (DAYOFWEEK(PAYDATE) = 3)
```

**Note**

Parentheses are not required when specifying a WHERE clause condition.

---

## DAYOFYEAR function

The DAYOFYEAR function returns a large integer in a range of 1 to 366 that represents the day of the year, where 1 is January 1.

```
DAYOFYEAR(expression)
```

The *expression* must return one of the following built-in data types:

- DATE
- TIMESTAMP
- TIMESTAMP WITH TIMEZONE
- Valid string representation of a date or timestamp with or without a time zone

**Note**

The *expression* cannot be specified as a constant.

If the argument can be null, the result can be null; if the argument’s value is null, the result is the null value.

### Example

```sql
UNLOAD FORMAT UNLOAD
SELECT MEMBERSHIP_NUM, DAYOFYEAR(ANNIVERSARY_DATE)
FROM REGION49.MEMBERSHIP_RECS
REORG TABLESPACE REGION49.MEMBERS7
DISCARD FROM TABLE REGION49.MEMBERSHIP_RECS
WHERE (DAYOFYEAR(ANNIVERSARY_DATE) > 360)
```
Note
Parentheses are not required when specifying a WHERE clause condition.

**DECFLOAT function**

The DECFLOAT function returns a DECFLOAT representation of a number using the precision specified in the second argument.

```
DECFLOAT( numericExpression ,precision )
```

Note
If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.

The arguments are defined as follows:

**numericExpression**

`numericExpression` is converted to a DECFLOAT from one of the following numerical data types:
- SMALLINT
- INTEGER
- BIGINT
- FLOAT
- REAL
- DECIMAL
- DECFLOAT

**stringExpression**

If the argument is a string expression, it can be in any external numerical format. It is converted directly to DECFLOAT with the specified precision. If the CCSID of the string is not in EBCDIC encoding scheme, it is cast to EBCDIC CCSID(37) before conversion takes place.

Other special strings are accepted and converted to their internal DECFLOAT values. These include:
- NAN
- SNAN
- INF
The precision can be either short (16) or long (34).

**DECIMAL or DEC function**

The DECIMAL function returns a decimal representation of a number.

```
{DECIMAL|DEC}( numericExpression precision, scale ) stringExpression
```

The arguments are defined as follows:

- **numericExpression**
  
  `numericExpression` is converted to a decimal data type with the specified precision and scale. The data type of the numeric expression can be:
  - SMALLINT
  - INTEGER
  - BIGINT
  - FLOAT
  - REAL
  - DECIMAL
  - DECFLOAT

- **stringExpression**

  If the argument is a string expression, it can be in any external numerical format. It is converted directly to decimal data type. If the string expression has a decimal point, the number is converted to the implied precision and scale, then rescaled to the specified precision and scale. No rounding takes place. Truncation of the fractional part may take place. A decimal point conversion overflow will be declared if a loss of precision is detected.

  If the CCSID of the string is not in EBCDIC encoding scheme, it is cast to EBCDIC CCSID (37) before conversion takes place.

  Other special strings are accepted and converted to their internal DECFLOAT values. These include:
  - NAN
- SNAN
- INF
- INFINITY

**precision**

The precision argument must be an integer constant with a value in the range of 1 through 31. The value of this second argument specifies the precision of the result.

The default value depends on the data type of the first argument, as follows:

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default precision value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>5</td>
</tr>
<tr>
<td>INTEGER</td>
<td>11</td>
</tr>
<tr>
<td>BIGINT</td>
<td>19</td>
</tr>
<tr>
<td>DECFLOAT</td>
<td>31</td>
</tr>
<tr>
<td>All other cases</td>
<td>15</td>
</tr>
</tbody>
</table>

**scale**

The scale argument must be an integer constant that is greater than or equal to zero and less than or equal to precision. The value specifies the scale of the result. The default value is 0.

**Note**

If the argument can be null, the result can be null; if the argument’s value is null, the result is the null value.

**DIGITS function**

The DIGITS function returns a character string representation of the absolute value of a number.

```
digits( numericExpression )
```

*numericExpression* must return a value that is a SMALLINT, BIGINT, INTEGER, or DECIMAL built-in numeric data type.
Table 5: Output lengths for the BIGINT function

<table>
<thead>
<tr>
<th>Data type</th>
<th>Output length</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>5</td>
</tr>
<tr>
<td>BIGINT</td>
<td>19</td>
</tr>
<tr>
<td>INTEGER</td>
<td>10</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>( p ) if the argument is a decimal with a precision of ( p )</td>
</tr>
</tbody>
</table>

*Note*
If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.

**FLOAT function**

The FLOAT function returns a floating-point representation of a number.

\[
\text{FLOAT}( \text{numericExpression}, \text{precision}, \text{stringExpression} )
\]

The arguments are defined as follows:

*numericExpression*

The numeric expression is converted to a floating point number of length (8) from one of the following numerical data types:
- SMALLINT
- INTEGER
- BIGINT
- FLOAT
- REAL
- DECIMAL
- DECFLOAT
- @ROWID

Decimal precision might be lost because a floating-point number is not an exact representation of a decimal field.
If the argument is a string expression, it can be in any external numerical format. It is first converted to its corresponding format, and then converted to a float. If the CCSID of the string is not in EBCDIC encoding scheme, the CCSID is cast to EBCDIC CCSID(37) before conversion takes place.

Float is capable of converting the value returned by the internal @ROWID function into a floating-point value between 0 and 1. This value can be used to generate random numbers.

The precision can be either short (16) or long (34).

Note

If the argument can be null, the result can be null; if the argument’s value is null, the result is the null value.

The HEX function returns a hexadecimal representation of a value as a fixed or variable character string.

expression must be an expression that returns a value of any of the following data types:

- SMALLINT
- INTEGER
- BIGINT
- FLOAT
- REAL
- DECIMAL
- DECFLOAT
- DATE
- TIME
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
CHAR, VARCHAR, GRAPHIC, VARGRAPHIC

Note

A character string must not have a maximum length greater than 4000. A graphic string must not have a maximum length greater than 2000. The HEX function is currently restricted to EBCDIC tables. If the argument can be null, the result can be null; if argument's value is null, the result is the null value.

The result is a string of hexadecimal digits. The first two represent the first byte of the argument, the next two represent the second byte of the argument, and so on. If the argument is a datetime value, the result is the hexadecimal representation of the internal form of the argument. If the argument is a numeric value, the result is the hexadecimal representation of the decoded form of the argument.

If the argument is a fixed-length string and the length of the result is less than 255, the result is a fixed-length string. Otherwise, the result is a variable-length string with a length attribute that depends on the following rules:

- If the argument is not a variable-length string, the length attribute of the result string is the same as the length of the result.
- If the argument is a variable character or binary string, the length attribute of the result string is twice the length attribute of the argument.
- If the argument is a variable graphic string, the length attribute of the result string is four times the length attribute of the argument.

**HOUR function**

The HOUR function returns an integer data type.

\[
\text{HOUR(expression)}
\]

The expression must return one of the following built-in data types:

- TIME
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
TIME DURATION
TIMESTAMP DURATION

Note
A time duration occurs as the result of subtracting two times; a timestamp duration occurs as the result of subtracting two timestamps of precision=6. The result is the hour portion of the derived time or time/timestamp duration.
A CHAR/VARCHAR argument is not accepted unless converted by a TIME, TIMESTAMP, or TIMESTAMP_TZ function.

Example
HOUR (TIME ('13.30.05'))
Will result in an integer of value = 13
It is invalid to specify HOUR ('13.30.05') since the argument is of type CHAR.

Example
HOUR (TIME ('13.30.05')-TIME ('12.02.01'))
Will result in an integer of value = 1

INTEGER or INT function

The INTEGER function returns an integer representation of a number. The argument can be either a string expression or a numeric expression.

{INTEGER,INT}(numericExpression,stringExpression)

The arguments are defined as follows:

numericExpression

numericExpression is converted to an integer from one of the following numerical data types:
- SMALLINT
- INTEGER
- BIGINT
- FLOAT
- REAL
- DECIMAL
- DECFLOAT
The numeric expression is not rounded before conversion and any fractional part is truncated. If the result exceeds the allowable values for an integer, an overflow condition will occur. The valid range is -2147483648 through 2147483647.

**Note**

If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.

If the argument is a constant, it must adhere to integer value rules. Specifying INT(1.5) causes an error because 1.5 is not an integer.

**stringExpression**

If the argument is a string expression, it can be in any external numerical format. It is first converted to its corresponding format, and then converted to an integer. If the CCSID of the string is not in EBCDIC encoding scheme, CCSID is cast to EBCDIC before conversion takes place.

**LENGTH function**

The LENGTH function returns the length of the argument's value as a large integer.

```sql
LENGTH( expression )
```

The result is the length of the argument. The length does not include the null indicator byte of column arguments that allow null values. The length of strings includes blanks. The length of a varying-length string is the actual length, not the maximum length.

The length of a graphic string including UTF-16 is the number of double-byte characters.

The length of all other values is the number of bytes used to represent the value:

<table>
<thead>
<tr>
<th>Data type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>2</td>
</tr>
<tr>
<td>INTEGER</td>
<td>4</td>
</tr>
<tr>
<td>BIGINT</td>
<td>8</td>
</tr>
<tr>
<td>DECIMAL with precision (p)</td>
<td>The integer part of ( \lceil \frac{p}{2} \rceil +1 )</td>
</tr>
<tr>
<td>DECFLOAT (34)</td>
<td>16</td>
</tr>
<tr>
<td>DECFLOAT (16)</td>
<td>8</td>
</tr>
</tbody>
</table>
### Data type

<table>
<thead>
<tr>
<th>Data type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single precision floating point</td>
<td>4</td>
</tr>
<tr>
<td>Double precision floating point</td>
<td>8</td>
</tr>
<tr>
<td>Strings</td>
<td>The length of the string</td>
</tr>
<tr>
<td>DATE</td>
<td>4</td>
</tr>
<tr>
<td>TIME</td>
<td>3</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>10</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIMEZONE</td>
<td>12</td>
</tr>
<tr>
<td>TIMESTAMP((p))</td>
<td>(7 + ((p +1)/2))</td>
</tr>
<tr>
<td>TIMESTAMP((p)) WITH TIMEZONE</td>
<td>(9 + ((p +1)/2))</td>
</tr>
</tbody>
</table>

### Note

If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.

### LOCATE function

The LOCATE function returns a large integer indicating the position of one string, `searchString`, in another, `sourceString`. The function returns 0 if `searchString` is a zero-length string or is not found, or if `start` is greater than the length of `sourceString`. By default, the search starts at the first character of the receiving string (that is, the value of `start` is 1), and continues to the end of the string.

```
LOCATE( searchString, sourceString, start )
```

The arguments are defined as follows:

- `searchString`
  
  `searchString` is an expression that specifies the string for which you are searching for. `searchString` must return a value that is a built-in single-byte character/varying string data type.

- `sourceString`
  
  `sourceString` must return a value that is a built-in single-byte character/ varying string data type.
**start**

=start is an expression that specifies the position within the searchString where the search is to start. start is expressed in bytes and must return an integer value that is greater than or equal to 1.

---

**Note**

If any argument can be null, the result can be null; if any argument's value is null, the result is the null value.

---

**LOWER or LCASE function**

The LOWER function returns a string in which all characters have been converted to lowercase characters.

\[ \text{LOWER}( \text{stringExpression} ) \]

\[ \text{LOWER}( \text{stringExpression}, \text{localeNameString} ) \]

\[ \text{LOWER}( \text{stringExpression}, \text{UNI}, \text{length} ) \]

---

**Note**

If the argument can be null, the result can be null; if argument's value is null, the result is the null value.

---

**LOWER or LCASE without UNI locale**

The arguments are defined as follows:

**stringExpression**

The stringExpression can be one of the following:

- CHAR SBCS in EBCDIC encoding scheme
- VARCHAR SBCS in EBCDIC encoding scheme

**localeNameString**

localeNameString is a string constant in EBCDIC that specifies a valid locale name. If you do not specify this argument, the utility treats the locale as blank.

The only accepted localeNameString types are:
- Blank—SBCS uppercase characters A-Z are converted to SBCS lowercase characters.

- EN_US—SBCS uppercase characters A-Z are converted to SBCS lowercase characters.

- DE_DE—SBCS Deutsch uppercase characters are converted to Deutsch SBCs lowercase characters.

**LOWER or LCASE with UNI locale**

The arguments are defined as follows:

`stringExpression`

`stringExpression` can be defined in the Unicode encoding scheme, and in one of the following data types:

- CHAR SBCS or mixed
- VARCHAR SBCS or mixed
- GRAPHIC or VARGRAPHIC

**UNI**

Currently the NGT SQL language supports only the UNI locale.

**length**

The `length` argument defines the length of the output expressed in bytes. The result string is padded when there is a varying source string.

--- **Note** ---

- `length` is supported only for the UNI locale.

- The value of `length` cannot exceed 255 for `stringExpression` when defined as a CHAR value, and cannot exceed 32704 for a varying string.

- The `length` argument must be a positive integer constant.

- The value of `length` must be bigger than the length of the source string. If less, the function returns an error.

**MICROSECOND function**

The MICROSECOND function accepts a single argument and returns an integer data type.
MICROSECOND(  expression  )

expression must return one of the following built-in data types:

- TIMESTAMP
- TIMESTAMP WITH TIMEZONE
- TIMESTAMP DURATION

A timestamp duration occurs as the result of subtracting two timestamps of precision=6. The result is the microsecond portion of the derived timestamp or timestamp duration.

**Note**
A CHAR/VARCHAR argument is not accepted unless converted by a TIMESTAMP or TIMESTAMP_TZ function.
If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.

**Example**
MICROSECOND (TIMESTAMP ('1991-12-31-13.30.05.123456'))
Will result in an integer of value = 123456
It is invalid to specify MICROSECOND ('1991-12-31-13.30.05.123456') since the argument is of type CHAR.

**Example**
MICROSECOND (TIMESTAMP ('1991-12-31-13.30.05.123456') - TIMESTAMP ('1991-12-31-13.30.05.123456'))
Will result in an integer of value = 0

**MINUTE function**

The MINUTE function accepts a single argument and returns an integer data type.

MINUTE(expression)

expression must return one of the following built-in data types:

- TIMESTAMP
- **TIMESTAMP WITH TIMEZONE**

- **TIMESTAMP DURATION**
  A timestamp duration occurs as the result of subtracting two timestamps of precision=6. The result is the minute portion of the derived timestamp or timestamp duration.

  **Note**
  A CHAR/VARCHAR argument is not accepted unless converted by a TIMESTAMP or TIMESTAMP_TZ functions.

  If the argument can be null, the result can be null; if the argument’s value is null, the result is the null value.

  **Example**
  \[
  \text{MINUTE (TIME ('13.30.05'))} \\
  \text{Will result in an integer of value = 30} \\
  \text{It is invalid to specify MINUTE ('13.30.05') since the argument is of type CHAR} \\
  \]

  **Example**
  \[
  \text{MINUTE (TIME ('13.30.05')-TIME ('12.02.01'))} \\
  \text{Will result in an integer of value = 28} \\
  \]

**MONTH function**

The MONTH function returns an integer data type.

\[
\text{MONTH( } expression \text{)}
\]

*expression* must return one of the following built-in data types:

- **TIMESTAMP**

- **TIMESTAMP WITH TIMEZONE**

- **TIMESTAMP DURATION**
  A timestamp duration occurs as the result of subtracting two timestamps of precision=6. The result is the month portion of the derived timestamp or timestamp duration.
**Note**

A CHAR/VARCHAR argument is not accepted unless converted by a TIMESTAMP or TIMESTAMP_TZ functions. If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.

**Example**

MONTH (DATE ('1991-12-31'))

Will result in an integer of value = 12

It is invalid to specify MONTH (‘1991-12-31’) since the argument is of type CHAR.

**Example**

MONTH (DATE ('1991-12-31') - DATE ('1995-12-31'))

Will result in an integer of value = 0

---

**REAL function**

The REAL function returns a short floating point representation of a long floating-point number.

```
REAL( numericExpression, stringExpression )
```

The arguments are defined as follows:

**numericExpression**

*numericExpression* must be of type float. Other data types are not allowed but can be converted to float using the FLOAT() function.

**stringExpression**

*stringExpression* can be in any external numerical format. It is first converted to its corresponding format, and then converted to a short floating point. If the CCSID is not in EBCDIC encoding scheme, CCSID is cast to EBCDIC CCSID(37) before conversion takes place.
REPEAT function

The REPEAT function returns a varying length character string that is composed of an argument concatenated to itself the specified number of times. The maximum length of the result is 32000 bytes.

\[ \text{REPEAT}( \text{stringExpression}, \text{integer} ) \]

The arguments are defined as follows:

\textit{stringExpression} 

\textit{stringExpression} is an expression that specifies the string to be repeated. The expression must return a value that is a built-in single-byte character/varying string. The argument cannot be a string constant.

\textit{integer} 

\textit{integer} must be a positive large integer value that specifies the number of times to repeat the string.

\textbf{Note} 

If any argument can be null, the result can be null; if any argument's value is null, the result is the null value.

SECOND function

The SECOND function accepts a single argument and returns an integer data type.

\[ \text{SECOND}( \text{expression} ) \]

The \textit{expression} must return one of the following built-in data types:

- \textbf{TIME}
- \textbf{TIME DURATION}
  - A time duration is the result of subtracting two times.
- \textbf{TIMESTAMP}
- \textbf{TIMESTAMP WITH TIMEZONE}
- **TIMESTAMP DURATION**
  
  A timestamp duration occurs as the result of subtracting two timestamps of precision=6. The result is the second portion of the derived time or time/timestamp duration.

  **Note**
  
  A CHAR/VARCHAR argument is not accepted unless converted by a TIME, TIMESTAMP, or TIMESTAMP_TZ function.

  If the argument can be null, the result can be null; if the argument’s value is null, the result is the null value.

  **Example**
  
  \[
  \text{SECOND (TIME ('13.30.05'))}
  \]

  Will result in an integer of value = 05

  It is invalid to specify SECOND ('13.30.05') since the argument is of type CHAR.

  **Example**
  
  \[
  \text{SECOND (TIME ('13.30.05')-TIME ('12.02.01'))}
  \]

  Will result in an integer of value = 4

---

**SMALLINT function**

The SMALLINT function returns a small integer data type.

\[
\text{SMALLINT( numericExpression, stringExpression )}
\]

The arguments are defined as follows:

- **numericExpression**

  *numericExpression* must be of data type INTEGER. Other data types are not allowed but can be converted to an integer using the INT() function.

  If the result of the conversion exceeds the allowable values for a SMALLINT, an overflow condition occurs. The valid range is -32768 through 32767.

  **Note**

  If the argument is a constant, then it must adhere to integer value rules. Specifying SMALLINT(1.5) causes an error because 1.5 is not an integer.
**stringExpression**

A string expression can be in any external numerical format. It is first converted to its corresponding format, then converted to a SMALLINT. If the CCSID of the string is not in EBCDIC encoding scheme, it is cast to EBCDIC CCSID(37) before conversion takes place.

**Note**

If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.

**SOUNDEX function**

The SOUNDEX function returns a four-character code that represents the sound of the single argument. The utility can compare the result to the results of the SOUNDEX function of other strings.

```
SOUNDEX( expression )
```

expression must return a character, or a varying character. The string is cast to Unicode SBCS CCSID 367 before generating the SOUNDEX code.

**Note**

You cannot specify expression as a constant string, but you can specify it as CHAR(expression,n).

The data type of the result is CHAR (4) in Unicode SBCS CCSID.

The SOUNDEX function is useful for finding strings for which the sound is known but the precise spelling is not. The function's assumptions about the way letters and combinations of letters sound can help you search for words with similar sounds. The SOUNDEX function produces the same code as the DB2 SOUNDEX function. The Special Census method generates the code.

**Example**

```sql
SELECT SOUNDEX(CHAR ('ashcraft', 8)) FROM SYSIBM.SYSDUMMY1
```

Results in 'A226'
**STRIP function**

The STRIP function removes blanks or other characters from the end, the beginning, or both ends of a string expression. The data type of the result is a varying length character string.

```
STRIP( stringExpression, direction, trimCharacter )
```

The length attribute of the result is the same as the length attribute of the string expression. The actual length of the result is the length of the string expression minus the number of characters removed. If all of the characters are removed, the result is an empty string.

The arguments are defined as follows:

- **stringExpression**
  
  `stringExpression` must be of type CHAR or VARCHAR.

- **direction**
  
  `direction` must be one of the following values:
  - B or Both (the default value)
  - T or Trailing
  - L or Leading

- **trimCharacter**

  `trimCharacter` is a single SBCS character (by default a blank). The utility converts the character to the CCSID of the table before the trim operations start.

  **Note**
  
  If the argument can be null, the result can be null; if argument's value is null, the result is the null value.

**SUBSTR function**

SUBSTR function returns a portion of a string of the same data type as `stringExpression`. 
The arguments are defined as follows:

\( \text{stringExpression} \)

\( \text{stringExpression} \) must be one of the following data types:
- CHAR
- VARCHAR
- GRAPHIC
- VARGRAPHIC

\( \text{startIntegerExpression} \)

\( \text{startIntegerExpression} \) specifies which position in \( \text{stringExpression} \) use as the first character of the substring. The value of the \( \text{startIntegerExpression} \) must be between 1 and the length attribute of \( \text{stringExpression} \). (The length attribute of a varying-length string is its maximum length.) A value of 1 indicates that the first character of the substring is the first character of \( \text{stringExpression} \).

\( \text{lengthIntegerExpression} \)

Specifying \( \text{lengthIntegerExpression} \) is optional. If you include it, the utility pads the substring on the right with the necessary number of characters to satisfy the length criteria. The utility uses the pad character for the current table.

The varying attribute of the result substring depends on \( \text{lengthIntegerExpression} \). If you omit \( \text{lengthIntegerExpression} \), the varying attribute of the substring is the same as \( \text{stringExpression} \), and the length of the result substring is the same as the source string.

If \( \text{lengthIntegerExpression} \) is a constant with a value of 255 or less, the resulting substring is a fixed character or graphic string of the specified length. If \( \text{lengthIntegerExpression} \) is not a constant the resulting substring is a varying character or graphic string with the same maximum length as \( \text{stringExpression} \).
Note
If any of the arguments can be null, the result can be null; if any argument's value is null, the result is the null value.

TIME function

The TIME function returns an internal representation of a time data type.

\[ \text{TIME}(\text{expression}) \]

expression must return one of the following built-in data types:

- TIME
  The result is the specified time.
- TIMESTAMP
  The result is the time part of the timestamp.
- TIMESTAMP WITH TIMEZONE
  expression is cast to TIMESTAMP WITHOUT TIME ZONE, and then the time portion is extracted.
- CHAR
  The string must be a valid string representation of a date or timestamp with or without time zone. If the encoding of the character string is not EBCDIC, it will be cast to an EBCDIC CCSID(37) before conversion takes place.

Note
If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.

TIMESTAMP function

The TIMESTAMP function returns a TIMESTAMP WITHOUT TIME ZONE value from its argument or arguments.
The rules for the arguments depend on whether you specify the second argument.

**If only one argument is specified without a precision of n**

The argument must be an expression that returns a value of one of the following built-in data types:

- DATE
- TIMESTAMP
- Character string representation of timestamp

*Note*
A time zone in a string representation of a timestamp results in an invalid format.
A character string with an actual length of 8 is assumed to be a TOD clock value.

A character string with an actual length of 14 represents a valid date and time in the form `yyyyxxddhhmmss`, where `yyyy` is the year, `xx` is the month, `dd` is the day, `hh` is the hour, `mm` is the minute, and `ss` is the seconds.

The resulting timestamp has a precision of 6.

**If only one argument is specified with a precision of n**

The argument must be an expression that returns a value of one of the following built-in data types:

- Date
- Timestamp
- Character string representation of timestamp

The resulting timestamp has a precision of `n`, where `n` is between 0 and 12 inclusive.

*Note*
A time zone in a string representation of a timestamp results in an invalid format.
If both arguments are specified

The first argument must be an expression that returns a value of one of the following built-in data types:

- Date
- Timestamp
- Character string representation of timestamp

The second argument must be an expression that returns a value of one of the following built-in data types:

- Time
- Character string representation of time

**Note**

If any of the arguments can be null, the result can be null; if any argument's value is null, the result is the null value.

**TIMESTAMP_TZ function**

The TIMESTAMP_TZ function returns a TIMESTAMP WITH TIME ZONE value from the input arguments.

```
TIMESTAMP_TZ( expression1, expression2 )
```

The arguments are defined as follows:

- **expression1**

  `expression1` is an expression that returns a value of one of the following built-in data types:
  - TIMESTAMP WITHOUT TIMEZONE
  - TIMESTAMP WITH TIMEZONE
  - Character string representation of timestamp with/out time zone
expression2

If you specify expression2, expression1 must be a timestamp without a time zone, or a string representation of a timestamp without a time zone.

expression2 returns a character string that represents a time zone in the format of ±th:tm with values ranging from -12:59 to +14:00. \( th \) represents the time zone hour, and \( tm \) represents the time zone minute.

You can specify expression2 as a constant integer in the range of 0 through 12. This value determines the precision of the result.

**Note**
If any argument can be null, the result can be null; if any argument's value is null, the result is the null value.

---

**TRANSLATE function**

The TRANSLATE function returns a fixed or varying character string that has a value in which one or more characters of the first argument might have been converted to other characters.

\[
\text{TRANSLATE}( \text{stringExpression}, \text{toString}, \text{fromString}, \text{pad} \cdot )
\]

The arguments are defined as follows:

**stringExpression**

stringExpression is an expression that specifies the string to be converted. stringExpression must return a value that is a built-in single-byte character string data type. stringExpression cannot be a string constant.

**toString**

toString is a fixed-length character constant that is assumed to be in the same encoding scheme as the stringExpression. The translate function might not work correctly if the encoding schemes are not the same. The maximum length of toString is 256 characters.

If you specify toString you must also specify fromString. If the length of toString is less than the length of fromString, toString is padded to the length of fromString with the pad or a blank. If the length of toString is greater than fromString, the extra characters in toString are ignored without warning.
fromString

*ftFromString* is a fixed-length character constant that is assumed to be in the same encoding scheme as the *stringExpression*. The translate function may not work correctly if the encoding schemes are not the same. Alternatively, you can specify the character constant as a hexadecimal constant. The maximum length of *fromString* is 256 characters.

pad

*pad* character is optional and defaults to an EBCDIC blank. You can specify *pad* as a character string constant or hexadecimal constant with a length not exceeding one character.

*fromString* is considered the input translate table, *tablei*. *toString* is the output translate table, *tableo*.

**Note**

If only *string expression* is specified, the translate function is treated as an UPPER function.

If the argument can be null, the result can be null; if argument's value is null, the result is the null value.

TRANSLATE searches *tablei* for each character in *stringExpression*. If it finds the character, the corresponding character in *tableo* is used in the result string; if *tablei* contains duplicates, TRANSLATE uses the first (leftmost) occurrence. If it does not find the character, then TRANSLATE uses the original character in *string*. The result string is always the same length as *string*.

**UPPER or UCASE function**

The UPPER or UCASE function returns a string in which all the characters have been converted to uppercase characters.

\[
\text{UPPER|UCASE}( \text{stringExpression}, \text{localeNameString} )
\]

\[
\text{UPPER|UCASE}( \text{stringExpression}, \text{UNI}, \text{length} )
\]

**Note**

If the argument can be null, the result can be null; if argument's value is null, the result is the null value.
UPPER or UCASE without UNI locale

The arguments are defined as follows:

\textit{stringExpression}

\textit{stringExpression} can be one of the following:

\begin{itemize}
  \item CHAR SBCS in EBCDIC encoding scheme
  \item VARCHAR SBCS in EBCDIC encoding scheme
\end{itemize}

\textit{localeNameString}

\textit{LocaleNameString} is a string constant in EBCDIC that specifies a valid locale name. You do not specify the \textit{localeNameString} argument, the utility treats the locale as blank.

The only accepted \textit{localeNameString} types are:

\begin{itemize}
  \item Blank—SBCS lowercase characters A-Z are converted to SBCS uppercase characters.
  \item EN_US—SBCS lowercase characters A-Z are converted to SBCS uppercase characters.
  \item DE_DE—SBCS \textit{Deutsch} lowercase characters are converted to \textit{Deutsch} SBCS uppercase characters.
\end{itemize}

UPPER or UCASE with UNI locale

The arguments are defined as follows:

\textit{stringExpression}

\textit{stringExpression} can be defined in the Unicode encoding scheme, and in one of the following data types:

\begin{itemize}
  \item CHAR SBCS or mixed
  \item VARCHAR SBCS or mixed
  \item GRAPHIC or VARGRAPHIC
\end{itemize}

\textit{UNI}

Currently the NGT SQL language supports only the UNI locale.

\textit{length}

The \textit{length} argument defines the length of the output expressed in bytes. The result string is padded when there is a varying source string.
Note

- `length` is supported only for the UNI locale.
- The value of `length` cannot exceed 255 for `stringExpression` when defined as a CHAR value, and cannot exceed 32704 for a varying string.
- The `length` argument must be a positive integer constant.
- The value of `length` must be bigger than the length of the source string. If less, the function returns an error.

VALUE/COALESCE/IF NULL function

The VALUE function argument list must have a homogeneous data type. That is, if the first argument is SMALLINT, the rest of the arguments must be of type SMALLINT.

A minimum of two expressions and max of 255 expressions can be specified. The first argument determines the data type of the result and therefore cannot be a constant.

VALUE( `expression1`, `expression2`, ...`expression255` )

COALESCE( `expression1`, `expression2`, ...`expression255` )

IFNULL( `expression1`, `expression2` )

Note

The function returns a non-null value of the first non-null expression and of the same data type.

The result can be null only if all arguments can be null. The result is null only if all arguments are null.

The argument's data types must be the same. If one is SMALLINT, all other arguments must be SMALLINT. If one is a decimal then all must be decimal but can have different precisions and scale; the largest precision and scale will be selected as the final result.
VARCHAR function

The VARCHAR function converts a fixed or varying length character string into a variable-length character string.

$$\text{VARCHAR}(\text{stringExpression}, \text{length})$$

The arguments are defined as follows:

stringExpression

stringExpression is a fixed or variable-length character string.

If stringExpression length exceeds its maximum length, the utility generates an error message and terminates the unit of work (UOW).

length

length is an integer constant of the maximum length of the result. If the value of length is shorter than the length of stringExpression, then the function truncates the result to the specified length, without warning.

Note

If the argument can be null, the result can be null; if argument's value is null, the result is the null value.

WEEK function

The WEEK function returns an integer in a range of 1 through 54 that represents the week of the year. The week starts with Sunday, and January 1 is always in the first week.

$$\text{WEEK}(\text{expression})$$

expression must return one of the following built-in data types:

- DATE
- TIMESTAMP
- TIMESTAMP WITH TIMEZONE
- Valid string representation of a date or timestamp with or without a time zone
Note

You cannot specify expression as a constant.

If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.

Example

```
UNLOAD FORMAT DSNTIAUL
SELECT LEAGUE_NUM, TEAM, WEEK(BOWL_DATES)
FROM BOWL2005.SUMMER_LEAGUES
REORG TABLESPACE BOWL2005.LEAGUES2
DISCARD FROM TABLE BOWL2005.SUMMER_LEAGUES
WHERE (WEEK(BOWL_DATES) BETWEEN 30 AND 45
```

YEAR function

The YEAR function returns an integer data type.

