BMC Next Generation Technology
General User Guide

Supporting

Version 11.2 of BMC Next Generation Technology Check for DB2 for z/OS
Version 11.2 of BMC Next Generation Technology Load for DB2 for z/OS
Version 11.2 of BMC Next Generation Technology LOBMaster for DB2 for z/OS
Version 11.2 of BMC Next Generation Technology Reorg for DB2 for z/OS
Version 11.2 of BMC Next Generation Technology Stats for DB2 for z/OS
Version 11.2 of BMC Next Generation Technology Unload for DB2 for z/OS
Version 11.2 of BMC Next Generation Technology Utility Manager for DB2 for z/OS

Revised February 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

Address  
BMC SOFTWARE INC  
2101 CITYWEST BLVD  
HOUSTON TX 77042-2827 USA

Telephone  
1 713 918 8800  
or  
1 800 841 2031

Fax  
1 713 918 8000

Outside United States and Canada

Telephone  
+01 713 918 8800

Fax  
+01 713 918 8000

© Copyright 2015 BMC Software, Inc.

BMC, BMC Software, and the BMC Software logo are the exclusive properties of BMC Software, Inc., are registered with the U.S. Patent and Trademark Office, and may be registered or pending registration in other countries. All other BMC trademarks, service marks, and logos may be registered or pending registration in the U.S. or in other countries. All other trademarks or registered trademarks are the property of their respective owners.

DB2, FlashCopy, IBM, 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 UNPUBLISHED—RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES.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, 2101 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
# Table of Contents

<table>
<thead>
<tr>
<th>Chapter 1</th>
<th>NGT Utility Concepts</th>
<th>1-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Wildcards</td>
<td>1-1</td>
<td></td>
</tr>
<tr>
<td>1.2 Utility Parameters</td>
<td>1-2</td>
<td></td>
</tr>
<tr>
<td>1.3 Journal</td>
<td>1-2</td>
<td></td>
</tr>
<tr>
<td>1.4 Applications</td>
<td>1-2</td>
<td></td>
</tr>
<tr>
<td>1.4.1 DEFINE</td>
<td>1-3</td>
<td></td>
</tr>
<tr>
<td>1.4.2 DELETE</td>
<td>1-7</td>
<td></td>
</tr>
<tr>
<td>1.4.3 RESPECIFY</td>
<td>1-8</td>
<td></td>
</tr>
<tr>
<td>1.4.4 UPDATE</td>
<td>1-9</td>
<td></td>
</tr>
<tr>
<td>1.4.5 DISPLAY</td>
<td>1-10</td>
<td></td>
</tr>
<tr>
<td>1.5 Servers</td>
<td>1-12</td>
<td></td>
</tr>
<tr>
<td>1.6 The Matrix</td>
<td>1-12</td>
<td></td>
</tr>
<tr>
<td>1.6.1 Matrix Control</td>
<td>1-12</td>
<td></td>
</tr>
<tr>
<td>1.7 Tape Commands</td>
<td>1-13</td>
<td></td>
</tr>
<tr>
<td>1.7.1 CDBTAPE</td>
<td>1-13</td>
<td></td>
</tr>
<tr>
<td>1.8 WHERE Clause Differences From DB2</td>
<td>1-20</td>
<td></td>
</tr>
<tr>
<td>1.8.1 Numeric comparisons</td>
<td>1-25</td>
<td></td>
</tr>
<tr>
<td>1.8.2 DECIMAL comparisons</td>
<td>1-25</td>
<td></td>
</tr>
<tr>
<td>1.8.3 Numeric String Comparisons</td>
<td>1-26</td>
<td></td>
</tr>
<tr>
<td>1.8.4 Comparing constants</td>
<td>1-26</td>
<td></td>
</tr>
<tr>
<td>1.8.5 Odd decimal precision</td>
<td>1-26</td>
<td></td>
</tr>
<tr>
<td>1.8.6 Labeled durations</td>
<td>1-26</td>
<td></td>
</tr>
<tr>
<td>1.8.7 Date durations</td>
<td>1-27</td>
<td></td>
</tr>
<tr>
<td>1.8.8 TIME durations</td>
<td>1-27</td>
<td></td>
</tr>
<tr>
<td>1.8.9 TIMESTAMP durations</td>
<td>1-27</td>
<td></td>
</tr>
<tr>
<td>1.8.10 DAYOFMONTH function</td>
<td>1-27</td>
<td></td>
</tr>
<tr>
<td>1.8.11 DAYOFWEEK function</td>
<td>1-28</td>
<td></td>
</tr>
<tr>
<td>1.8.12 DAYOFYEAR function</td>
<td>1-28</td>
<td></td>
</tr>
<tr>
<td>1.8.13 WEEK function</td>
<td>1-28</td>
<td></td>
</tr>
<tr>
<td>1.8.14 Restriction on the usage of NULL/NOT NULL predicate</td>
<td>1-29</td>
<td></td>
</tr>
<tr>
<td>1.8.15 DIGITS Function</td>
<td>1-29</td>
<td></td>
</tr>
<tr>
<td>1.8.16 LIKE Predicate</td>
<td>1-29</td>
<td></td>
</tr>
</tbody>
</table>
# Table Of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.8.17</td>
<td>VALUE Function</td>
<td>1-29</td>
</tr>
<tr>
<td>1.8.18</td>
<td>SOUNDEX Function</td>
<td>1-29</td>
</tr>
<tr>
<td>1.8.19</td>
<td>SUBSTR Function</td>
<td>1-30</td>
</tr>
<tr>
<td>1.8.20</td>
<td>EURO Conversion Function</td>
<td>1-30</td>
</tr>
<tr>
<td>1.8.21</td>
<td>NOT Operator</td>
<td>1-31</td>
</tr>
<tr>
<td>1.8.22</td>
<td>Reserved Words</td>
<td>1-31</td>
</tr>
<tr>
<td>1.8.23</td>
<td>IN Predicate</td>
<td>1-31</td>
</tr>
<tr>
<td>1.8.24</td>
<td>Trace Function</td>
<td>1-33</td>
</tr>
<tr>
<td>1.8.25</td>
<td>Subselects</td>
<td>1-34</td>
</tr>
<tr>
<td>1.9</td>
<td>Allocation Strategies</td>
<td>1-34</td>
</tr>
<tr>
<td>1.9.1</td>
<td>Parameters</td>
<td>1-34</td>
</tr>
<tr>
<td>1.9.2</td>
<td>Automation Control Points (Exits)</td>
<td>1-36</td>
</tr>
<tr>
<td>1.9.3</td>
<td>Space-Specific Defines</td>
<td>1-37</td>
</tr>
<tr>
<td>1.10</td>
<td>Managing the Model AMS define</td>
<td>1-39</td>
</tr>
<tr>
<td>1.10.1</td>
<td>CDBAMS DEFINE</td>
<td>1-39</td>
</tr>
<tr>
<td>1.10.2</td>
<td>CDBAMS DELETE</td>
<td>1-39</td>
</tr>
<tr>
<td>1.10.3</td>
<td>CDBAMS DISPLAY</td>
<td>1-39</td>
</tr>
</tbody>
</table>

## Chapter 2  JCL Setup

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>JCL Overview</td>
<td>2-1</td>
</tr>
<tr>
<td>2.2</td>
<td>Job Completion Codes/Return Codes</td>
<td>2-7</td>
</tr>
<tr>
<td>2.3</td>
<td>SYSSIN statements other than Utility Statements</td>
<td>2-8</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Comment</td>
<td>2-8</td>
</tr>
<tr>
<td>2.3.2</td>
<td>SQL</td>
<td>2-8</td>
</tr>
<tr>
<td>2.3.3</td>
<td>DB2 Commands</td>
<td>2-8</td>
</tr>
<tr>
<td>2.3.4</td>
<td>USERCMD</td>
<td>2-8</td>
</tr>
<tr>
<td>2.3.5</td>
<td>CDBPBG</td>
<td>2-9</td>
</tr>
<tr>
<td>2.3.6</td>
<td>Other</td>
<td>2-9</td>
</tr>
</tbody>
</table>

## Chapter 3  Servers

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Server Types</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2</td>
<td>Defining Servers</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Number of Servers</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Server Job Types: Batch Vs. Started Tasks</td>
<td>3-1</td>
</tr>
<tr>
<td>3.3</td>
<td>Defining Server JCL</td>
<td>3-1</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Batch Server Jobs</td>
<td>3-2</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Started Task Server Jobs</td>
<td>3-3</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Literals for Job and Step Name</td>
<td>3-3</td>
</tr>
<tr>
<td>3.4</td>
<td>Servers and Data Sharing</td>
<td>3-6</td>
</tr>
</tbody>
</table>
Table Of Contents

3.5 Servers Work Distribution and Balance of Work ........................................ 3-6
3.6 Server Modes and the +SVRMODE Parameter ........................................ 3-7
3.7 Reducing the number of Server jobs ..................................................... 3-8

Chapter 4 DB2 Real Time Statistics (RTS) .................................................... 4-1
4.1 NGT Copy ......................................................................................... 4-2
4.2 NGT Reorg ....................................................................................... 4-3
4.3 NGT Load ......................................................................................... 4-5
4.4 NGT Stats ....................................................................................... 4-7

Chapter 5 Parameters ................................................................................. 5-1
5.1 Setting Default Parameters ................................................................. 5-1
5.2 Parameter Syntax Rules ....................................................................... 5-1
5.3 Global Utility Parameters (UTLPARMS) ............................................ 5-3
5.4 Global Utility Alphabetical Parameter Reference ............................... 5-9
5.4.1 +AGELIMIT ................................................................................. 5-9
5.4.2 +CHECKDEFERRED ................................................................. 5-10
5.4.3 +CLEANUP ................................................................................ 5-11
5.4.4 +CONTROLPREFIX ................................................................. 5-12
5.4.5 +COPYPENDING ....................................................................... 5-13
5.4.6 +DEADLINE ............................................................................... 5-14
5.4.7 +DELETEOUTPUT ...................................................................... 5-15
5.4.8 +DELETEWORK ......................................................................... 5-16
5.4.9 +DIGITS ...................................................................................... 5-17
5.4.10 +DRAIN1 ................................................................................... 5-18
5.4.11 +DSCRL .................................................................................... 5-19
5.4.12 +EDPROCSUP .......................................................................... 5-20
5.4.13 +FASTSWITCH ........................................................................ 5-21
5.4.14 +FLASHCOPY .......................................................................... 5-22
5.4.15 +HRECALL ............................................................................... 5-23
5.4.16 +JOURNAL ............................................................................... 5-24
5.4.17 +LARGEOBJSIZE ...................................................................... 5-26
5.4.18 +MAXCONNECT ....................................................................... 5-27
5.4.19 +MAXIDLE ............................................................................... 5-28
5.4.20 +MAXMEMORY ....................................................................... 5-29
5.4.21 +MAXSERVERS ................................................................. 5-30
5.4.22 +MAXWAIT .............................................................................. 5-31
5.4.23 +MEMORYLIMIT ................................................................. 5-32
5.4.24 +MONITORRATE ..................................................................... 5-33
5.4.25 +ONERROR .............................................................................. 5-34
Table Of Contents

5.4.26 +OUTPUTAGE……………………………………………………………………………… 5-35
5.4.27 +PARMLIST……………………………………………………………………………… 5-36
5.4.28 +PLAN ………………………………………………………………………………… 5-37
5.4.29 +PSWDWTO ……………………………………………………………………………… 5-38
5.4.30 +QRETRY ………………………………………………………………………………… 5-39
5.4.31 +RACFID ………………………………………………………………………………… 5-40
5.4.32 +RERUN ………………………………………………………………………………….. 5-41
5.4.33 +REXXBUFR …………………………………………………………………………….. 5-42
5.4.34 +REXXTIME …………………………………………………………………………….. 5-43
5.4.35 +SQLID …………………………………………………………………………………. 5-44
5.4.36 +SVRDISK ……………………………………………………………………………… 5-45
5.4.37 +SVRDISKJC …………………………………………………………………………….. 5-46
5.4.38 +SVRJCLDISK …………………………………………………………………………… 5-47
5.4.39 +SVRJCLJOB …………………………………………………………………………… 5-48
5.4.40 +SVRJCLTAPE ………………………………………………………………………….. 5-49
5.4.41 +SVRMODE ……………………………………………………………………………… 5-50
5.4.42 +SVRPROC ……………………………………………………………………………… 5-51
5.4.43 +SVRPROCMSGCLAS ………………………………………………………………… 5-52
5.4.44 +SVRPROCMSGGLVL ………………………………………………………………… 5-53
5.4.45 +SVRPROC ……………………………………………………………………………… 5-54
5.4.46 +SVRSPECIAL ………………………………………………………………………….. 5-55
5.4.47 +SVRTAPE ………………………………………………………………………………… 5-56
5.4.48 +SVRTAPEJC ……………………………………………………………………………. 5-57
5.4.49 +SVRTYPE ……………………………………………………………………………… 5-58
5.4.50 +SYSINERROR ………………………………………………………………………….. 5-59
5.4.51 +SYSINORDER …………………………………………………………………………. 5-60
5.4.52 +TIMEEXEC …………………………………………………………………………… 5-61
5.4.53 +UIDHISTORY ………………………………………………………………………….. 5-62
5.4.54 +WAITTIMEOUT ………………………………………………………………………… 5-63
5.4.55 +WORKMODE ………………………………………………………………………….. 5-64
5.4.56 +WORKPREFIX …………………………………………………………………………. 5-65
5.4.57 +WORKUNIT …………………………………………………………………………… 5-66

Chapter 6  NGT Utility Reports ………………………………………………………………………… 6-1
6.1  NGT Utility Age Report …………………………………………………………………… 6-1
6.1.1  Age Report Fields ……………………………………………………………………… 6-1
6.2  OBJREPRT (Processed Objects Report) ………………………………………………… 6-2

Chapter 7  NGT Work Dataset Control ……………………………………………………………… 7-1
Table Of Contents

7.1 Overview of the NGT Work Datasets ................................................................. 7-1
7.2 Naming the NGT Work Datasets ...................................................................... 7-1
7.3 Controlling when NGT Work Datasets are Deleted ......................................... 7-1
7.4 Usage Notes for NGT Work Datasets ............................................................... 7-2
7.5 VSAM Work Datasets Deletion Review ......................................................... 7-2
7.6 Permanent Work Datasets and the CDBWORK Command ................................ 7-3
  7.6.1 CDBWORK Syntax .............................................................................. 7-4
  7.6.2 workoptions Syntax ........................................................................... 7-4
  7.6.3 CDBWORK Keywords ....................................................................... 7-4
  7.6.4 Usage With +WORKMODE Parameter ................................................. 7-6

Chapter 8 NGT Checkpoint Dataset Maintenance .................................................... 8-1
  8.1 NGT Checkpoint Reorganization ................................................................. 8-1
  8.2 Back Up NGT System Datasets .................................................................. 8-2
  8.2.1 LOGPDS ......................................................................................... 8-2
  8.2.2 CDBCKPT ..................................................................................... 8-2
  8.3 Restoring NGT System Datasets ................................................................. 8-2
    8.3.1 Restoring the LOGPDS ................................................................. 8-2
    8.3.2 Restoring the CDBCKPT ............................................................. 8-2

Chapter 9 CDBI (CDB Interactive) ......................................................................... 9-1
  9.1 Starting CDBI ......................................................................................... 9-1
    9.1.1 Environment Selection .................................................................... 9-2
  9.2 Display Utilities ....................................................................................... 9-4
    9.2.1 Data Columns .............................................................................. 9-4
    9.2.2 Line Commands ........................................................................... 9-5
    9.2.3 Primary Commands ...................................................................... 9-9
  9.2.4 CDBI – Display Utility error messages ............................................... 9-11
  9.2.5 CDBI – Display Utility warning messages ......................................... 9-12
  9.3 MESSAGES .......................................................................................... 9-12
  9.4 Display RTS ......................................................................................... 9-13
    9.4.1 Criteria Table ............................................................................... 9-13
    9.4.2 Exceptions Table .......................................................................... 9-13
  9.4.3 Schedule Table ............................................................................... 9-14
  9.5 Display AMS ......................................................................................... 9-15

Chapter 10 CDBDisp ......................................................................................... 10-1
  10.1 Input: DD Statements, Syntax, Keywords .................................................. 10-2
    10.1.1 DD Statements ............................................................................. 10-2
    10.1.2 SYSIN Syntax Diagrams ............................................................... 10-3
## Table Of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.3</td>
<td>Keywords</td>
<td>10-3</td>
</tr>
<tr>
<td>10.1.4</td>
<td>Display Matrix</td>
<td>10-6</td>
</tr>
<tr>
<td>10.1.5</td>
<td>Display Passwords</td>
<td>10-6</td>
</tr>
<tr>
<td>10.2</td>
<td>Report Columns</td>
<td>10-7</td>
</tr>
<tr>
<td>10.2.1</td>
<td>Display Utility report</td>
<td>10-7</td>
</tr>
<tr>
<td>10.2.2</td>
<td>Matrix report</td>
<td>10-9</td>
</tr>
<tr>
<td>10.2.3</td>
<td>Passwords report</td>
<td>10-9</td>
</tr>
<tr>
<td>10.2.4</td>
<td>Restore report</td>
<td>10-9</td>
</tr>
<tr>
<td>10.3</td>
<td>Examples</td>
<td>10-10</td>
</tr>
<tr>
<td>10.3.1</td>
<td>Show the Status of All Utilities</td>
<td>10-10</td>
</tr>
<tr>
<td>10.3.2</td>
<td>Show the Status of A Single Utility ID (UID)</td>
<td>10-10</td>
</tr>
<tr>
<td>10.3.3</td>
<td>Show the Status of Utility IDs with a Common Naming Pattern</td>
<td>10-10</td>
</tr>
<tr>
<td>10.3.4</td>
<td>Show the Status of Only Completed Utilities</td>
<td>10-10</td>
</tr>
<tr>
<td>10.3.5</td>
<td>Show the Status of Only Utilities in Error</td>
<td>10-10</td>
</tr>
<tr>
<td>10.3.6</td>
<td>Show the Status of Only Restartable Utilities</td>
<td>10-11</td>
</tr>
<tr>
<td>10.3.7</td>
<td>Show the Status of Utilities Currently Running</td>
<td>10-11</td>
</tr>
<tr>
<td>10.3.8</td>
<td>Show the Status of Utilities that have had Quickexit used</td>
<td>10-11</td>
</tr>
<tr>
<td>10.3.9</td>
<td>Show the Status of Utilities that have had Quickexit used, but failed</td>
<td>10-12</td>
</tr>
<tr>
<td>10.3.10</td>
<td>Show the Status of All Utilities that have been Forced (FORCEID)</td>
<td>10-12</td>
</tr>
<tr>
<td>10.3.11</td>
<td>Show the Status of All Utilities a certain number of Days Old</td>
<td>10-12</td>
</tr>
<tr>
<td>10.3.12</td>
<td>Show the Status of Utilities with Multiple Criteria</td>
<td>10-13</td>
</tr>
<tr>
<td>10.4</td>
<td>Relevant Automation</td>
<td>10-14</td>
</tr>
<tr>
<td>10.4.1</td>
<td>XSUTDISP</td>
<td>10-15</td>
</tr>
</tbody>
</table>

### Appendix A

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDB1DBUT</td>
<td>A-1</td>
</tr>
</tbody>
</table>

### Index

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X-1</td>
</tr>
</tbody>
</table>
NGT Utility Concepts

In this chapter, we present concepts that are common to 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 utilities.

Enabled New Function Mode Not Supported

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

1.1 Wildcards

Many keywords in NGT Utilities 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%

will generate 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".

NOTE

The use of wildcards with Reorg excludes objects in the DB2 catalog.
1.2 Utility Parameters

Each NGT utility 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 called “plus parms” because when they are specified they are preceded by a “+” sign.

When NGT utilities 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. 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.

1.3 Journal

NGT introduces Server technology where multiple jobs are working on a table space or set of table spaces. Each job writes a summary of its output to its usual DDs, and they each write the full detailed output to a VSAM NGT work dataset which is called the “Journal”.

This Journals primary use is for analysis by BMC Support when questions or issues arise. The various methods of getting the Journal level output to your job output or to a dataset for sending to support are discussed elsewhere in this manual.

1.4 Applications

An application is a unit that can define anything from a single table space all the way up to an entire DB2 subsystem. Its intent is to refer to a collection of database objects that are commonly related by a common entity, such as a dependent online application (such as found on the World Wide Web), or a software application (such as found on a network in an office).

However, an application can simply refer to a collection of database objects that have no logical relationship to one another. While this will not usually be the case, it is possible to define such an application, particularly for the purposes of disaster recovery.

NOTE:
Every command for defining or managing applications begins with the control word CDBAPPL.
1.4.1 DEFINE

The DEFINE command is used to define an application.

Syntax

```
CDBAPPL DEFINE ALL
    BYDATABASE
    THEREST
    DATABASE
    DBLIST
    DB
    TABLESPACE
    TS
    INCLUDE
    db-spec
    db-table-spec
    applname
    creator-table-spec
    INDEX
    IX
    INDEXSPACE
    IS
    EXCLUDE
    EX
    TEMPORARY
    db-spec
    db-table-spec
    db-index-spec
    creator-table-spec
    creator-index-spec
```

- **db-spec**
  
  ```
  (dbmask)
  ```

- **db-table-spec**
  
  ```
  (dbmask.tsmask)
  ```

- **creator-table-spec**
  
  ```
  (crmask.tblmask)
  ```
creator-index-spec

\[
\left( \text{crmask.ismask} \right)
\]

db-index-spec

\[
\left( \text{dbmask.ismask} \right)
\]
NGT Utility Concepts

Keywords

DEFINE
Defines a new application or set of applications.

ALL
Defines a single application that contains all user data bases in the DB2 subsystem.

BYDATABASE
Defines separate applications grouped by data base name. The application name in this case will be the data base name.

THEREST
Defines an application that contains all user data bases in the system that have not previously been defined using the DB keyword. The application name in this case will be "THEREST".

appname
Specifies the name of the application. The application name must be followed by one of the following operands:

DATABASE
Specifies a data base to be included in the named application. May also be abbreviated as DBLIST or DB.

TABLESPACE
Specifies a table space to be included in the named application. May also be abbreviated as TS; the keyword INCLUDE may also be substituted.

TABLE
Specifies a table to be included in the named application. May also be abbreviated as TB.

INDEX
Specifies an index to be included in the named application. May also be abbreviated as IX.

INDEXSPACE
Specifies an index space to be included in the named application. May also be abbreviated as IS.

Within parentheses, multiple names may be separated by commas, and wildcard identifiers may also be used.

EXCLUDE
Specifies an object, or list of objects, to be excluded from the current application definition. May also be abbreviated as EX.

☑️ IMPORTANT:
When using the EXCLUDE keyword with wildcards, selections must be level-consistent. If you use wildcards with the INDEX keyword to specify an INDEX, you may only use EXCLUDE INDEX; you cannot use EXCLUDE TABLESPACE, EXCLUDE TABLE or EXCLUDE APPLICATION. See Table 1-1 below for details.

<table>
<thead>
<tr>
<th>Keywords</th>
<th>Permitted</th>
<th>Not Permitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFINE ... DATABASE (with wildcard)</td>
<td>EXCLUDE DATABASE</td>
<td>EXCLUDE TABLESPACE EXCLUDE TABLE EXCLUDE INDEX EXCLUDE INDEXSPACE</td>
</tr>
<tr>
<td>DEFINE ... TABLESPACE (with wildcard)</td>
<td>EXCLUDE TABLESPACE</td>
<td>EXCLUDE DATABASE EXCLUDE TABLE</td>
</tr>
</tbody>
</table>
Table 1-1. CDBAPPL DEFINE/EXCLUDE Wildcard Level Consistency Table.

<table>
<thead>
<tr>
<th>Command</th>
<th>Exclude Level</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFINE … TABLE (with wildcard)</td>
<td>EXCLUDE TABLE</td>
<td>EXCLUDE DATABASE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXCLUDE TABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXCLUDE INDEX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXCLUDE INDEXSPACE</td>
</tr>
<tr>
<td>DEFINE … INDEX (with wildcard)</td>
<td>EXCLUDE INDEX</td>
<td>EXCLUDE DATABASE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXCLUDE TABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXCLUDE INDEX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXCLUDE INDEXSPACE</td>
</tr>
<tr>
<td>DEFINE … INDEXSPACE (with wildcard)</td>
<td>EXCLUDE INDEXSPACE</td>
<td>EXCLUDE DATABASE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXCLUDE TABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXCLUDE INDEX</td>
</tr>
</tbody>
</table>

TEMPORARY
Defines the application as an "on-the-fly" application. A temporary application can be used to perform "what-if" testing on a collection of database objects, or to limit the scope of processing to a few selected objects.

If you intend to define an application and use it in the same utility step, be sure to specify +SYSINORDER(HONOR) in your UTLPARMS DD. This will ensure that the NGT application definition is complete before we start the following statement.
1.4.2 DELETE

The DELETE command is used to remove an application definition from the system.

Syntax

```
CDBAPPL DELETE applname
```

Keywords

DELETE

Deletes an existing application.

applname

Specifies the name of the application to be deleted.
1.4.3 RESPECIFY
The RESPECIFY command deletes all current applications you have defined.

WARNING:
Running this statement will delete all current applications (but not the data base objects themselves). This is not a reversible process. If you use this command you will need to specify at least one DEFINE statement following RESPECIFY. If it is your intention to use other applications, while deleting selected ones, use the UPDATE statement instead (see Sec. 2.2.4, "UPDATE").

Syntax

```
CDBAPPL RESPECIFY
```

Keywords

RESPECIFY

Deletes all defined applications.
1.4.4 UPDATE

The UPDATE command is used to update an application definition.

Syntax

```
CDBAPPL UPDATE applname
```

Keywords

UPDATE

Update an existing application definition.

applname

Specifies the name of the application to be updated.

DATABASE

Specifies a database contained in the application. May also be abbreviated as DBLIST or DB.

TABLESPACE

Specifies a table space contained in the application. May also be abbreviated as TS; the keyword INCLUDE may also be substituted.

TABLE

Specifies a table contained in the application. May also be abbreviated as TB.

INDEX

Specifies an index contained in the application. May also be abbreviated as IX.

EXCLUDE

Specifies an object to be excluded from the current application definition. This is a repeatable keyword; place in front of each object to be excluded. May also be abbreviated as EX.
1.4.5 DISPLAY
The DISPLAY command is used to display application definition information.

Syntax

![Diagram of DISPLAY syntax]

Keywords

DISPLAY
Display application information.

ALL
Display information about all existing application definitions.

GENERATE
Generates the statements to Define all of your existing NGT Applications. Useful to back up your application definitions and define them in a new NGT Checkpoint dataset. Add an APPGENR8 DD to your job and the define statements will be written there in an executable form rather than to the SUMMARY report.

TEMPORARY
Display information only about existing temporary application definitions.

applname
Display information about an application having this name.

DATABASE
Display information grouped by application name. May also be abbreviated as DBLIST or DB.

TABLESPACE
Display information grouped by table space name. May also be abbreviated as TS; the keyword INCLUDE may also be substituted.

TABLE
Display information grouped by table name. May also be abbreviated as TB.

INDEX
Display information grouped by index name. May also be abbreviated as IX.

EXCLUDES
Display excluded objects only. May also be abbreviated as EX.

DETAIL
Display more detailed information.

NOTE
When applications are defined these definitions are kept in the CDBCKPT dataset. If the CDBCKPT is lost or redefined the application definitions will be lost.

Be sure to keep the statements used to define applications so they can be rerun to restore the application definitions at a later time.
1.5 **Servers**

NGT Utilities can expand parallelism and multitasking by invoking NGT Servers. NGT Servers are jobs automatically initiated by a master utility job. When a 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 details and discussion at length on the use of Servers, refer to Chapter 3, “Servers”.

1.6 **The Matrix**

In order to prevent contention between data base objects and the many jobs that may be running, NGT Utilities use an internal control called a **matrix**. The matrix prevents objects that have been flagged by processing as being in error from being utilized concurrently with other utility jobs, and helps ensure the integrity not only of the data base object in error, but also of the rest of processing.

As an example, suppose that in job COPY22, table space DB1.GLEDGER is being copied with NGT Copy, and experiences a problem – perhaps the resulting dataset image copy name is incorrect, there is no space available for the image copy, or a tape unit is unavailable. Table space DB1.GLEDGER is marked in the matrix as being in error and, thus, unavailable for processing.

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 COPY22 is corrected that same night and run to a successful completion, then table space DB1.GLEDGER would be freed from the matrix, allowing the 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.

1.6.1 **Matrix Control**

The contents of the matrix can be controlled using either of the following facilities.

**CDBDISP MATRIX**

Using the command CDBDISP MATRIX as a SYSIN statement will display the contents of all objects currently in the matrix. This is run just like any other NGT job.

**Automation Control Point XSUTMTRX**

Automation control points are, effectively, miniature 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
details on the use of XSUTMTRX and automation in general, see the *BMC Next Generation Technology Automation Reference Guide*.

1.7 **Tape Commands**

Prior to the introduction of Servers, tape details were provided in DD statements in the job, e.g., LP1 DD UNIT=CART,RETPD=9. Now, with Tape Server(s) doing the work, possibly on another LPAR, a new method is provided to supply 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, multiple drives with varying RETPD, EXPDT, etc.

The CDBTAPE statement makes it easy to specify all tape usage information in the SYSIN of the BMC Next Generation Technology (NGT) Reorg or NGT Copy products.

1.7.1 **CDBTAPE**

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

**Syntax**

```
CDBTAPE ( ... ICBackupSpec ... )

DDNAME( ddname )
LP
RP
LB
RB
UR

MAXTAPESERVERS( n )
```

**ICBackupSpec**

```
UNITNAME( name )
 DDNCOUNT( n )
  MAXVOLS( n )

RETPD( nnnn )
  EXPDT( date )
  STORCLAS( name )

USER1( string )
 USER2( string )
 USER3( string )
```

**Keywords**

CDBTAPE

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
ICBackupSpec
Required. Specification for tape unit. Must be enclosed in parentheses. See "ICBackupSpec Keywords", below.

DDNAME
To name a user DD, other than one of the four Copy destinations. For example Discard or Syspunch DDs. Use a USERn to provide the DSN, see following example.

UR rules are:

1) Use of the DDNCOUNT operand is invalid for this type of CDBTAPE statement. The default for this operand will always be treated as 1.

2) New parameter DDNAME(name) must be specified anywhere after the () for the UR. This parameter is invalid for all other types (LP / RP / LB / RB).

3) A user may define up to 36 different user defined DDs with CDBTAPE UR statements within a single SYSIN stream. (The 37th will result in an error.)

Example:

CDBTAPE
UR(
    UNITNAME(CART)
    RETPD(30)
    MAXVOLS(5)
    STORCLAS(MYSTOR)
)
DDNAME(TAPETEST)

MAXTAPESERVERS
Specifies the maximum number of tape servers to be used for the selected tape unit. A value from 1 to 32.

ICBackupSpec Keywords

UNITNAME
Required. Specifies the tape unit name.

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

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

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

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

**STORCLAS**

The storage class for any created tape files.

**USER1, USER2, USER3**

Specifies additional attributes for an image copy dataset (such as DCB information, management class, etc.).

**Example**

To add MGMTCLAS and DSN to an LP image copy DD, code:

\begin{verbatim}
CDBTAPE LP(
   UNITNAME(CART)
   USER1(MGMTCLAS=ACLAS)
)
COPY TABLESPACE DB.TS
\end{verbatim}

This creates an LP1 DD in the form:

\begin{verbatim}
//LP1 DD UNIT=(CART,,DEFER),VOL=(,,,255),MGMTCLAS=ACLAS
\end{verbatim}

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

**Example**

For LBI RB copies specify

\begin{verbatim}
CDBTAPE RB(
   UNITNAME(CTAPE)
   USER1(BLKSIZE=61400)
)
COPY TABLESPACE DB.TS
\end{verbatim}

This creates an RP1 DD in the form:

\begin{verbatim}
//RB1 DD UNIT=(CTAPE,,DEFER),VOL=(,,,255),BLKSIZE=61400
\end{verbatim}

**Usage Notes**

Since the CDBTAPE command runs at parser time in the master job, it is recommended that this command be placed as the first statement in the SYSin, ahead of any other NGT utility commands.

The CDBTAPE 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. As always, contact BMC Customer Support for assistance integrating this function into any existing automation scheme.

The CDBTAPE command generates the following global variables that are fetched by using the NGT \texttt{_gread()} function:

- \texttt{TAPE_LP_UNITNAME}
- \texttt{TAPE_LP_DDNCOUNT}
- \texttt{TAPE_LP_MAXVOLS}
- \texttt{TAPE_LP_RETPD}
- TAPE_LP_EXPDT
- TAPE_LP_STORCLAS
- TAPE_LP_UNITNAME
- TAPE_LP_DDNCOUNT
- TAPE_LP_MAXVOLS
- TAPE_LP_RETPD
- TAPE_LP_EXPDT
- TAPE_LP_STORCLAS
- TAPE_LP_UNITNAME
- TAPE_LP_DDNCOUNT
- TAPE_LP_MAXVOLS
- TAPE_LP_RETPD
- TAPE_LP_EXPDT
- TAPE_LP_STORCLAS
- TAPE_DEFAULT_DDNAME
- TAPE_LP_USER1
- TAPE_LP_USER2
- TAPE_LP_USER3
- TAPE_LP_USER1
- TAPE_LP_USER2
- TAPE_LP_USER3
- TAPE_LP_USER1
- TAPE_LP_USER2
- TAPE_LP_USER3
- TAPE_LP_USER1
- TAPE_LP_USER2
- TAPE_LP_USER3
- TAPE_LP_USER1
- TAPE_LP_USER2
- TAPE_LP_USER3
- TAPE_LP_USER1
- TAPE_LP_USER2
- TAPE_LP_USER3

The following global variables were added for UR

- TAPE_U'x'_DDNAME
- TAPE_U'x'_UNITNAME
- TAPE_U'x'_MAXVOLS
- TAPE_U'x'_RETPD
- TAPE_U'x'_EXPDT
- TAPE_U'x'_STORCLAS
- TAPE_U'x'_USER1
- TAPE_U'x'_USER2
- TAPE_U'x'_USER3

Where "x" is a single character from A to Z or from 0 to 9.
**UR variable example:**

```plaintext
cr    = _gread('TAPE_UA_DDNAME')
cr    = _gread('TAPE_UA_UNITNAME')
cr    = _gread('TAPE_UA_MAXVOLS')
cr    = _gread('TAPE_UA_RETPD')
cr    = _gread('TAPE_UA_EXPDT')
cr    = _gread('TAPE_UA_STORCLAS')
cr    = _gread('TAPE_UA_USER1')
```

For the example in DDNAME above, the these statements would retrieve the following values:

- `TAPE_UA_DDNAME = TAPETEST`
- `TAPE_UA_UNITNAME = CART`
- `TAPE_UA_MAXVOLS = 5`
- `TAPE_UA_RETPD = 30`
- `TAPE_UA_EXPDT = TAPE_UA_EXPDT`
- `TAPE_UA_STORCLAS = CDBSTOR`
- `TAPE_UA_USER1 = BLKSIZE=61400`

To determine the value of any of these variables, specify the name in single quotation marks and pass it as the sole argument to `_gread()`.  

**Examples:**

```plaintext
X=_gread('TAPE_LB_MAXVOLS')
X=_gread('TAPE_LP_UNITNAME')
X=_gread('TAPE_RB_RETPD')
```

A series of scope (global) variables are available to retrieve the values specified on the USER1, USER2 and USER3 keywords with the `_gread()` function in automation control point processing. For more information on the `_gread()` function, see the *BMC Next Generation Technology Automation Reference Guide*.

There are three variable names per USER keyword for a total of 12 variable names in all. If all 12 variables were in use in an automation control point simultaneously, it might look like this:

```plaintext
lpuser1 = _gread('tape_lp_user1') /* LP variables */
lpuser2 = _gread('tape_lp_user2')
luser3 = _gread('tape_lp_user3')

lbuser1 = _gread('tape_lb_user1') /* LB variables */
luser2 = _gread('tape_lb_user2')
luser3 = _gread('tape_lb_user3')

rpuser1 = _gread('tape_rp_user1') /* RP variables */
ruser2 = _gread('tape_rp_user2')
ruser3 = _gread('tape_rp_user3')

rbuser1 = _gread('tape_rb_user1') /* RB variables */
ruser2 = _gread('tape_rb_user2')
ruser3 = _gread('tape_rb_user3')
```
Example of CDBTAPE

Consider the following NGT Reorg job step, which also produces an image copy for each table space found:

```
//********************************************************
//*         REORG WITH CONCURRENT IMAGE COPY
//********************************************************
//REORG     EXEC PGM=CDBUTIL,PARM='DSN8,RRGJOB123,NORESTART'
//SYSPRINT  DD SYSOUT=*  
//SYSTSPRT  DD SYSOUT=*  
//SYSOUT    DD SYSOUT=*  
//SYSERROR  DD SYSOUT=*  
//JOURNAL   DD DUMMY
//SYSPUNCH  DD SYSOUT=*  
//*
//SYSSIN DD *
//     CDBTAPE LP(UNITNAME(TAPE1) DDNCOUNT(5)) LB(UNITNAME(TAPE2)) RP(UNITNAME(TAPE3) RETPD(90))
//     REORG TABLESPACE DB0001.TS%
//***
```

Note that the CDBTAPE command precedes the REORG TABLESPACE command. It specifies three tape units:

- **LP**, which will define up to 5 "LP" DD's to the TAPE1 drive, that is:

  ```
  //LP1 DD UNIT=TAPE1
  //LP2 DD UNIT=TAPE1
  //LP3 DD UNIT=TAPE1
  //LP4 DD UNIT=TAPE1
  //LP5 DD UNIT=TAPE1
  ```

- **LB**, which will define a single tape drive TAPE2 to "LB", and

- **RP**, which will define a single tape drive TAPE3 to "RP", with a retention period of 90 days for each image copy produced.

Use of the CDBTAPE command in this SYSIN input stream means that each of the values with names beginning with "TAPE_" will be available for use in the XSVRJCL automation control point. (Such values are also available in all other automation control points, but are most useful when tailoring the input JCL to include the dynamically-defined tape drives as typically done in XSVRJCL.)

**Note:** If you are running JES3, edit XSVRJCL to remove PRTY=9 from the jobcard.

In the XSVRJCL automation control point, for example, the following code could be added to generate the tape DD statements for LP, LB and RP:

```
lpd = _gread('tape_lp_ddncount') /* specified by DDNCOUNT keyword */
lpu = _gread('tape_lp_unitname') /* specified by UNITNAME keyword */
lpm = _gread('tape_lp_maxvols') /* specified by MAXVOLS keyword */
lrtpd = _gread('tape_lp_retpd')  /* specified by RETPD keyword */
lpexpdt = _gread('tape_lp_expdt') /* specified by EXPDT keyword */
lpstorc = _gread('tape_lp_storclas') /* specified by STORCLAS keyword */```
lbdd = _gread('tape_lb_ddncount') /* similar set of variables as */
... /* for the LP set above: lbu, */
... /* lbmax, lbretpd, lbexpdt, lbstorc */

rpdd = _gread('tape_rp_ddncount') /* similar set of variables as */
... /* for the LP set above: rpu, */
... /* rpmax, rpretpd, rpexpdt, rpstorc */

If lpdd <> 'TAPE_LP_DDNCOUNT' Then Do
  If datatype(lpdd,N) = 1 Then Do
    Do i = 1 to lpdd
      stmt = ''
      If lpu <> "TAPE_LP_UNITNAME" Then Do
        stmt = '/LP"i " DD UNIT="lpu",DEFER"
      If lpmax <> "TAPE_LP_MAXVOLS" Then ,
        stmt = stmt"VOL=(,,"lpmax")"
      If lpmax <> "TAPE_LP_MAXVOLS" Then ,
        stmt = stmt"RETPD="lprtpd"
      If lpexpt <> "TAPE_LP_EXPDT" Then ,
        stmt = stmt"EXPDT="lpexpdt"
      If lpstorc <> "TAPE_LP_STORCLAS" Then ,
        stmt = stmt"STORCLAS="lpstorc
      JCL.NLINE = stmt
      NLINE = NLINE + 1
    End
  End /* if datatype */
End

If lbdd <> 'TAPE_LP_DDNCOUNT' Then Do
  If datatype(lbdd,N) = 1 Then Do
    Do i = 1 to lbdd
      stmt = ''
      If lbu <> "TAPE_LP_UNITNAME" Then Do
        stmt = '/LP"i " DD UNIT="lbu",DEFER"
      If lbmax <> "TAPE_LP_MAXVOLS" Then ,
        stmt = stmt"VOL=(,,"lbmax")"
      If lbmax <> "TAPE_LP_MAXVOLS" Then ,
        stmt = stmt"RETPD="lbrtpd"
      If lbexpt <> "TAPE_LP_EXPDT" Then ,
        stmt = stmt"EXPDT="lbexpdt"
      If lbstorc <> "TAPE_LP_STORCLAS" Then ,
        stmt = stmt"STORCLAS="lbstorc
      JCL.NLINE = stmt
      NLINE = NLINE + 1
    End
  End /* if datatype */
End

If rpdd <> 'TAPE_LP_DDNCOUNT' Then Do
  If datatype(rpdd,N) = 1 Then Do
    Do i = 1 to rpdd
      stmt = ''
      If rpu <> "TAPE_LP_UNITNAME" Then Do
The resulting JCL stem variable will include DD statements for the newly-allocated tape drives for LP, LB and RP.

With the example from above:

```plaintext
CDBTAPE LP(UNITNAME(TAPE1) DDNCOUNT(5)) LB(UNITNAME(TAPE2)) RP(UNITNAME(TAPE3) RETPD(90))
```

assuming the first 67 lines of the stem variable JCL were already set, the newly added statements resulting from the CDBTAPE command above would look like this in the stem variable JCL:

```plaintext
JCL.68 = //LP1 DD UNIT=TAPE1
JCL.69 = //LP2 DD UNIT=TAPE1
JCL.70 = //LP3 DD UNIT=TAPE1
JCL.71 = //LP4 DD UNIT=TAPE1
JCL.72 = //LP5 DD UNIT=TAPE1
JCL.73 = //LB1 DD UNIT=TAPE2
JCL.74 = //RP5 DD UNIT=TAPE3,RETPD=90
```

The automation control points supplied in the NGT CNTL library, and copied to the CDBEXEC library as part of version 9 installation will provide for CDBTAPE use. It is not always necessary to use the CDBTAPE statement to make tape copies.

BMC Customer Support has automation control point logic available for the dynamic allocation of tape drives. Such logic can be used as is, or may be used as a starting point for creating a more customized scheme. Contact BMC Customer support for details and assistance.

## 1.8 WHERE Clause Differences From DB2

NGT Utilities include a powerful and functionally rich implementation of an SQL WHERE clause that may be used on the REORG Discard and UNLD statements. REORG Discard is a function statement that executes the NGT Reorg utility while deleting (and optionally archiving) rows from the table space. REORG Discard is part of the NGT Reorg product. UNLD runs the NGT Auto-Unload utility to extract rows from a DB2 Table, reformat them and write them to sequential files.

This section covers the syntax of the WHERE clause and lists all the differences between DB2's full implementation and NGTs implementation with this release.
The WHERE clause is always used to SELECT rows. With REORG Discard, rows are selected for deletion. With UNLD, rows are selected to be written to the output file.

The syntax for the NGT Utilities WHERE clause closely follows that of the SQL WHERE clause. There are a few minor differences but overall processing is the same, only much faster.

The following differences in NGT supported functions affects NGT Utilities support of Indexes on expressions. NGT Utilities do not support all functions nor do we support all combinations of data types when performing arithmetic on columns. If you have needs beyond what we currently support please contact BMC Customer Support; adding the support you require can often be accommodated quickly.

**NOTE**

SQL statements that are ordinarily coded using IBM's MOD function must instead use the operator `//` (double forward slash) as a substitute.

**Example:** Instead of coding a WHERE clause like this:

```sql
WHERE MOD(DAYS(CURRENT DATE), 2) = 1
```

The following statement must be used instead:

```sql
WHERE DAYS(CURRENT DATE) // 2 = 1
```
**Predicate Syntax**

\[
\text{predicate} \\
\text{AND} \\
\text{OR} \\
\text{predicate} \\
\text{(search-condition)} \\
\text{expression} \\
\text{expression} \\
\text{expression} \\
\text{expression} \\
\text{IS NOT}
\]

**NOTE**
DB2 does not support IS or IS NOT in the basic predicate, whereas NGT does.

**Expression Syntax**

\[
\text{IFNULL} \left( \text{expression}, \text{expression} \right) \\
\text{expression} \text{ IS NULL} \\
\text{NOT} \text{ expression} \\
\text{expression} \text{ BETWEEN expression AND expression}
\]

**NOTE**
DB2 supports host variables in the LIKE clause and ESCAPE clauses whereas NGT utilities do not.

**CASE Statement Syntax**

\[
\text{CASE} \\
\text{searched WHEN clause} \\
\text{ELSE NULL} \\
\text{ELSE RESULT expression} \\
\text{END}
\]

**Searched WHEN Clause**

\[
\text{WHEN} \text{ search condition} \text{ THEN} \\
\text{RESULT expression} \\
\text{NULL}
\]
Simple WHEN clause

expression WHEN search condition THEN RESULT expression

expression NOT IN (subselect THEN expression)

FILE(ddname) FORMAT(format) THEN expression IN () constant special-register

☑️ NOTE
DB2 supports host variable in the IN clause and NGT Utilities does not. Also, formats available with NGT Auto-Utilities are ARCHIVE and UNLOAD.

value syntax

function (expression)
constant
column-name
special-register
labeled-duration

operator syntax

+ - / *\

labeled duration syntax

function (expression)
constant
column-name
YEAR
YEARS
MONTH
MONTHS
DAY
DAYS
HOUR
HOURS
MINUTE
MINUTES
SECOND
SECONDS
MICROSECOND
MICROSECONDS
The following table delineates the features and functions included in the NGT implementation of the WHERE clause.

<table>
<thead>
<tr>
<th>Feature/Function</th>
<th>NGT</th>
<th>DB2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WHERE CLAUSE ELEMENTS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column to Column Compares</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Constant to Constant Compares</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Constant to Column Compares</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Host Variables</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>DATA MANIPULATION</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concatenation</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>CONSTANT DATA FORMATS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR, HEX, GRAPHIC</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INTEGER, DECIMAL, FLOAT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Mixed Data</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Date Formats: EUR/ISO/JIS/USA</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Time Formats: EUR/ISO/JIS/USA</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>DATE/TIME</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date/Time Durations</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Dayofmonth</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Dayofweek</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Dayofyear</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Timestamp Durations</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Labeled Durations</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Week</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>*<em>ARITHMETIC (+, <em>, /, //)</em></em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date/Time Arithmetic</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Decimal Arithmetic</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Float Arithmetic</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Integer Arithmetic</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>WHERE CLAUSE KEYWORDS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESCAPE, EXISTS Keywords</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>BETWEEN, IN, LIKE Keywords</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NOT BETWEEN, NOT IN, NOT LIKE Keywords</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IS NULL, IS NOT NULL Keywords</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CONCAT Keyword</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>FUNCTIONS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE(), TIME(), TIMESTAMP()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>YEAR(), MONTH(), DAY(), DAYS(), WEEK()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HOUR(), MINUTE(), SECOND(), MICROSECOND()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DAYOFWEEK(), DAYOFMONTH(), DAYOFYEAR()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INTEGER(), DECIMAL(), FLOAT(), REAL(), SMALLINT()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>BIGINT(), DECFLOAT()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VARCHAR()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VALUE()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LENGTH()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SOUNDEx()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBSTR(), STRIP()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CHAR(), HEX()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UPPER(), LOWER()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UCASE(), LCASE()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AVG(), MIN(), MAX(), SUM(), COUNT()</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DIGITS()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VARGRAPHIC()</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>REPEAT()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOCATE()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TRANSLATE()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IFNULL()</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EURO()</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>OPERATORS</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Numeric comparisons

When comparing two numerical expressions of different types, one will be converted to the type of the other and then the comparison will take place. The conversion rules are the same as outlined in SQL reference with the exceptions of FLOAT and DECIMAL. DB2 converts the decimal to float and then carries out the comparison in float. This technique will not produce correct results if the decimal precision is greater than 18. For example if the following two numbers are compared under DB2, the result will be EQUAL; this is incorrect:

1. DEC310=1234567890123456789012345678901
2. Float=.12345678901234568 E +31

To correct this, the float will be converted to a decimal number and then the comparison will be carried out in decimal. This technique may produce undesirable results when a floating point fraction cannot be represented exactly in decimal. In this case, BMC recommends the usage of either the FLOAT() or the DECIMAL() functions to control this process.

### DECIMAL comparisons

When comparing two decimal numbers having different precision and scale, the two numbers will be converted to a uniform precision and scale, which are derived from the following formula:
For example, if number1 is DEC(31,31) having a value of .31, and number2 is DEC(31,0) having a value of 31, then the uniform precision and scale will be:

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

\[
S = \max(S_1, S_2)
\]

Therefore number2 needs to be converted to DEC(31,31). This conversion, however, will result in an overflow.

1.8.3 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. For example, where INT1 is a numeric column, NGT allows

\[
\begin{align*}
\text{INT1} &= 1, \\
\text{INT1} &= '1', \text{ or } \\
\text{INT1} &= \text{X}'F1'.
\end{align*}
\]

1.8.4 Comparing constants

Comparing constants is **not** allowed in the NGT implementation and will result in a syntax error. For example,

\[
1 = 1 \text{ or } \quad 'A' = 'A'
\]

Neither of these comparisons, nor others like them, is permitted.

1.8.5 Odd decimal precision

All decimal numbers having an even precision will be treated with the next higher odd precision. This will affect the output of the DIGITS or CHAR function as well as the recognition of the DATE, TIME and TIMESTAMP durations. For example, DEC(6,0) will be treated as DEC(7,0).

1.8.6 Labeled durations

Unlike DB2 which internally treats a labeled duration as a DECIMAL(15,0) number, NGT labeled durations are large integers. Moreover, if a labeled duration is applied to a numerical expression whose type is not a large integer, it will cause a syntax error. To rectify the problem, apply the INTEGER() function on the expression before the labeled duration. For example, assuming that column 'DEC15' is a DECIMAL(15,0), then the expression:

\[
\text{CURRENT} \ \text{DATE} = \text{DATE}('1995-01-01') + \text{DEC15} \ \text{DAYS}
\]

will cause a syntax error because the labeled duration of DAYS can only be applied to a large integer.

Adding the INTEGER function results in:
1.8.7 Date durations
Any DECIMAL(8,0) or DECIMAL(9,0) field can be treated as a date duration by NGT utilities. DB2 supports only DEC(8,0).

Date durations can be negative or positive and are used to decrement or increment DATEs respectively. That is DATE+X where X is a date duration is equivalent to the expression:

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

In DB2, however, the above is not well-defined when X is a negative number.

1.8.8 TIME durations
Any DECIMAL(6,0) or DECIMAL(7,0) field can be treated as a TIME duration by NGT utilities. DB2 supports only DEC(6,0).

1.8.9 TIMESTAMP durations
Any DECIMAL(20,6) or DECIMAL(21,6) field can be treated as a TIMESTAMP duration by NGT utilities. DB2 supports only DEC(20,6).

1.8.10 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.

The argument must be a date, a timestamp, or a valid string representation of a date or timestamp. A string representation must not be a BLOB, CLOB, or DBCLOB and must have an actual length that is not greater than 255 bytes.

The result of the function is a large integer between 1 and 31, which represents the day part of the value. If the argument can be null, the result can be null; if the argument is null, the result is the null value.

Examples:

```
UNLD DATA 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
The condition in the WHERE clause for the REORG statement is enclosed in parentheses. Parentheses are required when specifying a WHERE clause condition.
1.8.11 DAYOFWEEK function

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

The argument must be a date, a timestamp, or a valid string representation of a date or timestamp. A string representation must not be a BLOB, CLOB, or DBCLOB and must have an actual length that is not greater than 255 bytes.

The result of the function is a large integer. If the argument can be null, the result can be null; If the argument is null, the result is the null value.

Example

```
UNLD DATA FORMAT(ASCIIDEL)
SELECT EMPLOYEE_NUM, SALARY, DAYOFWEEK(PAYDATE)
FROM SWREGION.ACCTS_PAYABLE
REORG TABLESPACE SWREGION.APTABLES
DISCARD FROM TABLE SWREGION.ACCTS_PAYABLE
WHERE (DAYOFWEEK(PAYDATE) = 3)
```

1.8.12 DAYOFYEAR function

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

The argument must be a date, a timestamp, or a valid string representation of a date or timestamp. A string representation must not be a BLOB, CLOB, or DBCLOB and must have an actual length that is not greater than 255 bytes.

The result of the function is a large integer. If the argument can be null, the result can be null; If the argument is null, the result is the null value.

Example

```
UNLD DATA FORMAT(ARCHIVE)
SELECT MEMBERSHIP_NUM, DAYOFYEAR(ANNIVERSARY_DATE)
FROM REGION49.MEMBERSHIP_RECS
REORG TABLESPACE REGION49.MEMBERS7
DISCARD FROM TABLE REGION49.MEMBERSHIP_RECS
WHERE (DAYOFYEAR(ANNIVERSARY_DATE) > 350)
```

1.8.13 WEEK function

The WEEK function returns an integer in a range of 1 to 54 that represents the week of the year. The week starts with Sunday.

The argument must be a date, a timestamp, or a valid string representation of a date or timestamp. A string representation must not be a CLOB and must have an actual length that is not greater than 255 bytes.

The result of the function is a large integer. If the argument can be null, the result can be null; If the argument is null, the result is the null value.
### Example

```sql
UNLD DATA FORMAT(DSN_TIAL)
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)
```

#### 1.8.14 Restriction on the usage of NULL/NOT NULL predicate

This restriction requires that the underlying column must have the NULL attribute. For example, the following expression will cause a syntax error if column 'COL1' was defined as NOT NULL:

```
COL1 IS NOT NULL
```

#### 1.8.15 DIGITS Function

DIGITS function output length will always be odd. For a SMALLINT, the output length will be 5. For an INTEGER, the output length will be 11. For a DECIMAL, the output length will be equal to \((P/2)*2+1\), where \(P\) is the precision.

#### 1.8.16 LIKE Predicate

LIKE predicate supports only character data (no graphics). It also supports patterns constructed dynamically such as:

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

where A1 and A2 are character columns.

#### 1.8.17 VALUE Function

VALUE function argument list must have a homogeneous data type. That is, if the first argument is a SMALLINT, then the rest of the arguments must be of type SMALLINT. Also, the first argument may not be a constant.

#### 1.8.18 SOUNDEX Function

Returns a four character code that represents the sound of the argument. Used for selecting similar sounding arguments. Useful when the spelling is unknown.
1.8.19 **SUBSTR Function**

SUBSTR function will produce a fixed length string (differs from IBM when the length is greater than 255). The circumvention to produce a varying length string is to specify the length as an expression of `nnn+0`, where `nnn` is the length.

SUBSTR(ACTIV_TEXT,801,500)

This expression returns a fixed-length string

SUBSTR(ACTIV_TEXT,801,500+0)

This expression returns a variable-length string

1.8.20 **EURO Conversion Function**

This function is designed to assist European customers in Euro currency conversions. Syntax of the new function is as follows:

EURO(arg1,arg2)

Where:

**arg1** is a decimal expression representing the source amount to be converted. If the source amount data type is not decimal then the decimal function must be used to convert it to the desired precision and scale, otherwise a syntax error will result.

**arg2** is a special code that identifies the currency codes of the source and result. The general form for this code is

'XXX_EUR'
or
'EUR_YYY'

Where:

**XXX** represents a three-letter currency code of the source amount.
**YYY** represents a three-letter currency code of the result amount.

The following list identifies all supported currency codes and the associated Euro conversion rate:

<table>
<thead>
<tr>
<th>Currency Code</th>
<th>Conversion Rate</th>
<th>Currency Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEF</td>
<td>40.3399</td>
<td>Belgian Francs</td>
</tr>
<tr>
<td>DEM</td>
<td>1.95583</td>
<td>German Marks</td>
</tr>
<tr>
<td>GRD</td>
<td>340.750</td>
<td>Greek Drachmas</td>
</tr>
<tr>
<td>ESP</td>
<td>166.386</td>
<td>Spanish Pesetas</td>
</tr>
<tr>
<td>FRF</td>
<td>6.55957</td>
<td>French Francs</td>
</tr>
<tr>
<td>IEP</td>
<td>0.787564</td>
<td>Irish Punts</td>
</tr>
<tr>
<td>ITL</td>
<td>1936.27</td>
<td>Italian Lire</td>
</tr>
<tr>
<td>LUF</td>
<td>40.3399</td>
<td>Luxembourg Francs</td>
</tr>
<tr>
<td>NLG</td>
<td>2.20371</td>
<td>Dutch Guilders</td>
</tr>
<tr>
<td>ATS</td>
<td>13.7603</td>
<td>Austrian Shillings</td>
</tr>
<tr>
<td>PTE</td>
<td>200.482</td>
<td>Portuguese Escudos</td>
</tr>
<tr>
<td>FIM</td>
<td>5.94573</td>
<td>Finnish Markka</td>
</tr>
</tbody>
</table>
The EURO function returns a decimal number having the same precision and scale as the source amount and a value derived from one of the following formulas:

If ARG2 is **XXX_EUR** then

\[
EURO = DECIMAL(ROUND(ARG1/R,2),P,S)
\]

where:

- \(P\) = Precision of ARG1
- \(S\) = Scale of ARG1
- \(R\) = Conversion rate associated with currency code XXX

If ARG2 is **EUR_YYY** then

\[
EURO = DECIMAL(ROUND(ARG1*R,2),P,S)
\]

where:

- \(P\) = Precision of ARG1
- \(S\) = Scale of ARG1
- \(R\) = Conversion rate associated with currency code YYY

**NOTE**

To Unload TBL1 and convert COL1 which has a currency code of FRF to EURO, code:

```
UNLD FROM TABLE TBL1 SELECT(EURO(COL1,FRF_EUR))
```

To Unload TBL1 and convert COL1 which has a currency code of FRF to IEP, code:

```
UNLD FROM TABLE TBL1
      SELECT(EURO(EURO(COL1,FRF_EUR),EUR_IEP))
```

**1.8.21 NOT Operator**

The NOT prefix operator is not supported. However, IS NOT NULL, NOT LIKE and NOT IN are supported, as is the relational comparison not equal (\(^=\)).

**1.8.22 Reserved Words**

There are no reserved words or keywords in the NGT utilities.

**1.8.23 IN Predicate**

IN predicate supports the standard DB2 subselect. To process the subselect, the NGT Utilities will extract the subselect statement, pass the SQL to DB2 for processing and collect the output. The output from DB2 is then filtered to exclude null and duplicate rows and then placed in an MVS/ESA data space. It is searched via a high speed search engine.

The data type of the IN predicate operands must match exactly. This means that the left hand operand of the IN predicate data type must match the right hand operand which is the result returned from DB2. Precision and scale of both operands must be the same, if not, then make them the same by using the INTEGER, FLOAT, or DECIMAL DB2 built-in functions. To improve performance when conversion is required, it is recommended that the right hand operand be converted to the type and length of the left hand operand.
For example, let's assume the WHERE clause was specified as follows:

\[
\text{WHERE C1 IN (SELECT C2 FROM TB1)}
\]

The following table shows the possible data type combinations of C1 and C2 as well as the action which needs to be performed.

<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(P,S)</td>
<td>DECIMAL(C1,P,S)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>REAL</td>
<td>(SELECT DECIMAL(C2,P,S) FROM TB1)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
<td>(SELECT DECIMAL(C2,P,S) 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>DECIMAL(P1,S1)</td>
<td>DECIMAL</td>
<td>DECIMAL(C1,P,S) IN (SELECT DECIMAL(C2,P,S) FROM TB1)</td>
</tr>
</tbody>
</table>

Where:
\[
p = \min(3, \max(s1, s2), \max(p1, -s1, -p2, s2)) \quad s = \max(s1, s2)
\]

**Table 1-3. IN Predicate Data Type Conversion Actions.**

To further expedite the search process or if the data does not reside in the DB2 table, use the external file technique. This technique utilizes a 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 as follows:

\[
\text{WHERE column_name IN (FILE(ddname) FORMAT(format))}
\]

Where:

- **column_name** is the left hand side 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 the null attribute of the first field of the external file.

- **ddname** is the name of the DD statement in the current jobstep that references the external data file.

- **format** is the internal format of the data in the external file. Code either FORMAT(UNLOAD) or FORMAT(ARCHIVE). For more information on unload formats, consult the NGT Unload reference manual.
Processing of the IN predicate having an external file will cause NGT Utilities 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 may contain several data fields or columns. However, only the first data field will be extracted from the file and the rest will be ignored. The first data field is recognized by using the same attributes of the left hand operand. This first field can be the concatenation of columns.

For example, assume File(DD1) contains the following records in HEX:

C1C2C3C4C5

Assume column C1 is VARCHAR. Executing:

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

will produce incorrect output because the record in the file does not look like a VARCHAR.

In one WHERE clause you can have up to 200 IN predicates having different files and/or subselects. Boolean logic can be used to form very complex pattern matching on large amounts of data at VERY high speeds. The only expense is in memory usage which, in certain cases if not sufficient, will cause a WHERE clause syntax error. In such an event, attempt to reduce the select data or add more paging datasets to your paging subsystem.

NOTE
If unloading to a file to be referenced by Reorg's WHERE … IN FILE(ddname) then be sure to specify the same +DIGITS parm on both the Unload job and the Reorg job.

1.8.24 Trace Function
A diagnostic trace function exists in the WHERE clause that can be used to resolve problems related to WHERE processing. This feature should be used under instructions from BMC Customer Support. The TRACE keyword, if used, must be coded immediately after the WHERE keyword and before any predicate. The syntax of the TRACE function is:

TRACE(arg1,arg2:arg3)

The first argument (arg1) is required and specifies the level of tracing requested. It can be one of the following:

I 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.

R Generates one line showing the row number and the result of the WHERE clause, (TRUE or FALSE).  

T Generates one line showing the number of the row when the result is TRUE.

F Generates one line showing the number of the row when the result is FALSE.

The second and third arguments, numeric values separated by a colon (:), specify the range of rows subject to tracing. arg2 is the starting row number and arg3 the ending row number. If neither is specified, all rows will be traced. If only arg2 is specified, then the default for arg3 is 2**31-1.

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

will be traced and print row numbers between 1 and 1000 whose result is FALSE.

WHERE TRACE(I,17:17) COL1=COL2 AND COL3=COL4

will print the entire WHERE logic for row number 17.

### 1.8.25 Subselects

With the Online version of our Reorg and Unload products (NGT Auto Reorg and NGT Unload) you may use subselects in your WHERE clause.

### 1.9 Allocation Strategies

An Allocation Strategy is the implementation of your requirements for dataset allocation. This includes dataset placement, allocation values (Primary quantity and Secondary quantity), dataset 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.

An Allocation Strategy includes where and how big the datasets should be defined, and whether the existing dataset should be written over or deleted and redefined.

**DB2 Compatibility**

Regardless of whether a dataset is user-managed or DB2-managed (STOGROUP defined), NGT utilities will delete and redefine the VSAM dataset. Therefore, it is vital to review this information.

It is possible to make the NGT Utilities DEFINE datasets precisely the way DB2 does, but that is not always the best way. The NGT Utilities offer allocation features that enhance the process. For example, unused space in a table space can be reclaimed and the DB2 Catalog updated with current PQTY and SQTY values 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 will use the terms `priqty` and `secqty` to refer to the VSAM allocation values for the dataset. When referring specifically to the DB2 catalog values for these terms, `pqty` and `sqty` will be used since these are the DB2 Catalog column names.

**Levels of Control**

Allocation can be managed at one of three levels, parameters, automation control points, and space-specific Defines. Each level is described in this chapter.

### 1.9.1 Parameters

There are two utility-specific parameters that control VSAM dataset allocation. These are `+ALLO` and `+VOLUMES`. They are utility-specific because there are separate parameters of each for Reorg, Load, Restore and PartRoll.

**+ALLO**

The `+ALLO` parameter is used to specify how the allocation values for the dataset are to be determined. The following options are available for most utilities:
NGT Utility Concepts

ALLO Use the current allocation (High-Allocated RBA) as the new PRIQTY.
USED Use the current used size (High-Used RBA) as the new PRIQTY.
DB2 Use the existing DB2 Catalog PQTY and SQTY values if large enough, else use upgrade to +ALLO(USED).
KEEPDB2 Force use of the existing DB2 Catalog PQTY and SQTY values.
MXIG Obtain and use largest contiguous area on volume for PQTY.
REUSE Reset the High Used RBA to 0; do not delete the dataset.

NOTE:
Check with the specific utility reference manual to see which variable/options are allowed for that utility. +ALLO is important even if you will be using one of the automation control point facilities described later. +ALLO will affect the value of the variables passed to the automation control point.

DEFINE Retry
When an NGT Utility does a DEFINE and it fails, the DEFINE is automatically retried with a smaller primary quantity. The MXIG processing is invoked to reduce the primary quantity to the size of the largest contiguous free extent on the first volume. This attempt will result in many utilities running successfully that would otherwise fail. It means that +ALLO(MXIG) does not need to be used in most cases. Any of the other +ALLO values can be used. If it turns out that the value is too large for the volume, the PRIQTY value will automatically be reduced.

SMS
If SMS is used for DB2 datasets, then the most important aspects of an allocation strategy are already defined to SMS and +VOLUMES(*) will be specified. All that is left is a choice of allocation values (PRIQTY and SECQTY). +ALLO may be all that's needed to complete an allocation strategy.

+VOLUMES
The +VOLUMES parameter specifies a list of volumes to use for the new dataset. You may specify up to nine volser 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 will be chosen from the list based on the space available on the volume. For example, +VOLUMES(/1,AAAAAA,BBBBBB,CCCCCC,DDDDDD) 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.

STOGROUPS
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 dataset.

When a DB2 space is DB2-managed (defined in a DB2 Storage Group), then the NGT Utilities will dynamically add volumes when the dataset needs to be extended. If, while writing to a DB2 space, one of the NGT utilities determines that the volume is full, the DB2 Catalog definition for the Storage Group will be accessed to find another volume. 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. Then the I/O will be retried, allowing the dataset to extend to the new volume.

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 that the variable &VOLUMES is included in the MANAGEMENTCLASS keyword. At run time, this variable is substituted with the value specified in the +VOLUMES parameter. For example, to use a management class named DB201, you would specify:

```plaintext
+VOLUMES(DB201)
DEFINE CLUSTER -
  (NAME('&DSNAMEC.') -
   LINEAR USE SPEED &ERASE.Erase -
   MANAGEMENTCLASS( -
   &VOLUMES )
   &UNIT(&PRI &SEC) SHAREOPTIONS(3 3)) -
DATA(NAME('&DSNAMED.'))
```

NOTE
As of z/OS V1R9 (with APAR OA21369), NGT allocates DB2 datasets with units in KB, no longer in records.

Figure 1-1. AMS DEFINE model for SMS support.

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.

NOTE
See section 1.10 below to customize the model AMS DEFINE statement for your installation or for a specific DB2 subsystem.

1.9.2 Automation Control Points (Exits)
The main automation control point for allocation is XSUTALLO. The //CDBEXEC DD statement must be included in the utility job to name the library that contains these execs. Refer to the BMC Next Generation Technology Automation Reference Guide for full detail on these automation control points.

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.

**STOGROUPs**
If you use DB2-managed datasets and want the utility to allocate datasets 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, _linspace() is provided to extract the space on a particular volume. Of course, 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.

**Updating the DB2 Catalog**
A user-defined automation automation control point can be used to update the DB2 catalog. Call BMC Customer Support for examples and considerations.

1.9.3 **Space-Specific Defines**
NGT Utilities allows you to specify IDCAMS DEFINE statements for individual DB2 spaces, thus affording you complete control over the volume and space parameters for the new DB2 datasets.

NGT Reorg will (optionally) internally issue an IDCAMS DELETE and DEFINE of the new table space and indexes. This internal DEFINE is controlled by you in one of two ways.

Normal processing uses the NGT model DEFINE. NGT Auto-Utilities reads this model and substitutes the "+" variables, and invokes IDCAMS internally to execute the statement. The values that are substituted for the "+" variables are determined from the settings of the +ALLO and +VOLUMES parameters, as well as by the dynamic allocation REXX execs.

Since these parameters are set for an entire run, which may include several utilities and involve many datasets, there may be a requirement to have more direct control over the DEFINE statement generation for some datasets. NGT Utilities allow you to specify DEFINE models for specific datasets. As before, substitution is still performed on the new model and IDCAMS is still invoked to execute the statement. Using a specific DEFINE allows you to decide which variables will be substituted on an individual dataset basis.