```
YEAR( expression )
```

expression must return one of the following built-in data types:

- DATE
- DATE DURATION
  A date duration occurs as the result of subtracting two dates.
- TIMESTAMP
- TIMESTAMP WITH TIMEZONE
- TIMESTAMP DURATION
  A timestamp duration occurs as the result of subtracting two timestamps with precision=6. The result is the year portion of the derived date or date/timestamp duration.

Note

A CHAR/VARCHAR argument is not accepted unless converted by a DATE, TIMESTAMP, or TIMESTAMP_TZ function.

If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.
**Example**

```
YEAR (DATE ('1991-12-31'))
Will result in an integer of value = 1991
```

It is invalid to specify `YEAR ('1991-12-31')` since the argument is of type `CHAR`.

**Example**

```
YEAR (DATE ('1991-12-31')-DATE ('1995-12-31'))
Will result in an integer of value = -4
```

---

**ZONED function**

The ZONED function returns a fixed-length character string that is composed of converting the decimal number to a string of digits without a decimal point, and with a sign byte at the right of the string depicted in a zoned format.

```
ZONED( decimalExpression )
```

The length of the result is $p$, where $p$ is the precision of the `decimalExpression`.

The ZONED function produces similar results as the DIGITS function but with special treatment at the rightmost byte, which is in the form $sn$, where:

- $s$ is the sign of the result expressed in a hexadecimal constant of X'D' if the result is negative, or X'F' if positive
- $n$ is the rightmost digit of the decimal expression

**Example**

Decimal number -123.4 results in hexadecimal 'F1F2F3D' or 123M.
+123.4 results in hexadecimal 'F1F2F3F4' or 1234

**Note**

The result is not translated to the current table's encoding scheme.
If the argument can be null, the result can be null; if the argument's value is null, the result is the null value.
@S370 function

The @S370 function encodes the numeric data represented by the character expression argument into IBM DB2 internal format.

```plaintext
@S370(decimalExpression)
```

This function is mainly used in the LOAD utility product to convert numeric data in decoded format to encoded format in order to perform arithmetic and other advanced functions.

@ROWID function

The @ROWID function generates a ROWID value consistent with DB2 ROWID column specifications. The function returns the result as VARCHAR(14) data type SBCS for bit data.

**Figure 10: @ROWID function syntax**

```
@ROWID()
```

---

**Note**

The utility might use the FLOAT function to convert the @ROWID function to a real number between 0 and 1 for use as a random number generator.

Date and time durations

NGT SQL language supports the following types of durations: labeled, date, time, and timestamp.

Labeled durations

NGT SQL labeled durations are large integers. In contrast, IBM DB2 internally treats labeled durations as DECIMAL(15,0) numbers.

In NGT SQL, applying a labeled duration to a numerical expression whose type is not a large integer causes a syntax error. To correct the error, you can apply the INTEGER() function on the expression, before the labeled duration.
**Example**

Assume that column 'DEC15' is a DECIMAL(15,0). The following expression causes a syntax error because you can apply the labeled duration (DAYS) only to a large integer:

```
CURRENT_DATE = DATE('1995-01-01') + DEC15 DAYS
```

Adding the INTEGER function corrects the error:

```
CURRENT_DATE = DATE('1995-01-01') + INTEGER(DEC15) DAYS
```

---

**Date durations**

NGT utility products can treat any DECIMAL(8,0) or DECIMAL(9,0) field as a date duration. Date durations can be negative or positive, and you can use them to decrement or increment DATES, respectively.

**Example**

If \( x \) is the date duration, \( \text{DATE} + x \) is equivalent to the following expression:

\[
\text{DATE} + \text{YEAR}(x) \text{YEARS} + \text{MONTH}(x) \text{MONTHS} + \text{DAY}(x) \text{DAYS}
\]

**Note**

DB2 supports only DEC(8,0). Also, negative dates are not well defined in DB2.

---

**Time durations**

NGT utility products can treat any DECIMAL(6,0) or DECIMAL(7,0) field as a TIME duration.

**Note**

DB2 supports only DEC(6,0).

---

**Timestamp durations**

NGT utility products can treat any DECIMAL(20,6) or DECIMAL(21,6) field as a TIMESTAMP duration.
Comparisons

NGT SQL language supports the following comparisons: numeric, decimal, numeric string, and constant.

Numeric comparisons

When you compare two numerical expressions of different types, NGT SQL converts one type to the type of the other and then compares them.

The conversion rules are the same as outlined in SQL reference with the exceptions of FLOAT and DECIMAL comparisons. IBM DB2 converts the decimal to FLOAT and then carries out the comparison in FLOAT format. This technique produces correct results if the decimal precision is greater than 18.

**Example**

If the following two numbers are compared under DB2, the result will be EQUAL; which is incorrect

DEC310=1234567890123456789012345678901
FLOAT=.12345678901234568 E +31

To correct this example, NGT SQL converts the float to a decimal number, and then compares them in decimal format. This technique might produce undesirable results when a floating-point fraction cannot be represented exactly in decimal format. In this case, BMC recommends using either the FLOAT() or the DECIMAL() function to control this process.

Decimal comparisons

When comparing two decimal numbers that have different precision and scale, NGT SQL converts the two numbers to a uniform precision and scale, which are derived from the following formulas:

- \[ P = \min(31, \max(S_1, S_2) + \max(P_1 - S_1, P_2 - S_2)) \]
- \[ S = \max(S_1, S_2) \]
Example

Assume that number 1 is DEC(31,31) and has a value of .31; number 2 is DEC(31,0) and has a value of 31. The uniform precision is:

\[
P = \min(31, \max(31,0) + \max(0,31)) = \min(31,62) = 31
\]

The uniform scale is:

\[
S = \max(31,0) = 31
\]

Therefore, number 2 needs to be converted to DEC(31,31). However, this conversion will result in an overflow.

Numeric string comparisons

DB2 requires numeric constants to be free of quotation marks and does not accept hexadecimal constants in numeric comparisons. NGT, however, is more flexible and does allow these specifications.

Example

If INT1 is a numeric column, NGT allows one of the following constants:

- INT1=1
- INT1='1'
- INT1=X'F1'

Constant comparison

NGT SQL does not support constant comparison, and results in a syntax errors.

For example, the following comparisons are not permitted:

- 1=1
- 'A'='A'

NGT SQL language keywords

NGT SQL language supports the following keywords: NOT, IN, and LIKE.
**NOT prefix operator**

NGT SQL does not support the NOT prefix operator but does support:

- IS NOT NULL
- NOT LIKE
- NOT IN
- The relational comparison not equal (^=)

**Reserved words**

The NGT utilities do not use reserved words or keywords.

**IN predicate**

In the NGT utility products, the IN predicate supports the standard DB2 subselect.

To process the subselect, the NGT utility products complete this process:

1. Extract the subselect statement.
2. Pass the SQL to DB2 for processing.
3. Collect the output.
4. Filter the output from DB2 to exclude null and duplicate rows.
5. Place the output in an IBM MVS/ESA data space.
6. Search the output by using a high-speed search engine.

The data type of the IN predicate operands must match exactly:

- The left operand of the IN predicate data type must match the right operand, which is the result that DB2 returned.
- The precision and scale of one operand must match the precision and scale of the other.
If the precision and scale are not the same, you can change them using the INTEGER, FLOAT, or DECIMAL DB2 built-in function. To improve performance when conversion is required, BMC recommends that you convert the right hand operand to the type and length of the left hand operand.

For example, assume that the WHERE clause is specified as follows:

```
WHERE C1 IN (SELECT C2 FROM TB1)
```

The following table shows the possible data type combinations of C1 and C2, and the action required.

**Table 6: IN predicate data type conversion actions**

<table>
<thead>
<tr>
<th>Data type C1</th>
<th>Data type C2</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>INTEGER</td>
<td>INTEGER(C1)</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>DECIMAL(P,S)</td>
<td>DECIMAL(C1,P,S)</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>REAL</td>
<td>(C1) IN (SELECT FLOAT(B) FROM TB1)</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>FLOAT</td>
<td>FLOAT(C1)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>SMALLINT</td>
<td>(SELECT INTEGER(C2) FROM TB1)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>DECIMAL(P,S)</td>
<td>DECIMAL(C1,P,S)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>REAL</td>
<td>FLOAT(C1) IN (SELECT FLOAT(B) FROM TB1)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>FLOAT</td>
<td>FLOAT(C1)</td>
</tr>
<tr>
<td>REAL</td>
<td>SMALLINT</td>
<td>FLOAT(C1) IN (SELECT FLOAT(B) FROM TB1)</td>
</tr>
<tr>
<td>REAL</td>
<td>INTEGER</td>
<td>FLOAT(C1) IN (SELECT FLOAT(B) FROM TB1)</td>
</tr>
<tr>
<td>REAL</td>
<td>DECIMAL</td>
<td>DECIMAL(C1,P,S)</td>
</tr>
<tr>
<td>REAL</td>
<td>FLOAT</td>
<td>FLOAT(C1)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>SMALLINT</td>
<td>(SELECT FLOAT(C2) FROM TB1)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>INTEGER</td>
<td>(SELECT FLOAT(C2) FROM TB1)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>DECIMAL</td>
<td>DECIMAL(C1,P,S)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>REAL</td>
<td>(SELECT FLOAT(C2) FROM TB1)</td>
</tr>
<tr>
<td>DECIMAL(P,S)</td>
<td>SMALLINT</td>
<td>(SELECT DECIMAL(C2,P,S) FROM TB1)</td>
</tr>
<tr>
<td>DECIMAL(P,S)</td>
<td>INTEGER</td>
<td>(SELECT DECIMAL(C2,P,S) FROM TB1)</td>
</tr>
<tr>
<td>DECIMAL(P,S)</td>
<td>REAL</td>
<td>(SELECT DECIMAL(C2,P,S) FROM TB1)</td>
</tr>
<tr>
<td>DECIMAL(P,S)</td>
<td>FLOAT</td>
<td>(SELECT DECIMAL(C2,P,S) FROM TB1)</td>
</tr>
<tr>
<td>Data type C1</td>
<td>Data type C2</td>
<td>Action</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DECIMAL(P1,S1)</td>
<td>DECIMAL</td>
<td>DECIMAL(C1,P,S) IN (SELECT DECIMAL(C2,P,S) FROM TB1 Where: p= min(31,max*s1,s2)+maxp1-s1,p2-s2) s=max(s1,s2)</td>
</tr>
</tbody>
</table>

To further accelerate the search process, or if the data does not reside in the DB2 table, use the external file technique. This technique utilizes non DB2 syntax and requires an input file that has been generated by either the NGT Unload utility or the DB2 REORG UNLOAD utility.

The full syntax of the feature is defined as follows:

```
WHERE columnName IN (FILE(ddname) FORMAT(format))
```

1. **columnName** is the left operand of the IN predicate. This operand must be a column name and cannot be a constant. The attributes of this column dictate the data type, length, and null attribute of the first field of the external file.

2. **ddname** is the name of the DD statement in the current job step that references the external data file.

3. **format** is the internal format of the data in the external file. Specify FORMAT(UNLOAD). For more information about unload formats, see the *BMC Next Generation Technology Unload for DB2 for z/OS Reference Manual*.

Processing an IN predicate by using an external file causes NGT utility products to complete these actions:

1. Read the specified DDNAME (which can be on DISK or TAPE).

2. Filter out duplicate and null rows.

3. Place the data in an MVS/ESA data space.

4. Searched the data via a high-speed search engine.

Processing of the IN predicate having an external file causes NGT utility products to read the specified DDNAME (which can be on DISK or TAPE), filter out duplicate and null rows, and place the data in an MVS/ESA data space. It is searched via a high speed search engine.

The IN predicate external file can contain several data fields or columns. However, only the first data field is extracted from the file. The remaining data fields are ignored. The first data field is recognized by using the same attributes of the left operand and can be the concatenation of columns.
Example

Assume that the following conditions exist:

- File(DD1) contains the following records in hexadecimal format:
  C1C2C3C4C5

- The data type for C1 is VARCHAR.

Executing the following expression returns incorrect output because the record in the file does not look like a VARCHAR:

```sql
WHERE C1 IN(FILE(DD1) FORMAT(ARCHIVE))
```

In one WHERE clause, you can have up to 200 IN predicates with different files, subselects, or both. You can use Boolean logic to form very complex pattern matching on large amounts of data at very high speeds. However, WHERE clause syntax errors might occur if memory space is exceeded; in that case attempt to reduce the select data or add more paging data sets to your paging subsystem.

Note

If you are unloading to a file to be referenced by the NGT Reorg WHERE … IN FILE(ddname), specify the same +DIGITS parameter on both the Unload job and the Reorg job.

LIKE predicate

The LIKE predicate supports only character data (no graphics).

It also supports patterns constructed dynamically, such as:

```sql
A1 LIKE SUBSTR(A2,1,1)||'%'||'CDE'
```

A1 and A2 represent character columns.

Restrictions using NULL/NOT NULL predicate

If you are using the NULL/NOT NULL predicate, you must ensure that the underlying column has the NULL attribute defined.
Example

The following expression causes a syntax error if column 'COL1' was defined as NOT NULL:

\texttt{COL1 IS NOT NULL}

SQL statements that are ordinarily coded using the IBM MOD function must instead use the operator \texttt{//} (double forward slash) as a substitute. For example instead of entering WHERE clause 

\texttt{MOD(DAYS(CURRENT DATE), 2) = 1}, enter \texttt{WHERE DAYS(CURRENT DATE) // 2 = 1}.

**Diagnosing the WHERE clause**

BMC provides a diagnostic trace function in the WHERE clause that can be used to resolve problems related to WHERE processing.

---

**WARNING**

Only use the trace function when directed to do so by BMC Customer Support. The TRACE keyword, if used, must be coded immediately after the WHERE keyword and before any predicate.

---

**Figure 11: TRACE function syntax**

\texttt{TRACE( arg1, arg2, arg3 )}

The arguments are defined as follows:

\textit{arg1}

The first argument (\textit{arg1}) is required and specifies the level of tracing requested. You can specify one of the following values for \textit{arg1}:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Generates detailed tracing showing the value of each operand in each executed predicate. This level generates one line of output per operand and could produce many lines of output. Use this trace level within a specified range of rows. See examples below.</td>
</tr>
<tr>
<td>R</td>
<td>Generates one line showing the row number and the result of the WHERE clause, ( TRUE or FALSE ).</td>
</tr>
<tr>
<td>T</td>
<td>Generates one line showing the number of the row when the result is TRUE.</td>
</tr>
<tr>
<td>F</td>
<td>Generates one line showing the number of the row when the result is FALSE.</td>
</tr>
</tbody>
</table>
arg2 and arg3

The second and third arguments (arg2 and arg3), specify the range of rows subject to tracing. These arguments must be numeric values separated by a colon (:).

arg2 is the starting row number and arg3 the ending row number. If neither value is specified, all rows are traced. If only arg2 is specified, then the default for arg3 is $2^{31}-1$.

Example
The following expression traces and prints row numbers between 1 and 1000 whose result is FALSE.

WHERE TRACE(F,1:1000) COL1=COL2 AND COL3=COL4

The following expression prints the entire WHERE logic for row number 17.

WHERE TRACE(I,17:17) COL1=COL2 AND COL3=COL4
This chapter describes the changes required for the EXEC card and DD names that are used by NGT utility products. Certain utilities might use additional DD names, which are described in that product's documentation.

**Tip**

If you are using JCL procedures (PROCS) to run your utilities, you might require only the following changes to implement NGT utility products:

- Change the program name
- STEBLIB changes in PROCS

### JCL overview

The following figure shows sample JCL for running NGT utility products.

Numbered lines are described in detail in Table 7 on page 104. The JCL is grouped to show related elements; the order of the DD statements is not significant.

**Figure 12: Basic JCL for NGT utility products**

```plaintext
//jobname JOB REGION=4M (1)
//*
//***************************************************************
//* REQUIRED STATEMENTS - MINIMUM FOR RUNNING A UTILITY
//***************************************************************
//UTILSTEP EXEC PGM=NGTUTIL,PARM='DB2B,MYREORG,RESTART' (2)
//STEPLIB DD DSN=NGT.UTILITY,DISP=SHR (3)
//SYSPRINT DD SYSOUT=* (4)
//SYSIN DD * (5)
utility-control-statements
utility-control-statements
utility-control-statements
//***************************************************************
//* GLOBAL PARAMETERS - USE TO OVERRIDE INSTALLATION OPTIONS
//***************************************************************
//UTLPARMS DD DSN=NGT.CNTL(PARMS),DISP=SHR (6)
//*
//***************************************************************
//* SPECIFIC UTILITY PARAMETERS - USE TO OVERRIDE INSTALLATION OPTIONS
//***************************************************************
//xxxPARMS DD DSN=NGT.CNTL(PARMSxxx),DISP=SHR (7)
```
Table 7: JCL descriptions

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1 | JOB name | *(optional)* Job statement name according to your installation standards. Include the REGION parameter on either your JOB statement or your EXEC statement to specify the region size (the amount of virtual storage used by the utility). For the best performance, BMC recommends that you specify REGION=0M to allocate the optimal amount available virtual storage to the utility job. If your data center does not permit you to specify REGION=0M, specify the amount that allows the most virtual storage both above and below the 16-megabyte line. Using a region size that is less than optimal risks the following potential issues:  
  - Running less efficiently, which could result in additional CPU and elapsed time  
  - Encountering memory failures or jobs that fail when new versions implement |
<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2  | EXEC statement  | Step name according to your installation's standards
You specify the NGT utility products positional parameters required to run utilities via the PARM= clause. You can use the following positional parameters: |
|    |                 | ■ *(required)*SSID
The target DB2 subsystem you wish to run against. |
|    |                 | ■ UID
This identifier is logged in the NGT utility products NGTCKPT control data sets and is used with other information as part of an enqueueing mechanism across your system with other NGT utility products jobs. The default UID is the jobname specified on the JOB card (not userid.jobname, which is the DB2 default). The example in Figure 12 on page 103 uses the default jobname. |
|    |                 | ■ *(optional)* Restart parameter
You can omit a restart parameter (in which case the default value is RESTART), or use one of following restart parameters:
— RESTART
— RESTART(PHASE)
— QUICKEXIT
— NORESTART
— TESTUID
— FORCEID
— MAINT

**RESTART**

NGT utility products to restart if a prior run with the same UID failed, or treat this run as if it had never started (that is, start at the beginning).

**RESTART(PHASE)**

This is equivalent to RESTART.

You can specify this parameter to maintain compatibility with the DB2 RESTART(PHASE) parameter.

**Example:** The input control stream is a set of 500 COPY statements and the run terminated after 100 had finished. If you specified RESTART, the utility restarts at COPY 101 and continues.
EXEC statement (continued) | QUICKEXIT

If a prior run with the same UID failed, QUICKEXIT completes the failed utility in the most efficient manner.

QUICKEXIT backs out to a point of consistency, or restarts and completes the failed utility, then terminates. If the prior utility did not fail, it terminates. For online utilities, you can use this option to clean up work data sets, and free up the UID for use by another job. For offline utilities, you can use this option to get your table spaces back online as quickly as possible.

The return code (rc variable) from a job with a restart parameter value of QUICKEXIT can take the following values:

- rc=0, if all objects completed
- rc=4, if any object was not processed
- rc=8, if QUICKEXIT failed, or the UID was not found
- rc=16, if QUICKEXIT failed while in the copy phase of reorganization

Example: You run the following statement: REORG TABLESPACE DB1.%, and DB1 contains table spaces TS1 and TS2. The return code will vary according to the QUICKEXIT run scenario:

- The reorganization fails while processing TS1, and TS2 had not yet started, and +CLEANUP(NO) was specified, then the UID is restartable
  If you submit QUICKEXIT, and it runs successfully (removing TS1 from the MATRIX/UTIL/SYNC, cleaning up work data sets and removing the UID), the job ends with rc=4. The return code is 4 is because the reorganization for TS1 and TS2 did not complete. Table spaces T1 and T2 are considered to be excluded.

- The reorganization completed for TS1, but failed while processing TS2 during the copy phase
  If you submit QUICKEXIT, it must finish the copy, as the reorganization has already completed. Unlike the previous scenario, it cannot simply remove the TS2 from the MATRIX/UTIL/SYNC, and clean up the work data sets.

- If QUICKEXIT completes the copy, it will then remove the object from the MATRIX/UTIL/SYNC, clean up the work data sets, and get rid of the UID, and the job ends with rc=0. The return code is 0 because all the object completed.

- If QUICKEXIT cannot complete the copy, the job ends with rc=16. The return code is 16 because the copy phase is SMC (Step Must Complete).
<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>EXEC statement (continued)</td>
<td>NORESTART</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Any prior utility with the same UID is QUICKEXITed, and the job starts from the beginning. This option will be honored only if it is determined by NGT utility products that no damage will be done to any of your DB2 data by ignoring restart. NORESTART is executed only if NGT utility products can ensure that no damage will occur to your DB2 data by ignoring a restart.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TESTUID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGT utility products attempt to locate an existing utility ID (UID) for the named DB2 subsystem (in SSID). If the UID does exist, NGTUTIL exits with RC=1. If the UID does not exist, NGTUTIL exits with RC=0. The process of verifying the UID does not execute the utility.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> A TESTUID must be performed using the same DB2 SSID with which the utility was originally executed.</td>
</tr>
<tr>
<td>2</td>
<td>EXEC statement (continued)</td>
<td>FORCEID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cleans up and removes the utility from the NGT Checkpoint, and deletes work data sets that conform to the +WORKPREFIX naming convention. Process members are also deleted from the NGT Checkpoint. FORCEID must be performed using the same DB2 subsystem ID with which the utility was originally executed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> Before using FORCEID, review the NGTDISP output and <strong>do not</strong> use FORCEID for a UID that is in a MUSTCOMPLETE state. FORCEID should be used as a last resort, for example, if QUICKEXIT and manual utility cleanup has already been performed. Do not use the FORCEID value without seeking instructions from BMC Customer Support.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAINT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tells NGT utility products to print a report listing the fixes installed on the target DB2 subsystem. Then the job ends without affecting any utility that is running.</td>
</tr>
<tr>
<td>#</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----</td>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>STEPLIB DD</td>
<td><em>(required)</em> STEPLIB or JOBLIB to access the NGT utility products load library</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This library must be APF-authorized.</td>
</tr>
<tr>
<td>4</td>
<td>SYSPRINT DD</td>
<td><em>(required)</em> SYSPRINT statement to capture utility output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>From SYSPRINT the utility output is transferred to the CDBPRINT DD data set. The SYSPRINT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>data set is cleared and released following each NGT Utility step.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The SYSPRINT data set provides a valuable diagnostic tool in the event of errors.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>However, some NGT utility products employ parallel processing, which can result in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>overlapping outputs in SYSPRINT. The SUMMARY DD statement contains a more</td>
</tr>
<tr>
<td></td>
<td></td>
<td>readable analysis of the utility execution.</td>
</tr>
<tr>
<td>5</td>
<td>SYSIN DD</td>
<td><em>(required)</em> The control statements for the utilities you are executing. These can include</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the RTS keyword, which will enable the NGT Utility Manager.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, see <em>BMC Next Generation Technology Utility Manager for DB2 for z/OS</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, see “SYSIN statements other than NGT utility products statements” on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>page 110.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You can concatenate multiple statements in one SYSIN file, however each statement must</td>
</tr>
<tr>
<td></td>
<td></td>
<td>process a different object. Alternatively, you can specify +SYSINORDER(HONOR) to process</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the objects in a single thread, rather than in parallel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>```</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REORG TABLESPACE NWDB.ARACCTS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PART 12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPY TABLESPACE NWDB.APX1022</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPYDDN(OUTCOPY)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECOVERYDDN(OUTRECV)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RUNSTATS TABLESPACE NWDB.INVENTORY</td>
</tr>
<tr>
<td>6</td>
<td>UTLPARMS DD</td>
<td><em>(optional)</em> Specifies general execution parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For a list of global parameters, see “Global utility alphabetical parameter reference” on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>page 155.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you do not specify UTLPARMS, NGT utility products uses the values that were defined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>during configuration for the specific DB2 subsystem.</td>
</tr>
<tr>
<td>7</td>
<td>Specific utility</td>
<td><em>(optional)</em> Contains parameters for a particular utility</td>
</tr>
<tr>
<td></td>
<td>parameters</td>
<td>There is one DD statement for each utility.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGT utility products first reads the Global Parameters from the //UTLPARMS DD statement.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Parameters from each of the //xxxPARMS DD statements specify options unique to the utility.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you do not specify parameters for a utility, NGT utility products uses the values that</td>
</tr>
<tr>
<td></td>
<td></td>
<td>were defined during configuration for the specific DB2 subsystem. For more information about</td>
</tr>
<tr>
<td></td>
<td></td>
<td>these parameters, refer to the specific NGT utility reference guide.</td>
</tr>
<tr>
<td>8</td>
<td>NGTAUTO DD</td>
<td><em>(optional)</em> Overrides the data set name for the NGTAUTO PDS that was specified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>during configuration for the DB2 subsystem.</td>
</tr>
<tr>
<td>#</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>--------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>9</td>
<td>CDBPRINT DD</td>
<td><em>(optional)</em> NGTPRINT data set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSPRINT output routinely goes to CDBPRINT, unless CDBPRINT is already allocated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All NGT utility products report output are written to CDBPRINT. All other non-NGT utility products report their output to SYSPRINT which is then copied to CDBPRINT. You do not need to supply this DD unless there are specific situations where you do not want to dynamically allocate this file.</td>
</tr>
<tr>
<td>10</td>
<td>SUMMARY DD</td>
<td><em>(optional)</em> SUMMARY statement to capture utility output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compared with SYSPRINT, SUMMARY contains a more organized analysis of the utility execution. It includes reports printed at the conclusion of a utility run, and does not reflect output overlapped from parallel processing. If this DD statement is specified, summary reports are not printed to SYSPRINT. Instead, they are printed to SUMMARY as each utility statement completes. If this DD statement is omitted, the summary reports are printed to SYSPRINT.</td>
</tr>
<tr>
<td>11</td>
<td>RUNSTATS DD</td>
<td><em>(optional)</em> RUNSTATS report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RUNSTATS contains a report showing RUNSTATS values, and messages affirming update of the DB2 Catalog with these values. This report is printed for any utility that also updates Catalog RUNSTATS values, such as REORG and LOAD.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If this DD statement is omitted, there is no RUNSTATS report on any statement (including SYSPRINT and SUMMARY).</td>
</tr>
<tr>
<td>12</td>
<td>SYSTSPRT DD</td>
<td><em>(optional)</em> SYSTSPRT dataset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Any messages sent to TSO are also sent to SYSTSPRT. This may include output from automation control points. REXX Displays from automation are only sent back to the job if the Journal level output is requested. With Journal level output these displays are sent to CDBPRINT unless this SYSTSPRT DD is included, then they are sent to SYSTSPRT.</td>
</tr>
<tr>
<td>13</td>
<td>SYSERROR DD</td>
<td><em>(optional)</em> error messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If this DD statement is not specified, error messages appear only in SYSPRINT. However, if SYSERROR is specified, error messages appear in SYSERROR and SYSPRINT.</td>
</tr>
<tr>
<td>14</td>
<td>LOGPDS DD</td>
<td>Overrides the data set name for the Log PDS that was specified during configuration for this DB2 subsystem</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional. The Log PDS contains diagnostic information in case of NGT utility products abends.</td>
</tr>
<tr>
<td>15</td>
<td>EDITPLIB DD</td>
<td><em>(optional)</em> If an Edit Proc is defined on an object that a utility is operating against, and the Edit Proc is in a load library that is not directly loadable, such as STEPLIB or LINKLIB, you may supply an //EDITPLIB to point NGT utility products to the Edit Proc library. If EDITPLIB is allocated, NGT utility products will attempt to load the Edit Proc from this library. If it cannot, NGT utility products will then perform an SVC 8 and attempt to let z/OS find the Edit Proc.</td>
</tr>
<tr>
<td>16</td>
<td>AGEREPT DD</td>
<td><em>(optional)</em> NGT utility products aging report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGT utility products aging report lists the utility IDs that are pending deletion.</td>
</tr>
</tbody>
</table>
Job completion codes/return codes

After utility execution, a completion code is returned to indicate whether execution was successful or not.

Table 8: Completion code/return code descriptions

<table>
<thead>
<tr>
<th>Return code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution</td>
</tr>
</tbody>
</table>
| 4           | Successful execution with one or more warnings
You can view the warning messages in SYSERROR.
Possible reasons for this code include:
  ■ An image copy was required.
  ■ Discards occurred (for LOAD).
  ■ No utility statements were read from SYSIN. |
| 8           | An error or an abend occurred
Take one of the following actions:
  ■ Correct the error and restart
  ■ Rerun the utility with the QUICKEXIT parameter. QUICKEXIT cleans up work data sets and insures that the objects return to their original state. |
| 16          | An error or ABEND occurred while the utility was in a Must Complete state
The table spaces and indexes are not usable by DB2. To ensure the usability and availability of space, rerun the utility with the RESTART parameter. |

SYSIN statements other than NGT utility products statements

The primary purpose of SYSIN is to provide NGT utility products statements. However, there are other statements that can be useful in SYSIN. Each new statement must begin on a new line, but the statement can begin anywhere and can continue over any number of lines.

The following topics describe SYSIN statements.
Comment

An asterisk (*) in column 1 denotes a comment. All lines beginning with asterisks are ignored.

**Example**

```
SYSIN DD *
* Reorg the XYZ Application
REORG TABLESPACE XYZDB.* RTS
```

SQL

The SQL keyword tells NGT utility products that a SQL statement follows. This SQL will be sent to DB2 for processing. The output will go to CDBPRINT.

**Example**

```
SYSIN DD *
SQL DELETE FROM tbcreator.tbname
LOAD RESUME YES
```

DB2 commands

DB2 Commands can be issued directly in the SYSIN DD statement of a DB2 utility job. The output is sent to CDBPRINT.

**Example**

```
SYSIN DD *
-DIS THD(*)
REORG TABLESPACE dbname.% RTS
```

USERCMD

USERCMD passes information to the XSUTSYIN automation control point. Similar to a comment, USERCMD has no direct effect on other statements in SYSIN.

**Example**

In this example VAR1='ABC123' is passed to XSUTSYIN prior to running Reorg.