Use of space-specific DEFINES is optional: you can use it for all spaces, for only some, or for none. Where you do not use them, the default model DEFINE is used.

**How to Use Space-Specific DEFINES**
The mechanism used to support space-specific DEFINES attempts to minimize conversion from your existing procedures. Some conversion is required to the JCL. For instance, in the case of Reorg, if you currently use UNLOAD(PAUSE), then you probably run REORG in three job steps: REORG UNLOAD(PAUSE), DEFINE, REORG UNLOAD(CONTINUE). Since NGT Reorg runs in a single jobstep, the DEFINE statements must be referenced in the REORG jobstep and not a jobstep of their own. NGT UTILITIES process everything in one Reorg job step and workS for User Defined VSAM as well as STOGROUP defined VSAM.

You may put the DEFINE s in separate members of a PDS (not a PDSE), or you may put them all in a sequential file. The format of the data in either case is the same and is discussed below. The PDS or the sequential file may have any block size, but must have RECFM=FB and LRECL=80.
If you use a partitioned dataset, then:

1. You must supply a DD statement naming the PDS. Add a DD statement to the JCL that runs CDBUTIL:

   ```
   //DEFINES   DD DISP=SHR,DSN=defines.pdsname
   ```

2. The member name in the PDS is the SPACE name. Each differently named space for which you want to use space-specific defines for must have its own member.

   The single member for the table space may contain one or more DEFINEs. For example, there can be one for the table space, or one for all partitions of a partitioned table space, or one for each partition of a partitioned table space.

   Similarly for an index, there can be one for each index, or one for all partitions of a partitioned index, or one for each partition of a partitioned index. Each index with a different SPACE name must have its own member of the PDS.

   Since there is only one member per SPACE and you might have the same SPACE name in multiple data bases, this member may contain DEFINEs for table spaces and indexes from several data bases.

If you use a sequential dataset, then:

1. You must supply a DD statement naming the file. Add a DD statement to the JCL that runs CDBUTIL:

   ```
   //DEFINES   DD DISP=SHR,DSN=defines.pdsname(mbrname)
   or       //DEFINES   DD DISP=SHR,DSN=defines.filename
   or       //DEFINES   DD *
   ```

   Note that the sequential file may be either a member of a PDS or a separate file. If "DD *" is used, then the DEFINE statements follow the DD statement.

2. The file may contain the DEFINEs for any or all of the table spaces and indexes being operated on in the utility step. There is no requirement that only one table space be run in each step.

**Format of the Statements**

The data from the PDS or sequential file consists of standard IDCAMS statements. All statements except DEFINE (e.g., ALTER or DELETE) are ignored. The Cluster and Data names in the DEFINE statement are scanned to identify the object being processed. Since NGT Utilities executes the DEFINEs internally and separately for each object, the DEFINE statement will be extracted from the member. The Cluster and Data names will be modified depending on the use of Fast switch to ensure the correct name is used.

**NOTE**

DEFINE statements must lie between columns 2 and 72.

**Found and Not Found**

When the particular DEFINE model for the dataset being allocated is found, messages NGTU055 and NGTU056 are issued verifying that the DEFINE being used is from the file. If from a PDS, the member name is shown, too.
If a DEFINE for the particular object is not found, then message NGTU057 is issued and the default model DEFINE is used.

1.10 Managing the Model AMS define

Starting with version 9.4 of NGT Utilities the optional custom model for the DB2 VSAM Define statements is kept in the NGT Checkpoint rather than the SYSPDS. The SYSPDS is no longer needed.

NGT Utilities have a model AMS DEFINE statement used when allocating DB2 VSAM datasets. 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.

A sample model AMS define is available in CNTL(JCLRXMLD). Copy and edit this member to suit your needs and define it to the NGT Checkpoint using the statements below.

1.10.1 CDBAMS DEFINE

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

```plaintext
CDBAMS DEFINE DDNAME(ddname) GLOBAL
```

The CDBAMS DEFINE statement will read the DB2 VSAM Define model statement from the specified ddname and put it in the NGT Checkpoint for use. If GLOBAL is specified this model will be used for all DB2 subsystems using this checkpoint dataset. Without GLOBAL the model override is only for the DB2 subsystem specified in the NGT Utility job running this CDBAMS statement.

1.10.2 CDBAMS DELETE

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

```plaintext
CDBAMS DELETE GLOBAL
```

The CDBAMS DELETE statement will remove 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.

1.10.3 CDBAMS DISPLAY

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

```plaintext
CDBAMS DISPLAY GLOBAL
```

The CDBAMS DISPLAY statement will report 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. For more on the CDBI menu see chapter 9 in this manual.
Chapter 2  **JCL Setup**

This chapter explains in detail the changes required for the EXEC card and most of the DD names that are used by NGT Utilities. Certain utilities may use additional DD names and they are described in that utility's documentation.

✅ **TIP:**

If you are using JCL procedures (PROCS) to run your utilities, then it might be that the only changes necessary in order to implement NGT Utilities are to make the program name change and any STEPLIB changes to the PROCS themselves.

### 2.1 JCL Overview

Figure 2-1 below shows sample JCL that will run NGT Utilities. Numbered lines are described in detail following the figure. The JCL has been grouped to show related elements; the order of the DD statements is not significant.

```jcl
//jobname JOB
B
//*******************************
//* REQUIRED STATEMENTS – MINIMUM FOR RUNNING A UTILITY
//*******************************
//UUTILSTEP EXEC PGM=CDBUTIL,PARM='DSNP,,'
//STEPLIB DD DSN=CDB.UTILITY,DISP=SHR
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *
//CDBCKPT DD DSN=CDB.CDBCKPT,DISP=SHR
//*******************************
//* GLOBAL PARAMETERS – USE TO OVERRIDE INSTALLATION OPTIONS
//*******************************
//UTLPARMS DD DSN=CDB.CNTL(PARMS),DISP=SHR
//*******************************
//* SPECIFIC UTILITY PARAMETERS – USE TO OVERRIDE INSTALLATION OPTIONS
//*******************************
//xxxPARMS DD DSN=CDB.CNTL(PARMSxxx),DISP=SHR
//*******************************
//* AUTOMATION SUPPORT (OPTIONAL)
//*******************************
//CDCEXEC DD DSN=CDB.REXX.EXEC,DISP=SHR
//*******************************
//* REPORT OUTPUT FILES (OPTIONAL)
//*******************************
//CDCPRINT DD SYSOUT=*  
//SUMMARY DD SYSOUT=*  
//RUNSTATS DD SYSOUT=*  
```


Figure 2-1. Basic JCL for NGT Utilities.

1 **JOB Statement**
   Code a job statement according to your installation's standards. A REGION size of 4-6M is recommended; usually, 4M will be sufficient. However, when working with large objects or multiple partitions and/or when using SORT, it may be necessary to code up to 6M. Some installations have found it easiest to code REGION=0M. Coding this causes no boundaries to be placed on the job when requesting storage below the line. This may not be available in your environment.

2 **EXEC Statement**
   Code a stepname according to your installation’s standards. The NGT Utilities positional parameters required to run utilities are specified via the PARM= clause. The positional parameters are summarized below.

   **SSID**
   This is the target DB2 subsystem you wish to run against. This is always required.

   **UID**
   This identifier is logged in the NGT Utilities CDBCKPT control datasets and is used with other information as part of an enqueuing mechanism across your system with other NGT Utilities jobs. The default uid is the jobname specified on the JOB card (not userid.jobname as is the DB2 default). This example in Figure 2-1 does not specify a uid, so the value used is the jobname.

   **RESTART**
   The values that can be used for the restart parameter are:

   **RESTART**
   Default. This tells NGT Utilities to restart if a prior run with the same uid failed, else treat this run as if it had never started, i.e., start at the beginning. RESTART(PHASE) can be specified to be compatible with the DB2 parameter, but is the same as RESTART. This parameter applies to restarting the stream of utility control statements as well as any utility that was functioning at the time of the premature termination. For
example, if the input control stream is a set of 500 COPY statements and the run terminated after 100 had finished, then RESTART will pick up at COPY 101 and continue.

**QUICKEXIT**

If a prior run with the same uid failed, complete the failed utility in the most efficient manner. That is, first either back out to a point of consistency or restart and complete the failed utility, whichever is required, and then terminate. If the prior utility did not fail, then terminate. For online utilities, use this option to clean up work datasets and free up the UID for use by another job. For offline utilities, use this option to get your table spaces back online as quickly as possible. In the example above, if the prior utility run had terminated after 100 of the 500 COPY statements had finished, the QUICKEXIT parameter specifies to complete any SYSCOPY posting required and reset the DB2 status for the table spaces, but then to stop. The final 400 COPY statements would not be processed.

The return code (rc variable) from a job with a restart parameter value of QUICKEXIT will always be 0 except in circumstances in which some objects are excluded (resulting in a return code of 4 instead).

**NORESTART**

Specifies that any prior utility with the same UID is to be QUICKEXITed and this job started from the beginning. This option will be honored only if it is determined by NGT Utilities that no damage will be done to any of your DB2 data by ignoring restart.

**TESTUID**

NGT Utilities will attempt to locate an existing utility ID (UID) for the named DB2 subsystem (in SSID) to see if it currently exists. If the UID does exist, perhaps left over from a previous failed utility, CDBUTIL will exit with RC=1 (Return Code = One). If the UID does not exist, CDBUTIL will exit with RC=0. This verifying of a UID does not execute the utility.

**NOTE**

A TESTUID must be performed using the very same DB2 subsystem ID with which the utility was originally executed.
FORCEID

NGT Utilities will clean up and remove the utility from the NGT Checkpoint. Work datasets are deleted, (those conforming to the +WORKPREFIX naming convention), process members are deleted from the NGT Checkpoint. This restart parameter is designed to be used as a last resort in the event the QUICKEXIT does not work, manually utility cleanup has already been performed or other drastic reasons.

NOTES

DO NOT USE THE FORCEID VALUE WITHOUT PERMISSION AND DIRECTION FROM BMC Customer Support.

Review CDBDISP output first and NEVER FORCEID a UID that is in a MUSTCOMPLETE state.

A FORCEID must be performed using the very same DB2 subsystem ID with which the utility was originally executed.

3 STEPLIB DD

Code a STEPLIB or a JOBLIB to access the NGT Utilities load library. This library must be APF-authorized.

4 SYSPRINT DD

Required. A SYSPRINT statement must be coded to capture utility output. The output is then transferred to the CDBPRINT DD data set. The SYSPRINT data set is cleared and released following each NGT Utility step. SYSPRINT contains a significant amount of detail output and is not always appropriate for perusal. For example, several of the utilities employ parallel processing. Output from this processing is often overlapped, making SYSPRINT difficult to follow. It does however, provide a valuable diagnostic tool in the event of errors. The SUMMARY DD statement, described later, contains a more readable analysis of the utility execution.

5 SYSIN DD

Required. The SYSIN DD contains the control statements for the utilities you are executing. See section 2.3 below for more details about what can be coded in SYSIN other than NGT Utility statements.

Multiple statements can be concatenated in one SYSIN file as long as each statement processes a different object as shown in the example below or +SYSLORDER(HONOR) is specified to process the objects single thread rather than in parallel.

```plaintext
REORG TABLESPACE NWDB.ARACCTS
   PART 12
COPY TABLESPACE NWDB.APX1022
   COPYDDN(OUTCOPY)
   RECOVERYDDN(OUTRECV)
RUNSTATS TABLESPACE NWDB.INVENTRY
```

6 CDBCKPT DD

Optional. Overrides the dataset name for the checkpoint PDS that was specified at configuration for this DB2 subsystem. The checkpoint PDS contains internal data used by NGT Utilities. This override is not recommended. If one job uses a different CDBCKPT dataset there
JCL Setup

could be issues due to two jobs trying to process the same object at the same time.

**Note:** Even if you do not code //CDBCKPT DD, the CDBCKPT dataset must exist and will be dynamically allocated by NGT Utilities. Its name will be obtained from the configuration member loaded at initialization. Each DB2 version MUST have its own CDBCKPT. In a data sharing environment, all members of that environment will share the same CDBCKPT (unless there are differing DB2 versions in the same data configuration). CDBCKPT datasets should not be placed under a compression software package.

7 **UTLPARMS DD**
Optional. Specifies general execution parameters. See Sec. 5.4, "Global Utility Alphabetical Parameter Reference", for a complete list of all global parameters. If not supplied, the values used will be those defined during configuration for the specific DB2 subsystem.

8 **Specific Utility Parameters**
Optional. Contains parameters for a particular utility. There is one such DD statement for each utility. If not supplied, the values used for that utility will be those defined during configuration for the specific DB2 system. Refer to the Input chapter in each utility reference for details on which parameters can be specified.

NGT Utilities first read Global Parameters from the //UTLPARMS DD statement; these are described in Sec. 5.4, "Global Utility Alphabetical Parameter Reference". Then parameters from each of the //xxxPARMS DD statements specify options unique to that utility; these are described in the REFERENCE document for that utility.

9 **CDBEXEC DD**
Optional. Overrides the dataset name for the CDBEXEC PDS that was specified during configuration for the DB2 subsystem.

10 **CDBPRINT DD**
Optional. SYSPRINT output routinely goes to CDBPRINT, unless CDBPRINT is already allocated. All NGT Utilities report output are written to CDBPRINT. All other non-NGT utilities report their output to SYSPRINT which is then copied to CDBPRINT. You do not need to supply this DD unless there are specific situations when you do not want dynamic allocation used for this file.

11 **SUMMARY DD**
Optional. 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 of the DB2 objects being processed. If this DD statement is included, summary reports will not be printed on SYSPRINT; they will be printed on SUMMARY as each utility statement completes. If this DD statement is not included, the summary reports will be on SYSPRINT instead.

12 **RUNSTATS DD**
Optional. RUNSTATS will contain 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. If this DD statement is omitted, there is no RUNSTATS report; it is never on SYSPRINT.

13 **SYSTSPRT DD**
Optional. 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
JCL Setup

CDBPRINT unless this SYSTSPRT DD is included, then they are sent to SYSTSPRT.

14 **SYERROR DD** Optional. SYERROR is used for NGT Utilities error messages. If this DD statement is not included, then error messages will be on SYSPRT only. On the other hand, if this is allocated, then error messages will be in both places.

15 **LOGPDS DD** Optional. Overrides the dataset name for the Log PDS that was specified during configuration for this DB2 subsystem. The Log PDS contains diagnostic information in case of NGT Utilities abends.

16 **SYSTRACE DD** Optional. This DD statement is used to turn on the NGT Utilities Parser trace. If this DD statement is included, the NGT Utilities Parser will write debugging messages to this file.

17 **EDITPLIB DD** Optional. 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 Utilities to the Edit Proc library. If EDITPLIB is allocated, NGT Utilities will attempt to load the Edit Proc from this library. If it cannot, NGT Utilities will then perform an SVC 8 and attempt to let MVS find the Edit Proc.

18 **AGEREPT DD** Optional. AGEREPT is used for the NGT Utilities Aging Report, a listing of which utility IDs are pending deletion.
2.2 Job Completion Codes/Return Codes

After utility execution, a completion code is returned to indicate whether execution was successful or not. One of the following codes from the table below is returned.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
</tbody>
</table>
| 4           | Successful execution with one or more warnings as described by messages issued to the SYSEXERROR file. Reasons for this code include but are not limited to:
  - An image copy was required
  - Discards occurred (for LOAD)
  - No utility statements were read from SYSIN |
| 8           | An error or an abend (abnormal end of processing) occurred. A rerun of the utility with restart parameter QUICKEXIT will clean up work datasets and insure that the objects are in 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. The utility must be rerun with restart parameter RESTART in order to ensure the usability and availability of spaces. |
2.3  **SYSIN statements other than Utility Statements**

The primary purpose of SYSIN is to provide NGT Utility 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 may continue over any number of lines.

2.3.1  **Comment**

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

Example:

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

2.3.2  **SQL**

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

Example:

```plaintext
SYSIN DD *
SQL DELETE FROM tbcreator.tbname
COPY TABLESPACE dbname.%
```

2.3.3  **DB2 Commands**

DB2 Commands can be issued directly in SYSIN of a DB2 utility job. The output will go to CDBPRINT.

Example:

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

2.3.4  **USERCMD**

USERCMD is used to pass information to the XSUTSYIN automation routine. It is different from a comment in that it is passed to the utility for BMC Next Generation Technology Automation (NGT Automation) to act on. It is similar to a comment in that it has no direct effect on other statements in SYSIN.

Example:

```plaintext
SYSIN DD *
USERCMD VAR1 = 'ABC123'
REORG TABLESPACE dbname.tname RTS
```

In this example VAR1='ABC123' is passed to XSUTSYIN prior to running Reorg. Contact BMC Customer Support about the use of USERCMD. It has valuable uses and nuances that need to be understood in a parallel processing environment.
2.3.5 **CDBPBG**

Partitions can be added to a PBG Universal Table Space without the need to Alter add part and Reorg.

Example:

```plaintext
SYSIN   DD *
CDBPBG TABLESPACE  dbname.tsname  PBGPARTS(new_parts)
```

Note: `dbname.tsname` must be a PBG UTS and `new_parts` must be greater than the current number of parts and less than MAXPARTS.

2.3.6 **Other**

There are other statements which are utility specific and documented in their corresponding manuals, such as:

CDBRTS – for defining the NGT Utility Manager internal tables, see the *BMC Next Generation Technology Utility Manager for DB2 for z/OS Reference Manual*.

CDBAPPL – for defining NGT Applications, see section 1.3 above.
3.1 Server Types
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.

3.2 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.

3.2.1 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.

**NOTE**
With server processing, servers are only started as needed, in order to reduce processing time and resource usage. For example, if a job requests five servers, the job will initiate with only one server, and will re-evaluate the need for more and possibly add one server per minute up to the maximum of five until processing is complete. If there isn't sufficient work to fill all five servers, then the job may complete with only one or two servers instead. 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 Sec. 0 (+SVRDISK).

3.2.2 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.

3.3 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.
3.3.1 Batch Server Jobs

JCL for the server jobs will be put in a PDS (partitioned dataset). At installation they are placed in the CDBEXEC PDS with the automation members. These could be put in an existing PDS such as a PARMLIB PDS. If moving them, be sure to specify where they are in the CDBSVR DD of the configuration job and rerun it. Some users will require different job cards for each Data Sharing group to specify system affinity. Additional job card members can be created, as many as are needed. Override the +SVRJCLJOB parameter in the JCL to specify which one to use.

This diagram 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.

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

Figure 3-1. Configuration Setup (CNFGdbid)

The PDS used for the server members can be overridden by specifying a CDBSVR DD statement in individual jobs. Or, just the member used for a job can be overridden 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 \texttt{+SVRJCLJOB} and:

- First, sees if it is a DDNAME in the job; if it is processing uses the library (member) specified to that DD.
- Second, looks for CDBSVR in the job to tell where the member is.
- Finally, looks for the member in the CDBSVR specified in the configuration member (RX11dbid).

This, of course, applies to \texttt{+SVRJCLDISK} and \texttt{+SVRJCLTAPE} as well.

\begin{quote}
\textbf{NOTE}

The recommended approach is to use the CDBSVR specification in the configuration to locate the Server model JCL. This way global changes can be made at the system level without changing any utility JCL. Only the +parms should be overridden in jobs to specify an alternate member.
\end{quote}

\subsection*{3.3.2 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 \texttt{PROCLIB member} is added and referenced by the UTLPARMS parameter \texttt{+SVRPROCD\{member\}}. When you request multiple servers, they will automatically be generated as started tasks with unique step names.

\begin{quote}
\textbf{NOTE}

The NGT CNTL library contains two \texttt{sample started task procedures} for use with the \texttt{+SVRPROCD\{member\}}. You may use this sample as is, or copy and modify it to customize it to your needs.
\end{quote}

\subsection*{3.3.3 Literals for Job and Step Name}

The master utility job will submit 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. NGTUtilities has provided literals to be used in these names. \texttt{You must use at least one of these literals in the server job name} or else two different jobs with the same name could be submitted (causing them to run serially rather than in parallel).

\begin{quote}
\textbf{NOTE:}

Remember that your generated literal, including all ",", ",", "(" and ")" characters, must not exceed 8 characters.

All literals (or symbolics) must end with a delimiter. That can be a ",", ",", ",", ",", or a space. Note that the "," (or period character) is special and is replaced as part of the symbolic (again normal MVS standards). The jobname/stepname produced is limited to 8 characters by MVS. If the master’s jobname is MSTRJOB1 and a DASD server is being started, then \texttt{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.
\end{quote}
Servers

&MJ The master job name.
&MJ(n) The first n characters of the master job name.
&MU The user ID of the master job.
&MU(n) The first n characters of the user ID of the master job.
&MJC The master job class, a single letter from A to Z or a single number from 0 to 9.
&JT The job type; either a D (for disk) or a T (for tape).
&SPEC A unique character string will be 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 &SPEC (e.g., A&SPEC, B&SPEC, C&SPEC, ... Z&SPEC).
&N A random number 0-9 will be inserted incremented each time a server is submitted from a job. If you have specified less than 9 servers, this literal will provide unique names.
&NN A random number 00-99 will be inserted incremented each time a server is submitted from a job. If you have specified less than 99 servers, this literal will provide unique names.
&NNN A random number 000-999 will be inserted incremented each time a server is submitted from a job. If you have specified less than 999 servers, this literal will provide unique names.
&A A letter starting at A and incrementing to Z is inserted each time a server is submitted from a job.

Examples

&MJ(2)&N 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.
&MU(3)&N 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.
&A&SPEC A unique name starting with "A" will be used.
SERV&NNN The name SERV001 through SERV999 will be used.
NGTSTP&A The name NGTSTPA through NGTSTPZ will be used.

NOTES

It is recommended 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 further information on XSVRJCL, please see the manual BMC Next Generation Technology Automation Reference Guide.
Servers

Figure 2-1 below shows a sample JCL skeleton. Numbered lines are described in detail following the figure.

```
//DSERV&NN EXEC PGM=CDBSERVR,PARM='&SVRPARM'
//*
/*************
//*REQUIRED DD CARDS
*************
// STEPLIB DD DSN=CDB.UTILITY,DISP=SHR
// SYSPRINT DD SYSOUT=*  
// SYSOUT DD SYSOUT=*  
//*
//EVERYTHING BELOW IS OPTIONAL
//*************************************************************************
// REPORT CONTROLS
//*************************************************************************
// SYSERROR DD SYSOUT=*  
// SUMMARY DD SYSOUT=*  
// RUNSTATS DD SYSOUT=*  
// SYSTSPRT DD SYSOUT=*  
// SYSTRACE DD SYSOUT=*  
// UTPRINT DD SYSOUT=*  
```

Figure 3-2. Server JCL Skeleton.

1 EXEC Statement
   On this statement, the step name MUST contain the literal &NN, and the values for PGM= and PARM= MUST be coded exactly as you see them in this example. Server processing will fail if this statement is not correctly coded. Please review this statement carefully and ensure it is correctly coded.