```
SYSIN DD *
USERCMD VAR1 = 'ABC123'
REORG TABLESPACE dbname.tsname RTS
```
Contact BMC Customer Support about the use of USERCMD. USERCMD has valuable uses and nuances that need to be understood in a parallel processing environment.

NGTPBG

You can add partitions to a PBG Universal Table Space without executing an ALTER ADD PARTITION statement and executing a Reorg.

Example

```
SYSIN DD *
NGTPBG TABLESPACE dbname.tsname PBGPARTS(new_#parts)
```

Note

Consider the following points:

- `dbname.tsname` must be a PBG UTS.
- `new_#parts` must be greater than the current number of parts and less than MAXPARTS.

OUTPUT statement and dynamic allocation of output data sets

This section discusses the dynamic allocation options that you can include in an OUTPUT statement in your NGT utility SYSIN data set.

The OUTPUT statement is a `utility-control-statement` that you enter after the `//SYSIN DD *` JCL statement in the SYSIN statement.

The OUTPUT command must precede a statement that refers to it. All statements in the SYSIN data set are processed sequentially, so a new output descriptor named in an OUTPUT statement is only available for statements that follow it. You can use more than one OUTPUT statement in a SYSIN data set, but each output descriptor must have a different name.

Example

```
OUTPUT copy01 UNIT unitName DSNAME copyDSName
OUTPUT copy02 UNIT unitName...
REORG TABLESPACE tsName COPYDDN(copy01,copy02)
```

The order of the OUTPUT commands before the utility statement is not important. For example, OUTPUT copy02 could precede OUTPUT copy01.
**OUTPUT syntax rules and diagram**

The following syntax diagram shows the options you can use with an OUTPUT statement to override the current default output descriptor values.

When you use an OUTPUT statement in the NGT utility SYSIN statement to override default output descriptor values, these rules apply:

- The statement must start with the OUTPUT keyword and the name of the descriptor you want to use to dynamically allocate your utility data sets.
- You can specify only options that apply to the media you use; that is, all of the options must apply either to disk data sets or to tape data sets.
- Any option that you do not specify will be used with the value specified in the current installation options module.
- An asterisk in column 1 in the SYSIN data set specifies that the line is a comment that will not be echoed in the SYSPRINT output.

![OUTPUT command syntax](image)

**Note**

Options specific to disk data sets and those specific to tape data sets are mutually exclusive: you cannot specify both disk copy data sets and tape copy data sets in the same OUTPUT statement. If you want to specify both disk and tape copy data sets in the same SYSIN data set and want to override default output descriptor values in both cases, you must use one OUTPUT statement for the disk data sets and another for the tape data sets. Also, the names of the descriptors must be different.

**Keywords**

**OUTPUT name**

Specify the OUTPUT keyword to introduce a new output descriptor name. The NGT utility creates the named descriptor and overrides the existing default values for the options specified in the OUTPUT statement. The value for name must not exceed 8 characters and must follow the rules for DD names.
OUTPUT syntax options common to disk and tape data sets

This section describes options you can use to write data sets to either disk or tape.

Figure 14: Syntax diagram of common options

| UNIT name | Specify UNIT and a new tape or disk unit name when you want to override the default unit named in the installation options module. The NGT utility installation default is SYSALLDA. If you specify a new name, the NGT utility uses it for UNIT. |
| DSNAMES dataSetName | Specify DSNAMES and a data set name (dataSetName) to set a new default data set name. |

You can construct dataSetName using the supported symbolic variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;ATTACH</td>
<td>DB2 group attachment name or subsystem ID (up to 4 characters)</td>
</tr>
<tr>
<td>&amp;DATE</td>
<td>Current date (in the form MMDDYY)</td>
</tr>
<tr>
<td>&amp;DATEJ</td>
<td>Current Julian date (in the form YYYYDDD)</td>
</tr>
<tr>
<td>&amp;DAY</td>
<td>Current day (in the format DD)</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&amp;DB</td>
<td>Database containing the space being used for this data set allocation (up to 8 characters)</td>
</tr>
<tr>
<td>&amp;DDNAME</td>
<td>DDname being used for this data set allocation (up to 8 characters)</td>
</tr>
<tr>
<td>&amp;GRPNM</td>
<td>DB2 data-sharing group name In a non-data-sharing environment, GRPNM contains the DB2 SSID. (up to 4 characters)</td>
</tr>
<tr>
<td>&amp;HOUR</td>
<td>Current hour (in the format HH)</td>
</tr>
<tr>
<td>&amp;JDATE</td>
<td>Current Julian date (in the form YYDDD)</td>
</tr>
<tr>
<td>&amp;JDAY</td>
<td>Current Julian day (in the format DDD)</td>
</tr>
<tr>
<td>&amp;JOBNAME</td>
<td>JOB name used in the JCL (up to 8 characters)</td>
</tr>
<tr>
<td>&amp;MIN</td>
<td>Current minute (in the format MM)</td>
</tr>
<tr>
<td>&amp;MINUTE</td>
<td>Current minute (in the format MM)</td>
</tr>
<tr>
<td>&amp;MONTH</td>
<td>Current month (in the format MM)</td>
</tr>
<tr>
<td>&amp;OBID</td>
<td>OBID of the table space (up to 4 figures)</td>
</tr>
<tr>
<td>&amp;PART</td>
<td>Partition being used for this data set allocation (up to 4 figures)</td>
</tr>
<tr>
<td>&amp;PART5</td>
<td>Partition being used for this data set allocation 5-character partition numbers as follows:</td>
</tr>
<tr>
<td></td>
<td>■ Partition 1 = 00001</td>
</tr>
<tr>
<td></td>
<td>■ Partition 10 = 00010</td>
</tr>
<tr>
<td></td>
<td>■ Partition 100 = 00100</td>
</tr>
<tr>
<td></td>
<td>■ Partition 1000 = 01000</td>
</tr>
<tr>
<td></td>
<td>■ Nonpartitioned = 0000</td>
</tr>
<tr>
<td>&amp;REPLACE</td>
<td>LOAD REPLACE being done (Y or N)</td>
</tr>
<tr>
<td>&amp;RESUME</td>
<td>LOAD RESUME being done (Y or N)</td>
</tr>
<tr>
<td>&amp;RTYPE</td>
<td>REORG type (TS or IX)</td>
</tr>
<tr>
<td>&amp;SEC</td>
<td>Current second (in the format SS)</td>
</tr>
<tr>
<td>&amp;SECOND</td>
<td>Current second (in the format SS)</td>
</tr>
<tr>
<td>&amp;SEQ</td>
<td>Sequence number (3 figure statement number)</td>
</tr>
<tr>
<td>&amp;SSID</td>
<td>The DB2 subsystem name or group name specified on the PARM=name attribute (up to 4 characters)</td>
</tr>
<tr>
<td>&amp;STEPNAME</td>
<td>STEP name used in the JCL (up to 8 characters)</td>
</tr>
<tr>
<td>&amp;TIME</td>
<td>Current time (in the form HHMMSS)</td>
</tr>
<tr>
<td>&amp;TIME4</td>
<td>Current time (in the format HHMM)</td>
</tr>
<tr>
<td>&amp;TS</td>
<td>Table space (up to 8 characters)</td>
</tr>
<tr>
<td>&amp;TSIX</td>
<td>Table space or index space (up to 8 characters)</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&amp;UID</td>
<td>Job user ID (up to 8 characters)</td>
</tr>
<tr>
<td>&amp;UNIQ</td>
<td>Unique characters (up to 8 characters)</td>
</tr>
<tr>
<td>&amp;USER1</td>
<td>The value of the +USER1(&lt;userVariable&gt;) parameter</td>
</tr>
<tr>
<td>&amp;USER2</td>
<td>The value of the +USER2(&lt;userVariable&gt;) parameter</td>
</tr>
<tr>
<td>&amp;USERID</td>
<td>Job user ID (up to 8 characters)</td>
</tr>
<tr>
<td>&amp;UTIL</td>
<td>BMC utility ID (up to 8 characters)</td>
</tr>
<tr>
<td>&amp;UTILPFX</td>
<td>BMC utility ID prefix (up to 8 characters)</td>
</tr>
<tr>
<td>&amp;UTILSFX</td>
<td>BMC utility ID suffix (up to 8 characters)</td>
</tr>
<tr>
<td>&amp;VCAT</td>
<td>VCATNAME specified in the DB2 catalog for the table space (up to 8 characters)</td>
</tr>
<tr>
<td>&amp;VER</td>
<td>The version (for example 1210) of the NGT utility being executed</td>
</tr>
<tr>
<td>&amp;VER2</td>
<td>The first two characters of the version (for example 12) of the NGT utility being executed.</td>
</tr>
<tr>
<td>&amp;VER3</td>
<td>The first 3 characters of the version (for example 121) of the NGT utility being executed.</td>
</tr>
<tr>
<td>&amp;YEAR</td>
<td>Current year (in the format YY)</td>
</tr>
</tbody>
</table>

**CATLG**

Specify CATLG and either YES or NO (as appropriate) to redefine the MVS catalog directive for the named descriptor. If you are using the NGT utility installation option defaults, this is CATLG YES.

If any SMS option (STORCLAS, DATACLAS, or MGMTCLAS) is used, the NGT utility forces CATLG YES.

**EXPOUT**

Specify EXPOUT YES or NO to tell the NGT utility to create a migration file for data migration. The default value is NO.

*Note*

Use of this feature requires one of the following valid passwords:

- A Recovery Management solution password
- A Database Administration solution password

EXPOT YES on the OUTPUT command used with an EXPORT command creates a migration file in NGT Copy. The migration file is then used by the NGT Recover MIGRATE and IMPORT commands for data movement. (For
more information about MIGRATE and IMPORT, see the *BMC Next Generation Technology Recover for DB2 for z/OS Reference Manual*.)

**MODELDCB** `dataSetName`

Specify **MODELDCB** and a cataloged data set name (`dataSetName`) to redefine the model DCB for the named descriptor. If you do not specify **MODELDCB** for the named descriptor, the NGT utility uses the installation option value (if any).

To specify that no model DCB be used, use **MODELDCB NONE**.

The specified model data set must be allocated on a mounted direct access volume; the NGT utility copies the DCB information from the data set label.

You can construct `dataSetName` using the supported symbolic variables. For more information, see **DSNAME** on page 114.

**VOLCNT** `integer`

To set **VOLCNT** `integer` for the named descriptor, specify the largest number of volumes you expect the NGT utility to process when copying a single data set. For disk data sets, this option limits the values of the VOLUME, LPVOLS, LBVOLS, RPVOLS, and RBVOLS options. For both tape and disk data sets, `integer` must be equal to or greater than the number of volumes produced for the single largest output copy, whether or not you use stacked output.

**Note**

For a nonspecific disk data set allocation, **VOLCNT** is ignored. **UNITCNT** should be used to request a multi-volume disk data set. Valid values for **UNITCNT** are 0 (zero) through 59. The default value is **UNITCNT**=0, which means the unit count will not be specified for the allocation.

If you do not specify **VOLCNT** for the named descriptor, the NGT utility uses the installation option value. If you are using the NGT utility installation option default, the default is 25. If the **VOLUMES** option is defined for disk data sets, the default is the number of volumes in the list. To use the MVS default, set VOLCNT to 0.

**Note**

If you are using SMS in your system, BMC recommends you use **VOLCNT** 0.

**ENCIPHER**

Use the **ENCIPHER** option to indicate whether you want to make encrypted copies.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENCIIPHER NO</td>
<td>ENCIIPHER NO is the default value and specifies that NGT Copy does not make encrypted copies.</td>
</tr>
<tr>
<td>ENCIIPHER YES</td>
<td>ENCIIPHER YES indicates that you want to create encrypted copies. ENCIIPHER YES in the OUTPUT command works with the COPY and COPY IMAGECOPY commands to encrypt all image copies sent to the device specified in the OUTPUT command.</td>
</tr>
</tbody>
</table>

**Note**

ENCIPHER is supported only by NGT Copy, and is ignored if it appears in an OUTPUT statement for all other NGT utilities.

Encryption is a feature of the Recovery Management for DB2 solution and requires a valid Recovery Management solution password.

ENCIPHER is incompatible with DSSNAP and with compressed indexes.

**BUFNO integer**

When you make copies using the utility, specify BUFNO for the named descriptor when you want to set a new value for the number of DB2 BSAM buffers. If you do not provide a new value, BUFNO defaults to the value of the BUFNO installation option. The NGT utility installation option default is 10. Valid values for BUFNO are in the range 0 through 99.

**STORCLAS name**

Specify STORCLAS name when you want to provide a new SMS storage class name for the named descriptor. The value of name must be a valid SMS storage class name, not exceeding eight characters.

If you do not specify STORCLAS for the named descriptor, the NGT utility uses the installation option value (if any). If you do not want to use a value for STORCLAS and a value exists in the current default output descriptor, specify STORCLAS NONE in the OUTPUT statement.

the NGT utility forces CATLG=YES when you specify STORCLAS.

**DATACLAS name**

Specify DATACLAS name when you want to provide a new SMS data class name for the named descriptor. The value of name must be a valid SMS data class name, not exceeding eight characters.

If you do not specify DATACLAS for the named descriptor, the NGT utility uses the installation option value (if any). If you do not want to use a value for DATACLAS and a value exists in the current default output descriptor, specify DATACLAS NONE in the OUTPUT statement.
The NGT utility forces CATLG=YES when you specify DATACLAS.

**MGMTCLAS name**

Specify MGMTCLAS name when you want to provide a new SMS management class name for the named descriptor. The value of name must be a valid SMS management class name, not exceeding eight characters.

If you do not specify MGMTCLAS for the named descriptor, the NGT utility uses the installation option value (if any). If you do not want to use a value for MGMTCLAS and a value exists in the current default output descriptor, specify MGMTCLAS NONE in the OUTPUT statement.

The NGT utility forces CATLG=YES when you specify MGMTCLAS.

**UNITCNT integer**

Use UNITCNT to specify the unit count used for dynamic allocation. Valid values are 0 (zero) to 59. The value 0 means the unit count will not be specified for the allocation.

If you do not specify UNITCNT for the named descriptor, the NGT utility uses the installation option value. If you are using the NGT utility installation option default, the default is 0. If you do not want to use a value for UNITCNT and a value exists in the current default output descriptor, specify UNITCNT 0 in the OUTPUT statement.

Specifying UNITCNT 2 for tape output will allocate two tape drives. When a tape volume is at the end of tape, the NGT utility begins writing on the second drive immediately. This allows you to eliminate time spent waiting for tape rewind.

Specifying UNITCNT 11 for disk output will allow the NGT utility to allocate a primary and 15 secondary extents on up to 11 volumes, which reduces the risk of getting B37 abends.

**GDGLIMIT integer**

This value is used for the GDG LIMIT if a base GDG needs to be defined.

**DISP dataSetName(argument1,argument2,argument3)**

The dataSetName takes three arguments that define the disposition of the specified data set:
Table 9: Initial status (argument)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLD</td>
<td>The data set exists before this step and this step requires exclusive (unshared) use of the data set.</td>
</tr>
<tr>
<td>SHR</td>
<td>The data set exists before this step and other jobs can share it.</td>
</tr>
<tr>
<td>NEW</td>
<td>A new data set is to be created in this step.</td>
</tr>
</tbody>
</table>
| MOD   | The behavior differs for new or existing data sets:  
  ■ The data set exists and records are to be added to the end of it.  
  Note: The data set must be sequential.  
  ■ A new data set is to be created. |

Table 10: Normal termination status (argument2)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| CATLG | If the step terminates normally, the system places an entry pointing to the data set in the catalog.  
  Note: The data set is kept on the volume. |
| KEEP  | The data set is kept on the volume if this step terminates normally. |
| DELETE| The data set is no longer needed if this step terminates normally. |

Table 11: Abnormal termination status (argument3)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| CATLG | If the step terminates abnormally, the system places an entry pointing to the data set in the catalog.  
  Note: The data set is kept on the volume. |
| KEEP  | The data set is kept on the volume if this step terminates abnormally. |
| DELETE| The data set is no longer needed if this step terminates abnormally. |

The default values differ for new or existing data sets:  
■ If the data set exists, the default values are MOD,KEEP,KEEP  
■ If data set is new, the default values are NEW,CATLG,DELETE

OUTPUT syntax options reserved for tape data sets

This section describes the options that apply only to copies written to tape.
The following OUTPUT tape options are supported only by NGT Copy and NGT Recover, and are ignored if they appear in an OUTPUT statement for all other NGT utilities:

- STACK
- REALDD
- STACK CABINET
- TRTCH

Figure 15: Syntax diagram of OUTPUT options reserved for tape data sets

ID (required if you are specifying multiple outputs) The system uses the ID keyword to distinguish between multiple user defined outputs.

LP/LB/RP/RB/OT/Ox | Ux

(a value is required, if you specify the ID keyword) Specify the ID of the tape unit, which can take the following values:

- LP is the Local Primary copy
- RP is the Remote Primary copy
- LP is the Local Backup copy
- RB is the Remote Backup copy
- OT is a user defined output
- Ox is a user defined output, where x can be any number between 1 and 9
- Ux is a user defined output, where x can be any alphanumeric character.
RETPD integer

Specify RETPD integer in the descriptor named after the OUTPUT keyword to set a new retention period (in days) for the current copy data set.

The value of integer must be in the range 1 through 9999.

Note
When EXPDT is specified on the OUTPUT statement, it takes precedence over RETPD. However, if EXPDT is not specified on the OUTPUT statement and RETPD is specified on the OUTPUT statement, RETPD overrides EXPDT in the installation options module.

EXPDT date

Specify EXPDT date in the descriptor named after the OUTPUT keyword to set a new expiration date for the current copy data set. The default value of the EXPDT installation option is 99000.

Note
When it is specified on the OUTPUT statement, EXPDT takes precedence over RETPD. However, if EXPDT is not specified on the OUTPUT statement and RETPD is specified on the OUTPUT statement, RETPD overrides EXPDT in the installation options module.

The value of date must be in the format YYDDD, YYYYDDD, or YYYY/DDD where YYYY is the 4-digit year, YY is the last two digits of the year, and DDD is the 3-digit Julian day (001 through 366).

Note
A date with a two-digit year is passed as is to dynamic allocation. For years beyond 1999, depending on your environment, this might not produce the appropriate result. BMC recommends using a four-digit year.

OUTPUT syntax options reserved for disk data sets

This section describes options that apply only to copy data sets written to disk devices.
Note
The following OUTPUT disk options are supported only by NGT Copy and NGT Recover, and are ignored if they appear in an OUTPUT statement for all other NGT utilities:
- STACK CABINET
- DSSNAP
- MIGRATE

Figure 16: Syntax diagram of OUTPUT options reserved for disk data sets

**SPACE** *(primary, secondary)* allocation unit

Specify SPACE to set the output allocation units (tracks or cylinders) for the named descriptor. Specify SPACE TRK to allocate the output in tracks. Specify SPACE CYL to allocate the output in cylinders. The default value for the installation option is CYL.

If you do not want primary and secondary space requirements *(primary, secondary)* to be automatically calculated, specify the values you want in parentheses, as in the following example: SPACE (200,100) CYL.

**PCTPRIM** integer

* The NGT utility provides these default values at installation time.
Specify PCTPRIM for the named descriptor to set a new value for the percentage of the total space that needs to be allocated as primary space. The default for this installation option is 100.

**Note**
For large table spaces, the primary allocation calculated by PCTPRIM might be too large. To override the calculated value, you can use MAXPRIM.

**MAXPRIM integer**

Specifying MAXPRIM for the named descriptor allows you to do the following things:

- Set a new value for the maximum amount of disk space (in the units specified by SPACE) that can be allocated as primary space
- Put an upper limit on the value calculated by PCTPRIM (in the case of large table spaces)

A nonzero value for integer establishes an upper limit for primary space allocation, while a value of zero specifies no limit.

The default for the installation option is 559.

**NBRSECD integer**

Specify NBRSECD for the named descriptor when you want to set a new value for the size of secondary allocations. After the primary allocation is calculated, the remaining space is secondary space and can be divided into from 1 to 15 parts. This is specified by integer, which must be in the range 1 through 15.

The default for the installation option is 10.

**Note**
The size of the secondary allocation cannot be less than 10% of the primary.

**LPVOLS (vol1, vol2, ...)**

Specify LPVOLS to provide a new list of disk volumes for storing local site primary copies for the named descriptor. The number of entries in the list must not exceed the value specified by VOLCNT in the named descriptor (see **VOLCNT on page 117**) . If the data set is uncataloged, the NGT utility
truncates the list recorded in SYSIBM.SYSCOPY to reflect the actual volumes used.

If you do not want to use a value for LPVOLS and there is an installation option default, specify LPVOLS(NONE).

--- WARNING ---
There must be enough space on the first specified volume to allocate the primary space required for the output data set.

**LBVOLS** *(vol1, vol2, ...)*

Specify LBVOLS to provide a new list of disk volumes for storing local site backup copies for the named descriptor. The number of entries in the list must not exceed the value specified by VOLCNT in the named descriptor (see VOLCNT on page 117). If the data set is uncataloged, the NGT utility truncates the list recorded in SYSIBM.SYSCOPY to reflect the actual volumes used.

If you do not want to use a value for LBVOLS and there is an installation option default, specify LBVOLS(NONE).

--- WARNING ---
There must be enough space on the first specified volume to allocate the primary space required for the output data set.

**RPVOLS** *(vol1, vol2, ...)*

Specify RPVOLS to provide a new list of disk volumes for storing recovery site primary copies for the named descriptor. The number of entries in the list must not exceed the value specified by VOLCNT in the named descriptor (see VOLCNT on page 117). If the data set is uncataloged, the NGT utility truncates the list recorded in SYSIBM.SYSCOPY to reflect the actual volumes used.

If you do not want to use a value for RPVOLS and there is an installation option default, specify RPVOLS(NONE).

--- WARNING ---
There must be enough space on the first specified volume to allocate the primary space required for the output data set.
**RBVOLS** *(vol1, vol2, ...)*

Specify RBVOLS to provide a new list of disk volumes for storing recovery site backup copies for the named descriptor. The number of entries in the list must not exceed the value specified by VOLCNT in the named descriptor (see [VOLCNT on page 117](#)). If the data set is uncataloged, the NGT utility truncates the list recorded in SYSIBM.SYSCOPY to reflect the actual volumes used.

If you do not want to use a value for RBVOLS and there is an installation option default, specify RBVOLS(NONE).

---

**WARNING**

There must be enough space on the first specified volume to allocate the primary space required for the output data set.

---

**VOLUMES** *(vol1, vol2, ...)*

Specify VOLUMES to provide a new list of default volumes for the named descriptor. The new list is used as the default list for LPVOLS, LBVOLS, RPVOLS, and RBVOLS for the named descriptor. The number of entries in the list must not exceed the value specified by VOLCNT for the named descriptor (see [VOLCNT on page 117](#)). If the data set is uncataloged, the NGT utility truncates the list recorded in SYSIBM.SYSCOPY to reflect the actual volumes used.

If you do not want to use a value for VOLUMES and there is an installation option default, specify VOLUMES(NONE).

---

**WARNING**

There must be enough space on the first specified volume to allocate the primary space required for the output data set. This is true for standard copies or Instant Snapshot copies.

---

**DISKRETN integer**

When you use dynamic allocation, specify DISKRETN integer in the descriptor named after the OUTPUT keyword to set a new retention period (in days) for the current disk copy data set. If you do not specify DISKRETN, no retention period will be specified.
The value of integer must be in the range 0 through 9999. A value of 0 indicates there is no retention of the disk copy data set.

**Note**
When DISKEXPD is specified, it takes precedence over DISKRETN.

**DISKEXPD date**

When you use dynamic allocation, specify DISKEXPD date in the descriptor named after the OUTPUT keyword to set a new expiration date for the current disk copy data set. If you do not specify DISKEXPD, no expiration date will be specified.

The value of date must be in the format YYDDD, YYYYDDD, or YYYY/DDD where YYYY is the 4-digit year, YY is the last two digits of the year, and DDD is the 3-digit Julian day (001 through 366).

**Note**
A date with a two-digit year is passed as is to dynamic allocation. For years beyond 1999, depending on your environment, this might not produce the appropriate result. BMC recommends using a four-digit year.

**Note**
When it is specified, DISKEXPD takes precedence over DISKRETN.

**EATTR**

Use EATTR to specify whether a data set supports extended attributes or not.

**Table 12: Values of EATTR**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EATTR NONE</td>
<td>Using NONE allows you to have your environment set up to use extended attributes. EATTR NONE specifies no value for EATTR and allows the value for EATTR to be set by an SMS DATACLAS. Using NONE allows you to have your environment set up to use extended attributes.</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EATTR OPT</td>
<td>EATTR OPT specifies that extended attributes are optional for the data set. You must set EATTR=OPT to allocate an extended format sequential data set. By using EATTR=OPT, the NGT utility supports sequential data sets in the cylinder-managed portion of EAVs. Extended format sequential data sets must be allocated on SMS-managed volumes and the size of the data set must be greater than the EAV break point, which is typically 10 cylinders.</td>
</tr>
<tr>
<td>EATTR NO</td>
<td>EATTR NO specifies that the data set cannot have extended attributes.</td>
</tr>
</tbody>
</table>

**Other**

There are other statements which are utility specific, such as NGTRTS, which defines the NGT Utility Manager internal tables. For more information, see the *BMC Next Generation Technology Utility Manager for DB2 for z/OS Reference Manual*. 


Servers

In this chapter we look at commands and facilities that apply to servers and server jobs.

There are two types of servers - disk servers and tape servers. Disk servers process all work other than accessing tapes. Tape servers process the reading and writing of work to tape. This separation insures utility processing tasks are not held up by tape delays.

Defining servers

This section explains how to control the submission of servers. Control over the number of each type of server, JCL skeletons, and manual submission are explained.

Number of servers

NGT servers are automatically initiated by the standard NGT utility job that is submitted. The utility job creates the JCL for the servers through the use of skeletons set up and declared by the user and submits the number of each type of server decided by the user.

With server processing, to reduce processing time and resource usage servers are only started as needed.

For example, if a job requests three servers, the job will initiate with only one server. After initialization the utility re-evaluates the need for more servers and might add one server per minute, up the maximum of three until processing is complete. If there isn't sufficient work to fill all three servers, the job might complete with only one or two servers. NGT processing is smart enough to know how many servers to start.

The number of disk servers is controlled by the global utility parameter +SVRDISK. For more information on this parameter, see “+SVRDISK” on page 192.
Server job types: batch vs started tasks

NGT servers can be run in one of two modes: batch and started tasks.

Batch jobs are automatically triggered by processing and always have an owner. They can be easily viewed in SDSF or other batch management software.

Started tasks, on the other hand, do not have an owner, and are run from members of your data center's specified PROCLIB. Disk and tape jobs may each have their own members for this purpose. In addition, started tasks do not require the use of initiators. The started task PROC name is determined during installation of the NGT utilities.

Defining server JCL

Servers can be batch jobs automatically submitted by the NGT Master job or Started Task steps automatically initiated by the NGT Master job.

Batch server jobs

JCL for the server jobs will be put in a PDS (partitioned data set). At installation they are placed in the NGTCNTL PDS with the automation members. These could be put in an existing PDS, such as a PARMLIB PDS. When moving server jobs, specify where the jobs are in the NGTSVR DD of the configuration job and rerun it. Some users will require different server jobname cards for each data sharing group to specify system affinity. You can create additional job card members, if required. To specify which job to use, override the +SVRJCLJOB parameter in the JCL.

The following figure shows the Server model JCL members on the right and how the Configuration job knows how to find them. All or part of this can be overridden in individual utility jobs.

Figure 17: Configuration Setup (CNFGdbid)

```
//NGTSRVR DD DISP=SHR,DSN=
//UTLPARMS DD*
+SVRJCLJOB(SVRJOBCD)
+SVRJCLJOB(SVRDISK)
+SVRJCLJOB(SVRTAPE)
```

PDS

SVRJOBCD

SVRDISK

SVRTAPE
You can override the PDS used for server members by specifying a NGTSVR DD statement in individual jobs. Alternatively, you can override only the member used for a job by specifying one of the +SVRJCL parameters in the utility jobs UTLPARMS DD.

Also, the use of these server members can be overridden completely by specifying a DD statement in the NGT utility job with the member name specified in each of the three +SVRJCL… parameters above. These DDs would reference a PDS dataset name (DSN) specifying a member needed for use. By specifying the following line in a utility job, the jobcard in member1 will be used.

```
//SVRJOBCD DD DISP=SHR,DSN=yourlibrary(member1)
```

In summary, processing reads the name specified in +SVRJCLJOB, +SVRJCLDISK or +SVRJCLTAPE and takes the following steps:

1. Verifies whether it is a DDNAME in the job; if it is processing uses the library (member) specified to that DD.

2. Looks for NGTSVR in the job to tell where the member is.

3. Looks for the member in the NGTSVR specified in the configuration member (RX11dbid).

4. Looks for the specific member (RX110GBL)

BMC recommends that you use the NGTSVR specification in the configuration to locate the server model JCL. Using this method you can make global changes at the system level, without changing any utility JCL. Only the utility parameters should be overridden in jobs to specify an alternate member.

### Started task server jobs

If started task servers are to be used, then the JOB statement, tape, and disk JCL mentioned above are not used. Instead, a PROCLIB member is added and referenced by the UTLPARMS parameter +SVRPROCD(member). When you request multiple servers, they are automatically generated as started tasks with unique step names.

**Note**

The NGT CNTL library contains two sample started task procedures for use with the +SVRPROCD parameter - member NGTSEVRD (for +SVRPROCD and disk servers). You may use this sample as is, or copy and modify it to customize it to your needs.
Literals

The master utility job submits several servers from the skeleton JCL. This requires variables to be used in the Job names and Step names, so that there is no contention with duplicates. NGT utility products has provided literals to be used in these names.

You must use at least one of these literals in the server job name, otherwise two different jobs with the same name could be submitted (causing them to run serially rather than in parallel).

---

**Note**

Generated literal, including all "&", ".", "(" and ")" characters, must not exceed 8 characters, and all literals (or symbolics) must end with a delimiter. That can be a ',', '.', '(.'), '&', '"", or a space.

The '.' (or period character) is special and is replaced as part of the symbolic (again normal MVS standards). The job name/step name produced is limited to 8 characters by MVS. If the master's jobname is MSTRJOB1 and a DASD server is being started, then &MJ(4)&JT&NN.P should be coded. This would produce a name of MSTRD01P which meets the MVS standards and is 8 characters long.

<table>
<thead>
<tr>
<th>Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;MJ</td>
<td>The master job name.</td>
</tr>
<tr>
<td>&amp;MJ(n)</td>
<td>The first n characters of the master job name.</td>
</tr>
<tr>
<td>&amp;MU</td>
<td>The user ID of the master job</td>
</tr>
<tr>
<td>&amp;MU(n)</td>
<td>The first n characters of the user ID of the master job.</td>
</tr>
<tr>
<td>&amp;MJC</td>
<td>The master job class, a single letter from A to Z or a single number from 0 to 9.</td>
</tr>
<tr>
<td>&amp;JT</td>
<td>The job type; either a D (for disk) or a T (for tape).</td>
</tr>
<tr>
<td>&amp;SPEC</td>
<td>A unique character string is inserted each time a server is submitted from a job. The proper use of this literal is to specify a single letter from A to Z, followed by &amp;SPEC (e.g., A&amp;SPEC, B&amp;SPEC, C&amp;SPEC, ... Z&amp;SPEC).</td>
</tr>
<tr>
<td>&amp;N</td>
<td>A random number 0-9 is inserted incremented each time a server is submitted from a job. If you have specified less than 9 servers, this literal provides unique names.</td>
</tr>
<tr>
<td>&amp;NNN</td>
<td>A random number 000-999 is inserted incremented each time a server is submitted from a job. If you have specified less than 999 servers, this literal provides unique names.</td>
</tr>
<tr>
<td>&amp;A</td>
<td>A letter starting at A and incrementing to Z is inserted each time a server is submitted from a job.</td>
</tr>
</tbody>
</table>
Table 13: Examples of literal usage

<table>
<thead>
<tr>
<th>Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;MJ(2)&amp;N</td>
<td>A unique name starting with the first two characters of the master job name and followed by a single random number from 0 to 9.</td>
</tr>
<tr>
<td>&amp;MU(3)&amp;N</td>
<td>A unique name starting with the first three characters of the master job's user ID and followed by a single random number from 0 to 9.</td>
</tr>
<tr>
<td>A&amp;SPEC</td>
<td>A unique name starting with A will be used.</td>
</tr>
<tr>
<td>SERV&amp;NNN</td>
<td>The name SERV001 through SERV999 will be used.</td>
</tr>
<tr>
<td>NGTSTP&amp;A</td>
<td>The name NGTSTPA through NGTSTPZ will be used.</td>
</tr>
</tbody>
</table>

**Tip**

BMC recommends that job name masks include a &NNN to guarantee unique server job names. If most of the 8 characters allowed have been used up, then &NN or &A may be substituted.

The automation control point XSVRJCL can provide additional functionality in the definition of literals. For more information on XSVRJCL, see the BMC Next Generation Technology Automation Reference Manual.

The following figure shows a sample JCL skeleton. Numbered lines are described in detail in Table 14 on page 134.

**Figure 18: Server JCL skeleton**

```jcl
//DSERV&NN EXEC PGM=NGTSERVR,PARM=&SVRPARM (1)

//*****************************************************************
//*REQUIRED DD CARDS
//*****************************************************************
//STEPLIB DD DSN=NGT.UTILITY,DISP=SHR (2)
//SYSPRINT DD SYSOUT=* (3)

//*****************************************************************
//*EVERYTHING BELOW IS OPTIONAL
//*****************************************************************
//SYSERROR DD SYSOUT=* (4)
//SUMMARY DD SYSOUT=* (5)
//RUNSTATS DD SYSOUT=* (6)
//SYSTSPRT DD SYSOUT=* (7)
//UTPRINT DD SYSOUT=* (8)
```
Table 14: JCL descriptions

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EXEC Statement</td>
<td>On this statement, the step name MUST contain the literal &amp;NN, and the values for PGM= and PARM= MUST be coded exactly as you see them in this example. <em>Server processing will fail if this statement is not correctly coded.</em> Please review this statement carefully and ensure that it is correctly coded.</td>
</tr>
<tr>
<td>2</td>
<td>STEPLIB DD</td>
<td>STEPLIB DSNs are provided to match those specified in the master job.</td>
</tr>
<tr>
<td>3</td>
<td>SYSPRINT DD</td>
<td><em>(required)</em> A SYSPRINT statement must be coded to capture utility output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>From SYSPRINT the utility output is transferred to the CDBPRINT DD data set. The SYSPRINT data set is cleared and released following each NGT utility step. The SYSPRINT dataset provides a valuable diagnostic tool in the event of errors. However, some NGT utility products employ parallel processing, which can result in overlapping outputs in SYSPRINT. The SUMMARY DD statement contains a more readable analysis of the utility execution.</td>
</tr>
<tr>
<td>4</td>
<td>SYSERROR DD</td>
<td><em>(optional)</em> SYSERROR is used for NGT server processing error messages.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If this DD statement is not specified, error messages appear only in SYSPRINT. However, if SYSERROR is specified, error messages appear in SYSERROR and SYSPRINT.</td>
</tr>
<tr>
<td>5</td>
<td>SUMMARY DD</td>
<td><em>(optional)</em> SUMMARY statement to capture utility output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compared with SYSPRINT, SUMMARY contains a more organized analysis of the utility execution. It includes reports printed at the conclusion of a utility run, and does not reflect output overlapped from parallel processing. If this DD statement is specified, summary reports are not printed to SYSPRINT. Instead, they are printed to SUMMARY as each utility statement completes. If this DD statement is omitted, the summary reports are printed to SYSPRINT.</td>
</tr>
<tr>
<td>6</td>
<td>RUNSTATS</td>
<td><em>(optional)</em> RUNSTATS report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RUNSTATS contains a report showing RUNSTATS values, and messages affirming update of the DB2 Catalog with these values. This report is printed for any utility that also updates Catalog RUNSTATS values, such as REORG and LOAD.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If this DD statement is omitted, there is no RUNSTATS report on any statement (including SYSPRINT and SUMMARY).</td>
</tr>
</tbody>
</table>
# Name | Description
---|---
7 | SYSTSPRT

*(optional) SYSTSPRT dataset
Any messages sent to TSO are also sent to SYSTSPRT. This may include output from automation control points. REXX Displays from automation are only sent back to the job if the Journal level output is requested. With Journal level output these displays are sent to CDBPRINT unless this SYSTSPRT DD is included, then they are sent to SYSTSPRT.

8 | UTPRINT DD

*(optional) Unless otherwise specified in the JCL, RUNSTATS DD, //SUMMARY, and //SYSERROR are dynamically allocated to match the specifications in //SYSPRINT.

---

## Servers and data sharing

In some data centers, you might need to run server jobs within specific LPARs. For example, your enterprise may have a three-way parallel sysplex with a data sharing group running within two of them. You can route the server job to either of the two valid LPARs.

If you are running batch jobs, you can use the +SVRDISKJC parameter to specify the job class.

For started tasks (STC), servers are started on the system where the master task is running.

If you cannot use the job class, immediately after the JOB statement specify the following line:

```ruby
/*JOBPARM S=(sys1,sys2 )
```

The two variables *sys1* and *sys2* represent the JES member names for the two LPARs in the MAS (JES spool environment).

### Note

The IBM SCHENV parameter is supported in server jobs when it is specified in the master job.

For details on the JOBPARM statement, see the *IBM MVS JCL Reference.*
NGT utility products are designed to distribute and balance work among servers, whether for one partitioned object or for many table spaces and indexes. This ability to distribute work even at the object level provides for the best possible balance of work, without requiring you to changing the job as objects change size.

For efficient use of memory and to reduce the runtime of any one object, the NGT server locks on to a partitioned table space or a database until that is complete. This enables the server to load a copy of only one database descriptor (DBD) at a time.

Procedurally, when three servers are requested, NGT will start one and then wait a minute and check to see if there is sufficient work on the queue; if so, it will start a second server. This will continue until a maximum of three servers are running. If there isn't enough work, all the work may be done with only two servers. If there is work to be done from many data bases and two servers are working on one large object that will leave one servers for other objects. This distribution of work is dynamic, and changes throughout the entire job.

There will be times when a server sits idle long enough to be ended by themaster. If at a later time its services are needed the master will start up another server to replace it.

It is possible for a large partitioned object to be use all the servers preventing smaller objects from starting; or conversely, many smaller objects can delay the start of a large table space. It may be beneficial to put the largest table spaces in a separate job by themselves.

**Server modes and the +SVRMODE parameter**

The global parameter +SVRMODE controls the type of servers used by processing.

You can select the following server modes:

- **External**—signifies that external server processing is to be used. External server processing spawns one or more server jobs and requires additional initiators to be available. This option is best for processing very large table spaces with very large non-partitioned indexes.

- **Internal**—signifies that all work is to be done in the Master job without Server jobs. This option is only for small objects and a small number of objects.
Mixed—signifies that a mixture of external and internal processing is to be used, determined by NGT processing logic. Processing examines or weights the size of all objects to be processed and uses the appropriate process accordingly.

**Note**

If you select Mixed mode and NGT processing determines that too much work exists for in-memory processing, then the processing mode is switched to External.

An external server is started whenever the monitor finds disk work waiting to be processed in the status file. In the case of +SVRMODE(MIXED), processing starts an internal server. The next time processing finds waiting work, processing starts an external server. This loop stays in place until the +SVRDISK(n) value is reached.

For more information on the +SVRMODE parameter, see “+SVRMODE” on page 197.

### Reducing the number of server jobs

Server jobs are most beneficial for spreading a large workload across separate jobs that can make use of separate LPARs of the SYSPLEX. Smaller jobs that don't require this can be expedited by not utilizing external server jobs.

There are three types of servers:

- **Tape servers:** If your utility reads or writes to tape there will always be a tape server. One tape server can allocate many tape drives so it is rare to ever need more than one tape server. A tape server will only be submitted if tape access is used.

- **Disk servers:** This is where most work is done, and without tasks tied up and delayed by tape I/O.

- **Special servers:** This currently is only used by Reorg when +SVRSPECIAL(YES) is specified. Currently this server can be eliminated by specifying +SVRSPECIAL(NO). However this will delay the completion of Reorgs and the deleting of work files. Specifying +SVRSPECIAL(I) for Internal is recommended. This provides the benefit of this special server without starting an external server and using an initiator.

BMC recommends:

- +SVRDISK(2), +SVRTAPE(1), +SVRSPECIAL(I) and +SVRMODE(INTERNAL).

- Specify +SVRDISK(1) for utilities processing a few small objects. Specify +SVRDISK(3) for utilities processing many objects or large table spaces.
Preparation for rolling IPL

NGT utilities can run server jobs on multiple LPARs, and have a NGT subsystem task (CDBSS) running on each LPAR. This section describes what happens to NGT utilities when the server is brought down, and what you should do in preparation and response.

Shut down process

1. Normally work on a LPAR is quiesced prior to shutting it down to prevent interruption of transactions and jobs. In this case the only concern is the CDBSS. As long as the subsystem task is up to see all the DB2s end, it knows that no DB2 changes were missed and utility work on other members of the data sharing group can continue. The same is true for startup, as long as CDBSS is started prior to DB2 it knows that no changes were missed.

2. The CDBSS will most likely have captured changes from DB2 transactions on objects being processed by NGT utilities. Upon shutdown of CDBSS these changes are passed to the CDBSS on the LPAR where the master job is running. This transfer is through a channel adapter where I/O is very fast. This transfer should only take a few seconds. If this process were to fail it would cause all related NGT utilities to fail in order to prevent the loss of any transactions.

3. If a server is actively working on an object when the DB2 subsystem is stopped, that work fails and the object is excluded from processing.

Work on the other objects will complete and in most cases the master will submit a replacement server on one of the remaining LPARs to complete the work using the number of servers that were originally planned.

Preparation and response

1. Set up your IPL procedures and configure NGT according to the information provided in the XIPL section of the BMC Next Generation Technology Subsystem Reference Manual. This configuration ensures that the DB2 subsystems are stopped before the CDBSS closes down, and are restarted after the CDBSS is restarted.

2. Take one of the following actions with the excluded objects caused by the closing of the DB2 subsystem:

   - Rerun the job to RESTART the UID and process it.
   - Perform QUICKEXIT to clean up and skip it; and this decision can be made in advance with the +CLEANUP parameter to avoid the need to RESTART or QUICKEXIT.
DB2 Real Time Statistics (RTS)

DB2 Real Time Statistics tables are updated in real time by DB2 and IBM Utilities. Likewise, NGT utilities update the DB2 Real Time Statistics tables for the objects they are processing in the DB2 catalog when they are run. In addition, NGT Automation allows the automation of NGT utilities based on the DB2 Real Time Statistics tables. For information on the NGT Automation utility, see the *BMC Next Generation Technology Automation Reference Manual*.

### Table 15: IBM RTS Objects

<table>
<thead>
<tr>
<th>Database</th>
<th>Table space</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNDB06</td>
<td>SYSRTSTS</td>
<td>SYSIBM.SYSTABLESPACESTATS</td>
</tr>
<tr>
<td>DSNDB06</td>
<td>SYSRTSTS</td>
<td>SYSIBM.SYSINDEXSPACESTATS</td>
</tr>
</tbody>
</table>

### NGT Copy

NGT Copy updates the following DB2 Real Time Statistics columns.

### Table 16: Fields updated in SYSTABLESPACESTATS by DBID, PSID, and partition

<table>
<thead>
<tr>
<th>Column name</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPYLASTTIME</td>
<td>Current timestamp</td>
</tr>
<tr>
<td>COPYUPDATEDPAGES</td>
<td>0</td>
</tr>
<tr>
<td>COPYCHANGES</td>
<td>0</td>
</tr>
<tr>
<td>COPYUPDATELRSN</td>
<td>NULL</td>
</tr>
<tr>
<td>COPYUPDATETIME</td>
<td>NULL</td>
</tr>
</tbody>
</table>

**Note**

These columns are also updated when NGT Reorg creates a copy.
Table 17: Fields Updated in SYSINDEXSPACESTATS by DBID, ISOBID, and partition

<table>
<thead>
<tr>
<th>Column name</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPYLASTTIME</td>
<td>Current timestamp</td>
</tr>
<tr>
<td>COPYUPDATEDPAGES</td>
<td>0</td>
</tr>
<tr>
<td>COPYCHANGES</td>
<td>0</td>
</tr>
<tr>
<td>COPYUPDATELRSN</td>
<td>NULL</td>
</tr>
<tr>
<td>COPYUPDATETIME</td>
<td>NULL</td>
</tr>
</tbody>
</table>

Note: These columns are also updated when NGT Reorg writes a copy.

NGT Reorg

NGT Reorg updates the following DB2 Real Time Statistics columns.

REORG TABLESPACE updates both the SYSTABLESPACESTATS and SYSINDEXSPACESTATS RTS tables; REORG INDEX only updates the SYSINDEXSPACESTATS table.

Table 18: Fields updated in SYSTABLESPACESTATS by DBID, PSID, and partition

<table>
<thead>
<tr>
<th>Column name</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTALROWS</td>
<td>Actual value</td>
</tr>
<tr>
<td>NACTIVE</td>
<td>Actual value</td>
</tr>
<tr>
<td>NPAGES</td>
<td>Actual value</td>
</tr>
<tr>
<td>SPACE</td>
<td>Actual value</td>
</tr>
<tr>
<td>EXTENTS a</td>
<td>Actual value</td>
</tr>
<tr>
<td>DATASIZE b</td>
<td>Actual value</td>
</tr>
<tr>
<td>REORGLASTTIME</td>
<td>Current timestamp</td>
</tr>
<tr>
<td>REORGINSERTS</td>
<td>0</td>
</tr>
<tr>
<td>REORGDDELETES</td>
<td>0</td>
</tr>
<tr>
<td>REORGUPDATES</td>
<td>0</td>
</tr>
<tr>
<td>REORGDISORGLOB</td>
<td>0</td>
</tr>
<tr>
<td>REORGUNCLUSTINS</td>
<td>0</td>
</tr>
</tbody>
</table>
NGT Reorg also creates a Full Image copy and sets the COPY RTS as specified under NGT Copy in this section.

When NGT Reorg is run with the Runstats(ALL) or (YES) option the STATS RTS columns are set as specified under NGT Stats in this section.

### Table 19: Fields updated in SYSINDEXSPACESTATS by DBID, ISOBID, and partition

<table>
<thead>
<tr>
<th>Column name</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTALENTRIES</td>
<td>Actual value</td>
</tr>
<tr>
<td>NLEVELS</td>
<td>Actual value</td>
</tr>
<tr>
<td>NACTIVE</td>
<td>Actual value</td>
</tr>
<tr>
<td>NPAGES</td>
<td>Actual value</td>
</tr>
<tr>
<td>NLEAF</td>
<td>Actual value</td>
</tr>
<tr>
<td>SPACE</td>
<td>Actual value</td>
</tr>
<tr>
<td>EXTENTS a</td>
<td>Actual value</td>
</tr>
<tr>
<td>REORGLASTTIME</td>
<td>Current timestamp</td>
</tr>
<tr>
<td>REORGINSERTS</td>
<td>0</td>
</tr>
<tr>
<td>REORGDELETES</td>
<td>0</td>
</tr>
<tr>
<td>REORGPAPPENDINSERT</td>
<td>0</td>
</tr>
<tr>
<td>REORGPSEUDODELETES</td>
<td>0</td>
</tr>
<tr>
<td>REORGMASSDELETE</td>
<td>0</td>
</tr>
<tr>
<td>REORGLEAFNEAR</td>
<td>0</td>
</tr>
<tr>
<td>REORGLEAFFAR</td>
<td>0</td>
</tr>
<tr>
<td>REORGNUMLEVELS</td>
<td>0</td>
</tr>
</tbody>
</table>

a For segmented multi-data set table spaces; IBM reports the extents of the last data set, but NGT utilities report the total extents of all data sets.

a For non-partitioned indexes in multiple pieces; IBM reports the extents of the last data set, but NGT utilities report the total extents of all data sets.
NGT Load

NGT Load updates the following DB2 Real Time Statistics columns.

### Table 20: Fields updated in SYSTABLESPACESTATS for a LOAD REPLACE * by DBID, PSID, and partition

<table>
<thead>
<tr>
<th>Column name</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTALROWS</td>
<td>Actual value</td>
</tr>
<tr>
<td>NACTIVE</td>
<td>Actual value</td>
</tr>
<tr>
<td>NPAGES</td>
<td>Actual value</td>
</tr>
<tr>
<td>SPACE</td>
<td>Actual value</td>
</tr>
<tr>
<td>EXTENTS</td>
<td>Actual value</td>
</tr>
<tr>
<td>DATASIZE</td>
<td>Actual value</td>
</tr>
<tr>
<td>LOADRLASTTIME</td>
<td>Current timestamp</td>
</tr>
<tr>
<td>REORGINSERTS</td>
<td>0</td>
</tr>
<tr>
<td>REORDELETE</td>
<td>0</td>
</tr>
<tr>
<td>REORGUPDATES</td>
<td>0</td>
</tr>
<tr>
<td>REORGDISORGLB</td>
<td>0</td>
</tr>
<tr>
<td>REORGUNCLUSTINS</td>
<td>0</td>
</tr>
<tr>
<td>REORGMASDELETE</td>
<td>0</td>
</tr>
<tr>
<td>REORGNEARINDREF</td>
<td>0</td>
</tr>
<tr>
<td>REORGFARINDREF</td>
<td>0</td>
</tr>
</tbody>
</table>

* For segmented multi-data set table spaces; IBM reports the extents of the last data set, NGT Utilities report the total extents of all data sets.

### Table 21: Fields updated in SYSINDEXSPACESTATS by DBID, ISOBID, and partition

<table>
<thead>
<tr>
<th>Column name</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTALENTRIES</td>
<td>Actual value</td>
</tr>
<tr>
<td>NLEVELS</td>
<td>Actual value</td>
</tr>
<tr>
<td>NACTIVE</td>
<td>Actual value</td>
</tr>
<tr>
<td>NPAGES</td>
<td>Actual value</td>
</tr>
<tr>
<td>NLEAF</td>
<td>Actual value</td>
</tr>
<tr>
<td>Column name</td>
<td>Settings</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>SPACE</td>
<td>Actual value</td>
</tr>
<tr>
<td>EXTENTS(^a)</td>
<td>Actual value</td>
</tr>
<tr>
<td>LOADRLASTTIME</td>
<td>Current timestamp</td>
</tr>
<tr>
<td>REORGINSERTS</td>
<td>0</td>
</tr>
<tr>
<td>REORGDELETES</td>
<td>0</td>
</tr>
<tr>
<td>REORGAPPENDINSERT</td>
<td>0</td>
</tr>
<tr>
<td>REORGPSEUDODELETES</td>
<td>0</td>
</tr>
<tr>
<td>REORGMASSDELETE</td>
<td>0</td>
</tr>
<tr>
<td>REORGLEAFNEAR</td>
<td>0</td>
</tr>
<tr>
<td>REORGLEAFFAR</td>
<td>0</td>
</tr>
<tr>
<td>REORGNUMLEVELS</td>
<td>0</td>
</tr>
</tbody>
</table>

\(^a\) For non-partitioned indexes in multiple pieces; IBM reports the extents of the last data set, but NGT utilities report the total extents of all data sets.

**NGT Stats**

NGT Stats updates the following DB2 Real Time Statistics columns.

Table 22: Fields updated in SYSTABLESPACESTATS by DBID, PSID, and partition

<table>
<thead>
<tr>
<th>Column name</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATSLASTTIME</td>
<td>Current timestamp</td>
</tr>
<tr>
<td>STATSINSERTS</td>
<td>0</td>
</tr>
<tr>
<td>STATSDELETES</td>
<td>0</td>
</tr>
<tr>
<td>STATSUPDATES</td>
<td>0</td>
</tr>
<tr>
<td>STATSMASSDELETE</td>
<td>0</td>
</tr>
</tbody>
</table>

These columns are also updated when other NGT utilities specify the RUNSTATS(ALL) or (YES) option.

Table 23: Fields updated in SYSINDEXSPACESTATS by DBID, ISOBID, and partition

<table>
<thead>
<tr>
<th>Column name</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATSLASTTIME</td>
<td>Current timestamp</td>
</tr>
<tr>
<td>Column name</td>
<td>Settings</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
</tr>
<tr>
<td>STATSINSERTS</td>
<td>0</td>
</tr>
<tr>
<td>STATSDELETES</td>
<td>0</td>
</tr>
<tr>
<td>STATSMASSDELETE</td>
<td>0</td>
</tr>
</tbody>
</table>

These columns are also updated when other NGT utilities specify the RUNSTATS(ALL) or (YES) option.
NGT parameters

This chapter describes parameters that apply when you use NGT utilities. Parameters specify options not represented by keywords for NGT utilities, and provide additional control for users.

Examples of options controlled by parameters include the following:

- The amount of time the master job waits for servers to start
- The amount of time to wait between quiesce operations if resources aren't available
- The high-level qualifier to be used for output data sets
- Whether Edit Procs can be run in Supervisor state

Some parameters apply to all NGT utilities (global parameters), and some parameters apply to a specific utility. However, not all NGT utilities have parameters.

Setting default parameters

You can set default parameters in the NGT Utility configuration to run utilities without having to specify the most commonly used parameter values. You can make changes and rerun the configuration whenever you want.

Syntax rules for parameters

The following rules apply to the syntax for parameters for the Next Generation Technology (NGT) products:

- All parameters must start with a plus symbol (+).
Parameters can start anywhere, but must be contained within the first 72 character positions of the line. Columns 73-80 are ignored and can contain sequence numbers.

You can code parameters over multiple lines. Continuation is automatic.

You can code multiple parameters on the same line.

You can code comments by enclosing the comment between /* (beginning) and */ (end). You can place comments anywhere, including in the middle of a word. The utility removes all comment strings before checking syntax.

**Note**

If you specify parameters by using a DD statement in your JCL, do not begin a comment with /* in column one. Placing /* in column one causes IBM MVS to generate a //SYSIN statement for any subsequent records in the data set unless you use DLM= to change the JCL comment delimiter.

You can nest comments, with no limit to the number of nested levels that you can use. Ensure that each */ has a corresponding */.

When specifying a series of values, you must separate them with commas. Blanks, wherever they are coded, have no significance. The utility removes all blanks before checking syntax.

If the same parameter is specified multiple times within the same input data set, the latest specification prevails.

Do not use the plus symbol inside parentheses for any parameter.

**Overview of global utility parameters (UTLPARMS)**

Global parameters specify options that apply to all NGT utilities. Default values are set at installation time, but you can override them on an individual job basis by specifying new values in the UTLPARMS DD statement.
Tip

To change parameters to incomplete utility-ids, you must include the NORESTART keyword in the EXEC statement. RESTART uses the parameters saved from the original UID execution.

You can also specify parameter values in a sequential data set or member of a partitioned data set. This is beneficial because you can change parameters for a series of jobs by altering one file rather than many.

You do not need to specify all the global utility parameters if you want to override only one some of them. You can specify individual parameters in the section of the job, and the defaults for the other parameters remain in effect.

The following table provides the global utility parameters in alphabetical order.

**Table 24: Global utility options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Default value</th>
<th>Use and value</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ALLERROR</td>
<td>YES</td>
<td>Sets the behavior of SYSERROR</td>
</tr>
<tr>
<td>+AGELIMIT</td>
<td>30</td>
<td>Max +OUTPUTAGE value; jobs cannot override this parameter</td>
</tr>
<tr>
<td>+CHECKDEFERRED</td>
<td>YES</td>
<td>Pre check and skip Deferred (DEFINE NO) objects</td>
</tr>
<tr>
<td>+CLEANUP</td>
<td>NO</td>
<td>Clean up work data sets that are not needed for restart after a failure. Can be overridden by XSVRXERR</td>
</tr>
<tr>
<td>+CONTROLPREFIX</td>
<td>No default</td>
<td>DSName prefix for INFO and OUT files</td>
</tr>
<tr>
<td>+COPYPENDING</td>
<td>YES</td>
<td>Set COPY PENDING? YES</td>
</tr>
<tr>
<td>+DEADLINE</td>
<td>0 (unlimited)</td>
<td>Limit in minutes placed on Change Apply</td>
</tr>
<tr>
<td>+DELETEOUTPUT</td>
<td>NO</td>
<td>Delete all work and output data sets when RC=0</td>
</tr>
<tr>
<td>+DELETEWORK</td>
<td>YES</td>
<td>Whether to delete work data sets</td>
</tr>
<tr>
<td>+DIGITS</td>
<td>CDB</td>
<td>Number formatting control</td>
</tr>
<tr>
<td>+DRAIN1</td>
<td>ALL</td>
<td>What to DRAIN with the initial drain ALL</td>
</tr>
<tr>
<td>+DSCRL</td>
<td>000100,100</td>
<td>Controls retry of DB2 commands (mmss\text{th},count)</td>
</tr>
<tr>
<td>+DSNUTILB</td>
<td>YES</td>
<td>Whether NGT Reorg calls DSNUTILB</td>
</tr>
<tr>
<td>+EDPROCSUP</td>
<td>NO</td>
<td>Whether to call Editprocs in supervisor mode</td>
</tr>
<tr>
<td>+FASTSWITCH</td>
<td>YES</td>
<td>Use faster shadow data set renaming? YES</td>
</tr>
<tr>
<td>+FLASHCOPY</td>
<td>No default</td>
<td>Preferred/Required if requested</td>
</tr>
<tr>
<td>+HRECALL</td>
<td>NO</td>
<td>Recall migrated DB2 data sets? YES</td>
</tr>
<tr>
<td>+IGNOBJSETPARTS</td>
<td>NO</td>
<td>Whether the NGT utility disregards the partition numbers defined in the object set</td>
</tr>
<tr>
<td>Option</td>
<td>Default value</td>
<td>Use and value</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>+JOURNAL</td>
<td>NO</td>
<td>Create the journal file associated with NGT utility processing.</td>
</tr>
<tr>
<td>+LARGEOBJSIZE</td>
<td>1048576</td>
<td>Run large objects in separate server</td>
</tr>
<tr>
<td>+LOGPDS</td>
<td>N/A</td>
<td>Defines the partitioned data set that stores diagnostic information in case a NGT utility abends.</td>
</tr>
<tr>
<td>+MAXCONNECT</td>
<td>32</td>
<td>Maximum number of CDBSS Connections</td>
</tr>
<tr>
<td>+MAXIDLE</td>
<td>10</td>
<td>Number of minutes a server will be idle before shutting down</td>
</tr>
<tr>
<td>+MAXMEMORY</td>
<td>5000</td>
<td>Maximum number of 1M blocks to allocate</td>
</tr>
<tr>
<td>+MAXSERVERS</td>
<td>8</td>
<td>Maximum number of disk and tape servers that jobs might spawn. Set at configuration time to limit users.</td>
</tr>
<tr>
<td>+MAXWAIT</td>
<td>10</td>
<td>Maximum time, in minutes, that processing waits before shutting down if no servers are available (1-60)</td>
</tr>
<tr>
<td>+MEMORYLIMIT</td>
<td>0</td>
<td>Maximum value for the +MAXMEMORY parameter</td>
</tr>
<tr>
<td>+MONITORRATE</td>
<td>10</td>
<td>Frequency of communication between Master and Server</td>
</tr>
<tr>
<td>+NGTAUTO1</td>
<td>N/A</td>
<td>Defines the name of the first NGT Automation Routine data set</td>
</tr>
<tr>
<td>+NGTAUTO2</td>
<td>N/A</td>
<td>Defines the name of the second NGT Automation Routine data set</td>
</tr>
<tr>
<td>+NGTCKPT</td>
<td>N.A</td>
<td>Defines the name of the NGT Checkpoint data set</td>
</tr>
<tr>
<td>+NGTSVR</td>
<td>N/A</td>
<td>Defines the name of the NGT Server JCL data set</td>
</tr>
<tr>
<td>+OBJESETSTOGROUP</td>
<td>N/A</td>
<td>Defines the name of the storage group used by the NGT Utility to store temporary objects</td>
</tr>
<tr>
<td>+ONERROR</td>
<td>ABEND</td>
<td>Action if a utility fails: ABEND</td>
</tr>
<tr>
<td>+OUTPUTAGE</td>
<td>5</td>
<td>How long to retain VSAM output, in days (1-365)</td>
</tr>
<tr>
<td>+OVERRIDEOUTPUT</td>
<td>NO</td>
<td>Determines whether NGT Automation Routines override OUTPUT command values.</td>
</tr>
<tr>
<td>+PARMLIST</td>
<td>YES</td>
<td>Print all parameters? YES</td>
</tr>
<tr>
<td>+PLAN</td>
<td>NGTPLAN</td>
<td>Plan name for DB2 access</td>
</tr>
<tr>
<td>+PSWDWTO</td>
<td>NO</td>
<td>Write password expiration messages to Operator Console</td>
</tr>
<tr>
<td>+QRETRY</td>
<td>002000,10</td>
<td>Controls retry of QUIESCE (*msssth,count)</td>
</tr>
<tr>
<td>+RACFID</td>
<td>None</td>
<td>User defined RACF ID</td>
</tr>
<tr>
<td>+RERUN</td>
<td>0</td>
<td>Restart Control (-n</td>
</tr>
<tr>
<td>+REXXBUFR</td>
<td>200</td>
<td>Buffer space for REXX variables in MB</td>
</tr>
<tr>
<td>Option</td>
<td>Default value</td>
<td>Use and value</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>+REXXTIME</td>
<td>10</td>
<td>Time limit for automation control points, in CPU seconds (1-32767)</td>
</tr>
<tr>
<td>+SQLID</td>
<td>None</td>
<td>Default SQL ID</td>
</tr>
<tr>
<td>+SVRDISK</td>
<td>2</td>
<td>Maximum number of disk servers to start? (0-32)</td>
</tr>
<tr>
<td>+SVRDISLKJC</td>
<td>Installation value</td>
<td>Disk Server Job Class</td>
</tr>
<tr>
<td>+SVRJCLDISK</td>
<td>Installation value</td>
<td>NGTSVR member name for disk servers</td>
</tr>
<tr>
<td>+SVRJCLJOB</td>
<td>Installation value</td>
<td>NGTSVR member name for JOB statement</td>
</tr>
<tr>
<td>+SVRJCLTAPE</td>
<td>Installation value</td>
<td>NGTSVR member name for tape servers</td>
</tr>
<tr>
<td>+SVRMODE</td>
<td>E</td>
<td>Internal, External, or Mixed Servers</td>
</tr>
<tr>
<td>+SVRPROCDC</td>
<td>Installation value</td>
<td>Name of PROCLIB member for disk tasks</td>
</tr>
<tr>
<td>+SVRPROCMSGCLAS</td>
<td>Installation value</td>
<td>Override the message class of a started task</td>
</tr>
<tr>
<td>+SVRPROCMSG_LVL</td>
<td>Installation value</td>
<td>Override the message level of a started task</td>
</tr>
<tr>
<td>+SVRPROCT</td>
<td>Installation value</td>
<td>Name of PROCLIB member for tape tasks</td>
</tr>
<tr>
<td>+SVRSPECIAL</td>
<td>Installation value</td>
<td>Use a special dedicated tape server for long processing runs? YES</td>
</tr>
<tr>
<td>+SVRTAPE</td>
<td>Installation value</td>
<td>Number of tape servers to start for server processing</td>
</tr>
<tr>
<td>+SVRTAPEJC</td>
<td>Installation value</td>
<td>Tape Server Job Class</td>
</tr>
<tr>
<td>+SVRTYPE</td>
<td>Job</td>
<td>Type of processing, batch job or started task? (JOB</td>
</tr>
<tr>
<td>+SYSINERROR</td>
<td>Abend</td>
<td>SYSIN Error Processing Control</td>
</tr>
<tr>
<td>+SYSINORDER</td>
<td>RANDOM</td>
<td>Whether to process SYSIN in order specified</td>
</tr>
<tr>
<td>+TIMEEXEC</td>
<td>NO</td>
<td>Whether to call XSUTTIME a second time</td>
</tr>
<tr>
<td>+UIDHISTORY</td>
<td>5</td>
<td>Days to keep UID History in Checkpoint for Display</td>
</tr>
<tr>
<td>+USER1()</td>
<td>N/A</td>
<td>User variable</td>
</tr>
<tr>
<td>+USER2()</td>
<td>N/A</td>
<td>User variable</td>
</tr>
<tr>
<td>+WAITTIMEOUT</td>
<td>ERROR</td>
<td>Cleanup setting for objects timed out by another NGT utility</td>
</tr>
<tr>
<td>+WORKMODE</td>
<td>TEMP</td>
<td>Permanent Work Dataset Control</td>
</tr>
<tr>
<td>+WORKPREFIX</td>
<td>Installation value</td>
<td>DSName prefix for temporary files</td>
</tr>
<tr>
<td>+XBMID</td>
<td>N/A</td>
<td>Specifies the XBM subsystem ID</td>
</tr>
</tbody>
</table>
## Parameters grouped by function

The following table groups the global utility parameters by logical function.

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System parameters</strong></td>
<td>+ALLERROR sets the behavior of SYSERROR.</td>
</tr>
<tr>
<td></td>
<td>+CONTROLPREFIX specifies a separate DSName prefix for the INFO and OUT files.</td>
</tr>
<tr>
<td></td>
<td>+EDPROCSUP whether to call Editprocs in supervisor mode.</td>
</tr>
<tr>
<td></td>
<td>+HRECALL specifies whether to recall migrated DB2 datasets.</td>
</tr>
<tr>
<td></td>
<td>+MAXCONNECT specifies the maximum number of CDBSS connections.</td>
</tr>
<tr>
<td></td>
<td>+MAXMEMORY specifies the maximum memory to allocate.</td>
</tr>
<tr>
<td></td>
<td>+MONITORRATE controls the frequency of checks by the master on any server jobs running.</td>
</tr>
<tr>
<td></td>
<td>+OBJSETSTOGROUP specifies the storage group used by a NGT utility to store temporary objects</td>
</tr>
<tr>
<td></td>
<td>+PSWDWTO controls whether password expiration messages are also written to the console.</td>
</tr>
<tr>
<td></td>
<td>+RACFID specifies a user defined RACF ID.</td>
</tr>
<tr>
<td></td>
<td>+REXXBUFR specifies the MB of memory to use for REXX variables.</td>
</tr>
<tr>
<td></td>
<td>+REXXTIME specifies the time limit for REXX automation routines.</td>
</tr>
<tr>
<td></td>
<td>+SQLID specifies a user defined default SQL ID.</td>
</tr>
<tr>
<td></td>
<td>+SVRSPECIAL specifies whether processing should use a special, dedicated server for producing image copies.</td>
</tr>
<tr>
<td></td>
<td>+WORKMODE specifies whether to use permanent or temporary work data sets.</td>
</tr>
<tr>
<td></td>
<td>+WORKPREFIX specifies DSName prefix for temporary files.</td>
</tr>
<tr>
<td>Function</td>
<td>Parameter</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DB2 Parameters</td>
<td>+COPYPENDING specifies whether or not Copy Pending should be set when appropriate. Used by NGT Load.</td>
</tr>
<tr>
<td></td>
<td>+DSCRL is used when issuing DB2 commands to set space access. When a -STOP command is issued, -DISPLAY commands must be issued to verify the status of the table space or index. +DSCRL( time,count ) controls the time interval between -DISPLAY commands and the number of retries before giving up.</td>
</tr>
<tr>
<td></td>
<td>+PLAN specifies the plan name used to interface to DB2. The installation of NGT utilities builds a BIND for this plan.</td>
</tr>
<tr>
<td></td>
<td>+QRETRY is like +DSCRL(time,count ) except that it controls re-execution of the DB2 QUIESCE utility if that utility fails to complete successfully.</td>
</tr>
<tr>
<td>Function</td>
<td>Parameter</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NGT utility control</td>
<td>This group of parameters controls the execution of NGT Utilities.</td>
</tr>
<tr>
<td></td>
<td>+AGELIMIT specifies the maximum number of days the +OUTPUTAGE parameter can be set to in a job.</td>
</tr>
<tr>
<td></td>
<td>+CHECKDEFERRED specifies whether to pre-check and skip DEFINE NO objects.</td>
</tr>
<tr>
<td></td>
<td>+CLEANUP specifies whether to cleanup work datasets not needed for restart after a failure.</td>
</tr>
<tr>
<td></td>
<td>+DEADLINE limits the number of minutes allowed for the Change Apply phase of NGT Reorg.</td>
</tr>
<tr>
<td></td>
<td>+DELETEOUTPUT specifies whether to delete all temp datasets and Output immediately when RC=0.</td>
</tr>
<tr>
<td></td>
<td>+DELETEWORK specifies whether or not temporary work datasets are deleted at the end of utility processing.</td>
</tr>
<tr>
<td></td>
<td>+DIGITS controls the number formatting.</td>
</tr>
<tr>
<td></td>
<td>+DRAIN1 specifies whether the initial drain is to Drain Writers or Drain All.</td>
</tr>
<tr>
<td></td>
<td>+DSCRL parameter verifies if the function completed after a -STOP or -START command was issued for an object.</td>
</tr>
<tr>
<td></td>
<td>+DSNUTILB parameter defines whether NGT Reorg calls DSNUTILB</td>
</tr>
<tr>
<td></td>
<td>+FASTSWITCH specifies to use faster-performance name switching for work shadow datasets.</td>
</tr>
<tr>
<td></td>
<td>+FLASHCOPY parameter specifies whether FlashCopy is preferred or required.</td>
</tr>
<tr>
<td></td>
<td>+IGNOBJSETPARTS parameter specifies whether the NGT utility disregards the partition numbers defined in the object set.</td>
</tr>
<tr>
<td></td>
<td>+JOURNAL specifies whether to include the full NGT output in the master job.</td>
</tr>
<tr>
<td></td>
<td>+LARGEOBJSIZE specifies the minimum object size for processing in its own dedicated server.</td>
</tr>
<tr>
<td></td>
<td>+MAXCONNECT specifies the maximum number of CDBSS connections.</td>
</tr>
<tr>
<td></td>
<td>+MAXMEMORY specifies the maximum number of 1M memory blocks to allocate.</td>
</tr>
<tr>
<td></td>
<td>+MEMORYLIMIT specifies the maximum value that can be specified for +MAXMEMORY.</td>
</tr>
<tr>
<td></td>
<td>+NGTAUTO1 and +NGTAUTO2 parameters override the NGT Automation routine data set name contained in the product configuration load module.</td>
</tr>
<tr>
<td></td>
<td>+NGTCKPPT parameter overrides the NGT Checkpoint data set.</td>
</tr>
<tr>
<td>Function</td>
<td>Parameter</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>NGT utility control (continued)</td>
<td>+OBJSETSTOGROUP defines the storage group used by a NGT utility to store temporary objects.</td>
</tr>
<tr>
<td></td>
<td>+ONERROR specifies action to perform if an individual utility fails to complete successfully.</td>
</tr>
<tr>
<td></td>
<td>+OUTPUTAGE specifies how long to keep output for a specific job.</td>
</tr>
<tr>
<td></td>
<td>+OVERRIDEOUTPUT specifies whether NGT Automation routines override OUTPUT command values.</td>
</tr>
<tr>
<td></td>
<td>+PARMLIST controls the listing of the input parameter statements.</td>
</tr>
<tr>
<td></td>
<td>+PLAN specifies the name of the DB2 plan to be used by NGT utilities.</td>
</tr>
<tr>
<td></td>
<td>+QRETRY specifies how long a NGT utility can hold a drain and how many times a drain can be requested</td>
</tr>
<tr>
<td></td>
<td>+RERUN parameter specifies what NGT utilities do on a restart attempt.</td>
</tr>
<tr>
<td></td>
<td>+SYSINERROR determines the action to take when a syntax error is detected in the SYSIN section of a utilities job.</td>
</tr>
<tr>
<td></td>
<td>+SYSINORDER specifies whether to process the SYSIN in the order specified or randomly.</td>
</tr>
<tr>
<td></td>
<td>+TIMEEXEC specifies whether to send control to the XSUTTIME automation routine a second time.</td>
</tr>
<tr>
<td></td>
<td>+UIDHISTORY specifies the number of days to keep the UID history in the checkpoint dataset for display.</td>
</tr>
<tr>
<td></td>
<td>+WAITTIMEOUT specifies whether to error or exclude objects that are timed out by another NGT utility.</td>
</tr>
<tr>
<td></td>
<td>+XBMID specifies the XBM subsystem ID used by NGT Copy when it is invoked by NGT Load.</td>
</tr>
<tr>
<td>Function</td>
<td>Parameter</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NGT server control</td>
<td>+MAXIDLE specifies the amount of time a server remains idle before shutting down.</td>
</tr>
<tr>
<td></td>
<td>+MAXSERVERS specifies the maximum number of disk and tape servers that processing may use.</td>
</tr>
<tr>
<td></td>
<td>+MAXWAIT specifies the maximum wait time when servers aren't immediately available.</td>
</tr>
<tr>
<td></td>
<td>+NGTsvr parameter overrides the name of the NGT server JCL data set.</td>
</tr>
<tr>
<td></td>
<td>+SVRDISK specifies the number of disk (or DASD) servers to start for server processing.</td>
</tr>
<tr>
<td></td>
<td>+SVRDISKJC specifies the job class to use for disk servers.</td>
</tr>
<tr>
<td></td>
<td>+SVRMODE specifies whether disk servers will be separate address spaces or within the submitted master job.</td>
</tr>
<tr>
<td></td>
<td>+SVRTAPE specifies the number of tape servers to start for server processing.</td>
</tr>
<tr>
<td></td>
<td>+SVRTAPEJC specifies the job class to use for tape servers.</td>
</tr>
<tr>
<td></td>
<td>+SVRTYPE specifies the type of job processing used, either a batch job or a started task.</td>
</tr>
<tr>
<td>All Servers</td>
<td></td>
</tr>
<tr>
<td>Batch Servers Only</td>
<td>+SVRjcldisk specifies the job name stem for disk (or DASD) server job JCL.</td>
</tr>
<tr>
<td></td>
<td>+SVRjcljob specifies the DD name associated with the server JCL’s JOB card.</td>
</tr>
<tr>
<td></td>
<td>+SVRjcltape specifies the job name stem for tape server job JCL.</td>
</tr>
<tr>
<td>Started Task Servers Only</td>
<td>+SVRprocd specifies the name of the PROCLIB member for disk (or DASD) processing.</td>
</tr>
<tr>
<td></td>
<td>+SVRprocmmsgclas overrides the message class of started task servers.</td>
</tr>
<tr>
<td></td>
<td>+SVRprocmmsglvl overrides the message class of started task servers.</td>
</tr>
<tr>
<td></td>
<td>+SVRproct specifies the name of the PROCLIB member for tape processing.</td>
</tr>
</tbody>
</table>
Global utility alphabetical parameter reference

This section lists the global utility parameters in alphabetical order.

**+AGELIMIT**

The +AGELIMIT parameter sets a cap on how long users can request their output be kept. This value may only be set at installation time.

The values are defined as follows:

30

*(default)* Values specified for the +OUTPUTAGE parameter may not exceed 30 days.

numberDays

A number of days ranging from 1 to 365.

**Example**

In the following example, values set for the +OUTPUTAGE parameter cannot exceed the set limit of 90 days.

+AGELIMIT(90)

**+ALLERROR**

The +ALLERROR parameter sets the behavior of SYSERROR.

The values are defined as follows:
YES

(syserror) SYSERROR shows all the error messages from all restarts since the UID began (first NORESTART run).

NO

SYSERROR shows only the error messages for the current job.

**+CHECKDEFERRED**

The +CHECKDEFERRED parameter indicates whether NGT should check for and ignore table spaces that are deferred upfront during SYSIN processing.

NO

Do not check for deferred table spaces.

YES

(default) Check for deferred table spaces.
You must use the full values YES or NO, you cannot abbreviate these values to Y or N.

**Example**

`+CHECKDEFERRED(NO)`

The utility processing does not check for deferred table spaces.

**+CLEANUP**

The +CLEANUP parameter is used to determine the default cleanup action for an object if the XSVRXERR automation control point is not in use. It also determines the default action if the XSVRXERR automation control point is in use, in other circumstances.

The values are defined as follows:

**NO**

*(default)* The object is processed as if ERROR was selected.

**YES**

The object is processed as if CLEANUP was selected.

+CLEANUP and +DELETEOUTPUT may have similar sounding names but their functions are different and which occurs depends on whether the outcome of a job was successful or not. +CLEANUP is for jobs that fail: when an object is placed on the SKIPPED portion of the ERROR queue (see the section on XSVRXERR in the *BMC Next Generation Technology Automation Reference Manual*, specifically the action variable, for details).

+DELETEOUTPUT, by comparison, is for jobs that have run successfully: it controls whether work data sets are deleted at the successful end of a job (that is, when it is marked COMPLETE). For more information, see “+DELETEOUTPUT” on page 161.
Example

+CLEANUP(NO)

If an error occurs, and the XSVRXERR automation control point is not in use, processing will behave as if ERROR was selected, meaning the object will not be removed from the matrix.

Example

+CLEANUP(YES)

If an error occurs, and the XSVRXERR automation control point is not in use, processing will behave as if CLEANUP was selected, meaning the object will be removed from the matrix (unless the object is in must-complete status).

+CONTROLPREFIX

The +CONTROLPREFIX parameter allocates INFO and OUTPUT data sets with the provided prefix string, instead of using the prefix value specified by +WORKPREFIX. This allows these two data sets to be in a different SMS pool since they will stay on the system up to the number of days specified with the parameter +OUTPUTAGE while the other data sets will be deleted upon successful completion of the job.

You can set the value of this parameter by using the XSUTGLOB NGT Automation control point. For more information, see the BMC Next Generation Technology Automation Reference Manual.

+CONTROLPREFIX( HLQ )

The value is defined as follows:

HLQ

Any valid MVS high-level qualifier or supported variable. You can use any of the following symbolic variables:
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;SSID</td>
<td>The DB2 subsystem name or group name specified on the PARM=name attribute</td>
</tr>
<tr>
<td>&amp;VER</td>
<td>The version (for example 1210) of the NGT utility being executed</td>
</tr>
<tr>
<td>&amp;VER3</td>
<td>The first three characters of the version (for example 121) of the NGT utility being executed</td>
</tr>
<tr>
<td>&amp;VER2</td>
<td>The first two characters of the version (for example 12) of the NGT utility being executed.</td>
</tr>
<tr>
<td>&amp;USER1</td>
<td>The value of the +USER1(userVariable) parameter</td>
</tr>
<tr>
<td>&amp;USER2</td>
<td>The value of the +USER2(userVariable) parameter</td>
</tr>
</tbody>
</table>

The HLQ is appended to a unique timestamp-based identifier, and the data set suffixes.

**Example**

In the following example, the INFO and OUTPUT files will be allocated with data set names beginning with ADMIN.NGT(tttttttt.uuuuuuuu represents the unique timestamp-based identifier):

- ADMIN74.NGTADMIN.ttttttt.uuuuuuuu.INFO001 (information data set for the CDBI Console)
- ADMIN74.NGTADMIN.ttttttt.uuuuuuuu.OUT001 (output data set for the CDBI Console)

**+COPYPENDING**

The +COPYPENDING parameter specifies whether to set the DB2 COPY PENDING status on a table space in circumstances when it would normally be set, e.g., after a LOAD, REPART, or RESTORE. This does not affect messages or return codes.

The values are defined as follows:
YES

*(default)* Set the COPY PENDING status.

NO

Does not set COPY PENDING. Use after a LOAD of a Read-Only table space.

**Example**

In the following example, COPY PENDING is set after a LOAD, REPART, or RESTORE.