2 STEPLIB DD
   Code a STEPLIB or a JOBLIB to access the NGT Utilities load library. This library must be APF-authorized. The XSVRJCL automation control point supplied with NGT Version 9 utilities dynamically adds the STEPLIB DDs to match the Master job that it serves.

3 SYSPRINT DD
   Required. A SYSPRINT statement must be coded to capture utility output. The output is then transferred to the CDBPRINT DD data set. The SYSPRINT data set is cleared and released following each NGT Utility step. SYSPRINT contains a significant amount of detail output and is not always appropriate for perusal. For example, several of the utilities employ parallel processing. Output from this processing is often overlapped, making SYSPRINT difficult to follow. It does however, provide a valuable diagnostic tool in the event of errors. The SUMMARY DD statement, described later, contains a more readable analysis of the utility execution.

4 SYSOUT DD
   This is used by the system SORT routine. If you are running DFSORT™ this is required. If you are running SYNCSORT™ this is optional. SORT is only called by NGT Load.

5 SYSERROR DD
   Optional. SYSERROR is used for NGT server processing error messages. If this DD statement is not included, then error messages will be on SYSPRINT only. On the other hand, if this is allocated, then error messages will be in both places.
6 SUMMARY DD  Optional. 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 of the DB2 objects being processed. If this DD statement is included, summary reports will not be printed on SYSPRINT; they will be printed on SUMMARY as each utility statement completes. If this DD statement is not included, the summary reports will be on SYSPRINT instead.

7 RUNSTATS DD  Optional. RUNSTATS will contain 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. If this DD statement is omitted, there is no RUNSTATS report; it is never on SYSPRINT.

8 SYSTSPRT DD  Optional. 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.

9 SYSTRACE DD  Optional. This DD statement is used to turn on the NGT server processing's Parser trace option. If this DD statement is included, the Parser will write debugging messages to this file.

10 UTPRINT DD  Optional. //RUNSTATS, //SUMMARY and //SYSERROR will be dynamically allocated matching //SYSPRINT, unless these same DD names are already coded in the JCL.

3.4 Servers and Data Sharing
In some data centers, there may be a 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.

It is possible to route the server job to either of the two valid LPARs.

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

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

If using the job class is not an option, then code the line

/*JOBPARM S=(sys1,sys2)

immediately after the JOB statement. In the above code segment, sys1 and sys2 are the JES member names for the 2 LPARs in the MAS (JES spool environment).

For details on the JOBPARM statement, see the IBM MVS JCL Reference.

3.5 Servers Work Distribution and Balance of Work
NGT Utilities were not designed to balance work among servers for a particular object. Servers are designed for efficient memory usage and parallel sysplex exploitation. Servers do balance
work in the sense that where multiple Reorgs jobs were previously run in parallel to gain throughput, while trying to maintain balance among jobs; those jobs can be combined to let NGT run that workload in parallel using multiple servers to balance the workload.

For efficient use of memory, when a server takes on work for a partitioned object it will load a copy of its DBD and then not accept work for another object. This is to avoid the loading of many large DBDs into a server address space. For non-partitioned objects servers similarly restrict themselves to a database.

Procedurally, when eight 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 eight servers are running. If there isn't enough work, all the work may be done with only three servers. If there is work to be done from many databases and three servers are working on one large object that will leave five servers for other objects. This distribution of work is dynamic, and changes throughout the entire job.

Similarly, if a 20-part table space is processed with eight servers with 15 tasks each, it will not need more than two or three servers. If NGT Utilities were to use more servers with fewer tasks each, this would waste memory and not run any faster.

When it's apparent that all the work for a particular partitioned table space is being done on three servers, the other servers should be busy processing other objects or be finished with the other work and no longer be needed. Both of these cases are by design and desirable.

Finally, it is possible for a large object to be "blocked" from the servers by many other smaller objects in the queue. There is currently no queue priority to force large objects to the front of the line. Because of this it may be beneficial to put the largest object at the top of the SYSIN, or place it in a separate job by itself.

3.6 Server Modes and the +SVRMODE Parameter

The global parameter +SVRMODE controls the type of servers used by processing. There are three possible values for the two 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 was designed to be used for jobs involving large numbers of objects, large objects, or significant amounts of database processing work.

Internal. 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.

Mixed. 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.

NOTE: If Mixed is selected and NGT processing determines that too much work exists for in-memory processing, then processing will be switched to External.

An external server will be started whenever the monitor finds disk work waiting to be processed in the status file. 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.

For more information on the +SVRMODE parameter, see Sec.5.4.41, "+SVRMODE".
3.7 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 15 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. For smaller jobs specify +SVRMODE(INTERNAL), this will prevent any Disk server jobs from being submitted.

**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.

Recommendation: Configure with +SVRDISK(1), +SVRTAPE(1) and +SVRSPECIAL(I); and to prevent external Disk servers specify +SVRMODE(INTERNAL). If your job is processing very large objects or many objects, external servers can help (+SVRMODE M or E) and maybe +SVRDISK(2). For most jobs +SVRDISK and +SVRTAPE can be 1 as configured, +SVRMODE is the key to controlling external servers.
Chapter 4  

DB2 Real Time Statistics (RTS)

DB2 Real Time Statistics tables are updated in real time by DB2 and the IBM Utilities. Likewise, the 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 product, see the BMC Next Generation Technology Automation Reference Manual.

This chapter presents the DB2 Real Time Statistics columns updated by each utility.

<table>
<thead>
<tr>
<th>Database</th>
<th>Tablespace</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNRTSDB</td>
<td>DSNRTSTS</td>
<td>SYSIBM.TABLESPACESTATS</td>
</tr>
<tr>
<td>DSNRTSDB</td>
<td>DSNRTSTS</td>
<td>SYSIBM.INDEXSPACESTATS</td>
</tr>
</tbody>
</table>

**IBM RTS Objects starting with DB2 V9** (Moved into DB2 Catalog and renamed)

<table>
<thead>
<tr>
<th>Database</th>
<th>Tablespace</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>
4.1 NGT Copy

Fields Updated in `SYSTABLESPACESTATS` by DBID, PSID, 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>

These columns are also updated when NGT Reorg writes a Copy.

Fields Updated in `SYSINDEXSPACESTATS` by DBID, ISOBID, 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>

These columns are also updated when NGT Reorg writes a Copy.
NGT Reorg

REORG TABLESPACE updates both RTS tables; REORG INDEX only updates the SYSINDEXSPACESTATS table.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Settings</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTALROWS</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>NACTIVE</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>NPAGES</td>
<td>Actual value</td>
<td>Introduced in DB2 Version 9</td>
</tr>
<tr>
<td>SPACE</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>EXTENTS</td>
<td>Actual value</td>
<td>See Note 1</td>
</tr>
<tr>
<td>DATASIZE</td>
<td>Actual value</td>
<td>See Note 2</td>
</tr>
<tr>
<td>REORGLASTTIME</td>
<td>Current timestamp</td>
<td></td>
</tr>
<tr>
<td>REORGINSERTS</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGDELETES</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGUPDATES</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGDISORGBLOB</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGUNCLUSTINS</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REОРGMASSDELETE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGNEARINDREF</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGFARINDREF</td>
<td>0</td>
<td></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.

**Note 1**
For segmented multi-dataset table spaces; IBM reports the extents of the last dataset, NGT reports the total extents of all datasets.

**Note 2**
Reorg only updates DATASIZE when +RUNSTATS(ALL) or (YES) is specified in RRGPARMS.
Fields Updated in **SYSINDEXSPACESTATS** by DBID, ISOBID, Partition

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Settings</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTALENTRIES</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>NLEVELS</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>NACTIVE</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>NPAGES</td>
<td>Actual value</td>
<td>Introduced in DB2 Version 9</td>
</tr>
<tr>
<td>NLEAF</td>
<td>Actual value</td>
<td>Introduced in DB2 Version 9</td>
</tr>
<tr>
<td>SPACE</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>EXTENTS</td>
<td>Actual value</td>
<td>See Note 1</td>
</tr>
<tr>
<td>REORGLASTTIME</td>
<td>Current timestamp</td>
<td></td>
</tr>
<tr>
<td>REORGINSERTS</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGDELETES</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGAPPENDINSERT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGPSEUDODELETES</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGMASSDELETE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGLEAFNEAR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGLEAFFAR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGNUMLEVELS</td>
<td>0</td>
<td></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.

**Note 1**

For non-partitioned indexes in multiple pieces; IBM reports the extents of the last dataset, NGT Utilities report the total extents of all datasets.
### 4.3 NGT Load

Fields Updated in `SYSTABLESPACESTATS` for a `LOAD REPLACE*` by DBID, PSID, Partition

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Settings</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTALROWS</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>NACTIVE</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>NPAGES</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>SPACE</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>EXTENTS</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>DATASIZE</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>LOADRLASTTIME</td>
<td>Current timestamp</td>
<td></td>
</tr>
<tr>
<td>REORGINSERTS</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGDELETES</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGUPDATES</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGDISORGLOB</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGUNCLUSTINS</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGMASSDELETE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGNEARINDREF</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGFARINDREF</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1**
For segmented multi-dataset table spaces; IBM reports the extents of the last dataset, NGT Utilities report the total extents of all datasets.
Fields Updated in **SYSINDEXSPACESTATS** by DBID, ISOBID, Partition

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Settings</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTALENTRIES</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>NLEVELS</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>NACTIVE</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>NPAGES</td>
<td>Actual value</td>
<td>Introduced in DB2 Version 9</td>
</tr>
<tr>
<td>NLEAF</td>
<td>Actual value</td>
<td>Introduced in DB2 Version 9</td>
</tr>
<tr>
<td>SPACE</td>
<td>Actual value</td>
<td></td>
</tr>
<tr>
<td>EXTENTS</td>
<td>Actual value</td>
<td>See Note 1</td>
</tr>
<tr>
<td>LOADRLASTTIME</td>
<td>Current timestamp</td>
<td></td>
</tr>
<tr>
<td>REORGINSERTS</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGDELETES</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGAPPENDINSERT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGPSEUDODELETES</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGMASSDELETE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGLEAFNEAR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGLEAFFAR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>REORGNUMLEVELS</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

* Because NGT-Load produces reorganized indexes after a LOAD RESUME, the RTS Reorg stats in SYSINDEXSPACESTATS will be updated as with Reorg index.

**Note 1**
For non-partitioned indexes in multiple pieces; IBM reports the extents of the last dataset, NGT Utilities report the total extents of all datasets.
4.4 **NGT Stats**

Fields Updated in **SYSTABLESPACESTATS** by DBID, PSID, 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.

Fields Updated in **SYINDEXSPACESTATS** by DBID, ISOBID, 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>STATSMASSDELETE</td>
<td>0</td>
</tr>
</tbody>
</table>

These columns are also updated when other NGT utilities specify the RUNSTATS(ALL) or (YES) option.
Parameters

Parameters specify options not represented by keywords for NGT Utilities. Parameters provide tremendous, additional control for users of NGT Utilities. Examples of options controlled by parameters are 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 datasets, and whether Edit Procs can be run in Supervisor state or not.

There are parameters that apply to all NGT Utilities (global parameters), and then there are additional parameters that are specific to each utility.

**NOTE:**
Not all NGT Utilities have parameters.

5.1 Setting Default Parameters

Setting default parameters allows users to run the utilities without having to specify the most commonly used parameter values each job submitted.

5.2 Parameter Syntax Rules

Parameters coded in the global parameters DD name //UTLPARMS or in any of the parameter files for a specific utility must follow the following simple rules.

- **All parameters must start with a plus sign (+).**
- Parameters can start anywhere, but must be contained within the first 72 character positions of the line. Columns 73-80 are ignored and may contain sequence numbers.
- **Parameters may be coded on multiple lines.** Continuation is automatic.
- Multiple parameters may be coded on the same line.
- Specify comments by embedding them between /* */ pairs. Comments can be placed anywhere, even in the middle of a word. All comment strings are removed before syntax checking begins.
  - Note: If specifying parameters via a DD * statement in your JCL, be careful not to begin a comment start (/*) in column 1. If you do so, MVS will internally generate a //SYSIN for any subsequent records in the dataset unless you use DLM= to change the JCL comment delimiter.
- Comments may be nested. There is no (practical) limit to the nesting level of the comments, however, for each /*, there must be a corresponding */.
- You may specify no value for a parameter by just coding open and close parenthesis with no value, such as +JOURNAL(). This will cause the default value (from the NGTUtilities installation) to be used.
- Blanks, literally coded anywhere, have no significance. All blanks are removed before syntax checking begins. This means that a series of values must be separated by commas.
- If the same parameter is specified multiple times within the same input dataset, the latest specification takes precedence (or "wins").
- Do not use the plus sign inside the parenthesis for any global or product parameter.

On the next page are some examples.
Parameters

Parameter Specification Examples

**Plus Signs**

<table>
<thead>
<tr>
<th>RIGHT</th>
<th>WRONG</th>
</tr>
</thead>
<tbody>
<tr>
<td>//UTLPARMS DD *</td>
<td>//UTLPARMS DD *</td>
</tr>
<tr>
<td>+COPYPENDING(NO)</td>
<td>+COPYPENDING(NO)</td>
</tr>
<tr>
<td>+AGELIMIT(30)</td>
<td>AGELIMIT(30)</td>
</tr>
<tr>
<td>+PLAN(CDBPLAN)</td>
<td>PLAN(CDBPLAN)</td>
</tr>
<tr>
<td>+MAXWAIT(10)</td>
<td>MAXWAIT(10)</td>
</tr>
<tr>
<td>All four parameters have the plus sign.</td>
<td>The middle two parameters are missing the plus sign.</td>
</tr>
</tbody>
</table>

**Comments**

<table>
<thead>
<tr>
<th>RIGHT</th>
<th>WRONG</th>
</tr>
</thead>
<tbody>
<tr>
<td>//UTLPARMS DD *</td>
<td>//UTLPARMS DD *</td>
</tr>
<tr>
<td>/* THIS PLAN WAS BOUND BY */</td>
<td>THIS PLAN WAS BOUND BY</td>
</tr>
<tr>
<td>/* KELLY RAYMOND */</td>
<td>KELLY RAYMOND</td>
</tr>
<tr>
<td>+PLAN(CDBPLAN)</td>
<td>PLAN(CDBPLAN)</td>
</tr>
<tr>
<td>+MAXWAIT(10)</td>
<td>MAXWAIT(10)</td>
</tr>
<tr>
<td>Comments properly placed within /* */ marks.</td>
<td>Comments missing /* */ marks. Processing will interpret this as a syntax error.</td>
</tr>
</tbody>
</table>

**Latest-Use Precedence**

**EXAMPLE**

<table>
<thead>
<tr>
<th>//RRGPARMS DD *</th>
<th>//RRGPARMS DD *</th>
</tr>
</thead>
<tbody>
<tr>
<td>+FREEPAGE(20)</td>
<td></td>
</tr>
<tr>
<td>+ALLO(USED)</td>
<td></td>
</tr>
<tr>
<td>+PCTFREE(15)</td>
<td></td>
</tr>
<tr>
<td>+RUNSTATS(YES)</td>
<td></td>
</tr>
<tr>
<td>+ALLO(ALLO)</td>
<td></td>
</tr>
</tbody>
</table>

For the parameter ALLO, the second value, "ALLO", will be used for REORG processing.
Blank Lines

<table>
<thead>
<tr>
<th>EXAMPLE</th>
</tr>
</thead>
</table>
| //RRGPARMS DD *
  +MAXTASKS(15)

+RUNSTATS(YES)

+PCTFREE(10)
+FREEPAGE(-1)

Blanks can be used to group parameters any way you wish.

Blank Parameters

<table>
<thead>
<tr>
<th>EXAMPLE</th>
</tr>
</thead>
</table>
| //RRGPARMS DD *
  +FREEPAGE()
  +PCTFREE()

A blank value tells processing to use the installation default for that parameter.

5.3 Global Utility Parameters (UTLPARMS)

Global parameters specify options that apply to all job runs of NGT Utilities. Default values are set at installation time but may be overridden on an individual job basis by specifying new values in the UTLPARMS DD statement.

✔️ TIPS:

- Parameter values may also be specified in a sequential dataset or member of a partitioned dataset. The benefit this serves is that parameters for a series of jobs can be conveniently changed by altering one file rather than many.
- You do not need to specify all the global utility parameters if you only want to override only one or two of them. Simply specify the one or two parameters by themselves in the section of the job. The defaults for the other parameters will remain in effect.

The table on the next page lists all the global utility parameters in alphabetical order.
<table>
<thead>
<tr>
<th>Option</th>
<th>Default Value</th>
<th>Use and Value</th>
</tr>
</thead>
<tbody>
<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>YES</td>
<td>Clean up work datasets 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>+CPYPENDING</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 datasets when RC=0</td>
</tr>
<tr>
<td>+DELETEWORK</td>
<td>YES</td>
<td>Delete work datasets? YES</td>
</tr>
<tr>
<td>+DIGITS</td>
<td>installation value</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 (mmssth,count)</td>
</tr>
<tr>
<td>+EDPROCSUP</td>
<td>NO</td>
<td>Whether to call Editprocs in supervisor mode</td>
</tr>
<tr>
<td>+FASTSWITCH</td>
<td>NO</td>
<td>Use faster shadow dataset renaming? YES</td>
</tr>
<tr>
<td>+FLASHCOPY</td>
<td>Required</td>
<td>Preferred/Required if requested</td>
</tr>
<tr>
<td>+HRECALL</td>
<td>NO</td>
<td>Recall migrated DB2 datasets? YES</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>+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>1024</td>
<td>Maximum number of 1M blocks to allocate</td>
</tr>
<tr>
<td>+MAXSERVERS</td>
<td>none for disk servers, none for tape server</td>
<td>Maximum number of disk and tape servers that jobs may 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 +MAXMEMORY parm</td>
</tr>
<tr>
<td>+MONITORRATE</td>
<td>10</td>
<td>Frequency of communication between Master and Server</td>
</tr>
<tr>
<td>+ONERROR</td>
<td>ABEND</td>
<td>Action if a utility fails -- ABEND</td>
</tr>
<tr>
<td>+OUTPUTAGE</td>
<td>installation value</td>
<td>How long to retain output, in days (1-365)</td>
</tr>
<tr>
<td>+PARMLIST</td>
<td>YES</td>
<td>Print all parameters? YES</td>
</tr>
<tr>
<td>+PLAN</td>
<td>CDBPLAN</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>0020000,10</td>
<td>Controls retry of QUIESCE (mmssth,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>1</td>
<td>Buffer space for REXX variables in MB</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>
</tbody>
</table>

continues next page
<table>
<thead>
<tr>
<th>Option</th>
<th>Default Value</th>
<th>Use and Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>+SVRDISK</td>
<td>installation value</td>
<td>Maximum number of disk servers to start? (0-32)</td>
</tr>
<tr>
<td>+SVRDISKJC</td>
<td>Installation value</td>
<td>Disk Server Job Class</td>
</tr>
<tr>
<td>+SVRMODE</td>
<td>E</td>
<td>Internal, External, or Mixed Servers</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>+SVRTAPEJC</td>
<td>Installation value</td>
<td>Tape Server Job Class</td>
</tr>
<tr>
<td>+SVRTYPE</td>
<td>installation value</td>
<td>Type of processing, batch job or started task? (JOB</td>
</tr>
<tr>
<td>+SVRPROC</td>
<td>installation value</td>
<td>Name of PROCLIB member for disk tasks</td>
</tr>
<tr>
<td>+SVRPROCT</td>
<td>installation value</td>
<td>Name of PROCLIB member for tape tasks</td>
</tr>
<tr>
<td>+SVRCLEIDISK</td>
<td>installation value</td>
<td>CDBSVR member name for disk servers</td>
</tr>
<tr>
<td>+SVRCLJOB</td>
<td>installation value</td>
<td>CDBSVR member name for JOB statement</td>
</tr>
<tr>
<td>+SVRCLTAPE</td>
<td>installation value</td>
<td>CDBSVR member name for tape servers</td>
</tr>
<tr>
<td>+SVRPROCMSGCLAS</td>
<td>installation value</td>
<td>Override the message class of a started task</td>
</tr>
<tr>
<td>+SVRPROCMSGGLVL</td>
<td>installation value</td>
<td>Override the message level of a started task</td>
</tr>
<tr>
<td>+SYSINERROR</td>
<td>Installation value</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>+UIIDHISTORY</td>
<td>Installation value</td>
<td>Days to keep UID History in Checkpoint for Display</td>
</tr>
<tr>
<td>+WAITTIMEOUT</td>
<td>Installation value</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>+WORKUNIT</td>
<td>SYSDA</td>
<td>Unit name for temporary files - any value acceptable to MVS</td>
</tr>
</tbody>
</table>

Table 5-1. Global Utility Options.
Parameters

Parameters Grouped By Function
To better understand the significance of the many global parameters provided for NGT Utilities processing, the Table 4-2 below groups them all by logical function.

| System Parameters | +CONTROLPREFIX specifies a separate DSName prefix for the INFO and OUT files. |
|                   | +EDPROCSUP whether to call Editprocs in supervisor mode. |
|                   | +HRECALL specifies whether to recall migrated DB2 datasets. |
|                   | +MAXCONNECT specifies the maximum number of CDBSS connections. |
|                   | +MAXMEMORY specifies the maximum memory to allocate. |
|                   | +MONITORRATE controls the frequency of checks by the master on any server jobs running. |
|                   | +PSWDWTO controls whether password expiration messages are also written to the console. |
|                   | +RACFID specifies a user defined RACF ID. |
|                   | +REXXBUFR specifies the MB of memory to use for REXX variables. |
|                   | +REXXTIME specifies the time limit for REXX automation routines. |
|                   | +SQLID specifies a user defined default SQL ID. |
|                   | +SVRSPECIAL specifies whether processing should use a special, dedicated server for producing image copies. |
|                   | +WORKMODE specifies whether to use permanent or temporary work datasets. |
|                   | +WORKPREFIX specifies DSName prefix for temporary files. |
|                   | +WORKUNIT specifies the unit name for temporary files. |

| DB2 Parameters | +COPYPENDING specifies whether or not Copy Pending should be set when appropriate. Used by LOAD utility. |
|                | +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. |
|                | +PLAN specifies the plan name used to interface to DB2. NGT Utilities installation builds a BIND for this plan. |
|                | +QRETRY is like +DSCRL( time,count ) except that it controls re-execution of the DB2 QUIESCE utility if that utility fails to complete successfully. |

| NGT Utility Control | +AGELIMIT specifies the maximum number of days the +OUTPUTAGE parameter can be set to in a job. |
|                    | +CHECKDEFERRED specifies whether to pre-check and skip DEFINE NO objects. |
|                    | +CLEANUP specifies whether to cleanup work datasets not needed for restart after a failure. |
|                    | +DEADLINE limits the number of minutes allowed for the Change Apply phase of NGT Reorg. |
|                    | +DELETEOUTPUT specifies whether to delete all temp datasets and Output immediately when RC=0. |
## Parameters

| +DELETEWORK | specifies whether or not temporary work datasets are deleted at the end of utility processing. |
| +DIGITS | controls the number formatting. |
| +DRAIN1 | specifies whether the initial drain is to Drain Writers or Drain All. |
| +FASTSWITCH | specifies to use faster-performance name switching for work shadow datasets. |
| +JOURNAL | specifies whether to include the full NGT output in the master job. |
| +LARGEOBJSIZE | specifies the minimum object size for processing in its own dedicated server. |
| +MAXCONNECT | specifies the maximum number of CDBSS connections. |
| +MAXMEMORY | specifies the maximum number of 1M memory blocks to allocate. |
| +MAXSERVERS | specifies the maximum number of disk and tape servers that processing may use. |
| +MAXWAIT | specifies the maximum wait time when servers aren’t immediately available. |
| +MEMORYLIMIT | specifies the maximum value that can be specified for +MAXMEMORY. |
| +ONERROR | specifies action to perform if an individual utility fails to complete successfully. |
| +OUTPUTAGE | specifies how long to keep output for a specific job. |
| +PARMLIST | controls the listing of the input parameter statements. |
| +SYSINERROR | determines the action to take when a syntax error is detected in the SYSIN section of a utilities job. |
| +SYSINORDER | specifies whether to process the SYSIN in the order specified or randomly. |
| +TIMEEXEC | specifies whether to send control to the XSUTTIME automation routine a second time. |
| +UIDHISTORY | specifies the number of days to keep the UID history in the checkpoint dataset for display. |
| +WAITTIMEOUT | specifies whether to error or exclude objects that are timed out by another NGT utility. |

### NGT Server Control
This group of parameters controls NGT server processing.

### All Servers
- +SVRDISK specifies the number of disk (or DASD) servers to start for server processing.
- +SVRDISKJC specifies the job class to use for disk servers.
- +SVRMODE specifies whether disk servers will be separate address spaces or within the submitted master job.
- +SVRTAPE specifies the number of tape servers to start for server processing.
- +SVRTAPEJC specifies the job class to use for tape servers.
- +SVRTYPE specifies the type of job processing used, either a batch job or a started task.