```
+COPYPENDING(YES)
```

**+DEADLINE**

The +DEADLINE parameter specifies the number of minutes allowed for the Change Apply phase of NGT Reorg.

You can set the value of this parameter by using the XSUTGLOB NGT Automation control point. For more information, see the *BMC Next Generation Technology Automation Reference Manual*.

**Note**

+DEADLINE is used *only* with NGT Reorg.

The values are defined as follows:

0

*(default)* A value of 0 indicates no deadline is to be used.

*numberMinutes*

A value from 0 to 255, indicating the number of minutes.
If the deadline is reached without Change Apply completing, processing issues message NGTR859 CHANGE APPLY DEADLINE LIMIT OF \( xxx \) MINUTES REACHED and the job terminates.

**Example**

In the following example, NGT Reorg terminates processing if Change Apply cannot complete in five minutes.

```
+DEADLINE(5)
```

**+DELETEOUTPUT**

The +DELETEOUTPUT parameter value is checked when a job completes and is marked as COMPLETE. If it is equal to YES, then all server files (WK, INFO, OUT, and product), the Status IDB, and the Utility Entry are deleted. Message NGTB268 is written to CDBPRINT indicating this has occurred. Issues the user may have concerning a job with this parameter set to YES may have to re-run the job since the Journal output will have been purged.

The values are defined as follows:

**NO**

*(default)* Even after successful completion, all output data sets will be kept.

**YES**

At successful completion, all output data sets will be deleted.

**Example**

In the following example, output files are deleted at the completion of a job.

```
+DELETEOUTPUT(YES)
```
+DELETEWORK

The +DELETEWORK parameter controls the disposition of utility work data sets. These include //SYSUT1 and the +TAPEDD data sets. These data sets must be allocated and catalogued, since they are critical to restart of the utility in case of failure. However, when the utility completes successfully, these datasets are of no further use and can be deleted.

The values are defined as follows:

YES

*(default)* At successful completion, all work data sets will be uncatalogued and deleted.

NO

Even after successful completion, the work data sets will be kept.

**Example**

In the following example, work files are deleted at termination.

```
+DELETEWORK(YES)
```

+DIGITS

The +DIGITS parameter specifies how numerical columns are to be handled. The DIGITS keyword, specified in //UTLPARMS, enforces compatibility between certain DB2 SQL function and NGT SQL functions.
The values are defined as follows:

**CDB**

*(default)* Use NGT format, see below.

**IBM**

Use IBM format.

The **CHAR( )** and **DIGITS( )** functions as documented in the WHERE clause guide differ in output from the standard DB2 SQL functions. To enforce compatibility between NGT and DB2 version of the utility, these options have been introduced to automatically change the output of those functions to a format compatible with DB2.

Under NGT own implementation, the **DIGITS** function generates an odd number of digits by taking the next highest odd precision of its argument, if even. For example, these numerics exhibit the following behaviors with **+DIGITS(CDB)**:

- An **INTEGER** generates 11 digits
- A **SMALLINT** generates 5 digits
- A **DECIMAL** generates **P+N** digits where **P** is the precision of the argument and **n=0**
- If **p** is odd or **n=1** otherwise

Under the DB2 implementation or **+DIGITS(IBM)**, numeric columns are generated as follows:

- An **INTEGER** generates 10 digits
- A **SMALLINT** generates 5 digits
- A **DECIMAL** generates **P** digits where **P** is the precision of the decimal
**WARNING**

If you are an existing customer of NGT utilities and have used +DIGITS() in any of your job streams, then *DO NOT* change the default. Changing the default may cause your existing jobs to fail or produce incorrect results. Also, coding the +DIGITS() option affects *all* products that use the WHERE clause processor.

---

**Example**

In the following example, the NGT approach is used to handle numeric values for this utility job.

```plaintext
+DIGITS(CDB)
```

---

**+DRAIN1**

The +DRAIN1 parameter allows you to control how drains are issued by processing.

The +DRAIN1 parameter affects both the initial drain and the final drain, and the final drain will be an iteration of drains if needed.

The initial drain will be writers only or both readers and writers based on +DRAIN1.

The second drain, or series of drains will either:

- +DRAIN(ALL) Drain both readers and writers together each time.
- +DRAIN(WTR) Drain WTR only and when change apply completes, append a drain RDR to complete the process.

By specifying +DRAIN(WTR) we must append a drain RDR to the existing drain WTRs. With some application SQL this can cause a deadlock where the reader drain can never be obtained. This condition is addressed by +DRAIN(ALL). With ALL we must drain both for each iteration because we don't know which iteration will be the last one. So +DRAIN(WTR) is less intrusive but may fail at the end, +DRAIN(ALL) will succeed as long as the object is drainable.
The values are defined as follows:

**ALL**

*(default)* Issues DRAIN(ALL) commands. Processing waits on *all* (both reader and writer) claims.

**WTR**

Issues DRAIN(WTR) commands. Processing waits on *reader* claims only.

Drain writers only until the last iteration. Then change apply would drain readers while holding the writers' drain for the previous iteration.

**+DSCRL**

The +DSCRL parameter is used to verify that, after a -STOP or -START command has been issued for an object, the function completed.

+DSCRL determines how often and how many times the status should be checked. Before a NGT utility starts a function, it issues DB2 -STOP and -START commands to put the table space in a state suitable for the Utility. If, by the end of the time limit specified in DSCRL, the table space is not in the desired state, an error message is issued and the Utility fails.

When a command is issued (e.g., -STOP DATABASE(xxx) SPACENAM(yyy)) DB2 schedules the request and then acknowledges that the command has been scheduled. A DB2 acknowledgement means only that the command was scheduled, not that it has completed. If another program holds a lock on the table space, the table space will go into a STOP PENDING status which could last for seconds, minutes, or even hours if it is a batch program which holds the lock.

After the -STOP command is issued, the NGT utility issues -DISPLAY commands until DB2 shows the desired state or until the time specified in DSCRL expires, whichever comes first.
**timeAttempts**

Time between display commands where the following subfields exist (default = 000100):

- MM: Minutes
- SS: Seconds
- T: Tenths of a second
- H: Hundredths of a second

**numAttempts**

Maximum number of display attempts (default = 100)

If the number of retries is exceeded, a message is issued and the NGT utility terminates with return code 8.

**Example**

In the following example, the first operand tells the NGT utility to retry every second. The second operand specifies a maximum of 120 retries. This combination of parameters would make the NGT utility wait for up to two minutes before terminating the NGT utility on this object.

```
+DSCRL(000100,120)
```

**+DSNUTILB**

The +DSNUTILB parameter defines whether NGT Reorg calls DSNUTILB. DSNUTILB enables NGT Reorg to process those object that it cannot natively process.

```
+DSNUTILB(YES
               NO
)
```

The values are defined as follows:
YES

NGT REORG calls DSNUTILB to process an object that NGT Reorg does not natively support.

NO

NGT REORG does not call DSNUTILB when it encounters an unsupported object, and the object is not processed.

**+EDPROCSUP**

The +EDPROCSUP parameter specifies whether to call Editprocs in supervisor state. Normally, this is not required and performance is enhanced if the value is NO. However, some Editprocs require running in Supervisor State, so the parameter must be set to YES. This parameter can also be used to skip all Editproc routine processing. This should only be done if the Editprocs do not change the DB2 data in any way or damaged data could result.

The values are defined as follows:

YES

Call Editprocs in supervisor state.

NO

*(default)* Call Editprocs in program state. This uses significantly less CPU time.

**+FASTSWITCH**

The +FASTSWITCH parameter indicates whether to use I/J name switching instead of the A/B name switching previously used by NGT utilities.

If you are rebalancing partitions in NGT Reorg, +FASTSWITCH must be YES.
The values are defined as follows:

**NO**

Do not use Fast Switch.

**YES**

*(default)* Use Fast Switch.

+FASTSWITCH(YES) updates the DB2 catalog and DB2 directory to reflect the current I/J name. The utility does not perform any IDCAMS RENAMES.

---

**+FLASHCOPY**

The +FLASHCOPY parameter specifies whether FlashCopy is preferred or required.

The values are defined as follows:

**REQUIRED**

Specifies that the copy to be taken with Reorg must be a FlashCopy.

If the volumes do not support FlashCopy, the reorganization will fail.
**PREFERRED**

Specifies that the copy to be taken with Reorg is be a FlashCopy if the volumes support FlashCopy, otherwise use standard copy.

If the volumes do not support FlashCopy, a regular copy will be made with the reorganization.

**+HRECALL**

The +HRECALL parameter indicates whether or not migrated VSAM data sets are to be automatically recalled.

![Diagram showing +HRECALL parameter with values NO and YES]

The values are defined as follows:

**NO**

*(default)* Do not recall migrated volumes.

This causes the utility to stop processing unless +ONERROR(CONTINUE) is specified.

**Note**

Processing will issue a message and continue even if +ONERROR(CONTINUE) is not coded.

**YES**

Recall migrated volumes. This may increase the utility's elapsed time.

**+IGNOBJSETPARTS**

The +IGNOBJSETPARTS parameter is used to determine whether the NGT utility disregards the partition numbers defined in the object set.
The values are defined as follows:

**No**

*(default)* The NGT utility only processes the partitions specified in the object set.

**Yes**

The NGT utility processes *all* partitions in the object set, disregarding any partition definitions.

### +JOURNAL

The +JOURNAL parameter controls the creation of the journal file associated with the NGT utility processing.

*Note*

The journal contains trace messages for the exclusive use of BMC Customer Support. Many of these messages are not documented. This is by design and does not signify a problem with the software or the manual.

The values are defined as follows:

**NO**

*(default)* No journal prints.
YES

Journal prints to JRNLOUT DD. This DD may be defined to SYSOUT or to a static data set. If a static data set is used, the DCB must be specified with the following attributes: RECFM=FBA, LRECL=133, and the BLKSIZE should be set to 0 in order that the system will set a default value based on the DASD type.

ONERROR

If an error occurs, and the utility is left in a restartable state, send journal output to the JRNLOUT DD.

Examples

In the following example, a journal is sent to JRNLOUT.

```
//JRNLOUT DD SYSOUT=*  
//UTLPARMS DD *  
+JOURNAL(YES)
```

In the following example, a journal is sent to the data set USER.NGT.JOURNAL

```
//JRNLOUT DD DISP=SHR,DSN=USER.NGT.JOURNAL 
//UTLPARMS DD *  
+JOURNAL(YES)
```

In the following example, a journal is sent to the data set USER.NGT.JOURNAL and the utility ID ACCTG2009 if an error occurs and the utility is marked restartable.

```
//NGT1233 EXEC PGM=NGTUTIL,PARM='DB2T,ACCTG2009,RESTART' 
//JRNLOUT DD DISP=SHR,DSN=USER.NGT.JOURNAL 
//UTLPARMS DD *  
+JOURNAL(ONERROR)
```

+LARGEOBJSIZE

The +LARGEOBJSIZE parameter specifies the number of 4K pages in an object. When processing begins, if the size of an object exceeds this size, that object will be processed by itself rather than in parallel with other objects.
The values are defined as follows:

8589934592

*(default)*

Objects exceeding 8589934592 4K pages (32 GB) are processed individually.

**maxPages**

Threshold for the number of pages in an object; objects exceeding this number of pages will be processed individually. Minimum valid value is 1. Maximum valid value is 17179869184 4K pages (64 GB).

**Example**

In the following example, any object 2048000 4K pages or larger will be processed by itself.

+LARGEOBJSIZE(2048000)

**+LOGPDS**

The +LOGPDS parameter overrides the log partitioned data set name contained in the product configuration load module set, and lets you define the partitioned data set that stores diagnostic information in case a NGT utility abends.

During NGT utility execution, the data set name is allocated to the LOGPDS DD. If the +LOGPDS parameter contains variables, these are resolved prior to the allocation.

The *dataSetName* can contain the following variables:
Variable | Description
--- | ---
&SSID | The DB2 subsystem name or group name specified on the PARM= name attribute
&VER | The version (for example 1210) of the NGT utility being executed
&VER3 | The first three characters of the version (for example 121) of the NGT utility being executed
&VER2 | The first two characters of the version (for example 12) of the NGT utility being executed.
&USER1 | The value of the +USER1(userVariable) parameter
&USER2 | The value of the +USER2(userVariable) parameter

**+MAXCONNECT**

The +MAXCONNECT parameter specifies the maximum number of connections to the NGT Subsystem (CDBSS) that the master can start.

You can set the value of this parameter by using the XSUTGLOB NGT Automation control point. For more information, see the *BMC Next Generation Technology Automation Reference Manual*.

The values are defined as follows:

- **Note**
  
  +MAXCONNECT has no association with or effect on DB2 threads.

- **32**
  
  *(default)* Processing will use a maximum of 32 connections to the CDBSS process.
**maxNumberConnections**

An integer representing the maximum number of connections to the CDBSS process. The maximum value you may specify is 128.

**Example**

In the following example, processing will allow a maximum of 32 connections to the NGT Subsystem.

```
+MAXCONNECT(32)
```

**+MAXIDLE**

The +MAXIDLE parameter specifies the amount of time a server remains idle before shutting down.

When a server job is idle for the specified number of minutes, the master will instruct it to shut down. This parameter can be useful for a job that uses many servers, and only one of those servers is used towards the end of processing. The unnecessary server jobs end, which frees up their initiators.

The values are defined as follows:

10

*(default)* Server will remain idle for 10 minutes before shutting down. Job will continue its work.

**maxIdleTime**

An integer representing the amount of time in minutes a server will remain idle before shutting down.
Example

In the following example, processing will allow a maximum server to remain idle for 20 minutes before shutting the server down because it is not being used.

+MAXIDLE(20)

+MAXMEMORY

The +MAXMEMORY parameter specifies the number of 1M blocks of memory to allocate for processing by each server. If insufficient memory exists for processing, the message WAITING FOR MEMORY is issued to CDBPRINT until the condition is resolved.

You can set the value of this parameter by using the XSUTGLOB NGT Automation control point. For more information, see the BMC Next Generation Technology Automation Reference Manual.

The values are defined as follows:

**numberMemBlocks**

An integer from 100 to 65535 representing the number of 1M blocks (default is 5000, which allocates 5G of memory).

For prior NGT installations configured when a minimum value of 2 was valid, the configuration job will need to be rerun. If a value for +MAXMEMORY is not explicitly specified at that point, the new default value will be 100.

Example

In the following example, processing will allocate 2048 x 1M (or 2 G) of memory for Unload processing.

+MAXMEMORY(2048)
**+MAXSERVERS**

The +MAXSERVERS parameter specifies the maximum number of disk and tape servers for processing. This parameter is set at configuration time and may be used to prevent other users from utilizing excessive server jobs.

\[ +\text{MAXSERVERS}(\text{maxDisk},\text{maxTape}) \]

*maxDisk*

Maximum number of disk servers. Maximum valid value is 32 (default value is 8).

*maxTape*

Maximum number of tape servers. Maximum valid value is 32 (default value is 2).

**Examples**

In the following example, jobs may not use more than 10 disk servers and 5 tape servers.

\[ +\text{MAXSERVERS}(10,5) \]

In the following example, jobs may not use tape servers at all.

\[ +\text{MAXSERVERS}(4,0) \]

**+MAXWAIT**

The +MAXWAIT parameter specifies the maximum wait time, in minutes, before processing shuts down if no servers are available. Lack of available servers may be due to servers in error, or an insufficient number of servers being defined.

You can set the value of this parameter by using the XSUTGLOB NGT Automation control point. For more information, see the *BMC Next Generation Technology Automation Reference Manual*. 
10

(default) Processing will wait a maximum of 10 minutes before shutting down if no servers are available.

**maxWaitTime**

A value, in minutes, from 1 through 60, indicating the maximum time processing will wait before shutting down, if no servers are available.

**Example**

In the following example, processing waits a maximum of 20 minutes if servers are unavailable, otherwise shut down processing.

```
+MAXWAIT(20)
```

**+MEMORYLIMIT**

The +MEMORYLIMIT parameter specifies the maximum value that can be used with the +MAXMEMORY parameter. This was designed to prevent the specification of a +MAXMEMORY parameter value so high as to impede performance.

**Note**

You can only set this parameter during installation.

The values are defined as follows:
**maxMemoryLimit**

A value in 1M blocks (default value is 0).

**Example**

In the following example, the +MAXMEMORY parameter cannot be set higher than 4G for any job.

```
+MEMORYLIMIT(4096)
```

**+MONITORRATE**

The +MONITORRATE parameter controls the frequency of server job checks. The NGT utilities processing routinely queries activity between the master job and any server jobs running.

The values are defined as follows:

10

*(default)* A 10-second delay occurs between server job checks.

**timeDelay**

The time delay between server job checks, in seconds.

**Examples**

In the following example, server jobs are checked every minute (60 seconds).

```
+MONITORRATE(60)
```

In the following example, server jobs are checked every second. This could have significant performance impacts.