### Batch Servers Only
- +SVRJCLDISK specifies the job name stem for disk (or DASD) server job JCL.
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+SVRJCLJOB</td>
<td>specifies the DD name associated with the server JCL's JOB card.</td>
</tr>
<tr>
<td>+SVRJCLTAPE</td>
<td>specifies the job name stem for tape server job JCL.</td>
</tr>
</tbody>
</table>

**Started Task Servers Only**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+SVRPROCDD</td>
<td>specifies the name of the PROCLIB member for disk (or DASD) processing.</td>
</tr>
<tr>
<td>+SVRPROCMSCAS</td>
<td>overrides the message class of started task servers</td>
</tr>
<tr>
<td>+SVRPROCMSCGLVL</td>
<td>overrides the message class of started task servers</td>
</tr>
<tr>
<td>+SVRPROCT</td>
<td>specifies the name of the PROCLIB member for tape processing.</td>
</tr>
</tbody>
</table>

| Table 5-2. Parameters Grouped By Function. |
5.4 Global Utility Alphabetical Parameter Reference

5.4.1 +AGELIMIT
Output Age Cap Control

Syntax

\[ +\text{AGELIMIT}(\text{number}) \]

Description

Sets a cap on how long users can request their output be kept. This value may only be set at installation time.

Operands

- **30**
  - Default. Values specified for the +OUTPUTAGE parameter may not exceed 30 days.

- **number**
  - A number of days ranging from 1 to 365. Values specified for the +OUTPUTAGE parameter may not exceed this number of days.

Example

\[ +\text{AGELIMIT}(90) \]

Values set for the +OUTPUTAGE parameter may not exceed the set limit of 90 days.
5.4.2 +CHECKDEFERRED
Deferred Table space Control

Syntax

\[ +\text{CHECKDEFERRED}(\text{NO}) \]

Description

The +CHECKDEFERRED parameter indicates whether NGT should check for and ignore table spaces that are deferred during wildcard expansion prior to the utility processing starting.

Operands

<table>
<thead>
<tr>
<th>Operand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>Default. Do not check for deferred table spaces.</td>
</tr>
<tr>
<td>YES</td>
<td>Check for deferred table spaces.</td>
</tr>
</tbody>
</table>

Note: These operand values may not be abbreviated to "Y" or "N". The full values "YES" or "NO" must be used.

Example

\[ +\text{CHECKDEFERRED(NO)} \]

RTMAN processing will not check for deferred table spaces.
5.4.3  
**+CLEANUP**

Cleanup Action Control

### Syntax

```
+CLEANUP ( OFF )
```

### Description

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.

### Operands

- **NO**  
  Default. Object will be processed as if ERROR was selected.

- **YES**  
  Object will be processed as if CLEANUP was selected.

**Note:** These operand values may not be abbreviated to "Y" or "N". The full values "NO" or "YES" must be used.

### Usage Notes

`+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 BMC Next Generation Technology Automation Reference Manual manual section on XSVRXERR, specifically the **action** variable, for details). `+DELETEOUTPUT`, by comparison, is for jobs that have run successfully: it controls whether work datasets are deleted at the successful end of a job (that is, when it is marked COMPLETE). Compare with `+DELETEOUTPUT`, Sec. 5.4.7.

### 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.

- **+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).
5.4.4  +CONTROLPREFIX
INFO and OUTPUT File Allocation Control

Syntax

```
+CONTROLPREFIX ( string )
```

Description

+CONTROLPREFIX, if specified, allocates INFO and OUTPUT files with the provided prefix string instead of using the prefix value specified by +WORKPREFIX. This allows these two 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 datasets will be deleted upon successful completion of the job.

Operands

- **string**: Any valid MVS high-level qualifier.

Example

```
+CONTROLPREFIX(ADMIN.NGT)
```

INFO and OUTPUT files will be allocated with dataset names beginning with "ADMIN.NGT".
5.4.5  +COPYPENDING
Copy Pending Flag Control

Syntax

+COPYPENDING ( NO YES )

Description

+COPYPENDING 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.

Operands

YES Set the COPY PENDING status.
NO Does not set COPY PENDING. Use after a LOAD of a Read-Only table space.

Example

+COPYPENDING(YES)

Set COPY PENDING after a LOAD, REPART, or RESTORE. This is the default.
5.4.6  +DEADLINE
NGT Reorg Change Apply Control

Syntax

+DEADLINE(integer 255)

Description

+DEADLINE specifies the number of minutes allowed for the Change Apply phase of NGT Reorg. Note that this parameter is not used with any other product other than NGT Reorg.

Operands

255 Default, 255 minutes.
integer A value from 0 to 255, indicating the number of minutes. A value of 0 indicates no deadline is to be used; job will continue to run retry processing until successful or terminated by the user.

Usage Notes

If the deadline is reached without a successful DRAIN(ALL), processing issues message NGTR858 ("Dispatching Change Apply work...") and the job terminates.

Example

+DEADLINE(5)

If a DRAIN(ALL) operation is unsuccessful, run retry processing for a maximum of 5 minutes before failing.
5.4.7  **+DELETEOUTPUT**  
Output Deletion Control

**Syntax**

```
+DELETEOUTPUT( [NO] [YES] )
```

**Description**

When a job completes and is marked as COMPLETE then the value of +DELETEOUTPUT is checked. If it’s 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.

**Operands**

<table>
<thead>
<tr>
<th>Operand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>Default. Even after successful completion, all output datasets will be kept.</td>
</tr>
<tr>
<td>YES</td>
<td>At successful completion, all output datasets will be deleted.</td>
</tr>
</tbody>
</table>

**Example**

```
+DELETEOUTPUT(YES)
```

Delete output files at the completion of a job.
5.4.8  +DELETEWORK
Work Dataset Deletion Control

Syntax

+DELETEWORK(YES NO)

Description

+DELETEWORK controls the disposition of utility work datasets. These include //SYSUT1 and the +TAPEDD datasets. These datasets 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.

Operands

YES Default. At successful completion, all work datasets will be uncatalogued and deleted.

NO Even after successful completion, the work datasets will be kept.

Example

+DELETEWORK(YES)

Delete work files at termination.
5.4.9  **+DIGITS**

**Number Formatting Control**

**Syntax**

```
+DIGITS(CDB, IBM)
```

**Description**

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.

**Operands**

- **CDB**  Default. Use NGT format (see "Usage Notes" below).
- **IBM**  Use IBM's format.

**Example**

```
+DIGITS(CDB)
```

Use the NGT approach to handling numeric values for this utility job.

**Usage Notes**

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, two new parameters 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 DB2's implementation or **+DIGITS(IBM)**, numeric columns are generated as follows:

- 10 digits for an INTEGER
- 5 digits for a SMALLINT
- P digits for a DECIMAL where P is the precision of the decimal

**WARNING:**

If you are an existing customer of BMC 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.
Parameters

5.4.10  +DRAIN1
Drain Control

Syntax

\[ +\text{DRAIN1}(\text{WTR/FULL}) \]

Description

+DRAIN1 allows control over 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.

Operands

<table>
<thead>
<tr>
<th>ALL</th>
<th>Default. Issues DRAIN(ALL) commands. Processing waits on all (both reader and writer) claims.</th>
</tr>
</thead>
<tbody>
<tr>
<td>WTR</td>
<td>Issues DRAIN(WTR) commands. Processing waits on reader claims only.</td>
</tr>
</tbody>
</table>

Examples

+DRAIN1(ALL)

+DRAIN1(ALL) will drain both readers and writers.

+DRAIN1(WTR)

+DRAIN1(WTR) will drain writers only until the last iteration. Then change apply would drain readers while holding the writers’ drain for the previous iteration.
5.4.11  +DSCRL
Display SPACE Command Retry Limits

Syntax

```
+DSCRL( MMSSTH, NNN )
```

Description

+DSCRL 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 NGT Utilities start 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, NGT Utilities issues. -DISPLAY commands until DB2 shows the desired state or until the time specified in DSCRL expires, whichever comes first.

Operands

<table>
<thead>
<tr>
<th>MMSSTH</th>
<th>Time between display commands where the following subfields exist:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MM</td>
<td>Minutes</td>
</tr>
<tr>
<td>SS</td>
<td>Seconds</td>
</tr>
<tr>
<td>T</td>
<td>Tenths of a Second</td>
</tr>
<tr>
<td>H</td>
<td>Hundredths of a Second</td>
</tr>
</tbody>
</table>

| NNN    | Maximum number of display attempts                            |

If the number of retries is exceeded, a message is issued and the Utility terminates with return code 8.

Example

```
+DSCRL(000100,120)
```

The first operand tells NGT Utilities to retry every second. The second operand specifies a maximum of 120 retries. This combination of parameters would make NGT Utilities wait for up to two minutes before terminating the Utility on this object.
5.4.12  **+EDPROCSUP**

Whether to call Editprocs in supervisor state.

**Syntax**

```
+EDPROCSUP( Yes/No )
```

**Description**

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.

**Operands**

- **YES**  
  Call Editprocs in supervisor state.

- **NO**  
  Call Editprocs in program state.

**Example**

```
+EDPROCSUP(NO)
```

Run any Editproc in program state. This uses significantly less CPU time.
5.4.13  
**+FASTSWITCH**

DB2 I-J Name Switching Control

**Syntax**

```
+FASTSWITCH( NO YES )
```

**Description**

+FASTSWITCH indicates whether or not to use supported "I/J" name switching instead of the "A/B" name switching previously used by NGT Utilities. ("I/J" name switching has been supported since DB2 Version 7.0.

The benefit of +FASTSWITCH(YES) is much better performance since the "A" to "X" and "B" to "A" renames formerly used by NGT Utilities will not be done. Instead, a DB2 Catalog field and the DB2 Directory are updated to reflect the current "I"/"J" name. No IDCAMS RENAMES are done at all.

**Operands**

- **NO**  
  Do not use Fast Switch.

- **YES**  
  (default). Use Fast Switch.

**Example**

```
+FASTSWITCH(YES)
```

Use the new faster Rename process.
5.4.14 +FLASHCOPY
To specify whether Flashcopy is Preferred or Required

Syntax

```
+FLASHCOPY(REQUIRED)  
+FLASHCOPY(PREFERRED)
```

Description
With REQUIRED if the volumes do not support flashcopy then the Reorg will fail.
With PREFERRED if the volumes do not support flashcopy then a regular non-flashcopy will be made with the Reorg.

Operands

- REQUIRED (default) Specifies that the copy to be taken with Reorg must be a FLASHCOPY.
- PREFERRED Specifies that the copy to be taken with Reorg is be be a Flashcopy if the volumes support Flashcopy, otherwise use standard copy.

Example

```
+FLASHCOPY(PREFERRED)
```

Use Flashcopy if supported by the volumes, otherwise use the standard copy.
5.4.15 **+HRECALL**  
**VSAM Dataset Migration Control**

**Syntax**

```
+HRECALL (YES NO)
```

**Description**

+HRECALL indicates whether or not migrated VSAM datasets are to be automatically recalled.

**Operands**

- **NO**  
  Default. Do not recall migrated volumes.

- **YES**  
  Recall migrated volumes.

**Examples**

```
+HRECALL(NO)
```

No VSAM datasets are to be recalled if migrated. 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.

```
+HRECALL(YES)
```

Any VSAM datasets needed and migrated will be recalled. This may increase the utility's elapsed time.
5.4.16  

**+JOURNAL**  
Journal Control  

**Syntax**  

```
+JOURNAL(NO)  
+JOURNAL(YES)  
+JOURNAL(ONERROR)
```

**Description**  

+JOURNAL controls the creation of the journal file associated with the NGT utility processing.

**IMPORTANT**  

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.

**Operands**  

- **NO**  
  Default. No journal will be printed.
- **YES**  
  Journal will be printed to JRNLOUT DD. This DD may be defined to SYSOUT or to a static dataset. If a static dataset 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**  

```bash  
//JRNLOUT DD SYSOUT=*  
//UTLPARMS DD *  
+JOURNAL(YES)  
A journal will be sent to JRNLOUT.

//JRNLOUT DD DISP=SHR,DSN=USER.NGT.JOURNAL  
//UTLPARMS DD *  
+JOURNAL(YES)  
A journal will be sent to the dataset USER.NGT.JOURNAL.
```
Parameters

//CDB1233 EXEC PGM=CDBUTIL,PARM='DB2T,ACCTG2009,RESTART'
//JRNLOUT DD DISP=SHR,DSN=USER.NGT.JOURNAL
//UTLPARMS DD *
+JOURNAL(ONERROR)
A journal will be sent to the dataset USER.NGT.JOURNAL and the utility ID ACCTG2009 if an error occurs and the utility is marked restartable.
5.4.17  +LARGEOBJSIZE
Large Objects Control

Syntax

+LARGEOBJSIZE(\(\text{number}\))

Description

+LARGEOBJSIZE 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.

Operands

1048576 Default. Objects exceeding 1048576 4K pages (4 G) will be processed individually.

number 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 2097152 (8 G).

Examples

+LARGEOBJSIZE(2048000)

Any object 2048000 4K pages or larger will be processed by itself.
5.4.18 \textbf{+MAXCONNECT} \\
Maximum Number of CDBSS Connections

\textbf{Syntax}

\begin{align*}
\text{+MAXCONNECT(} & \text{integer} \text{)}
\end{align*}

\textbf{Description}

Specifies the maximum number of connections to the NGT Subsystem (CDBSS) that the master can start. (NOTE: +MAXCONNECT has no association with or effect on DB2 threads.)

\textbf{Operands}

- \textbf{64} \quad \text{Default. Processing will use a maximum of 64 connections to the CDBSS process. The maximum value you may specify is 128.}
- \textbf{n} \quad \text{An integer representing the maximum number of connections to the CDBSS process.}

\textbf{Example}

\begin{verbatim}
+MAXCONNECT(32)
\end{verbatim}

Processing will allow a maximum of 32 connections to the NGT Subsystem.
5.4.19  +MAXIDLE
Amount of time Server will remain idle before shutting down

Syntax

+MAXIDLE( 10 integer )

Description

When a Server job has set idle for n minutes, the Master will instruct it to shut down. This parm can be useful for a job that uses many Servers but only one of those Servers is used towards the end of processing. The no longer needed Server jobs will end, freeing up their initiators.

Operands

10  Default. Server will remain idle for 10 minutes before shutting down. Job will continue its work.

n  An integer representing the amount of time in minutes a server will remain idle before shutting down.

Example

+MAXIDLE(20)

Processing will allow a maximum Server to remain idle for 20 minutes before shutting the server down because it is not being used
Parameters

5.4.20  +MAXMEMORY
Maximum Amount of 1M Blocks Of Memory To Allocate

Syntax

[+MAXMEMORY( integer )]

Description

Specifies the number of 1M blocks of memory to allocate for processing by each Server. In other words, a value of 1024 means that processing will allocate 1024 x 1M, or 1G of memory.

If insufficient memory exists for processing, the message "WAITING FOR MEMORY" will be issued to CDBPRINT until the condition is resolved.

Operands

1000   Default. Processing will allocate 1024 x 1M = 1G of memory.

n   An integer from 100 to 65535 representing the number of 1M blocks.

Usage Notes

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

+MAXMEMORY(2048 )

Processing will allocate 2048 x 1M (or 2 G) of memory for Unload processing.
5.4.21  

+MAXSERVERS

Maximum Number of Servers

Syntax

```
+MAXSERVERS(integer1, integer2)
```

Description

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.

Operands

- **integer1** Maximum number of disk servers. Maximum valid value is 32.
- **integer2** Maximum number of tape servers. Maximum valid value is 32.

Examples

```
+MAXSERVERS(10, 5)
```

Jobs may not use more than 10 disk servers and 5 tape servers.

```
+MAXSERVERS(4, 0)
```

Jobs may not use tape servers at all.
5.4.22  +MAXWAIT
Maximum Wait Time When Servers Unavailable

Syntax

```
+MAXWAIT( 10 n )
```

Description

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.

Operands

10  Default. Processing will wait a maximum of 10 minutes before shutting down if no servers are available.

n  A value, in minutes, from 1 through 60, indicating the maximum time processing will wait before shutting down, if no servers are available.

Example

```
+MAXWAIT(20)
```

Wait a maximum of 20 minutes if servers are unavailable, otherwise shut down processing.
5.4.23  
**+MEMORYLIMIT**

Maximum Memory Limit Parameter Control

**Syntax**

```
+MEMORYLIMIT( n )
```

**Description**

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.

**Operands**

- **n**  
  A value, in 1M blocks.

**Usage Notes**

This parameter may only be set at configuration time when the utilities are being installed.

**Example**

```
+MEMORYLIMIT(4096)
```

The `+MAXMEMORY` parameter cannot be set higher than 4G for any job.
5.4.24 +MONITORRATE
Server Monitoring Frequency Control

Syntax

```
+MONITORRATE(n)
```

Description

NGT Utilities processing routinely queries activity between the master job and any server jobs running. The +MONITORRATE parameter controls the frequency of these checks.

Operands

10 Default. A 10-second delay will occur between server job checks.

n The time delay between server job checks, in seconds.

Examples

```
+MONITORRATE(10)
```

Server jobs will be checked every 10 seconds.

```
+MONITORRATE(1)
```

Server jobs will be checked every second. This could have significant performance impacts.

```
+MONITORRATE(60)
```

Server jobs will be checked every minute (60 seconds).
5.4.25 +ONERROR
Action Taken In Case of Abend Or Error

Syntax

+ONERROR (ABEND CONTINUE STOP)

Description

The +ONERROR determines the action to be taken by NGT Utilities when an error occurs while attempting to execute a Utility.

Operands

ABEND If an error is encountered, the task ABENDs. The user ABEND code issued is either 8 or 16, depending on the severity of the problem.

CONTINUE If an error is encountered during processing of this object, NGT Utilities 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 Utilities notify 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

+ONERROR(CONTINUE)

If NGT Utilities encounter an error when processing this object, continue with the next statement.
5.4.26 +OUTPUTAGE

VSAM Output File Retention

Syntax

+OUTPUTAGE( n )

Description

Specifies how long to retain a VSAM output file. For further details, see Sec. 1.5, "Servers". This value may not exceed the value of +AGELIMIT.

Operands

7 Default. VSAM output retained for 7 days.

n An integer from 1 to 365 representing the number of days.

Example

+OUTPUTAGE(7) - set in current job
+AGELIMIT(30) - set at installation time

Retain VSAM output for a week. Output age is within the 30-day limit set at installation time.

+OUTPUTAGE(20) - set in current job
+AGELIMIT(14) - set at installation time

Output retention period of 20 days exceeds 14-day limit set at installation time. This job would result in an error condition.
5.4.27 **+PARMLIST**  
Parameter List Print Control

**Syntax**

```
+PARMLIST(YES
NO)
```

**Description**

Specifies whether or not the values of the product parameters will print in the SYSPRINT listing.

**Operands**

- **YES**  
  Default. Print out the values of the parameters in the job output.
- **NO**  
  Do not print out the values of the parameters.

Global parameters are not suppressed.

**Example**

```
+PARMLIST(NO)
```

Suppress printing of utility parameter values.
Parameters

5.4.28 +PLAN
NGT Utilities DB2 Plan Name

Syntax

+PLAN(planname)

Description

Specifies the name of the plan to be used by NGT Utilities.

Operands

planname Character string representing the name of the NGT plan.

Example

+PLAN(MYPLAN)

Use MYPLAN as the plan name for NGT Utilities.
5.4.29  

**+PSWDWTO**  
Password expire Write To Operator

**Syntax**

```
+PSWDWTO (                   )
```

**Description**

When +PSWDWTO(YES) is specified we will issue a WTO specifying which product(s) passwords are within 21 days of expiring. When +PSWDWTO(NO) the messages will only be written to the individual jobs output. Default value is NO.

**Example**

```
+PSWDWTO(YES)
```

Write messages like the following to the operator console:

+NGTZ166 Reorg PASSWORD (xxxxxxxxxxxxxxxx) TO EXPIRE IN 21 DAYS.
+NGTZ166 RTS PASSWORD (xxxxxxxxxxxxxxxx) TO EXPIRE IN 21 DAYS.
5.4.30  

+QRETRY
QUIESCE Retry Limits

Syntax

+QRETRY(MMSSTH, NNN)

Description

+QRETRY specifies how long NGT can hold the Drain and how many times the Drain can be requested. Initially the Drain is only to establish a Point-In-Time, later Drains are to apply changes accumulated during the utility processing.

The Drain is retried when a drain fails to occur in the allotted time, this produces message:
NGTC267 DRAIN(WTR) FAILED.

The Drain is retried when all the changes cannot be applied in the remaining drain time, this produces message:
NGTC999 FETCH TIMER EXPIRED

Operands

MMSSTH  Time between display commands where the following subfields exist:

MM       Minutes
SS       Seconds
T        Tenths of a Second
H        Hundredths of a Second

NNN      Maximum number of display attempts

If the number of retries is exceeded, a message is issued and the Utility terminates with a return code of 8.

Example

+QRETRY(002000, 30)

The first operand tells NGT Utilities to run the QUIESCE for twenty seconds. The second operand specifies a maximum of 30 retries.
Parameters

5.4.31  +RACFID
User-Defined RACF ID For DB2 Objects

Syntax

+RACFID(racfid)

Description

+RACFID specifies a user defined RACF ID to be used by the NGT Utilities 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.