```
+MONITORRATE(1)
```
The +NGTAUTO1 parameter overrides the NGT Automation routine data set name contained in the product configuration load module set, and lets you configure a specific NGT Automation routine set for each DB2 subsystem where you install the NGT utilities.

During NGT utility execution, the data set name is allocated to the NGTAUTO DD. If the +NGTAUTO1 parameter contains variables, these are resolved prior to the allocation.

The dataSetName can contain the following variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;SSID</td>
<td>The DB2 subsystem name or group name specified on the PARM=name attribute</td>
</tr>
<tr>
<td>&amp;VER</td>
<td>The version (for example 1210) of the NGT Utilities being executed</td>
</tr>
<tr>
<td>&amp;VER3</td>
<td>The first three characters of the version (for example 121) of the NGT utility being executed</td>
</tr>
<tr>
<td>&amp;VER2</td>
<td>The first two characters of the version (for example 12) of the NGT utility being executed.</td>
</tr>
<tr>
<td>&amp;USER1</td>
<td>The value of the +USER1(userVariable) parameter</td>
</tr>
<tr>
<td>&amp;USER2</td>
<td>The value of the +USER2(userVariable) parameter</td>
</tr>
</tbody>
</table>

The +NGTAUTO2 parameter lets you configure a second NGT Automation routine set for each DB2 subsystem where you install the NGT utilities. The data set specified in +NGTAUTO2 is concatenated behind the data set specified in +NGTAUTO1.
During NGT utility execution, the data set name is allocated to the NGTAUTO DD. If the +NGTAUTO2 parameter contains variables, these are resolved prior to the allocation.

The `dataSetName` can contain the following variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;SSID</td>
<td>The DB2 subsystem name or group name specified on the <code>PARM=name</code> attribute</td>
</tr>
<tr>
<td>&amp;VER</td>
<td>The version (for example 1210) of the NGT Utilities being executed</td>
</tr>
<tr>
<td>&amp;VER3</td>
<td>The first three characters of the version (for example 121) of the NGT utility being executed</td>
</tr>
<tr>
<td>&amp;VER2</td>
<td>The first two characters of the version (for example 12) of the NGT utility being executed.</td>
</tr>
<tr>
<td>&amp;USER1</td>
<td>The value of the +USER1(<code>userVariable</code>) parameter</td>
</tr>
<tr>
<td>&amp;USER2</td>
<td>The value of the +USER2(<code>userVariable</code>) parameter</td>
</tr>
</tbody>
</table>

**+NGTCKPT**

The `+NGTCKPT` parameter overrides the NGT Checkpoint data set name contained in the product configuration load module set, and lets you configure a specific NGT Checkpoint data set for each DB2 subsystem where you install NGT utilities.

During NGT utility execution, the data set name is allocated to the NGTCKPT DD. If the `+NGTCKPT` parameter contains variables, these are resolved prior to the allocation.
The *dataSetName* can contain the following variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;SSID</td>
<td>The DB2 subsystem name or group name specified on the PARM= attribute</td>
</tr>
<tr>
<td>&amp;VER</td>
<td>The version (for example 1210) of the NGT utility being executed</td>
</tr>
<tr>
<td>&amp;VER3</td>
<td>The first three characters of the version (for example 121) of the NGT utility being executed</td>
</tr>
<tr>
<td>&amp;VER2</td>
<td>The first two characters of the version (for example 12) of the NGT utility being executed.</td>
</tr>
<tr>
<td>&amp;USER1</td>
<td>The value of the +USER1(<em>userVariable</em>) parameter</td>
</tr>
<tr>
<td>&amp;USER2</td>
<td>The value of the +USER2(<em>userVariable</em>) parameter</td>
</tr>
</tbody>
</table>

**+NGTSVR**

The +NGTSVR parameter overrides the name of the NGT server JCL data set contained in the product configuration load module set, and lets you configure a specific NGT server JCL data set for each DB2 subsystem where you install the NGT utilities. The NGTSVR data set contains the JCL needed to start the external servers.

During NGT utility execution, the data set name is allocated to the NGTSVR DD. If the +NGTSVR parameter contains variables, these are resolved prior to the allocation.
The `dataSetName` can contain the following variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;SSID</td>
<td>The DB2 subsystem name or group name specified on the PARM=\textit{name} attribute</td>
</tr>
<tr>
<td>&amp;VER</td>
<td>The version (for example \textit{1210}) of the NGT utility being executed</td>
</tr>
<tr>
<td>&amp;VER3</td>
<td>The first three characters of the version (for example \textit{121}) of the NGT utility being executed</td>
</tr>
<tr>
<td>&amp;VER2</td>
<td>The first two characters of the version (for example \textit{12}) of the NGT utility being executed.</td>
</tr>
<tr>
<td>&amp;USER1</td>
<td>The value of the +USER1(\textit{userVariable}) parameter</td>
</tr>
<tr>
<td>&amp;USER2</td>
<td>The value of the +USER2(\textit{userVariable}) parameter</td>
</tr>
</tbody>
</table>

\textbf{+OBJSETSTOGROUP}

The `+OBJSETSTOGROUP` defines the storage group used by a NGT utility to store temporary objects.

```
+OBJSETSTOGROUP (storageGroup)
```

The values are defined as follows:

\textbf{storageGroup}

A character string representing the name of the storage group.

The NGT Utility removes embedded blanks from the \textit{storageGroup} character string. (ABC DE is treated as ABCDE)

\textbf{+ONERROR}

The `+ONERROR` parameter determines the action that an NGT utility takes when an error occurs while attempting to execute a utility.
The values are defined as follows:

**ABEND**

*(default)* If an error is encountered, the task abends with a return code of either 8 or 16, depending on the severity of the problem.

**CONTINUE**

If an error is encountered during processing of this object, the NGT utility issues messages, performs cleanup of the current function and continues processing with the next statement. The final return code will be at least 8.

**STOP**

If an error is encountered, The NGT utility notifies the user and terminates with a condition code of 8 or 16 depending on the severity of the error. It should be noted that cleanup will not be attempted. If cleanup is desired, the utility must be rerun with PARM = QUICKEXIT.

**Example**

In the following example, if the NGT utility encounter an error when processing this object, continue with the next statement.

```
+ONERROR(CONTINUE)
```

**+OUTPUTAGE**

The +OUTPUTAGE parameter specifies how long to retain a VSAM output file. This value cannot exceed the value of +AGELIMIT.
The values are defined as follows:

**numberDays**

An integer from 1 to 365 representing the number of days that VSAM output is retained (default value is 5).

**Examples**

In the following example, VSAM output is retained for a week. Output age is within the 30-day limit set at installation time.

```
+OUTPUTAGE(7) - set in current job
+AGELIMIT(30) - set at installation time
```

In the following example, the output retention period of 20 days exceeds 14-day limit set at installation time. This job would result in an error condition.

```
+OUTPUTAGE(20) - set in current job
+AGELIMIT(14) - set at installation time
```

**+OVERRIDEOUTPUT**

The +OVERRIDEOUTPUT parameter determines whether NGT Automation Routines override OUTPUT command values.

**Figure 19: +OVERRIDEOUTPUT parameter syntax**
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>NGT Automation Routines cannot override OUTPUT command values. If NO is specified, the NGT data set allocation Automation Routines will not be called. (This is the default value)</td>
</tr>
<tr>
<td>YES</td>
<td>NGT Automation Routines can override OUTPUT command values. If YES is specified, the NGT data set allocation Automation Routines will be called and allowed to change allocation values.</td>
</tr>
<tr>
<td>XSDN</td>
<td>NGT Automation Routines can override OUTPUT command values, except for the data set name. If YES is specified, the NGT data set allocation Automation Routines will be called and allowed to change any allocation values, except for the data set name.</td>
</tr>
</tbody>
</table>

**+PARMLIST**

The +PARMLIST parameter specifies whether or not the values of the product parameters will print in the SYSPRINT listing.

```
+PARMLIST(NO  YES)
```

The values are defined as follows:

YES

*(default)* Print the values of the product parameters in the job output.

NO

Do not print the values of the product parameters.

*Note*  
Global parameters are not suppressed.

**+PLAN**

The +PLAN parameter specifies the name of the DB2 plan to be used by NGT utilities.
The values are defined as follows:

planName

Character string representing the name of the NGT plan (default = NGTPLAN).

Example

In the following example, MYPLAN is used as the plan name for NGT utilities.

+PLAN(MYPLAN)

+PSWDWTO

The +PSWDWTO password specifies how NGT notifies you of expiring passwords.

The values are defined as follows:

NO

(default) Writes message to the individual jobs output.

YES

Issues a WTO (Write To Operator) specifying which product passwords are within 21 days of expiring.
**Example**

+PSWDWTO(YES) writes messages to the operator console, as in the following example:

```
+NGTZ166 Reorg PASSWORD (xxxxxxxxxxxxxxxx) TO EXPIRE IN 21 DAYS.
+NGTZ166 RTS PASSWORD (xxxxxxxxxxxxxxxx) TO EXPIRE IN 21 DAYS.
```

**+QRETRY**

The +QRETRY parameter specifies how long an NGT utility can hold a drain and how many times a drain can be requested. Initially, the drain only establishes a Point-In-Time. Later drains apply changes accumulated during the utility processing.

```
+QRETRY( duration,numTries )
```

The values are defined as follows:

**duration**

Duration that the NGT utility holds a drain. The time is expressed as MMSSTH:

- MM Minutes
- SS Seconds
- T Tenths of a Second
- H Hundredths of a Second

**numTries**

Accepts a three digit integer (default =10). If the number of retries is exceeded, a message is issued, and the utility terminates with a return code of 8.

The drain is retried when a drain fails to occur in the allotted time. The NGT utility issues the following message:
Example

In the following example, the NGT utility holds the drain for twenty seconds and retries a maximum of 30 times.

```
+QRETRY(002000,30)
```

+RACFID

The +RACFID parameter specifies the ID that NGT utilities use for access authority. If none is specified then it will default to the submitter's user ID. This parameter can only be set in the NGT configuration job CNTL(CNFGssid) and cannot be set or overridden in individual jobs.

The values are defined as follows:

racfID

Authorization ID for NGT to use to access DB2 objects.

&DBM1ID

The product uses the user ID of the DB2 DBM1 address space.

Restrictions

+RACFID does not apply to allocation on sequential work or copy data sets. This parameter can only be set during configuration, which requires AMASPZAP authority. Use of this parameter outside configuration jobs is ignored.
Example

In the following example, NGT utilities use authorization ID DB2AUTH to access DB2 objects.

```
+RACFID(DB2AUTH)
```

In the following example, NGT utilities use the utility submitters authorization to access DB2 objects.

```
+RACFID()
```

+RERUN

The +RERUN parameter specifies what NGT utilities do on a restart attempt.

This parameter is only used when the JCL EXEC parameter is RESTART. It is ignored when using NORESTART or QUICKEXIT on the JCL EXEC statement.

A zero value will not affect restart logic. A value less than 0 instructs NGT utilities to perform a QUICKEXIT on the n\textsuperscript{th} restart attempt. A value greater than zero tells NGT utilities to perform a NORESTART on the n\textsuperscript{th} restart attempt.

The values are defined as follows:

0

*(default)* Tells NGT utilities to continue to apply normal restart logic on all subsequent restart attempts.

-integer

This is a value from -1 through -32767. On the (absolute value of) -n attempts, NGT utilities will internally change the RESTART parameter to a QUICKEXIT parameter and issue a message to that effect. It will then perform the QUICKEXIT processing.
**integer**

This is a value from 1 through 32767. On the \(n\)th restart attempt, NGT utilities internally change the RESTART parameter into a NORESTART parameter. It will issue a message and then perform the NORESTART processing. There are certain cases in which NGT utilities are unable to change to NORESTART (i.e. A previous Reorg failed and the original object must be recovered before any processing can occur. In this case a message indicating a QUICKEXIT processing was taken will be issued.

**Note**

Do not use a plus sign to denote a positive value.

---

**Example**

In the following example, if the RESTART parameter is specified, the job will attempt 5 restarts and then will change the parameter to a QUICKEXIT and finish processing.

```plaintext
+RERUN(-5)
```

**+REXXBUFR**

The +REXXBUFR parameter specifies the REXX variable buffer size in megabytes (M).

The values are defined as follows:

- **200**
  
  *(default)* Uses REXX variable buffer size of 200 M.

- **bufferSize**
  
  An integer from 1 to 200.
Example

In the following example, a REXX variable buffer of size 50 M is specified.

+REXXBUF(50)

+REXXTIME

The +REXXTIME parameter specifies how many CPU seconds the NGT Automation Utility automation control points are allowed to use before being cancelled.

The values are defined as follows:

10

*(default)* Up to ten seconds can be used.

CPUSeconds

Any number of seconds from 1 to 32767. This allows up to a little over 9 hours of CPU time per exec invocation. Each time an automation control point is called; the timer is reset and starts over.

Example

In the following example, NGT allows every automation control point to use up to 25 CPU seconds of processing before being cut off.

+REXXTIME(25)

+SQLID

The +SQLID parameter specifies the secondary authid that NGT utilities use for SQL. This is equivalent to SET CURRENT SQLID. Your security product, usually RACF, is checked to verify that the user is in the group specified in +SQLID.
This parameter can only be set in the NGT Configuration job CNTL(CNFGssid), and cannot be set or overridden in individual jobs.

The values are defined as follows:

sqlID

A valid SQLID for the DB2 system on which a utility is being run.

&DBM1ID

The product uses the user ID of the DB2 DBM1 address space.

+SVRDISK

The +SVRDISK parameter specifies the maximum number of disk (or DASD) servers to start for server processing.

You can set the value of this parameter by using the XSUTGLOB NGT Automation control point. For more information, see the *BMC Next Generation Technology Automation Reference Manual*.

**Note**

If +SVRMODE(Mixed) is used, the internal server does count as one of the disk servers.
The values are defined as follows:

$maxDisks$

A number from 1 to 32 indicating the maximum number of disk servers to be started (default =2).

-1

Signifies a disk server should not be started.

**Example**

In the following example, a maximum of 2 disk servers are used for server processing. Both the internal and external servers count toward this maximum.

```
+SVRDISK(2)
```

**+SVRDISKJC**

The +SVRDISKJC parameter specifies the job class for the disk server job.

You can code a $&/C$ symbolic on the JOB statement.

```
// CLASS=&JC.. NOTIFY=...
```

The value for the job class is provided by this parameter. If $&/C$ is specified in the JOB statement and this parameter is omitted, then the default job class of $A$ is used.
The values are defined as follows:

A

Default job class for disk servers.

x

The job class for the disk servers.

Example

In the following example, the server job card, pointed to by the +SVRJCLJOB parameter, contains CLASS=&JC. The server will run with CLASS=D.

+SVRDISKJJC(D)

+SVRJCLDISK

The +SVRJCLDISK parameter specifies the DD name to override the location of the disk (or DASD) server job JCL. Normally there would be no DD in the job and this would name the member in the NGTSVR PDS specified in the NGT configuration.

+SVRJCLDISK(ddname)

The values are defined as follows:

SVRDISK

Default DD or member name for the disk server JCL.
ddname

An alternative DD or member name for the disk server JCL. If the ddname specified in this parameter is not valid, message NGTY118 is issued indicating the ddname was not found, and processing terminates with a return code of 8.

Example

+SVRJCLDISK(DSERVJCL)

DSERVJCL normally names a member in the PDS named by the NGTSVR DD allocated in the NGT installation configuration job and can be overridden in the utility's JCL.

This whole scheme can be overridden in a utility job by adding the DD name DSERVJCL pointing to a data set containing the disk server JCL. This is accomplished by way of the following JCL change:

//DSERVJCL DD DSN=prefix.NGTSVR,DISP=SHR

Otherwise, without this specified DSERVJCL, disk server JCL will come from member DSERVJCL from the NGTAUTO dataset specified in the job (if it was not changed at installation time).

+SVRJCLJOB

The +SVRJCLJOB parameter specifies the DD name to override the location of the server JCL’s JOB card. Normally there would be no DD and this would name the member in the NGTSVR PDS that is specified in the NGT configuration.

The values are defined as follows:

SVRJOBCD

Default DD or member name for the server job card.
**ddname**

An alternative DD name or member name for the JCL associated with the server JCL's JOB card. If the *ddname* specified in this parameter is not valid, message NGTY118 is issued indicating the *ddname* was not found, and processing terminates with a return code of 8.

**Example**

```+SVRJCLJOB(SJOBJCL)```

SJOBJCL normally names a member in the PDS named by the NGTSVR DD allocated in the NGT installation configuration job and can be overridden in your utility JCL.

This whole scheme can be overridden in a utility job by adding the DD name SJOBJCL pointing to a data set containing the disk server JCL. This is accomplished by way of the following JCL change:

```//SJOBJCL DD DSN=prefix.NGTSVR,DISP=SHR```

Otherwise, without this specified SJOBJCL DD, the disk server JCL will come from member SJOBJCL from the NGTAUTO dataset (if it was not changed at installation time).

**+SVRJCLTAPE**

The +SVRJCLTAPE parameter specifies the DD name to override the location of the tape server job JCL. Normally there would be no DD and this would name the member in the NGTSVR PDS that is specified in the NGT Configuration.

```
+SVRJCLTAPE(ddname SVRTAPE)
```

The values are defined as follows:

**SVRTAPE**

Default DD or member name for the tape server JCL.
ddname

An alternative DD or member name for the tape server JCL. If the ddname specified in this parameter is not valid, message NGTY118 is issued indicating the ddname was not found, and processing terminates with a return code of 8.

Example

+SVRJCLTAPE(TSERVJCL)

TSERVJCL normally names a member in the PDS named by the NGTSVR DD allocated in the NGT installation configuration job and can be overridden in your utility JCL.

This whole scheme can be overridden in your utility job by adding the DD name TSERVJCL pointing to a data set containing the disk server JCL. This would be accomplished by the following JCL change:

//TSERVJCL DD DSN=prefix.NGTSVR,DISP=SHR

Otherwise, without this specified TSERVJCL, the disk server JCL will come from member TSERVJCL from the NGTAUTO data set (if it was not changed at installation time).

+SVRMODE

The +SVRMODE parameter specifies the mode in which servers are spawned for processing.

The values are defined as follows:

MIXED or M

(default) Signifies that a mixture of external and internal processing is to be used, determined by NGT processing logic. Processing examines or weighs
the size of all objects to be processed and uses the appropriate process accordingly.

In the case of +SVRMODE(MIXED) processing will first start an internal server. The next time processing finds waiting work, processing will start an external server. This loop stays in place until the +SVRDISK(n) value is reached.

EXTERNAL or E

Signifies that external server processing is to be used. External server processing spawns one or more server jobs and requires additional initiators to be available. This option was designed to be used for jobs involving large numbers of objects, large objects, or significant amounts of data base processing work.

INTERNAL or I

Signifies that internal server processing is to be used. Internal server processing handles all processing in memory, reducing system resource usage. This option was designed to be used for jobs involving small numbers of objects, a single object, or short jobs. This mode is suitable for shops desiring maximum throughput on small tasks.

Examples

In the following example, processing uses internal servers.

+SVRMODE(INTERNAL)

In the following example, processing proceeds in mixed server mode, determining whether to use internal or external servers on its own.

+SVRMODE(M)

In the following example, processing uses external servers.

+SVRMODE(EXTERNAL)

+SVRPROCD

The +SVRPROCD parameter specifies the name of the PROCLIB member for disk (or DASD) processing when a started task is used (+SVRTYPE(STC)).
This parameter only applies if +SVRTYPE(STC) is specified (run jobs as started tasks). If jobs are submitted in regular batch mode (+SVRTYPE(JOB)) then this parameter is ignored.

The values are defined as follows:

**procName**

The name of a PROCLIB member containing code for a started task.

The NGT CNTL library contains a *sample started task procedure* for use with this parameter - member NGTSEVRD. You may use this sample as is, or copy and modify it to customize it to your needs.

**Example**

In the following example, when processing is run with +SVRTYPE(STC) indicating a started task is to be run, member CODISK in the PROCLIB library will be called.

```
+SVRPROC(CODISK)
```

Contact your system administrator for the name of your PROCLIB data set as well as for access and content.

**+SVRPROCMSGCLAS**

The +SVRPROCMSGCLAS parameter specifies the Message Class for started task server jobs. Often the output for Started Tasks are not retained, this allows the message class to be changed so it is available for diagnostics. See also: +SVRPROCMSGVL.
The values are defined as follows:

\[ x \]

The desired message class.

**Example**

In the following example, the message class for the NGT server started tasks will be "A" rather then the installation default for started tasks.

\[ +SVRPROCMSGCLAS(A) \]

---

**+SVRPROCMSGCLAS**

The +SVRPROCMSGCLAS parameter specifies the message level for started task server jobs. Often the output for started tasks are not retained, this allows the message class to be changed so it is available for diagnostics.

See also “+SVRPROCMSGCLAS” on page 199.

---

The values are defined as follows:

\[ x,y \]

The desired message level.
Example

In the following example, the message level for the NGT server started tasks is 1,1 rather than the installation default for started tasks.

+SVRPROCMSGGLVL(1,1)

+SVRPROCT

The +SVRPROCT parameter specifies the name of the PROCLIB member for tape processing when a started task is used (+SVRTYPE(STC)).

Note

This parameter only applies if +SVRTYPE(STC) is specified (run jobs as started tasks). If jobs are submitted in regular batch mode (+SVRTYPE(JOB)) then this parameter is ignored.

+SVRPROCT(procName)

The values are defined as follows:

procName

The name of a PROCLIB member containing code for a started task.

Note

The NGT CNTL library contains a sample started task procedure for use with this parameter - member CDBSEVRT. You may use this sample as is, or copy and modify it to customize it to your needs.

Example

In the following example, when processing is run with +SVRTYPE(STC) indicating a started task is to be run, member COTAPE in the PROCLIB library will be called. Contact your system administrator for the name of your PROCLIB dataset as well as for access and content.

+SVRPROCT(COTAPE)
+SVRSPECIAL

The +SVRSPECIAL parameter controls special server processing and is designed to boost throughput.

+SVRSPECIAL(YES) starts an additional server (not one limited by the number of servers specified with +MAXSERVERS) *exclusively for image copies* after the first reorg has completed.

The reorg process is not complete until the copy is complete. Using a special server allows the copies to be processed as early as possible, concurrent with reorgs. This reduces overall reorg time thus reducing the application changes that have to be gathered and applied before the reorg is complete. Using +SVRSPECIAL will give a significant performance improvement when many reorgs are done in one job and/or many application changes are happening during the utility run.

The values are defined as follows:

**YES or Y**

Initiate a special copy server job right after the first Reorg has completed.

**NO or N**

Do not initiate a special copy server job. Use a regular server for the copy after all other Reorg servers have completed.

**EXTERNAL or E**

Same as YES, Initiate a special copy server job right after the first Reorg has completed; this server will be a separate job.

**INTERNAL or I**

*(default)* Initiate a special copy server job right after the first Reorg has completed; but run this server internally within the Master job. A separate job is not submitted.
Tip
+SVRSPECIAL(INTERNAL) is good for typical jobs, it will use available sub-tasks to process copies.
+SVRSPECIAL(YES) or EXTERNAL is good for jobs with many large datasets to be copied where all sub-tasks are busy processing Reorgs.

Examples

In the following example, after the first Reorg has completed, NGT immediately begins a copy server.

+SVRSPECIAL(YES)

In the following example, NGT uses a regular server for copy after the last Reorg has completed.

+SVRSPECIAL(NO)

In the following example, NGT uses an internal server to manage copies immediately after the first Reorg has completed.

+SVRSPECIAL(INTERNAL)

+SVRTAPE

The +SVRTAPE parameter specifies the number of tape servers to start for server processing.

You can set the value of this parameter by using the XSUTGLOB NGT Automation control point. For more information, see the BMC Next Generation Technology Automation Reference Manual.

Note
At least one tape server must be started for tape jobs. If +SVRTAPE(0) is specified, any image copies produced go to disk (DASD).
The values are defined as follows:

*integer*

Specifies the number of tape servers to be utilized.

**Examples**

In the following example, 3 tape servers are used for processing.

```
+SVRTAPE(3)
```

In the following example, no tape servers are used for processing.

```
+SVRTAPE(0)
```

**+SVRTAPEJC**

The +SVRTAPEJC parameter specifies the job class for the tape server job.

A new &JC. symbolic may now be coded on the JOB statement.

```
// CLASS=&JC., NOTIFY=...
```

The value for the job class is provided by this parameter. If &JC. is specified in the JOB statement and this parameter is omitted, then the default job class of A will be used.

The values are defined as follows:

* A
  
  Default job class for tape servers.

  * x
    
    The job class for the tape servers.
Example

In the following example, the server job card, pointed to by the +SVRJCLJOB parameter, contains CLASS=&JC. The server will run with CLASS=B.

+SVRTAPEJC(B)

+SVRTYPE

The +SVRTYPE parameter specifies the type of server processing.

The values are defined as follows:

JOB

(default) Servers run as batch jobs.

STC

Servers run as started tasks. Use of this operand requires the additional use of the +SVRPROC and +SVRPROCT parameters.

Started-task servers (STC) have some restrictions not found with batch job servers. For instance, STC servers must run on the same LPAR as the master which limits the intended distribution of servers across a sysplex.

Example

In the following example, batch jobs are used for server processing.

+SVRTYPE(JOB)

In the following example, started tasks are used for server processing. The PROCLIB will use member CODISK for disk tasks and member COTAPE for tape tasks.

+SVRTYPE(STC)
+SVRPROC(CODISK)
+SVRPROCT(COTAPE)
+SYSINERROR

The +SYSINERROR parameter determines the action to take when a syntax error is detected in the SYSIN section of a utilities job.

The values are defined as follows:

CONTINUE

*(default)* Errors in SYSIN statements will be ignored. Statements without syntax errors will be processed.

ABEND

Stops processing SYSIN statements immediately and terminates the currently running utility as soon as possible.

STOP

Stops processing SYSIN statements immediately but allows any utility jobs already in progress to complete.

Example

In the following example, when a syntax error is detected in the SYSIN statements, processing will not read any more statements and will terminate the currently running utility as quickly as possible.

+SYSINERROR(ABEND)

+SYSINORDER

The +SYSINORDER parameter controls the order in which SYSIN statements are processed. For NGT Load usage in conjunction with NGT Unload only.
The values are defined as follows:

**RANDOM or R**

*(default)* NGT processing will determine the order in which SYSIN statements are processed.

**HONOR or H**

NGT processing will follow the order - as exactly specified - for processing SYSIN statements.

**Example**

Consider the following unload statements in a SYSIN part of a job file:

```
//SYSIN DD *
LOAD DATA INTO TABLE USR001.EXPENSES
LOAD DATA INTO TABLE USR001.PAYROLL
LOAD DATA INTO TABLE USR001.GEN_LEDGER
UNLOAD FROM TABLE USR001.EXPENSES
UNLOAD FROM TABLE USR001.PAYROLL
UNLOAD FROM TABLE USR001.GEN_LEDGER
```

If the following is specified:

In the following example, the EXPENSES, PAYROLL, and GEN_LEDGER tables are unloaded in any order, and this order can vary from job to job. NGT processing determines the best order in order to optimize performance.

```
+SYSINORDER(RANDOM)
```

In the following example, the tables are unloaded in the order specified: EXPENSES, PAYROLL, GEN_LEDGER.

```
+SYSINORDER(HONOR)
```
**+TIMEEXEC**

The +TIMEEXEC parameter controls a second invocation of automation exit XSUTTIME. NGT can make two time checks for deadline processing using of the XSUTTIME automation exit if you define +TIMEEXEC YES.

The values are defined as follows:

**NO**

*(default) XSUTTIME is called once, after all SYSIN statements have been processed.*

**YES**

XSUTTIME is called twice: after all SYSIN statements have been processed and again prior to BFACC (before access processing).

**Example**

In the following example, the system time will be checked after all SYSIN statements have been processed, and again before BFACC is initiated. If the system time is after the deadline set in automation exit XSUTTIME, processing will be terminated.

```
+TIMEEXEC(YES)
```

**+UIDHISTORY**

The +UIDHISTORY parameter specifies the number of days a utility ID history will be kept. A utility ID history is kept in the NGT Checkpoint dataset.

```
+UIDHISTORY(0 numberDays)
```
The values are defined as follows:

**numberDays**

A value from 1 to 180 days (default value is 5).

0

Specifying a value of 0 will not maintain UID history beyond the value specified for +OUTPUTAGE.

**Examples**

In the following example, a history of utility IDs is kept for 30 days.

```
+UIDHISTORY(30)
```

In the following example, a history of utility IDs will not be kept beyond +OUTPUTAGE value.

```
+UIDHISTORY(0)
```

**+USER1**

+USER1 defines a user defined variable that can be used by the following load configuration parameters:

- +LOGPDS
- +NGTAUTO1
- +NGTAUTO2
- +NGTCKPT
- +NGTSVR
+USER2

+USER2 defines a user defined variable that can be used by the following load configuration parameters:

- +LOGPDS
- +NGTAUTO1
- +NGTAUTO2
- +NGTCKPT
- +NGTSVR

+WAITTIMEOUT

The +WAITTIMEOUT parameter specifies whether the object is to be placed in the ERROR queue to be available for restart processing (ERROR), or to skip the object and move it to the exclude queue making restart unnecessary for this object.

- EXCLUDE
  - Put timed out object in EXCLUDE queue making RESTART or QUICKEXIT unnecessary for this object.

- ERROR
  - Put timed out object in ERROR queue making RESTART or QUICKEXIT necessary to complete the utility.
**Example**

In the following example, an object that is timed out and waiting for another NGT utility to release it is put in the EXCLUDE queue, and the object is skipped.

---

**+WAITTIMEOUT(EXCLUDE)**

---

**+WORKMODE**

The +WORKMODE parameter provides the capability of defining permanent work data sets which can improve utility performance by eliminating the portion of processing where work data sets are defined.

![Diagram](image)

The values are defined as follows:

**TEMP**

*(default)* Use temporary work data sets.

**PERM**

Use permanent work data sets and fail the job if none are available. You must specify a name for a group of permanent work data sets.

**ANY**

Use permanent work data sets if available. If none are available, use temporary work data sets. If NGT determines permanent work data sets will be used, you must specify a name for a group of work data sets.

**name**

The name of the work data sets. The default is DEFAULT.
The +WORKPREFIX parameter specifies the data set name prefix for dynamic allocation of VSAM work (temporary) data sets.

You can set the value of this parameter by using the XSUTGLOB NGT Automation control point. For more information, see the *BMC Next Generation Technology Automation Reference Manual*.

The following data sets are created during processing:

- Information data set for the CDBI Console. The suffix is `.INFO001`.
- Output data set for the CDBI Console. The suffix is `.OUT001`.
- Scratch data set used for general processing. The suffix is `.WK00001`.
- Copy data set for Reorg. The suffix is `.RGTS0001`.

**Note**

It is important that the DASD pool used for the +WORKPREFIX data sets be extended-addressability enabled.

The +WORKPREFIX parameter is saved across restarts. The value for +WORKPREFIX cannot be changed on a restart.

The +CONTROLPREFIX parameter can be used to provide a different prefix for the INFO and OUT data sets. This allows these two to be in a different SMS pool since they will stay on the system up to +OUTPUTAGE days while the other two will be deleted upon successful completion of the job.

See also “+OUTPUTAGE” on page 183 and “+CONTROLPREFIX” on page 158.