Operands

racfid authorization id for NGT to use to access DB2 objects.

Restrictions

Does not apply to allocation on sequential work or copy datasets. Can only be set during configuration, which requires AMASPZAP authority. Use of this parameter outside configuration jobs is ignored.

Example

+RACFID(DB2AUTH)

NGT Utilities will use authorization ID DB2AUTH to access DB2 objects.

+RACFID()

NGT Utilities will use the utility submitters authorization to access DB2 objects.
5.4.32 +RERUN

Restart Control

Syntax

\[ +\text{RERUN}(\quad \theta \quad n) \]

Description

Specifies what NGT-Utilities should 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 the NGT Utilities to perform a QUICKEXIT on the \( n \)th restart attempt. A value greater than zero tells the NGT Utilities to perform a NORESTART on the \( n \)th restart attempt.

Operands

0  Defaults. Tells NGT Utilities to continue to apply normal restart logic on all subsequent restart attempts.

-\( n \)  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.

\( n \)  This is a value from 1 through 32767. On the \( n \)th restart attempt, NGT Utilities will 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 will be 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

\[ +\text{RERUN}(-5) \]

If the RESTART parameter is specified, the job will attempt 5 restarts and then will change the parameter to a QUICKEXIT and finish processing.
5.4.33  
+REXXBUFR

REXX Variable Buffer Control

Syntax

```
+REXXBUFR integer
200
)
```

Description

Specifies the REXX variable buffer size. The minimum value that can be specified (and is 1 (for 1 megabyte). The maximum value is 200 (200 megabytes).

Operands

200 Default. Uses REXX variable buffer size of 200 M.
integer Any number from 1 to 200.

Example

```
+REXXBUFR(50)
```

Specifies a REXX variable buffer of size 50 M.
5.4.34  

**+REXXTIME**  
Time Limit For Automation control points

**Syntax**

```
+REXXTIME(seconds 10)
```

**Description**

Specifies how many CPU seconds the NGT Automation Utility automation control points are allowed to use before being cancelled.

**Operands**

- **10**  
  Up to ten seconds can be used. This is the default value.

- **seconds**  
  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**

```
+REXXTIME(25)
```

Allows every automation control point to use up to 25 CPU seconds of processing before being cut off.
5.4.35  

+SQLID
Default SQLID Control

Syntax

+SQLID( sqlid )

Description

Specifies the secondary authid NGT utilities will use for SQL. This is equivalent to SET CURRENT SQLID. Your security product, i.e. RACF, will be checked to ensure the user is in the group specified in +SQLID. +SQLID is generally not set to any value in the configuration; it is specified in jobs as needed.

Operands

sqlid  
A valid SQLID for the DB2 system on which a utility is being run.

Example

+SQLID(ADMIN32)

When running the CDBRTS DEFINE statement to create the NGT Utility Manager tables, use ADMIN32 authority to create the tables.
Parameters

5.4.36  +SVRDISK
Maximum Number Of Disk Units To Use For Server Control

Syntax

+SVRDISK(n)

Description

Specifies the maximum number of disk (or DASD) servers to start for server processing. For general information on servers, refer back to Sec. 1.5, "Servers".

If +SVRMODE(Mixed) is used, the internal server does count as one of the disk servers.

Operands

1  Default. Indicates a maximum of 1 disk server will be started.
   n  A number from 1 to 32 indicating the maximum number of disk servers to be started.
   -1  Signifies a disk server should not be started.

Examples

+SVRDISK(2)

Use a maximum of 2 disk servers for server processing. Both the internal and external servers count toward this maximum.
5.4.37 +SVRDISKJC
Job Class For Disk Server Job

Syntax

\[+SVRDISKJC(x)\]

Description

Specifies the job class for disk server job. A new "&JC" symbolic may now be coded on the JOB statement, i.e., // 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. For general information on servers, refer back to Sec. 2.2, "Servers".

Operands

A  Default job class for disk servers.
X  The job class for the disk servers.

Example

+SVRDISKJC(D)

Server job card, pointed to by the +SVRJCLJOB parameter, contains CLASS=&JC. Server will run with CLASS=D.
5.4.38 +SVRJCLDISK
DD Name For Disk Server Job JCL

Syntax

+SVRJCLDISK(ddname)

Description

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 CDBSVR PDS specified in the NGT configuration.

Operands

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 CDBSVR 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 dataset containing the disk server JCL. This is accomplished by way of the following JCL change:

//DSERVJCL DD DSN=prefix.CDBSVR,DISP=SHR

Otherwise, without this specified DSERVJCL, disk server JCL will come from member DSERVJCL from the CDBEXEC dataset specified in the job (if it was not changed at installation time).
5.4.39  +SVRJCLJOB
Job Card JCL DD Name

Syntax

+SVRJCLJOB (ddname) SVRJOBCD

Description

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 CDBSVR PDS that is specified in the NGT configuration.

Operands

SVRJOBCD
Default DD or member name for the server job card.

name
An alternative DD name or member name for the JCL associated with the server JCL's JOB card. If the name 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 CDBSVR 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 dataset containing the disk server JCL. This is accomplished by way of the following JCL change:

//SJOBJCL DD DSN=prefix.CDBSVR,DISP=SHR

Otherwise, without this specified SJOBJCL DD, the disk server JCL will come from member SJOBJCL from the CDBEXEC dataset (if it was not changed at installation time).
5.4.40  +SVRJCLTAPE
DD Name For Tape Server Job JCL

Syntax

\[ +SVRJCLTAPE( ddname ) \]

Description

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 CDBSVR PDS that is specified in the NGT Configuration.

Operands

SVRTAPE Default DD or member name for the tape server JCL.

name An alternative DD or member name for the tape server JCL. If the \textit{ddname} specified in this parameter is not valid, message NGTY118 is issued indicating the \textit{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 CDBSVR 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 dataset containing the disk server JCL. This would be accomplished by the following JCL change:

\[ //TSERVJCL DD DSN=prefix.CDBSVR,DISP=SHR \]

Otherwise, without this specified TSERVJCL, the disk server JCL will come from member TSERVJCL from the CDBEXEC dataset (if it was not changed at installation time).
5.4.41 +SVRMODE
Server Mode Control

Syntax

![Diagram of Syntax]

Description

Specifies the mode in which servers are spawned for processing.

Operands

- **MIXED**
  - 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. A value of 'M' may be substituted for 'MIXED'.

- **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 was designed to be used for jobs involving large numbers of objects, large objects, or significant amounts of data base processing work. A value of 'E' may be substituted for 'EXTERNAL'.

- **INTERNAL**
  - 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. A value of 'I' may be substituted for 'INTERNAL'.

Usage Notes

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.

Examples

- **+SVRMODE(INTERNAL)**
  - Processing will use internal servers.

- **+SVRMODE(M)**
  - Processing will proceed in mixed server mode, determining whether to use internal or external servers on its own.
### Parameters

<table>
<thead>
<tr>
<th>+SVRMODE (EXTERNAL)</th>
</tr>
</thead>
</table>

Processing will use external servers.
5.4.42  +SVRPROCD
Name of PROCLIB Member For Disk (DASD) Processing

Syntax

+SVRPROCD( procname )

Description

Specifies the name of the PROCLIB member for disk (or DASD) processing when a started task is used (+SVRTYPE(STC)). For general information on servers, refer back to Sec. 1.5, "Servers".

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.

Operands

procname  The name of a PROCLIB member containing code for a started task.

Example

+SVRPROCD(CODISK)

When processing is run with +SVRTYPE(STC) indicating a started task is to be run, member CODISK 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.

NOTE
The NGT CNTL library contains a sample started task procedure for use with this parameter – member CDBSEVRD. You may use this sample as is, or copy and modify it to customize it to your needs.
5.4.43  **+SVRPROCMSGCLAS**  
Message Class For Started Task Servers

**Syntax**

```
+SVRPROCMSGCLAS ( x )
```

**Description**

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: **+SVRPROCMSGGLVL**. For general information on servers, refer back to Sec. 2.2, "Servers".

**Operands**

- **x**  
  The desired message class.

**Example**

```
+SVRPROCMSGCLAS(A)
```

The Message Class for the NGT Server Started Tasks will be “A” rather then the installation default for Started Tasks.
5.4.44  +SVRPROCMSGLVL
Message Level For Started Task Servers

Syntax

+SVRPROCMSGLVL( x,y )

Description

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. For general information on servers, refer back to Sec. 2.2, "Servers".

Operands

x, y  The desired message level.

Example

+SVRPROCMSGLVL(1,1)

The Message Level for the NGT Server Started Tasks will be “1,1” rather then the installation default for Started Tasks.
5.4.45  +SVRPROCT
Name of PROCLIB Member For Tape Processing

Syntax

+SVRPROCT( procname )

Description

Specifies the name of the PROCLIB member for tape processing when a started task is used (+SVRTYPE(STC)). For general information on servers, refer back to Sec. 1.5, "Servers".

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.

Operands

procname  The name of a PROCLIB member containing code for a started task.

Example

+SVRPROCT(COTAPE)

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.

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.
Parameters

5.4.46  +SVRSPECIAL
Special Server Job Processing Control

Syntax

\[
+SVRSPECIAL( \begin{array}{c}
\text{NO} \\
\text{YES} \\
\text{EXTERNAL} \\
\text{INTERNAL}
\end{array} )
\]

Description

+SVRSPECIAL 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.

Operands

YES, Y  Default. Initiate a special copy server job right after the first Reorg has completed.

NO, N  Do not initiate a special copy server job. Use a regular server for the copy after all other Reorg servers have completed.

EXTERNAL, 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, I  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.

Recommendation

+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

+SVRSPECIAL(YES)

After the first Reorg has completed, immediately begin a copy server.
### Parameters

- **SVRSPECIAL (NO)**
  
  Use a regular server for copy after the last Reorg has completed.

- **SVRSPECIAL (INTERNAL)**
  
  Use an internal server to manage copies immediately after the first Reorg has completed.
5.4.47  +SVRTAPE
Number Of Tape Units To Use For Server Control

Syntax

+SVRTAPE( n )

Description

Specifies the number of tape servers to start for server processing.

☑️ NOTE
At least one tape server MUST be started for tape jobs. If +SVRTAPE(0) is specified, any image copies produced will go to disk (DASD).

Operands

n Specifies the number of tape servers to be utilized.

Examples

+SVRTAPE(3)

Use 3 tape servers for processing.

+SVRTAPE(0)

Do not use tape servers for processing.
5.4.48  **+SVRTAPEJC**

Job Class for Tape Server Job

**Syntax**

```
+SVRTAPEJC(x)
```

**Description**

Specifies the job class for tape server job. A new "+JC" symbolic may now be coded on the JOB statement, i.e., // 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. For general information on servers, refer back to Sec. 2.2, "Servers".

**Operands**

- **A**  Default job class for tape servers.
- **x**  The job class for the tape servers.

**Example**

```
+SVRTAPEJC(B)
```

Server job card, pointed to by the +SVRJCLJOB parameter, contains CLASS=JC. Server will run with CLASS=B.
5.4.49 +SVRTYPE
Type of Server Processing

Syntax

```
+SVRTYPE(JOB)
```

Description

Specifies the type of server processing. For general information on servers, refer back to Sec. 1.5, "Servers".

Operands

- **JOB**
  Default. Servers are to be run as batch jobs.
- **STC**
  Servers are to be run as started tasks. Use of this operand requires the additional use of the +SVRPROCD and +SVRPROCT parameters. See Sec. 4.4.27, "+SVRPROCD", and Sec. 4.4.28, "+SVRPROCT", for details.

Usage Notes

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.

Examples

```
+SVRTYPE(JOB)
```

Use batch jobs for server processing.

```
+SVRTYPE(STC)
+SVRPROCD(CODISK)
+SVRPROCT(COTAPE)
```

Use started tasks for server processing; the PROCLIB will use member CODISK for disk tasks and member COTAPE for tape tasks.
5.4.50 +SYSINERROR
SYSIN Error Processing Control

Syntax

![Syntax Diagram]

Description

Determines the action to take when a syntax error is detected in the SYSIN section of a utilities job.

Operands

- **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.

Examples

```
+SYSINERROR(ABEND)
```

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.
5.4.51  
+SYSINORDER  
SYSIN Order Processing Control

Syntax

```
+SYSINORDER( RANDOM )
+SYSINORDER( HONOR )
```

Description

For NGT Load usage in conjunction with NGT Unload only. Controls the order in which SYSIN statements are processed.

Operands

- **RANDOM**: Default. NGT processing will determine the order in which SYSIN statements are processed. The value "R" may also be used.
- **HONOR**: NGT processing will follow the order – as exactly specified – for processing SYSIN statements. The value "H" may also be used.

Examples

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
UNLD DATA FROM TABLE USR001.EXPENSES
UNLD DATA FROM TABLE USR001.PAYROLL
UNLD DATA FROM TABLE USR001.GEN_LEDGER
```

If the following is specified:

```
+SYSINORDER(RANDOM)
```

The three tables above may be unloaded in any order, and this order may vary from job to job. NGT processing determines the best order in order to optimize performance.

On the other hand, if the following is specified instead:

```
+SYSINORDER(HONOR)
```

The three tables above will be unloaded in the exact order specified: EXPENSES, PAYROLL and GEN_LEDGER.
5.4.52 +TIMEEXEC
XSUTTIME/Deadline Processing Control

Syntax

```
+TIMEEXEC ( NO  YES )
```

Description

Controls a second invocation of automation exit XSUTTIME. Note that processing makes two time checks for deadline processing by way of the XSUTTIME automation exit; see “YES” operand value below.

Operands

- **NO**  Default. XSUTTIME will be called only once, after all SYSIN statements have been processed.
- **YES**  XSUTTIME will be called twice – once after all SYSIN statements have been processed, and once again prior to BFACC (before access) processing.

Example

```
+TIMEEXEC(YES)
```

The system time will be checked after all SYSIN statements have been processed, and again before BFACC (before access) processing is initiated. If the system time is after the deadline set in automation exit XSUTTIME, processing will be terminated.
5.4.53 **+UIDHISTORY**
Utility ID History Control

**Syntax**

```
+UIDHISTORY(integer 0)
```

**Description**

Specifies the number of days a utility ID history will be kept. A utility ID history is kept in the NGT Checkpoint dataset.

**Operands**

- **integer**
  - A value from 1 to 180 days.
- **0**
  - Specifying a value of 0 will not maintain UID history beyond +OUTPUTAGE.

**Examples**

```
+UIDHISTORY(30)
```

Keep a history of utility IDs for 30 days.

```
+UIDHISTORY(0)
```

Because a value of 0 was specified, a history of utility IDs will not be kept beyond +OUTPUTAGE.
5.4.54 **+WAITTIMEOUT**

Set restart option of objects not available because another NGT utility has not released it.

**Syntax**

```
+WAITTIMEOUT ( EXCLUDE ERROR )
```

**Description**

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.

**Operands**

- **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**

```
+WAITTIMEOUT(EXCLUDE)
```

Put any object that is timed out waiting for another NGT utility to release it in the EXCLUDE queue so the object will be skipped.

**NOTE**

This parameter can be overridden by setting the return code (RC) in automation routine XSUTMTRX. The XSUTMTRX we originally shipped with V930 did override +WAITTIMEOUT in many instances. The more recent XSUTMTRX in CNTL does not.
5.4.55  

**+WORKMODE**  
Permanent Work Dataset Control

**Syntax**

```
+WORKMODE(
  TEMP,
  PERM,
  ANY,
  name
)
```

**Description**

+WORKMODE provides the capability of defining permanent work datasets which can improve utility performance by eliminating the portion of processing where work datasets are defined.

**Operands**

- **TEMP**  
  Default. Means to use temporary work datasets (as is done currently).

- **PERM**  
  Means to use permanent work datasets and if none are available, then fail the job. Requires specification of a name for a group of permanent work datasets; see **name** below.

- **ANY**  
  Means to use permanent work datasets if available and if none are available to then use temporary work datasets. Requires specification of a name for a group of work datasets if processing determines permanent work datasets will be used; see **name** below.

- **name**  
  The name of the work datasets set for use (default is DEFAULT).
Parameters

5.4.56  +WORKPREFIX

Dataset Name Prefix For VSAM Work Datasets

Syntax

+WORKPREFIX( charstring )

Description

+WORKPREFIX specifies the dataset name prefix for dynamic allocation of VSAM work (temporary) datasets. Four such datasets are created during processing:

- Information file for the CDBI Console; file name ends with .INFO001
- Output file for the CDBI Console; file name ends with .OUT001
- Scratch file used for general processing; file name ends with .WK00001
- Copy file for Reorg; file name ends with .RGTS001

Up to 15 characters can be specified for the work prefix, which is appended to the four file names above. The entire work dataset name, including all qualifiers, may not exceed 42 characters or IDCAMS errors may result.

If a value of "*" (asterisk) is coded, it will be substituted with the user ID associated with the JOB. If a value of "*charstring" is coded, the "*" will be substituted with the user ID associated with the JOB and the value of charstring will be appended. A "*" may only be specified in the first character position.

NOTES

It is important that the DASD pool used for the +WORKPREFIX datasets 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 datasets. 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.

Operands

charstring   A string of up to 15 characters.
Parameters

Examples

+WORKPREFIX(ADMIN74.NGTJOBS)

With the work prefix ADMIN74.NGTJOBS, the following four files will be created during processing:

- **ADMIN74.NGTJOBS.INFO001** (information file for the CDBI Console)
- **ADMIN74.NGTJOBS.OUT001** (output file for the CDBI Console)
- **ADMIN74.NGTJOBS.WK00001** (scratch file for processing)
- **ADMIN74.NGTJOBS.RGTS001** (copy file)
Parameters

5.4.57  +WORKUNIT
Unit Name for Temporary Files

Syntax