```
+WORKPREFIX(workPrefix)
```

The single value is defined as follows:

`workPrefix`:

The `workPrefix` is appended to a unique timestamp-based identifier, and the four data set suffixes.
You can specify up to 15 characters or use the following supported symbolic variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;SSID</td>
<td>The DB2 subsystem name or group name specified on the PARM= parameter attribute</td>
</tr>
<tr>
<td>&amp;VER</td>
<td>The version (for example 1210) of the NGT utility being executed</td>
</tr>
<tr>
<td>&amp;VER3</td>
<td>The first three characters of the version (for example 121) of the NGT utility being executed</td>
</tr>
<tr>
<td>&amp;VER2</td>
<td>The first two characters of the version (for example 12) of the NGT utility being executed</td>
</tr>
<tr>
<td>&amp;USER1</td>
<td>The value of the +USER1 parameter</td>
</tr>
<tr>
<td>&amp;USER2</td>
<td>The value of the +USER2 parameter</td>
</tr>
</tbody>
</table>

**Note**

After the variable is resolved, workPrefix cannot exceed 15 characters. Before resolution, workPrefix including variable names cannot exceed 32 characters.

The entire work data set name, including all qualifiers, cannot exceed 42 characters or IDCAMS errors may result.

An * (asterisk) is substituted with the user ID associated with the JOB.

**Example**

If you enter *workPrefix, the * (asterisk) is substituted with the user ID associated with the JOB and the value of workPrefix is appended.

An *(asterisk) can only be specified for the first character.

**Example**

In the following example, the following data sets are created during processing (ttttttt.uuuuuuuu represents the unique timestamp-based identifier):

- ADMIN74.NGTJOBS.ttttttt.uuuuuuuu.INFO001 (information data set for the CDBI Console)
- ADMIN74.NGTJOBS.ttttttt.uuuuuuuu.OUT001 (output data set for the CDBI Console)
- ADMIN74.NGTJOBS.ttttttt.uuuuuuuu.WK00001 (scratch data set for processing)
- ADMIN74.NGTJOBS.Nttttttt.uuuuuuuu.RGTS0001 (copy data set)
The +XBMID parameter specifies the XBM subsystem ID used by NGT Copy when it is invoked by NGT Load to make a copy of the object that it is processing.

The +XBMID parameter is optional and overrides the value of the XBMID installation option.

+XBMID is required only in the following circumstances:

- You did not specify the XBMID in the NGT Copy options module in the installation process
- There are multiple XBM subsystems running and you want to direct copies to a specific XBM subsystem.

The values are defined as follows:

**XBMSSID**

The XBM subsystem ID is the unique identifier you specified when you installed XBM

**XBMGroup**

The *XBMGroup* is the name of the XBM coupling facility group defined to the XBM subsystem.

**Note**

NGT Copy supports only alphanumeric characters for the specification of XBMID.
NGT utility reports

This chapter describes the reports and messages produced by NGT utilities that are common to multiple utilities.

NGT utility age report

The NGT checkpoint data set maintains information about utilities that have been run. The global parameters +OUTPUTAGE and +UIDHIST define how long the product keeps this information. The NGTDISP statement is used to report on current and past NGT utility IDs (UIDs).

To verify why a UID is still running, you can request a Utility Age report by adding //AGEREPRT DD SYSOUT=* to the NGT utility job. The Utility Age report is displayed:

![Figure 20: Sample Utility Age report](image)

Age report fields

**AGE-FAILED UTILITY-ID**

The Utility ID of the original utility.

**JOBNAME**

The job name that the utility ran in.
JOB #

The JES job number of the utility job.

STEPNAME

The stepname that the utility ran in.

DB2ID

The DB2 subsystem that the utility ran on.

STARTED

The date and time that the utility-ID originally started. Note: for restarted utilities this will reflect the original utilities start date and time, not the restart date and time.

AGE VALUE

The +OUTPUTAGE specified when the utility originally started.

DATASET PREFIX

The value specified in +WORKPREFIX in the original utility job. This is the HLQ for the NGT work data sets.

DATASET QUALIFIER

The work data set qualifiers generated by NGT to make these work data sets unique to this utility-ID.

AGE-FAILED REASON

The reason that the work data sets for this utility-ID have not been aged off. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTHORIZATION</td>
<td>The userid is not authorized to delete the work data sets</td>
</tr>
<tr>
<td>DB2ID MISMATCH</td>
<td>The current DB2ID does not match the DB2ID that the utility was run on</td>
</tr>
<tr>
<td>NOT COMPLETE</td>
<td>The utility-ID has not finished and should be restarted or quickexited</td>
</tr>
<tr>
<td>OUTPUTAGE VALUE</td>
<td>The +OUTPUTAGE value specified in the original utility-ID has not been reached</td>
</tr>
</tbody>
</table>
Processed objects report (OBJREPRT)

OBJREPRT shows a report listing all table spaces and indexes that have been processed, and their status at completion.

This report is automatically generated to the OBJREPRT DD if a SUMMARY DD SYSOUT=* statement is in the JCL. If the SUMMARY report is sent to a data set (using a DSN= keyword), an OBJREPRT DD SYSOUT=* statement is also required in order to generate the report.

**Note**

Objects that NGT Reorg cannot process, and that generate an NGTZ322 error message in SYSERROR, are not listed in the OBJREPRT.

The following figure provides an example of this report:

**Figure 21: OBJREPRT report**

<table>
<thead>
<tr>
<th>STMT #</th>
<th>UTILITY TYPE</th>
<th>DATABASE/CREATOR NAME</th>
<th>TABLESPACE/INDEX NAME</th>
<th>OBJECT TYPE</th>
<th>PART(S)</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>REORG</td>
<td>DBASMITH</td>
<td>XYZSIM</td>
<td>TS</td>
<td></td>
<td>COMPLETED</td>
</tr>
<tr>
<td>2</td>
<td>REORG</td>
<td>DBASMITH</td>
<td>XYZSEG</td>
<td>TS</td>
<td></td>
<td>COMPLETED</td>
</tr>
<tr>
<td>3</td>
<td>COPY</td>
<td>DBASMITH</td>
<td>RN2381</td>
<td>TS</td>
<td>1-10</td>
<td>EXCLUDED</td>
</tr>
<tr>
<td>4</td>
<td>NGTSTA</td>
<td>DBASMITH</td>
<td>A2388A</td>
<td>TS</td>
<td></td>
<td>COMPLETED</td>
</tr>
</tbody>
</table>

Each entry in the report corresponds to the statements in the NGT job's SYSIN section (a search for "STATEMENT NUMBER" will provide these). The following columns are included in this report:

**Utility Type**

An abbreviation for the name of the NGT utility run.

**Database/Creator Name**

The name of the data base or creator for the object listed.

**Tablespace/Index Name**

The name of the index for the object listed.
Object Type

The type of object processed - TS for table space or IX for index.

Part(s)

If the table space or index was partitioned, this column will show which range of parts was processed.

Status

The status of the processed object at job or step completion.

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPLETED</td>
<td>Object has been processed successfully.</td>
</tr>
<tr>
<td>EXCLUDED</td>
<td>Object failed to process and +CLEANUP(YES) or XSVRXERR set ACTION=CLEANUP.</td>
</tr>
<tr>
<td></td>
<td>Cleanup has occurred so no restart or quickexit is needed or possible to</td>
</tr>
<tr>
<td></td>
<td>complete processing of these objects.</td>
</tr>
<tr>
<td>SELECTED BY RTS FOR PROCESSING</td>
<td>Self-explanatory, followed by post processing status.</td>
</tr>
<tr>
<td>IN-PROCESS</td>
<td>Object did not complete, possibly due to some error, since Cleanup</td>
</tr>
<tr>
<td></td>
<td>processing did not occur, the object is available for restarting. Cleanup</td>
</tr>
<tr>
<td></td>
<td>processing would not occur because it was not requested or because the</td>
</tr>
<tr>
<td></td>
<td>object is in a must complete state preventing cleanup.</td>
</tr>
<tr>
<td></td>
<td>Object may have never started processing due to previous errors.</td>
</tr>
<tr>
<td></td>
<td>Objects that are In-Process will be processed upon restart.</td>
</tr>
</tbody>
</table>

For any utility that processes all indexes, or uses wildcards, the Data base/Creator Name column will show the indication *WILDCARD*, followed by a list of objects matching the wildcard naming mask.

The following figure shows an example of this:

Figure 22: OBJREPRRT Report Showing Wildcard Expansion

<table>
<thead>
<tr>
<th>STMT</th>
<th>UTILITY</th>
<th>DATABASE/CREATOR</th>
<th>TABLESPACE/INDEX</th>
<th>OBJECT NAME</th>
<th>TYPE</th>
<th>PART(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COPY</td>
<td><em>WILDCARD</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH</td>
<td>ABCSIM</td>
<td>TS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH</td>
<td>ABCSEG</td>
<td>TS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH</td>
<td>ABCP04</td>
<td>TS</td>
<td>1-4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH</td>
<td>ABCP16</td>
<td>TS</td>
<td>1-16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH</td>
<td>ABCP64</td>
<td>TS</td>
<td>1-64</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH</td>
<td>ABCPFF</td>
<td>TS</td>
<td>1-255</td>
<td></td>
</tr>
</tbody>
</table>

For any utility run with RTS, the status column will indicate how RTS determined the objects would be processed. If the message EXCLUDED BY RTS appears, check the RTS Excluded Table space section of the SUMMARY report in the output.
The following figures show examples of these situations:

**Figure 23: OBJREPRPT Report With RTS Results**

<table>
<thead>
<tr>
<th>STMT #</th>
<th>UTILITY TYPE</th>
<th>DATABASE/CREATOR NAME</th>
<th>TABLESPACE/INDEX NAME</th>
<th>OBJECT TYPE</th>
<th>PART(S)</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COPY</td>
<td><em>WILDCARD</em> DSN9DCAT</td>
<td>ABCP04</td>
<td>TS</td>
<td>1-4</td>
<td>COMPLETED</td>
</tr>
<tr>
<td></td>
<td>PROCESSING</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>COPY</td>
<td>DSN9DCAT</td>
<td>ABCP16</td>
<td>TS</td>
<td>1-16</td>
<td>SELECTED BY RTS FOR PROCESSING</td>
</tr>
<tr>
<td></td>
<td>PROCESSING</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>COPY</td>
<td>DSN9DCAT</td>
<td>ABCP64</td>
<td>TS</td>
<td>1-64</td>
<td>SELECTED BY RTS FOR PROCESSING</td>
</tr>
<tr>
<td></td>
<td>PROCESSING</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note**

The objects not reported on the OBJREPRPT were excluded by Utility Manager and never sent to the utility. These will be reported in the SUMMARY in the *Excluded* reports.

**Figure 24: RTS Excluded table space report**

<table>
<thead>
<tr>
<th>DATABASE NAME</th>
<th>TABLESPACE NAME</th>
<th>PART(S)</th>
<th>EXCLUDE REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSN9DCAT</td>
<td>ABCSIM</td>
<td></td>
<td>PASSED ALL RTS CRITERIA</td>
</tr>
<tr>
<td>DSN9DCAT</td>
<td>ABCSEG</td>
<td></td>
<td>PASSED ALL RTS CRITERIA</td>
</tr>
</tbody>
</table>

Reasons for RTS exclusion for table spaces ABCSIM and ABCSEG are shown in the Exclude Reason column.
NGT work data set control

This chapter describes the work data sets used by NGT utilities, as well as the parameters that name them and control how long they are kept on a mainframe system.

Overview of the NGT work data sets

You can provide high-level qualifiers for work data sets.

The prefixes NGT uses for the last node of the work data sets include the following:

**WK**
- Work data sets. These data sets might be needed for restart.

**RGTS**
- Concurrent copy temporary data sets. These data sets are used for Reorg table space only.

**INFO**
- Environmental information. These data sets are used for NGTDISP etc.

**OUT**
- Journal output data set. These data sets are used by the console and NGT1DBUT.

Naming the NGT work data sets

All the NGT work data sets are qualified by the HLQ that you supply in +WORKPREFIX. However, if you want the two INFO and OUT data sets, which are kept for a few days, to be in a separate pool you can use the +CONTROLPREFIX global parameter to provide a separate HLQ for them.

- +WORKPREFIX (prefix for INFO, WK, OUT, and RGTS)
Controlling when NGT work data sets are deleted

The new WK and RGTS data sets are deleted when a job is successful. The new INFO and OUT data sets are kept for a defined number of days. This number of days is controlled by the +OUTPUTAGE global parameter.

You can use the following global parameters to define when NGT work data sets are deleted:

- **+OUTPUTAGE** (number of days to keep output for this job)
  The +OUTPUTAGE global parameter controls how long output is kept for each job, and therefore can be overridden in jobs. The +AGELIMIT global parameter is to set a maximum number of days that can be specified for +OUTPUTAGE. This lets the system control how much total output will be kept. +AGELIMIT cannot be overridden in individual jobs.

- **+AGELIMIT** (maximum days to keep output)

- **+DELETEOUTPUT** (delete INFO, OUT, WK when successful job)
  The +DELETEOUTPUT global parameter is an override that instructs the system control to delete the output immediately when the job is successful. If +DELETEOUTPUT is set to YES the JOURNAL output will not be available to the console or to NGT1DBUT. +DELETEWORK defaults to YES.

- **+DELETEWORK** (delete WK files when successful job)

Usage notes for NGT work data sets

The WK, RGTS, INFO, and OUT, are VSAM data sets. This allows for shared access between the Master and Server jobs. The SMS pool where these data sets go to, defined by +WORKPREFIX and optionally +CONTROLPREFIX, should be extended-addressability enabled. This will reduce the number of extents and therefore the below the line storage required by your jobs to manage those extents.
Note

It is important that:

- The DASD pool used for the +WORKPREFIX data sets be extended-addressability enabled.
- There are no system automation to delete NGT work data sets. NGT work data sets are required for UID restart and they will be deleted by NGT.

Aging NGT work data sets

NGT work data sets are automatically aged. When processing has ended successfully and +DELETEOUTPUT is set to (N) the .INFO and .OUT data sets are not deleted (only the .RGTS and .WK files are deleted). The .INFO and .OUT data sets are kept for +OUTPUTAGE(n) days.

The deletion of these aged out data sets UIDs is performed whenever a NGT utility runs.

The following message is displayed in the output:

NGTZ117 STARTED AGE PROCESSING FOR COMPLETED ENTRIES
The following figure illustrates the VSAM work data sets deletion process:

**Figure 25: Disposition of VSAM work data sets**

**Alternate Method of Aging NGT work data sets**

Automatically deleting the work data sets through the standard aging process works in most situations. If the data sets are not being aged in a timely manner, this alternate method might be useful. Migrating the .INFO and .OUT data sets using hierarchical storage management (HSM) can delay the age processing, which holds up the deletion schedule.

To ensure age processing completes, make the following changes to the jobs:
1  Delay the migration by scheduling a daily utility with a AGEFIRST DD statement and a null SYSIN command. This tells the NGT utility to perform the age processing before processing the (null) SYSIN.

Example

```
//NGTAGE   EXEC PGM=NGTUTIL,PARM='ssid,,NORESTART'
//UTPRINT  DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//AGEFIRST DD DUMMY
//AGEREPORT DD SYSOUT=*  
//SYSIN DD *  
*NGT AGING JOB
```

This job ends with RC=4 because this job only has a comment for the SYSIN, if you prefer RC=0, add a simple display statement such as: NGTDISP

2  *(optional)* Add a SKIPAGE DD to each NGT utility job.

## Permanent work data sets and the NGTWORK command

A pool of permanent work data sets can be defined then used and reused by utilities. If used, this is only recommended for utilities which process small objects or don’t require very large work data sets.

Definition of permanent work data sets is performed with the NGTWORK command.
NGTWORK command syntax

The +WORKMODE parameter must be used with the NGTWORK command.

ALL DETAIL
Displays details on all defined work data sets.

DEFINE
Defines a work data set with the specified options.

DELETE
Deletes a work data set.

DISPLAY
Displays the options for a particular work data set specified by name.

FORCE
Deletes the set even though some of the data sets are assigned to utilities.

NAME
8-character name to identify a set of work data sets. The default is DEFAULT.
RELEASE

Releases a work data set.

SET

The number of the set to be released.

workoptions block

See “workoptions block” on page 227.

XSUTDBMG

Tells the NGTWORK command whether or not to use the XSUTDBMG automation exit for all allocations of permanent work data sets. IF YES, NGTWORK uses the XSUTDBMG automation exit for all allocations of permanent work data sets, even if other options are specified on the NGTWORK command in the SYSIN.

workoptions block

This topic describes the workoptions syntax for the DEFINE keyword.
Note

All keywords are required.

Figure 27: workoptions syntax block

```plaintext
NAME

8-character name to identify a set of work data sets. The default is DEFAULT.

NUMBER

The number of work data sets to allocate in this set. The default is 5.

PREFIX

The prefix to use when allocating the work data sets.

WORK_PRIMARY

The primary allocation of the work data sets. The default is 500 M.

WORK_SECONDARY

The secondary allocation of the work data sets. The default is 1500 M.
```
RGTS_PRIMARY

The primary allocation of the RGTS work data sets. The default is 500 M.

RGTS_SECONDARY

The secondary allocation of the RGTS work data sets. The default is 1500 M.

INFO_PRIMARY

The primary allocation of the info data sets. The default is 50 M.

INFO_SECONDARY

The secondary allocation of the info data sets. The default is 500 M.

OUT_PRIMARY

The primary allocation of the out (journal) data sets. The default is 50 M.

OUT_SECONDARY

The secondary allocation of the out (journal) data sets. The default is 500 M.

STORCLAS

The SMS storage class to use allocating the data sets.

MGMTCLAS

The SMS management class to use when allocating the data sets.

DATACLAS

The SMS data class to use allocating the data sets.

VOLUMES

The DASD volumes to use allocating the data sets. One or more volume serials may be specified, separated by commas. For SMS volumes, one or more asterisks (*) may be specified, separated by commas.

UNIT

The unit name to use when allocating the data sets.

MAXSIZE

The maximum size to which the work data sets are allowed to grow before reallocation. The default is 0. Work files can grow to any size.
Example of NGTWORK JCL

NGTWORK is submitted like any NGT product statement would be using the same JCL.

The following JCL defines a set of permanent workfile data sets with the group name WORKSET1 and has the following characteristics:

- It will use up to 10 workfile data sets.
- The workfile will have a primary allocation of 1000 M with 750 M extents.
- The RGTS file will have a primary allocation of 1000 M with 750 M extents.
- The INFO file will have a primary allocation of 500 M with 250 M extents.
- The OUT file, or journal, will have a primary allocation of 2000 M with 1000 M extents.
- Storage will be in the AMCLASS storage class, the ADMIN33 management class, and the NGTDATA data class.
- Processing will use up to 4 SMS volumes on 3390 units.
- The maximum size of any workfile is 5000 M.

```
//STEPLIB DD ...
//CDBUSTEP EXEC PGM=NGTUTIL,PARM='SSID,,RESTART'
//SYSPRINT DD SYSOUT=* 
//UTPRINT DD SYSOUT=* 
//SYSIN DD *
NGTWORK DEFINE NAME(WORKSET1) 
  NUMBER(10) PREFIX(NGT.PERMWORK) 
  WORK_PRIMARY(1000 M) WORK_SECONDARY(750 M) 
  RGTS_PRIMARY(1000 M) RGTS_SECONDARY(750 M) 
  INFO_PRIMARY(500 M) INFO_SECONDARY(250 M) 
  OUT_PRIMARY(2000 M) OUT_SECONDARY(1000 M) 
  STORCLAS(AMCLASS) MGMTCLAS(ADMIN33) DATACLAS(NGTDATA) 
  VOLUME(*,*,*,*) 
  UNIT(3390) 
  MAXSIZE(5000 M)
```

Using NGTWORK with +WORKMODE

Once permanent work data sets have been defined with the NGTWORK command, their usage can be switched on or off using the +WORKMODE global parameter.

For more information about the +WORKMODE parameter, see “+WORKMODE” on page 211.
The NGT Checkpoint repository stores information for NGT utilities. The repository has a directory and internal files. The directory maps internal files in the repository and is implemented as a DB2 TYPE2 index. The index also maps free space in the repository.

Reorganizing the NGT Checkpoint repository

The NGT Checkpoint repository can be maintained with one simple job which should be run every six months or monthly on very busy systems. You can run this job concurrently with NGT utilities, as it is not disruptive.

The job contains the following statements:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMG REPORT</td>
<td>Run the DBMG REPORT statement before and after the reorganization statements. This is required for diagnosis in the case of errors.</td>
</tr>
<tr>
<td>DBMG CHECKIX</td>
<td>BMC recommends that you run DBMG CHECKIX, before you run DBMG REORG or DBMG COMPACT, to ensure the validity of the directory index. <strong>Note:</strong> If the CHECKIX fails the statements that follow the CHECKIX will not be executed; send the job output to BMC Customer Support for diagnosis.</td>
</tr>
<tr>
<td>DBMG OPTIMIZE</td>
<td>Run DBMG OPTIMIZE to reorganize the other indexes in the checkpoint repository.</td>
</tr>
<tr>
<td>DBMG COMPACT</td>
<td>The directory also contains free space records. After intensive use, run DBMG COMPACT to compact those records to improve performance and reduce the size of the index.</td>
</tr>
<tr>
<td>DBMG REORG</td>
<td>Run DBMG REORG to reorganize the directory index to speed up the process and reduce its size due to page splits</td>
</tr>
</tbody>
</table>
Example

```bash
//DBMG  EXEC PGM=NGT1DBMG
//STEPLIB DD DISP=SHR,DSN=your.ngt.loadlib
//SYSPRINT DD SYSOUT=* 
//CDBPRINT DD SYSOUT=* 
//SYSIN DD *
  DBMG REPORT  LOGDS(your.ngtckpt)
  DBMG CHECKIX LOGDS(your.ngtckpt)
  DBMG OPTIMIZE LOGDS(your.ngtckpt)
  DBMG COMPACT LOGDS(your.ngtckpt)
  DBMG REORG  LOGDS(your.ngtckpt)
  DBMG REPORT  LOGDS(your.ngtckpt)
```

---

### Backing up the NGT system data sets

You must backup the LOGPDS and NGTCKPT NGT system data sets.

#### Backing up the LOGPDS data set

The LOGPDS is a partitioned data set (PDS) that is used to write a mini-dump when a NGT product abends with an 0C3 due to unexpected results from one of its integrity checks.

Periodically you should delete the older members from LOGPDS, the data set members are only needed to document current issues. Periodically, you can redefine LOGPDS.

---

**Tip**

Place the LOGPDS data set on a volume that is backed up periodically in order to facilitate disaster recovery.

---

#### Backing up the NGTCKPT data set

The NGTCKPT (also known as checkpoint) data set contains checkpoint and restart information for NGT utility products and must be backed up. It keeps track of active and restartable NGT products, registration of NGT RTS tables, and the Access Method Services (AMS) override. NGTCKPT, is critical to a NGT product environment, but requires little additional maintenance other than periodic backing up.

The NGTCKPT data set is a linear VSAM data sets with a custom format so REPRO cannot be used on it. This data set should be placed on volumes that are backed up periodically for disaster recovery using IBM's DFDSS, FDR, or DASD-specific full volume backup utilities.
If mirrored DASD is used for business continuation then the NGTCKPT should be placed on mirrored DASD so the product environment can continue at the alternate site.

## Restoring NGT System data sets

In a disaster recovery situation, you must restore the LOGPDS and NGTCKPT NGT data sets.

### Restoring the LOGPDS data set

Restore the LOGPDS data set from a backup volume or redefine it. The mini-dumps kept in the LOGPDS are not critical to executing NGT utility products; they are only required for diagnostic purposes.

### Restoring the NGTCKPT data set

In a disaster recovery situation the NGTCKPT data set is critical to the continued execution of BMC Next Generation Technology (NGT) utility products. BMC recommends that you restore NGTCKPT because the RTS and NGT Application information are not lost. However because everything is restored to some prior point-in-time, NGT product restart is not possible.

For restoring the NGTCKPT there are two options:

- If you have a backup of NGTCKPT.
  1. Restore NGTCKPT from backup in the same way that you restore LOGPDS.
  2. Use the NGTDISP product to check for NGT utility product ID's that were in progress at the time of the backup and resubmit any applicable NGT utility product jobs with the restart parameter set to FORCEID.

  **Note**
  
  Always consult with BMC Customer Support before restarting a NGT product with FORCEID as this could cause data loss if the product is in a must-complete state.

- If you do not have a backup of the NGTCKPT, you can redefine NGTCKPT as a new installation.
  
  If you redefine NGTCKPT, any NGT Application definitions, NGT RTS definitions, and AMS overrides are lost.
  
  To restore these definitions, take the following actions:
— Recover the NGT RTS DB2 tables and redefine them to NGT.
For more information, see the section referring to defining the RTS tables in the *BMC Next Generation Technology Utility Manager for DB2 for z/OS Reference Manual*.

— Redefine the AMS override with the NGTAMS command.
For more information, see “Managing the model AMS define” on page 39.
NGTDISP

The NGTDISP command lets you display reports showing the current status of NGT utilities, using either the NGT Matrix and Utility internal databases or the BMC BMCUTIL and BMCSYNC common tables. It also enables you to use the XSUTDISP automation control point, which lets you automatically select and execute QUICKEXIT JCL for utilities and DELETE statements for data sets.

For more information, see “XSUTDISP” on page 253.

Note
NGTDISP runs only in the master address space.

NGTDISP input

This section covers the DD statements, SYSIN input (statement syntax), keywords, and parameters.

NGTDISP DD statements

The SYSIN DD, SUMMARY DD, and optionally the NGTAUTO DD statements are used in the NGTDISP utility.

SYSIN DD

The SYSIN DD statement is required for use with this utility.

```sql
//SYSIN DD *
  (statement)
  (statement)
  (statement)
```

For more details of the SYSIN statement syntax, see “SYSIN syntax diagrams” on page 237.
**SUMMARY DD**

The SUMMARY DD statement is required DD statement for use with this utility. Output may be sent to the job stream, or to an output data set.

```plaintext
//SUMMARY DD *
```

**NGTAUTO DD**

If you wish to use this utility with automation control points, specify an NGTAUTO DD statement. This statement should reference a partitioned data set containing the automation exits you wish to use.

**Example**

```plaintext
//NGTAUTO DD DISP=SHR,DSN=NGT.AUTOMATN.CTRL.PTS
```

For more information on automation control points, see the *BMC Next Generation Technology Automation Reference Manual*.
SYSIN syntax diagrams

Use the following diagrams to define the NGTDISP utility.

Figure 28: NGTDISP display utility report syntax

Figure 29: NGTDISP Matrix report syntax

Figure 30: NGTDISP Passwords report syntax
Keywords

The following keywords are valid for the NGTDISP SYSIN statements.

Display Utility report keywords

NGTDISP

statement identifier

UTIL

A utility ID, either a single name or a naming pattern specified with wildcards.

COMPLETED

Selects utilities that have completed with a return code of 0.

ERROR

Selects utilities that ended with a non-zero return code and are not restartable.
NORESTART

Selects utilities that have had a cleanup performed or failed prior to starting any work that can be restarted.

RESTARTABLE

Selects utilities that are not currently running, that may be restarted with either RESTART or QUICKEXIT. These include utilities that are in must complete status.

MUSTCOMPLETE

Selects utilities that are in must complete status.

ACTIVE

Selects utilities that are currently executing.

QE

Selects utilities where a QUICKEXIT has been run.

QEFAILED

Selects utilities where a QUICKEXIT has been run, but which completed with a return code of greater than 4.

FORCEID

Selects utilities for which a FORCEID has been run.

RUNNING

Identical to ACTIVE. Selects utilities that are currently executing.

AGE

Selects utilities older than the specified number of days.

RPT

Default. Sends report results to the SUMMARY DD.

NOREPORT

Inhibits generation of report.

UNMATCHEDUIDS

Displays all unmatched utility IDs in the system (those in either of the SYSPDS or the checkpoint data sets, but not in the other). If this keyword is
coded with GENQE, then the XSUTDISP automation control point will be called in which FORCEID steps can be generated to remove the unmatched utility IDs.

For more details of XSUTDISP, see “XSUTDISP” on page 253.

**GENQE**

Causes the automation control point XSUTDISP to be invoked, and to generate a QUICKEXIT. For more details of XSUTDISP, see “XSUTDISP” on page 253.

**GENDELETE**

Causes the automation control point XSUTDISP to be invoked, and to generate DELETE statements. For more details of XSUTDISP, see “XSUTDISP” on page 253.

**ORDER BY**

Causes the report to be sorted by the specified field. You can sort by:
- UID
- STATUS
- STARTED
- ENDED
- JOBNAME

**Display Matrix report keywords**

**NGTDISP MATRIX**

produces a report of all entries in the matrix. Use to show any NGT locks held on objects by NGT utility IDs.

**Display Passwords report keywords**

**NGTDISP PASSWORDS**

Produces a report of all installed products with their passwords, expiration dates, and the number of days until those expiration dates.

**NGTDISP BMCUTIL**

Produces a report of the entries in the BMCUTIL table, you can use additional keywords to filter the entries that are displayed.
UID

Filters the entries displayed according to the user IDs defined by the \textit{pattern}.

UTILITY

Filters the entries displayed according to the utilities defined by the \textit{pattern}.

UTILITYID

Filters the entries displayed according to the utility IDs defined by the \textit{pattern}.

\textit{pattern}

Create a pattern using wildcards to filter the BMCUTIL results displayed in the report.

Display BMCSYNC report keywords

NGTDISP BMCSYNC

Produces a report of the entries in the BMCSYNC table, you can use additional keywords to filter the entries that are displayed.

UID

Filters the entries displayed according to the user IDs defined by the \textit{pattern}.

UTILITY

Filters the entries displayed according to the utilities defined by the \textit{pattern}.

UTILITYID

Filters the entries displayed according to the utility IDs defined by the \textit{pattern}.

\textit{pattern}

Create a pattern using wildcards to filter the BMCUTIL results displayed in the report.

NGTDISP report columns

This section describes the columns of the reports generated by NGTDISP.
This is a sample display utility report produced by NGTDISP.

You can also view this information in the BMCSYNC report. For more information, see “BMCUTIL report” on page 244.

### Display utility report columns

The columns in a NGTDISP report are as follows:

#### UTILITY-ID

Specifies the utility ID. Note that certain utility ID values may be preceded by an asterisk character in the first column, immediately preceding them (see the first three rows in the sample report in Display utility report columns This is a sample display utility report produced by NGTDISP. ) – this signifies the utility has completed. Utility ID values without an asterisk (such as for the last row in the sample report in Display utility report columns This is a sample display utility report produced by NGTDISP. ) were not fully completed at the time the report was compiled.

#### UTILITY STATUS

The status of the associated utility ID.

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPLETED OK</td>
<td>Return code of 0 or 4</td>
</tr>
<tr>
<td>RESTARTABLE</td>
<td>Utility ID is incomplete, can be restarted</td>
</tr>
<tr>
<td>FORCEID RUN</td>
<td>A FORCEID has been run on this utility ID</td>
</tr>
<tr>
<td>MUST COMPLETE</td>
<td>Utility ID is in a MUST COMPLETE state. Quickeexit or Restart must be run.</td>
</tr>
<tr>
<td>NO RESTART</td>
<td>Utility ID is not in a restartable state. Either a cleanup has occurred or utility work did not get started.</td>
</tr>
<tr>
<td>QEXIT ERROR</td>
<td>QUICKEXIT was attempted but encountered an error</td>
</tr>
<tr>
<td>QUICKEXIT</td>
<td>Utility ID completed via QUICKEXIT</td>
</tr>
</tbody>
</table>

The status of the associated utility ID.
### Status Description

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOB RUNNING</td>
<td>Utility ID was active when NGTDISP ran.</td>
</tr>
</tbody>
</table>

**STARTED**

The date and time at which the utility was started.

**ENDED**

The date and time at which the utility ended, if completed. If the utility is still active, this field will not have a value. Utilities with an abend (RC=255) will not have an end time.

**JOBNAME**

The name of the job associated with the utility ID.

**JOB #**

The job number associated with the utility ID.

**STEPNAME**

The step name associated with the utility ID.

**RET CODE**

The highest return code for the job step. The return code for abends is 255.

**DATASET QUALIFIER**

The high-level qualifier for the VSAM work data set.

### Matrix report columns

This is a sample SUMMARY report from a NGTDISP MATRIX. It shows what DB2 objects have a shared or exclusive lock within NGT. It also shows what UID is holding or waiting for that lock.

*Note*

You can also view this information in the BMCSYNC report. For more information, see “BMCSYNC report” on page 246.

The NGT Matrix is used to prevent two NGT utilities from processing the same object at the same time. This NGT Lock does not affect DB2 or other non-NGT object processing.
Passwords report

This is a sample SUMMARY report from a NGTDISP PASSWORDS. It shows the password for each NGT utility as well as when and how long before it expires.

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>LOCAL EXPIRE</th>
<th>EXPIRE</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>PASSWORD</td>
<td>PASSWORD DATE</td>
<td>DAYS</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>Check Index</td>
<td>A52EBF69B4B11860</td>
<td>10/15/2012</td>
<td>10</td>
</tr>
<tr>
<td>Copy</td>
<td>262119F194A8B1CA</td>
<td>10/15/2012</td>
<td>10</td>
</tr>
<tr>
<td>Load</td>
<td>251AA493B0F60A1E</td>
<td>10/15/2012</td>
<td>10</td>
</tr>
<tr>
<td>Rebuild Index</td>
<td>B59719264BA57BFO</td>
<td>10/15/2012</td>
<td>10</td>
</tr>
<tr>
<td>Restore</td>
<td>7051E12EB3D0301A</td>
<td>10/15/2012</td>
<td>10</td>
</tr>
<tr>
<td>Reorg</td>
<td>9E838D7CF6C9751F</td>
<td>10/15/2012</td>
<td>10</td>
</tr>
<tr>
<td>RTS</td>
<td>267B063ED4C6EA5</td>
<td>10/15/2012</td>
<td>10</td>
</tr>
<tr>
<td>Stats</td>
<td>BFFF7FB4116B9C9F</td>
<td>10/15/2012</td>
<td>10</td>
</tr>
<tr>
<td>Unlock</td>
<td>7F4B5B713F02FA51</td>
<td>10/15/2012</td>
<td>10</td>
</tr>
</tbody>
</table>

BMCUTIL report

The BMCUTIL table contains information about utilities that are currently running or started. The utilities use the table to control the use of utility IDs. Each BMC utility must have a unique ID for restart purposes. If you have more than one BMC utility installed, all of these utilities should share the same BMCUTIL table.

The BMCUTIL report contains the utility ID, utility type, execution status, restart option (for a restarted job), user ID, and starting timestamp of the utility.

Table 25: BMCUTIL report column values

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILID</td>
<td>CHAR(16)</td>
<td>Utility identifier</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| TYPE        | CHAR(8)   | Type of the executing utility:  
- CHECK  
- COPY  
- STATS  
- LOAD  
- NGT  
- RECOVER  
- REORG  
- UNLOAD |
| STATUS      | CHAR(1)   | Execution status of the utility:  
- A (active, not executing command)  
- I (initializing)  
- P (pausing or pause-stopped)  
- S (stopped)  
- T (terminating)  
- X (executing command) |
| PHASE       | CHAR(8)   | Current phase of the utility |
| RESTART     | CHAR(1)   | Restart option:  
- N (not restart)  
- P (RESTART(PHASE))  
- Y (RESTART) |
| ID          | CHAR(8)   | User ID executing the utility |
| START TIMESTAMP | TIMESTAMP | Starting timestamp of the utility |

**Figure 34: BMCUTIL sample report**

<table>
<thead>
<tr>
<th>UTILITY</th>
<th>UTILITY</th>
<th>STATUS</th>
<th>RESTART</th>
<th>USER</th>
<th>START TIMESTAMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFRC005A.SNAP2</td>
<td>COPY</td>
<td>S</td>
<td></td>
<td>MVSJDP3</td>
<td>2014-10-17-17.04.15</td>
</tr>
<tr>
<td>AMUIX68</td>
<td>LOAD</td>
<td>S</td>
<td>N</td>
<td>IDA</td>
<td>2014-10-16-12.45.44</td>
</tr>
<tr>
<td>BMCRECOVE1</td>
<td>COPY</td>
<td>S</td>
<td>N</td>
<td>RDATGP1</td>
<td>2014-11-20-10.40.30</td>
</tr>
<tr>
<td>JRG$JB.RG1</td>
<td>REORG</td>
<td>S</td>
<td>N</td>
<td>RDAJXB3</td>
<td>2014-07-23-12.35.28</td>
</tr>
<tr>
<td>LK1VC4K</td>
<td>REORG</td>
<td>S</td>
<td>N</td>
<td>RDABXN</td>
<td>2014-08-01-15.22.48</td>
</tr>
<tr>
<td>LKPLINV</td>
<td>REORG</td>
<td>S</td>
<td>N</td>
<td>RDABXN</td>
<td>2014-12-08-14.45.21</td>
</tr>
<tr>
<td>LKTTM01</td>
<td>REORG</td>
<td>S</td>
<td>N</td>
<td>RDABXN</td>
<td>2014-09-30-13.33.12</td>
</tr>
<tr>
<td>V686D003.BR</td>
<td>RECOVER</td>
<td>N</td>
<td></td>
<td>RDAJXM</td>
<td>2014-11-13-13.34.06</td>
</tr>
<tr>
<td>RGL000005</td>
<td>NGT</td>
<td>A</td>
<td>N</td>
<td>MYSRECI</td>
<td>2015-08-07-10.26.50</td>
</tr>
<tr>
<td>RGDSP900</td>
<td>NGT</td>
<td>A</td>
<td>N</td>
<td>MYSRECI</td>
<td>2015-08-07-16.10.27</td>
</tr>
</tbody>
</table>
BMCSYNC report

The BMCSYNC table contains information about the status of the objects that the currently executing utilities are accessing.

The BMCSYNC table synchronizes and controls access to DB2 spaces by concurrently executing BMC utility products. If you have more than one BMC utility installed, all of these utilities should share the same BMCSYNC table.

The BMCSYNC report contains the utility ID, utility type, object names, partition number, object type, and sharing-level access for each utility included in the report.

Table 26: BMCSYNC report column values

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILITY ID</td>
<td>CHAR(16)</td>
<td>Utility identifier</td>
</tr>
<tr>
<td>UTILITY TYPE</td>
<td>CHAR(8)</td>
<td>Name of the executing utility:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ CHECK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ COPY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ STATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ LOAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ NGT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ RECOVER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ REORG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ UNLOAD</td>
</tr>
<tr>
<td>DB/CRE8</td>
<td>CHAR(8)</td>
<td>Database name or creator name</td>
</tr>
<tr>
<td>SP/TBL/IX</td>
<td>CHAR(18)</td>
<td>Space, table, or index name</td>
</tr>
<tr>
<td>PART</td>
<td>SMALLINT</td>
<td>Physical partition number:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Null or 0 for a single data set nonpartitioned space</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Data set number for a multi-data-set, nonpartitioned space</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Partition number for a partitioned space (all products except NGT Recover and RECOVERY MANAGER) The value is null or 0 for any nonpartitioned space.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| KIND        | CHAR(2)   | Type of object:  
  ■ IP (index partition)  
  ■ IX (index)  
  ■ TB (table)  
  ■ TP (table space partition)  
  ■ TS (table space)  
  ■ DD, DW, D1, D2 (dynamic work file allocation)  
  ■ CI (copy information)  
  ■ RD (restart data set block) |
| SHARE LEVEL | CHAR(1)   | Degree to which utilities can share this object:  
  ■ Blank means that no status is requested, and any other utility can obtain any status.  
  ■ S allows sharing among any number of SHRLEVEL S utilities.  
  ■ X indicates that exclusive control is required. No other utility can run with SHRLEVEL X. |

**Figure 35: Sample BMCSYNC report**

<table>
<thead>
<tr>
<th>UTILITY ID</th>
<th>UTILITY TYPE</th>
<th>DB/CRE6</th>
<th>SP/TBL/IX</th>
<th>PART</th>
<th>KIND</th>
<th>LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGRGRL</td>
<td>REORG</td>
<td>DBVBREG</td>
<td>RGTS</td>
<td>1</td>
<td>TP</td>
<td>X</td>
</tr>
<tr>
<td>RGRGRL</td>
<td>REORG</td>
<td>DBVBREG</td>
<td>RGTS</td>
<td>1</td>
<td>LK</td>
<td>X</td>
</tr>
<tr>
<td>RGRGRL</td>
<td>REORG</td>
<td>DBVBREG</td>
<td>RGTS</td>
<td>2</td>
<td>TP</td>
<td>X</td>
</tr>
<tr>
<td>RGRGRL</td>
<td>REORG</td>
<td>DBVBREG</td>
<td>RGTSX1</td>
<td>2</td>
<td>LK</td>
<td>X</td>
</tr>
<tr>
<td>RGRGRL</td>
<td>REORG</td>
<td>DBVBREG</td>
<td>RGTSX1</td>
<td>3</td>
<td>TP</td>
<td>X</td>
</tr>
<tr>
<td>RGRGRL</td>
<td>REORG</td>
<td>DBVBREG</td>
<td>RGTS</td>
<td>4</td>
<td>TP</td>
<td>X</td>
</tr>
</tbody>
</table>

**Examples**

This section covers the syntax for specific examples of NGTDISP for displaying utilities. The other reports available with NGTDISP have few keywords and are self-explanatory.

**Display the status of all utilities**

To produce a report showing the status of all utilities, simply specify the NGTDISP keyword without any other keywords following it.

```bash
//SYSIN DD *
NGTDISP BMCUTIL UID(%STREORG%)
```
Display the status of a single utility ID (UID)

To produce a report showing the status of a single utility ID (UID), use the UTIL keyword.

```
//SYSIN DD *
NGT_DISP UTIL(BANKSTATMNTS)
NGT_DISP UTIL(WILMINGTON49)
NGT_DISP UTIL(CME18)
```

Display the status of utility IDs with a common naming pattern

You can also check the status of a set of similarly-named utility IDs by using a naming pattern mask. The wildcards % or * may be used.

```
//SYSIN DD *
NGT_DISP UTIL(SWREGION%)
NGT_DISP UTIL(*A320)
NGT_DISP UTIL(%STREORG%)
```

Display the status of only completed utilities

To produce a report showing the status of only completed utilities, specify the COMPLETED keyword (RC = 0)

```
//SYSIN DD *
NGT_DISP COMPLETED
```

You can combine this keyword with the UTIL keyword to check for completed utilities with a common naming pattern, as in:

```
//SYSIN DD *
NGT_DISP UTIL(AMBACKUPS%) COMPLETED
```

Display the status of only utilities in error

To produce a report showing the status of only utilities that did not complete with a return code of 0 or 4, specify the ERROR keyword.

```
//SYSIN DD *
NGT_DISP ERROR
```

You can combine this keyword with the UTIL keyword to check for utilities in error with a common naming pattern, as in:

```
//SYSIN DD *
NGT_DISP UTIL(COPIES909%) ERROR
```
**Display the status of only restartable utilities**

To produce a report showing the status of only restartable utilities, use the **RESTARTABLE** keyword.

```
//SYSIN DD *
NGTDISP RESTARTABLE
```

You can combine this keyword with the **UTIL** keyword to check for utilities in error with a common naming pattern, as in:

```
//SYSIN DD *
NGTDISP UTIL(TXREGION%) RESTARTABLE
```

**Display the status of utilities currently running**

NGTDISP can be run while utilities are active in other jobs. To produce a report showing the status of all active utilities, specify the **ACTIVE** keyword.

```
//SYSIN DD *
NGTDISP ACTIVE
```

You can combine this keyword with the **UTIL** keyword to check for active utilities with a common naming pattern, as in:

```
//SYSIN DD *
NGTDISP UTIL(COPYPACIF%) ACTIVE
```

You can also use the keyword **RUNNING** if you prefer, to display all running utilities, as in:

```
//SYSIN DD *
NGTDISP RUNNING
```

Or, for a list of all running utilities meeting a specific naming pattern:

```
//SYSIN DD *
NGTDISP UTIL(%PETRO55) RUNNING
```

**Note**

The **ACTIVE** and **RUNNING** keywords are identical in meaning.

**Display the status of utilities that have had Quickexit used**

You can also print the status of utilities that have had a **QUICKEXIT** run against them. This is accomplished with the **QE** keyword.

```
//SYSIN DD *
NGTDISP QE
```
You can combine this keyword with the UTIL keyword to check for utilities with a common naming pattern that have had a QUICKEXIT, as in:

```
//SYSIN DD *
NGTDISP UTIL(*BATCH*) QE
```

**Display the status of utilities that have had Quickexit used, but failed**

In addition, you can print the status of utilities that have had a QUICKEXIT run against them, but which had a return code of greater than 4. This is accomplished with the QEFAILED keyword.

```
//SYSIN DD *
NGTDISP QEFAILED
```

Again, you can combine this keyword with the UTIL keyword to check for utilities with a common naming pattern that have had a QUICKEXIT failure, as in:

```
//SYSIN DD *
NGTDISP UTIL(RRG%) QEFAILED
```

**Display the status of all utilities that have been forced (FORCEID)**

To produce a report showing the status of all utilities that have been forced to exit by use of the FORCEID parameter on the EXEC statement, use the FORCEID keyword.

```
//SYSIN DD *
NGTDISP FORCEID
```

The FORCEID keyword can be combined with the UTIL keyword to check for utilities with a common naming pattern that have had a forced exit, as in:

```
//SYSIN DD *
NGTDISP UTIL(HVY%) FORCEID
```

**Display the status of all utilities a certain number of days old**

You can display the status of all utilities a certain number of days old by using the AGE parameter. For example, to check utilities that are at least a week old, or 7 days old, you would type:

```
//SYSIN DD *
NGTDISP AGE(7)
```
To check utilities that are at least a two weeks old, or 14 days old, you would type:

```//SYSIN DD *
NGTDISP AGE(14)``

The AGE keyword can be combined with the UTIL keyword to check for utilities with a common naming pattern that are of a certain age, as in:

```//SYSIN DD *
NGTDISP UTIL(STM%) AGE(3)
NGTDISP UTIL(EEH%) AGE(5)
NGTDISP UTIL(DELIV%) AGE(21)``

**Display the status of utilities with multiple criteria**

NGTDISP is very flexible. You can specify multiple criteria for printing utilities. Here are some examples combining more than two selection criteria.

This first example displays utility IDs starting with R55 for which the QUICKEXIT failed, and which are at least 7 days old.

```//SYSIN DD *
NGTDISP UTIL(R55%) QEFAILED AGE(7)``

This second example displays all utility IDs that are currently running, restartable, or completed.

```//SYSIN DD *
NGTDISP RUNNING RESTARTABLE COMPLETED`'

This same example with a naming pattern would look like this:

```//SYSIN DD *
NGTDISP UTIL(EASTC%) RUNNING RESTARTABLE COMPLETED`'

**Display all entries from the BMCUTIL table**

To produce a report showing all entries from the BMCUTIL table, simply specify the BMCUTIL keyword without any other keywords following it.

```//SYSIN DD *
NGTDISP BMCUTIL`'
Display the status of utility IDs with a common naming pattern from the BMCUTIL table

To produce a report from the BMCUTIL table, specify the BMCUTIL keyword followed by the optional UID|UTILITY|UTILITYID keywords and the naming pattern mask.

//SYSIN DD *
NGTDISP BMCUTIL UID(%STREORG%)

Display all entries from the BMCSYNC table

To produce a report showing all entries from the BMCUTIL table, simply specify the BMCSYNC keyword without any other keywords following it.

//SYSIN DD *
NGTDISP BMCSYNC

Display the status of utility IDs with a common naming pattern from the BMCSYNC table

To produce a report from the BMCSYNC table, specify the BMCSYNC keyword followed by the optional UID|UTILITY|UTILITYID keywords and the naming pattern mask.

//SYSIN DD *
NGTDISP BMCSYNC UID(%STREORG%)

Relevant automation

NGT automation control points are integrated into all NGT utilities and can greatly enhance and expand your processing options. This chapter describes the automation exits that BMC recommends that you incorporate into your processing, though NGT Automation control points are optional.

The presence of the NGTAUTO DD statement triggers automation control point processing, as in this example:

//NGTAUTO DD DISP=SHR,DSN=NGT.AUTO.CTRL.PTS

The automation control point data set will contain one member with each of the exits you choose to use.

In listing the exits below, be aware that these are not the only exits you can use with this product. These should serve only as a basic recommendation for automation.
This recommendation should be used to spawn more ideas or give insight into how others are utilizing this powerful facility. You may incorporate the use of any of the automation control point you wish for whatever additional uses you require. Furthermore, where this section lists more than one automation control point, you are not constrained to using all such listed exits as a set; you may exclude as many of the listed automation control points as you need.

The following standard NGT Automation control points may be used with this utility:

<table>
<thead>
<tr>
<th>NGT Automation control points</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSUTDISP</td>
<td>Creates QUICKEXIT and DELETE statements. For more information, see “XSUTDISP” on page 253.</td>
</tr>
<tr>
<td>XSUTGLOB</td>
<td>Sets global variables that are then used by other automation points. Also can be used to set the value of certain NGT parameters.</td>
</tr>
<tr>
<td></td>
<td>For more information, see the BMC Next Generation Technology Automation Reference Manual.</td>
</tr>
<tr>
<td>XSUT0000</td>
<td>Called before any processing starts. Aborts a run before it starts.</td>
</tr>
<tr>
<td>XSUTSYIN</td>
<td>Called before the parser reads SYSIN. It has full access to the SYSIN and can modify it. Overrides or disallows some input parameters.</td>
</tr>
<tr>
<td>XSUTTERM</td>
<td>Called after processing has finished. Used, for example, to insert run statistics into a DB2 table or to e-mail an audit report at the completion of processing, automatically.</td>
</tr>
</tbody>
</table>

**XSUTDISP**

A sample XSUTDISP is supplied in the NGT CNTL library. This XSUTDISP is required to use NGTDISP to generate QUICKEXIT or DELETE jobs.

XSUTDISP is used to trigger QUICKEXITS, DELETES, or other actions in processing, it is called when the keywords GENQE or GENEDELETES are encountered in the SYSIN for any utility job.

**Table 27: Variables**

<table>
<thead>
<tr>
<th>Name</th>
<th>Contains</th>
<th>Permitted values</th>
</tr>
</thead>
</table>
| action| Code to trigger either the generation of QUICKEXITS or DELETES. | GEND  
<p>|       |                                       | GENQ             |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Contains</th>
<th>Permitted values</th>
</tr>
</thead>
<tbody>
<tr>
<td>uid</td>
<td>Utility ID.</td>
<td></td>
</tr>
<tr>
<td>jobname</td>
<td>Utility job name.</td>
<td></td>
</tr>
<tr>
<td>stepname</td>
<td>Utility step name.</td>
<td></td>
</tr>
<tr>
<td>jobid</td>
<td>Utility job ID.</td>
<td></td>
</tr>
<tr>
<td>utility_rc</td>
<td>Utility return code.</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>Status value</td>
<td>RESTARTABLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUICKEXIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JOB RUNNING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMPLETED OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FORCEID RUN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QEXIT ERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NO RESTART</td>
</tr>
<tr>
<td>work_prefix</td>
<td>Utility work data set</td>
<td></td>
</tr>
<tr>
<td></td>
<td>prefix</td>
<td></td>
</tr>
<tr>
<td>start_date</td>
<td>Utility start date.</td>
<td></td>
</tr>
<tr>
<td>start_time</td>
<td>Utility start time.</td>
<td></td>
</tr>
<tr>
<td>end_date</td>
<td>Utility end date.</td>
<td></td>
</tr>
<tr>
<td>end_time</td>
<td>Utility end time.</td>
<td></td>
</tr>
</tbody>
</table>

All of the variables above are issued in stems. For details on stem variables, see the IBM TSO/E REXX Reference.

For more information on the use of automation in your processing, see the *BMC Next Generation Technology Automation Reference Manual*
Using NGT1DBUT to copy NGT VSAM work files

You can use the NGT1DBUT program to copy the NGT VSAM work files (.OUT and .INFO) to sequential data sets. You can then view or send these sequential data sets to BMC Customer Support for further analysis. NGT1DBUT supports two SYSIN statements, one to dump the work files, and one to extract them.

NGT1DBUT syntax

DBUT DUMP utility command syntax

```
DBUT DUMP - JOBNAME(jobName) -JESID(jesID)

OUTDD (OUTFILE)
OUTDD(outDD)

JOBNAME
(required) A 1- to 8-byte job name.

JESID
(required) An 8-byte JES ID.
```
**STEPNAME**


**OUTDD**

*(optional)* A 1- to 8-byte DD name in the JCL. The default ddname is OUTFILE.

---

**DBUT EXTRACT utility command syntax**

```
DBUT EXTRACT  FILE  ( INFO  JRNL )
```

```
INDD (INFLE)

INDD(inDD)
```

```
OUTDD (OUTFILE)

OUTDD(outDD)
```

**FILE**

*(required)* Which NGT work file to extract, INFO or JRNL. JRNL, known as the Journal, refers to the .OUT work file.

**INDD**

*(optional)* The NGT work file data set name. The default ddname is INFILE.

**OUTDD**

*(optional)* A 1- to 8-byte DD name in the JCL. The default ddname is OUTFILE.

---

**NGT1DBUT usage notes**

Take note of the following considerations when you configure the NT1DBUT command.

- The optional STEPNAME keyword allows a single step of a multi-step job to be DUMPed, independently of other steps within the job.
The STEPNAME keyword can be omitted or present for single step jobs, but if omitted for multistep jobs, the entire job including all associated steps will be DUMPed.

If multiple steps within the same job are to be dumped, then it is required that these be dumped to separate output data sets, because of the excessive amounts of data, and in order to facilitate ease of diagnosis by development.

In data sharing installations in which two or more DB2 subsystems are shared under a common group name, the group name should be specified rather than the name of an individual DB2 subsystem in the group.

NGT1DBUT sample JCL

This is an example of a NGT1DBUT JCL job.

```cbas
//NGT1DBUT JOB . . .
//JOBLIB DD DISP=SHR, DSN=your.ngt.loadlib
//STEP01 EXEC PGM=NGT1DBUT, PARM='DSNA,,NORESTART'
//SYSPRINT DD SYSOUT=* 
//OUTFILE DD DISP=(,CATLG), DSN=your.journal.datasetname,
// UNIT=SYSDA, SPACE=(CYL,(5,50),RLSE) 
//SYSSIN DD *
//DBUT DUMP JOBNAME(jobname) JESID(JOB31569)

//NGT1DBUT JOB . . .
//JOBLIB DD DISP=SHR, DSN=your.ngt.loadlib
//NGT1DBUT EXEC PGM=NGT1DBUT, PARM='DSNA,,NORESTART'
//CDBPRINT DD SYSOUT=* 
//INFILE DD DISP=SHR, DSN=workprefix.MMBNINJG.OGAEFLAA.OUT001
//OUTFILE DD UNIT=SYSDA, DISP=(NEW,CATLG,DELETE),
// DSN=your.journal.datasetname,
// SPACE=(TRK,(1,5),RLSE) 
//SYSSIN DD *
//DBUT EXTRACT FILE(JRNL)

//
```

Appendix A  Using NGT1DBUT to copy NGT VSAM work files 257
Index

_Index( ) 38
&JT 132
&MJ 132
&MJC 132
&MU 132
&N 132
&SPEc 132
+AGELIMIT 147
+ALERROR 147
+ALLO 33
+CHECKDEFERRED 147
+CLEANUP 147
+CONTROLPREFIX 147, 212
+COPYPENDING 147
+DEADLINE 147
+DELETEOUTPUT 147
+DELETEWORK 147
+DIGITS 147
+DRAIN1 147
+DSCRL 147
+DSNUTILB 147
+EDPROCSUP 147
+FASTSWITCH 147
+FLASHCOPY 147
+HRECALL 147
+IGNOBJSETPARTS 147
+JOURNAL 148
+LARGEOBJSIZE 148
+LOGPDS 148
+MAXCONNECT 148
+MAXIDLE 148
+MAXMEMORY 148
+MAXSERVERS 148
+MAXWAIT 148
+NGTAUTO1 148
+NGTAUTO2 148
+NGTCKPT 148
+NGTSVR 148
+OBJESETSTOGROUP 148
+ONERROR 148
+OUTPUTAGE 148, 212
+OVERRIDEOUTPUT 148
+PARMLIST 148
+PLAN 148
+PSWDWTO 148
+QRETRY 148
+RACFID 148
+RERUN 148
+REXXBUFR 148
+REXXTIME 149
+SQI.LID 149
+SVRDISK 149
+SVRDISKJC 149
+SVRJCLDISK 149
+SVRJCLJOB 149
+SVRJCLTAPE 149
+SVRMODE 149
+SVRPROC 149
+SVRPROCMSGCLAS 149
+SVRPROCMSGCLVL 149
+SVRPROCT 149
+SVRSPECIAL 149
+SVRTAPE 149
+SVRTAPEJC 149
+SVRTYPE 149
+SYSINERROR 149
+SYSINORDER 149
+TIMEEXEC 149
+UIDHISTORY 149
+USER1 149
+USER2 149
+VOLUMES 33
+WAITTIMEOUT 149
+WORKMODE 149
+WORKPREFIX 107, 149
+XBMID 149
0C3
  abend  232

A
  A/B name switching  167
  ABEND  110
  access, shared  20
  AGEREPRT  109, 215
  ALLO  33, 35
  AMS  35
  AMS Define  39
  ARCHIVE  41

B
  BLOB  61
  BMC utilities
    running concurrently  24
  BMCSYNC table
    contents  20
    maintaining  23
    running utilities concurrently  24
  BMCUTIL table
    contents  18
    maintaining  20
  BUFNO
    OUTPUT command  118

C
  CATLG
    OUTPUT command  116
  CDBPRINT  109
  CDBR858  160
  changing global parameters  146
  checkpoint  232
  CLOB  61, 89
  COALESCE function  88
  comments in SYSIN  111
  conventions, documentation  12
  creator ID  15

D
  DATACLAS
    OUTPUT command  118
    date durations  93
  DAYOFMONTH function  61
  DAYOFWEEK function  61
  DAYOFYEAR function  62
  DB2  35
  DB2 operand  33
  DB2-managed  32
  DBCLOB  61
  DBMG CHECKIX  231
  DBMG COMPACT  231
  DBMG OPTIMIZE  231
  DBMG REORG  231
  DBMG REPORT  231
  DECIMAL  94, 96
  DEFINE data sets  32
  DFDSS  232
  DIGITS function  65
  DISKEXP
    OUTPUT command  127
  DISKRETN
    OUTPUT command  126
    documentation information  11
  DRAIN(ALL)  160
  DSNAME
    OUTPUT command  114
  DSSNAP
    OUTPUT command  123
    durations
      date  93
      labeled  92
      time  93
      timestamp  93
    Dynamic Allocation  38

E
  EATTR
    OUTPUT command  127
  EDITPLIB  109
  ENCIPHER
    OUTPUT command  117
  ESCAPE clause  41
  EXEC  134
  EXEC statement  105
EXPDT
  OUTPUT command 122
EXPOUT
  OUTPUT command 116

F
FDR 232
FLOAT 94, 96
FORCEID 107
  function
    trace 100
  functions
    COALESCE 88
    DAYOFMONTH 61
    DAYOFWEEK 61
    DAYOFYEAR 62
    DIGITS 65
    IFNULL 88
    SOUNDEX 79
    SUBSTR 80
    VALUE 88
    WEEK 89

G
GLOBAL 39

H
high-used RBA 35

I
I/J name switching 167
ICBackupSpec 27
IFNULL function 88
IN predicate 96
  index on expression 41
INTEGER 96
IS 41
IS NOT 41

J
JCL procedures 103
JOB statement 104

K
KEEPDB2 33, 35

L
labeled durations 92
LBVOLS
  OUTPUT command 125
LIKE clause 41
LIKE keyword 15
LIKE predicate 99
LOGPDS 109
LPVOLS
  OUTPUT command 124

M
MANAGEMENTCLASS 33
MAXPRIM
  OUTPUT command 124
MGMTCLAS
  OUTPUT command 119
MIGRATE
  OUTPUT command 123
  mini-dump 232
Model AMS DEFINE 33
MODELDCB
  OUTPUT command 117
multitasking 17
Must Complete state 110
MVS JCL SPACE 35
MXIG 33, 35

N
NBRSECD
  OUTPUT command 124
NGTAPPL 128
NGTAUTO 38, 108
NGTCKPT 233
NGTPBG 112
NGTRTS 128
NORESTART 107
NULL/NOT NULL predicate 99
O

OUTPUT command
BUFNO 118
CATLG 116
DATACLAS 118
DISKEXPD 127
DISKRETN 126
DSNAME 114
DSSNAP 123
EATTR 127
ENCIPHER 117
EXPDT 122
EXPOUT 116
LBVOLS 125
LPVOLS 124
MAXPRIM 124
MGMTCLAS 119
MIGRATE 123
MODELDCB 117
NBRSECD 124
PCTPRIM 123
RBVOLS 126
REALDD 121
RETPD 122
RPVOLS 125
SPACE 123
STACK 121
STACK CABINET 121, 123
STORCLAS 118
TRTCH 121
UNIT 114
UNITCNT 119
VOLCNT 117
VOLUMES 126

P

parallelism 17
parameters for a particular utility 108
PBRPARTS 112
PCTPRIM
  OUTPUT command 123
PDS 232
PDSE 232
PQTY 32, 35
predicates

Q

QUICKEXIT 106, 110

R

RBVOLS
  OUTPUT command 126
REALDD
  OUTPUT command 121
REGION 104
related publications 11
REPRO 232
reserved words 96
RESTART 105
RESTART(PHASE) 105
restoration 233
RETPD
  OUTPUT command 122
REUSE 33, 35
RPVOLS
  OUTPUT command 125
RTS Tables 233
running BMC utilities concurrently 24
RUNSTATS 109, 134

S

sample JCL skeleton 133
secondary quantity 32
secqty 32
server types 129
SHRLEVEL 20
SORT 104
SOUNDEX function 79
SPACE
  OUTPUT command 123
SQL in SYSIN 111
SQL statements
deleing rows from the BMCSYNC table 20, 23
SQTQ 32
SSID 105
STACK
  OUTPUT command 121
STACK CABINET
  OUTPUT command 121, 123
STEPLIB 103, 108, 134
STOGROUP 32
STORCLAS
  OUTPUT command 118
SUBSTR function 80
SUMMARY 15, 109, 134
SVC 8 109
SVRDISK 129
symbolic variables 212
SYSERROR 109, 134
SYSIN 108
SYSIN comments 111
SYSIN SQL 111
SYSIN USERCMD 111
SYSPDS 39
SYSPRINT 15, 108, 134
SYSTSPRT 109, 135

T
  tables, BMC
    BMCSYNC 20
    BMCUTIL 18
  TESTUID 107
  time durations 93
  timestamp durations 93
  trace function 100
  TRTCH
    OUTPUT command 121

U
  UID 105
  UNIT
  OUTPUT command 114
  UNITCNT
    OUTPUT command 119
  UNLOAD 41
  updating the DB2 Catalog 32
  USED 33, 35
  user-managed 32
  USERCMD in SYSIN 111
  UTLPARMS 108
  UPRINT 135

V
  VALUE function 88
  VOLUMES
    OUTPUT command 117
    OUTPUT command 126
  VSAM catalog 35
  VSAM cluster 35
  VSAM datasets 232

W
  WEEK function 89
  WHERE clause 15, 41, 100
  wildcards 15

X
  XSUTALLO 38