```
+WORKUNIT( string )
```

Description

+WORKUNIT specifies a unit name to be used for sequential temporary files. +WORKUNIT passes the unit name to XSUTDYNM for dynamic allocation. Only Restore uses sequential temporary files, those temporary files are for building indexes.

With NGT the standard work files are VSAM and +WORKUNIT does not apply. If SMS is not used then VOLUMES can be specified in XSUTDBMG.

Operands

<table>
<thead>
<tr>
<th>SYSDA</th>
<th>Default.</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>Any value acceptable to MVS may be supplied.</td>
</tr>
</tbody>
</table>
NGT Utility Reports

This chapter defines the reports generated by NGT Utilities that are common to multiple utilities.

6.1 NGT Utility Age Report

The NGT Checkpoint dataset maintains information about utilities that have been run. How long the product keeps this information depends on the parms +OUTPUTAGE and +UIDHIST. The CDBDISP statement is used to report on current and past NGT utility-ids.

If you have questions as to why a UID has not been aged off there is an optional utility age report you can request from any NGT utility job. Just add a //AGEREPRT DD SYSOUT=* to any NGT utility job and you will get the following report.

Figure 6-1. Sample Age Report.

<table>
<thead>
<tr>
<th>AGE-FAILED</th>
<th>UTILITY-ID</th>
<th>JOBNAME</th>
<th>JOB #</th>
<th>STEPNAME</th>
<th>DB2ID</th>
<th>STARTED</th>
<th>AGE VALUE</th>
<th>DATASET PREFIX</th>
<th>DATASET QUALIFIER</th>
<th>AGE-FAILED</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NGTPS02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.1.1 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
Reports

The value specified in +WORKPREFIX in the original utility job. This is the HLQ for the NGT work datasets.

DATASET QUALIFIER
The work dataset qualifiers generated by NGT to make these work datasets unique to this utility-ID.

AGE-FAILED REASON
The reason that the work datasets for this utility-ID have not been aged off. Possible values are:

- AUTHORIZATION
  The userid is not authorized to delete the work datasets
- DB2ID MISMATCH
  The current DB2ID does not match the DB2ID that the utility was run on
- NOT COMPLETE
  The utility-ID has not finished and should be restarted or quickexited
- OUTPUTAGE VALUE
  The +OUTPUTAGE value specified in the original utility-ID has not been reached

6.2 OBJREPRT (Processed Objects Report)
OBJREPRT shows a report listing all table spaces and indexes that have been processed, and their status at completion. An example of this report is shown in Figure 1-4. 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 dataset (using a DSN= keyword), an OBJREPRT DD SYSOUT=* statement is also required in order to generate the 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><em>WILDCARD</em></td>
<td>XYZSIM</td>
<td>TS</td>
<td></td>
<td>COMPLETED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH</td>
<td>XYZSEG</td>
<td>TS</td>
<td></td>
<td>COMPLETED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH</td>
<td>XYZP04</td>
<td>TS</td>
<td>1-4</td>
<td>COMPLETED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH</td>
<td>XYZP16</td>
<td>TS</td>
<td>1-16</td>
<td>COMPLETED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH</td>
<td>XYZP64</td>
<td>TS</td>
<td>1-64</td>
<td>COMPLETED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH</td>
<td>XYZPFF</td>
<td>TS</td>
<td>1-255</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>2</td>
<td>REORG</td>
<td>DBASMITH</td>
<td>AGRE05X1</td>
<td>IX</td>
<td>1-6</td>
<td>ERROR</td>
</tr>
<tr>
<td>3</td>
<td>COPY</td>
<td>DBASMITH</td>
<td>AGRE06</td>
<td>TS</td>
<td></td>
<td>EXCLUDED</td>
</tr>
<tr>
<td>4</td>
<td>CDBSTA</td>
<td>DBASMITH</td>
<td>RN2381</td>
<td>TS</td>
<td>1-10</td>
<td>EXCLUDED</td>
</tr>
<tr>
<td>5</td>
<td>LOAD</td>
<td>DBASMITH</td>
<td>RN2382</td>
<td>IX</td>
<td></td>
<td>EXCLUDED</td>
</tr>
</tbody>
</table>

Figure 6-1. OBJREPRT Report.

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 database 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.
Parameters

Status

The status of the processed object at job or step completion.

COMPLETED
Object has been processed successfully.

EXCLUDED
Object failed to process and +CLEANUP(YES) or XSVRXERR set ACTION=CLEANUP. Cleanup has occurred so no restart or quickexit is needed or possible to complete processing of these objects.

SELECTED BY RTS FOR PROCESSING
Self-explanatory, followed by post processing status.

IN-PROCESS
Object did not complete, possibly due to some error, since Cleanup processing did not occur, the object is available for restarting. Cleanup processing would not occur because it was not requested or because the object is in a must complete state preventing cleanup.

Object may have never started processing due to previous errors.

Objects that are In-Process will be processed upon restart.
For any utility that processes all indexes, or uses wildcards, the Database/Creator Name column will show the indication "*WILDCARD*", followed by a list of objects matching the wildcard naming mask. An example of this is shown in Figure 1-5.

<table>
<thead>
<tr>
<th>STMT #</th>
<th>UTILITY TYPE</th>
<th>DATABASE/CREATOR NAME</th>
<th>TABLESPACE/INDEX NAME</th>
<th>OBJECT TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COPY</td>
<td><em>WILDCARD</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH ABCSIM</td>
<td></td>
<td>TS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH ABCSEG</td>
<td></td>
<td>TS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH ABCP04</td>
<td></td>
<td>TS 1-4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBASMITH ABCP16</td>
<td></td>
<td>TS 1-16</td>
</tr>
</tbody>
</table>

Figure 6-2. OBJREPRT Report Showing Wildcard Expansion.

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 Tablespace Section of the SUMMARY report in the output. An example of this situation is shown in Figures 1-6 and 1-7.

<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>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COPY</td>
<td><em>WILDCARD</em></td>
<td></td>
<td></td>
<td>COMPLETED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSN9DCAT ABCSIM</td>
<td></td>
<td>TS</td>
<td>EXCLUDED BY RTS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSN9DCAT ABCSEG</td>
<td></td>
<td>TS</td>
<td>EXCLUDED BY RTS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSN9DCAT ABCP04</td>
<td></td>
<td>TS 1-4</td>
<td>SELECTED BY RTS FOR PROCESSING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSN9DCAT ABCP16</td>
<td></td>
<td>TS 1-16</td>
<td>SELECTED BY RTS FOR PROCESSING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSN9DCAT ABCP64</td>
<td></td>
<td>TS 1-64</td>
<td>SELECTED BY RTS FOR PROCESSING</td>
</tr>
</tbody>
</table>

Figure 6-3. OBJREPRT Report With RTS Results. Note the "EXCLUDED BY RTS" messages for tablespaces ABCSIM and ABCSEG.

-- CDB/UTILITIES NGT V9.2.0 -- FOR CDB
-- U.S. PATENTS: 5408654, 5517641, 5758357, 5887274

RTS COPY EXCLUDED TABLESPACE REPORT (STATEMENT 1)

<table>
<thead>
<tr>
<th>DATABASE NAME</th>
<th>TABLESPACE NAME</th>
<th>EXCLUDE PART(S)</th>
<th>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>

Figure 6-4. RTS Excluded Tablespace Report. Reasons for RTS exclusion for tablespaces ABCSIM and ABCSEG in Figure 1-6 are shown in the Exclude Reason column.
Chapter 7

NGT Work Dataset Control

This chapter defines the work datasets used by BMC NGT Version 9 and the parameters that name them and control how long they are kept on a mainframe system.

7.1 Overview of the NGT Work Datasets

In prior versions of NGT utilities we allocated datasets qualified with the high-level qualifier you supplied in +WORKPREFIX. These datasets were work datasets with a last node of WK*. NGT Version 9 utilities have these same datasets but the last node is not always WK*. The following are the four new prefixes used for the last node of the work datasets:

- **WK** - work datasets, may be needed for restart
- **RGTS** - concurrent copy temporary datasets used for Reorg table space only
- **INFO** - environmental information used for CDBDISP etc.
- **OUT** - journal output dataset, for console and CDB1DBUT

7.2 Naming the NGT Work Datasets

All of these datasets are qualified by the HLQ you supply in +WORKPREFIX. However, if you want the two datasets that are kept for a few days to be in a separate pool you can optionally provide a separate HLQ for them. This would be done with the +CONTROLPREFIX global parameter.

- **+WORKPREFIX** (prefix for INFO, WRK OUT, and RGTS)
- **+CONTROLPREFIX** (optional prefix for INFO and OUT only)

7.3 Controlling when NGT Work Datasets are Deleted

The new WK and RGTS datasets are deleted when a job is successful. The new INFO and OUT datasets are kept for some number of days. This number of days is controlled by the +OUTPUTAGE global parameter.

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.

The +DELETEOUTPUT global parameter is an override to tell us 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 CDB1DBUT. +DELETEWORK is not new to NGT V9, it tells us to delete the WK and RGTS datasets upon successful completion of the job; +DELETEWORK defaults to YES.
Usage Notes for NGT Work Datasets

Starting with NGT V9 all of these work datasets; WK, RGTS, INFO, and OUT, are VSAM datasets. This allows for shared access between the Master and Server jobs. The SMS pool where these dataset 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 datasets be extended-addressability enabled.

VSAM Work Datasets Deletion Review

When processing has ended successfully and +DELETEOUTPUT is set to (N) the .INFO and .OUT datasets are not deleted (the .RGTS and .WK files are deleted). The .INFO and .OUT datasets will be kept until a job is run specifying +OUTPUTAGE(x) where x is number of days. .INFO and .OUT datasets from previous jobs that meet this age specification are then deleted. This process is similar to the MODIFY command in DB2. For example, if a successful job has been run specifying +DELETEOUTPUT(N) on March 15, 2009 at 2:00pm the .INFO and .OUT datasets for those jobs will be kept. If the next job run is not until March 17th at 3:00pm and +OUTPUTAGE(2) is specified, then this message will be seen in the output:

NGTZ117 STARTED AGE PROCESSING FOR COMPLETED ENTRIES

This processing deletes all files associated with successful runs that are older than the number of days specified in +OUTPUTAGE(x). In the above example, .INFO and .OUT files would be deleted from the March 15th job as the files are more than 2 days old. Any other outstanding files meeting that criteria will also be aged off with the same process.

Figure 7-1 on the next page helps illustrate this.
7.6 **Permanent Work Datasets and the CDBWORK Command**

In order to provide optimal performance for processing smaller amounts of work, permanent work datasets may be utilized instead of temporary work datasets, which are automatically generated and deleted from job to job. Using permanent datasets eliminates the stage of processing in which temporary datasets are created, and this omission can provide some savings in processing resources.

Definition of permanent work datasets is performed with the CDBWORK command, shown on the next page.
7.6.1 **CDBWORK Syntax**

The `+WORKMODE` parameter must be used with the `CDBWORK` command. Syntax is as follows:

```
CDBWORK
  DEFINE NAME(name)
  DELETE NAME(name)
  RELEASE NAME(name)
  DISPLAY NAME(name) ALL DETAIL
  FORCE
  SET(integer) workoptions
  XSUTDBMG YES NO
```

- **DEFINE** Means to define a work dataset with the specified options.
- **DELETE** Means to delete a work dataset.
- **RELEASE** Means to release a work dataset.
- **DISPLAY** Means to display the options for a particular work dataset specified by name, or details on all defined work datasets.
- **XSUTDBMG** Tells the CDBWORK command to either use (YES) or not use (NO) the XSUTDBMG automation exit for all allocations of permanent work datasets. If YES is specified, then processing will use the XSUTDBMG exit even if other options are specified on the CDBWORK command in the SYSIN.

7.6.2 **workoptions Syntax**

Syntax for `workoptions` for the DEFINE keyword is as follows. Note that all keywords are required.

```
NUMBER(integer) PREFIX(name)
WORK_PRIMARY(memory_size) WORK_SECONDARY(memory_size)
RGTS_PRIMARY(memory_size) RGTS_SECONDARY(memory_size)
INFO_PRIMARY(memory_size) INFO_SECONDARY(memory_size)
OUT_PRIMARY(memory_size) OUT_SECONDARY(memory_size)
STORCLAS(class) MGMTCLAS(class) DATACLAS(class)
VOLUME(volser) UNIT(unitname) MAXSIZE(memory_size)
```

7.6.3 **CDBWORK Keywords**

**NAME**
8-character name to identify a set of work datasets. 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.
NGT Work Dataset Control

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 datasets. The default is 500 M.

RGTS_SECONDARY
The secondary allocation of the RGTS work datasets. 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 datasets.

MGMTCLAS
The SMS management class to use when allocating the datasets.

DATACLAS
The SMS data class to use allocating the datasets.

VOLUMES
The DASD volumes to use allocating the datasets. 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 datasets are allowed to grow before reallocation. The default is 0. Work files can grow to any size.

FORCE
Deletes the set even though some of the data set are assigned to utilities.

SET
The number of the set to be released.
Example

```
//STEPLIB   DD ...
//CDBUSTEP EXEC PGM=CDBUTIL,PARM='SSID,,RESTART'
//SYSPRINT DD SYSOUT=
//UTPRINT   DD SYSOUT=
//SYSIN     DD *
CDBWORK 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)
```

CDBWORK is submitted just like any NGT utility statement would be using the same JCL. The above instruction defines a set of permanent workfile datasets with the group name WORKSET1. It has the following characteristics:

- It will use up to 10 workfile datasets.
- 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.

### 7.6.4 Usage With +WORKMODE Parameter

Once permanent work datasets have been defined with the CDBWORK command, their usage can be switched on or off using the +WORKMODE global parameter. For details on the +WORKMODE parameter, see Section 5.4.54, "+WORKMODE".
Chapter 8

**NGT Checkpoint Dataset Maintenance**

The CDBCKPT is a NGT repository used to hold pertinent information for the NGT utilities.

The NGT Checkpoint has a directory and internal files. The directory maps internal files in the repository and it is implemented as a DB2 TYPE2 index. The index also maps free space in the repository.

8.1 **NGT Checkpoint Reorganization**

The NGT Checkpoint repository can be maintained with one simple job which should be run every six months. See sample job below.

The directory index can be reorganized to speed up the process and reduce its size due to page splits. The `DBMG REORG` statement is for this purpose.

The directory also contains free space records. After excessive use those records need to be compacted in order to improve performance and reduce the size of the index. The `DBMG COMPACT` statement is for this purpose.

Before running REORG or COMPACT, it is recommended to run `DBMG CHECKIX` to ensure the validity of the directory index.

In the checkpoint repository there are other indexes that need to be reorganized. The `DBMG OPTIMIZE` statement is for this purpose.

Finally, there is a `DBMG REPORT` statement run before and after the reorganization statements; this is for NGT diagnosis in the case of errors.

**SAMPLE JOB**

Running this job twice a year, monthly on very busy systems, will maintain the performance and size of your CDBCKPT dataset. This job is not disruptive; it can run concurrent with NGT utilities.

```
//DBMG  EXEC PGM=CDB1DBMG
//STEPLIB DD DISP=SHR,DSN=your.ngt.loadlib
//SYSPRINT DD SYSOUT=*  
//CDBPRINT DD SYSOUT=*  
//SYSLIN DD *  
DBMG REPORT  LOGDS(your.cdbckpt)  
DBMG CHECKIX LOGDS(your.cdbckpt)  
DBMG OPTIMIZE LOGDS(your.cdbckpt)  
DBMG COMPACT LOGDS(your.cdbckpt)  
DBMG REORG LOGDS(your.cdbckpt)  
DBMG REPORT LOGDS(your.cdbckpt)  
```

If the CHECKIX fails the statements that follow the CHECKIX will not be executed; send the job output to BMC Customer Support for diagnosis.
8.2 **Back Up NGT System Datasets**

There are special considerations for backing up and restoring datasets used by NGT Utilities for its processing – the LOGPDS and CDBCKPT datasets. Care must be exercised in performing backup or restore operations, as failure to perform them correctly may cause an interruption or failure of NGT utilities.

8.2.1 **LOGPDS**

The LOGPDS is a PDS (it can also be defined as a PDSE) that is used to write a mini-dump when a NGT utility abends with an 0C3 due to unexpected results from one of its many integrity checks.

This PDS should have its older members deleted periodically. The members are only needed to document current issues. This PDS can be periodically redefined. Place the LOGPDS dataset on a volume that is backed up periodically in order to facilitate disaster recovery.

8.2.2 **CDBCKPT**

The CDBCKPT (also known as "checkpoint") dataset keeps track of active and restartable NGT utilities, registration of NGT RTS tables, registration of NGT applications, and registration of optional model AMS statements. This dataset, although critical to a NGT utility environment, requires little additional maintenance other than periodic backing up.

The CDBCKPT dataset is a linear VSAM datasets with a custom format so REPRO cannot be used on them. This dataset 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 NGT CDBCKPT should be placed on mirrored DASD so the utility environment can continue at the alternate site.

8.3 **Restoring NGT System Datasets**

8.3.1 **Restoring the LOGPDS**

In a disaster recovery situation when NGT datasets need to be restored the LOGPDS is easy; simply restore it from a backup volume or redefine it. The mini dumps kept in the LOGPDS are not critical to executing utilities, they are only for diagnostic purposes.

8.3.2 **Restoring the CDBCKPT**

In a disaster recovery situation the CDBCKPT dataset is critical to the continued execution of utilities. Restoring this dataset is best because the RTS and NGT Application information will not be lost. However because everything is restored to some prior point-in-time, utility restart will not be possible.

For restoring the CDBCKPT there are two options:

- **Restore them from backup just like the LOGPDS.** Then use the CDBDISP product to check for utility ID’s that were “in-flight” (that is, in progress) at the time of the backup and
resubmit any applicable utility jobs with the restart parameter set to **FORCEID**. This would be the normal way to restore a NGT system. **NOTE:** Always confer with BMC Customer Support before restarting a utility with FORCEID as this could cause data loss if the utility is in a must-complete state.

- **Redefine the datasets.** Alternatively, if backups don’t exist, these datasets can be redefined as if a new install were being performed. **NOTE:** If this is done, any NGT Utility definitions, BGT RTS definitions, NGT AMS definitions, and NGT AMS definitions will be lost.

The NGT RTS DB2 tables will be recovered and can be redefined to NGT; see the section in the *BMC Next Generation Technology Automation Reference Manual* entitled, "Creating the RTS Tables". The NGT utility will have to be manually redefined. For further information on defining applications, see Sec. 1.3 "Applications". Any NGT AMS overrides can be redefined from their original member, see Sec. 1.10 "Managing NGT AMS Define".
Chapter 9

**CDBI (CDB Interactive)**

This chapter covers the fundamentals of operation for CDB Interactive.

**Overview**

CDBI is an ISPF-based menu where NGT applications are made available. Here you can:

1. Display utilities and view their output.
2. View NGT error messages.
3. Display and manage your NGT RTS control tables
4. Display any AMS overrides you may use.

The advantage to viewing NGT Utility output through this interface is that here you see the output for the Master job and all Server jobs as one output.

Utility output is available through this interface depending on parms +OUTPUTAGE and +DELETEOUTPUT. +UIDHISTORY allows the utility list available to this facility to be maintained after the output is aged off.

Details on these parms are available in chapter 5 Parameters.

**Installation**

This new ISPF based display utility requires a listener be started. Refer to the *BMC Next Generation Technology Installation Guide* for instructions on starting the listener(s).

Once the Listeners are in place to pass data to this facility all that is needed is the editing of two members in the NGT CNTL library. See the *BMC Next Generation Technology Installation Guide* for details.

### 9.1 Starting CDBI

Now this new facility can be started with the simple command or can be selected from an ISPF menu; however you chose to set it up. For instance:

```plaintext
EX ‘yourCDBhlq.CNTL(CDBUTDSP)’
```

Or put the following REXX exec in a library in your SYSPROC or SYSEXEC concatenation with member name CDBI:

```plaintext
/*REXX*/
ADDRESS TSO "EX ‘yourCDBhlq.CNTL(CDBUTDSP)’"
```

Then CDBI is started by typing **CDBI** from the ISPF Command Shell or TSO CDBI from any panels command line. The fist thing you see is the CDBI menu.
The rest of this chapter will discuss the first option; Display Utility.

9.1.1 Environment Selection

When Display Utility is initially started, a choice of environment must be made. (If there is only one environment defined, this screen will still appear.)

To choose an environment, place an "S" on a line corresponding to the DB2 subsystem ID (SSID) and checkpoint dataset desired for operation.

Figure 9-1. CDBI Environment Selection.

To choose an environment, place an "S" on a line corresponding to the DB2 subsystem ID (SSID) and checkpoint dataset desired for operation.
The ALL option – which always appears at the top of the list – indicates that all DB2 subsystems will be utilized. Placing an "S" by this option will allow the monitoring of every DB2 subsystem concurrently.

**NOTE:** Some checkpoint datasets will be associated with multiple DB2 subsystems, thus appearing on more than one line.
9.2 Display Utilities

The home screen for Display Utility is a listing of all NGT utility IDs (UIDs).

Figure 9-2. Display Utilities Home Screen.

9.2.1 Data Columns

This list consists of the following columns:

- **SEL**: Selection. Use to select a particular utility ID.
- **SSID**: DB2 Subsystem ID. If "ALL" was selected on the Environment Selection screen, there will be multiple SSID values in this column.
- **UID**: Utility ID. A utility ID of up to 18 characters.
- **MASTER JOB (ID)**: Name and number for the NGT master job (servers are not listed).
- **START TIME**: The time at which a master job was submitted originally.
- **RC**: Return code from the master job.
- **STATUS**: The status of the master job. Usually this field will be blank when it is non-blank it will show codes indicating how a job was terminated. This column may have one of the following values:
  - **FORCE**: UID was forced, using the FORCEID value for the Restart parameter.
9.2.2 Line Commands

The following line commands may be specified for any UID by placing one of the following characters in the SEL field on any line in the list:

D  **Details.** Shows the details of a utility, including any objects processed, the product used, and its completion status. Figure 8-3 shows an example.

I  **SYSIN Statements.** A listing of SYSIN statements for a particular master job. This screen presents the DB2 subsystem ID (SSID), the name of the product run (along with a product code number), the entire text of a SYSIN statement, and the length of the statement. Where there is more than one SYSIN statement on this screen, use the NEXT and PREV keys (usually PF8 and PF7) to scroll through to successive SYSIN statements, as only one SYSIN statement is shown per screen. Figure 8-6 shows an example.

M  **Master Job Information.** Shows the following for the selected UID: DB2 subsystem ID, system name, job name, ASID, server ID number, JES job number, the last change time for the job, start time, RID, and the number of servers that the master started. Also included are several status flags. Figure 8-4 shows an example.

O  **Output Listings.** This command presents a pop-up menu from which specific output listings may be selected. Figure 8-8 shows an example of this submenu. The number of output lines may be limited by specifying a value in the **MAX Lines** entry field.

Q  **Quickexit Command.** This command allows the immediate termination of any utility. A special message will appear, asking for confirmation of the Quickexit operation. Placing a "Y" in the "YES I DO" field and pressing Enter will Quickexit the selected utility. Figure 8-7 shows an example of this command.

V  **Server Job Information.** Shows the following for the selected UID: DB2 subsystem ID, system name, job name, ASID, server ID number, JES job number, the last change time for the job, start time, RID, and the number of servers that the master started. Also included are several status flags. Figure 8-5 shows an example.

---

Q-EXITED  UID was quickexit, using the QUICKEXIT value for the Restart parameter.

NA  No output was available for this UID.
Figure 9-3. Details Screen.

Figure 9-4. Master Job Information Screen.
Figure 9-5. Server Job Information Screen.

Figure 9-6. SYSIN Information Screen.
Figure 9-7. Quickexit Confirmation Screen.

Figure 9-8. Output Listing Pop-Up Menu.
9.2.3 **Primary Commands**

In addition to the line commands, there are several actions that may be typed at the command line prompt (CMD ==>):

- **R** Refresh. Refresh the display with the latest information.

- **FILT** Filter Results. Use this option to reduce the number of lines viewed by specifying a selection criteria. Jobs may be filtered according to:
  - Whether output is currently available
  - Whether certain jobs have a specific return code
  - Only jobs which have a utility in error
  - Only jobs in must-complete status
  - Only jobs that have had Quickexit specified
  - Only jobs that have been terminated with FORCEID

  An example of the Filter Results screen is shown in Figure 9-9.

- **MAT** Matrix. Use this option to display all enqueues and other resources being held by processing.

- **ENV** Environment. Use this option to display all enqueues and other resources being held by processing. An example of the Filter Results screen is shown in Figure 8-10.

The **END** command may be typed at the command prompt to terminate Display Utility.
Figure 9-9. Filter Results Screen.

Figure 9-10. Environment Display.
### 9.2.4 CDBI – Display Utility error messages

In the table below, the first column is the error code displayed in the Display Utility panel, the second column is the NGT message that shows in the Listener started task, the third column is its description.

<table>
<thead>
<tr>
<th>Panel</th>
<th>Listener</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>E30</td>
<td>NGTY330</td>
<td>TCP/IP RECEIVE BUFFER ERROR ($*32).</td>
</tr>
<tr>
<td>E31</td>
<td>NGTY543</td>
<td>TCP/IP UNABLE TO ACCESS UTILITY FILE.</td>
</tr>
<tr>
<td>E32</td>
<td>NGTY544</td>
<td>TCP/IP REQUESTED ENTRIES NOT FOUND IN UTILITY FILE.</td>
</tr>
<tr>
<td>E33</td>
<td>NGTY545</td>
<td>TCP/IP UNABLE TO ACCESS STATUS FILE WITH KEY ($*32).</td>
</tr>
<tr>
<td>E34</td>
<td>NGTY546</td>
<td>TCP/IP REQUESTED ENTRIES NOT FOUND IN STATUS FILE WITH ($*32).</td>
</tr>
<tr>
<td>E35</td>
<td>NGTY547</td>
<td>TCP/IP INFORMATION FILE ($*44) DOES NOT EXIST.</td>
</tr>
<tr>
<td>E36</td>
<td>NGTY548</td>
<td>TCP/IP UNABLE TO ACCESS INFORMATION FILE ($*44).</td>
</tr>
<tr>
<td>E37</td>
<td>NGTY549</td>
<td>TCP/IP REQUESTED ENTRIES NOT FOUND IN INFO FILE ($*44).</td>
</tr>
<tr>
<td>E38</td>
<td>NGTY550</td>
<td>TCP/IP INFO FILE ($*44) HAS MORE THAN ONE MASTER RECORD.</td>
</tr>
<tr>
<td>E39</td>
<td>NGTY551</td>
<td>TCP/IP OUTPUT FILE ($*44) DOES NOT EXIST.</td>
</tr>
<tr>
<td>E40</td>
<td>NGTY552</td>
<td>TCP/IP UNABLE TO ACCESS OUTPUT FILE ($*44).</td>
</tr>
<tr>
<td>E41</td>
<td>NGTY553</td>
<td>TCP/IP REQUESTED ENTRIES NOT FOUND IN OUTPUT FILE ($*44).</td>
</tr>
<tr>
<td>E42</td>
<td>NGTY555</td>
<td>TCP/IP INFO FILE ($*44) HAS INVALID SYSIN SEQUENCE NUMBER.</td>
</tr>
<tr>
<td>E43</td>
<td>NGTY558</td>
<td>TCP/IP IS UNABLE TO ACCESS DB2 SSID ($$$), WITH PLAN NAME ($$$$$$).</td>
</tr>
<tr>
<td>E44</td>
<td>NGTY559</td>
<td>TCP/IP INFO FILE ($*44) HAS MISMATCHING SERVER RECORDS.</td>
</tr>
<tr>
<td>E45</td>
<td>NGTY562</td>
<td>TCP/IP UNABLE TO ACCESS SYSPDS ONES FILE.</td>
</tr>
<tr>
<td>E46</td>
<td>NGTY563</td>
<td>TCP/IP REQUESTED ENTRIES NOT FOUND IN SYSPDS ONES FILE.</td>
</tr>
<tr>
<td>E47</td>
<td>NGTY564</td>
<td>TCP/IP UNABLE TO ACCESS OBJECT REGISTRATION FILE.</td>
</tr>
<tr>
<td>E48</td>
<td>NGTY565</td>
<td>TCP/IP REQUESTED ENTRIES NOT FOUND IN OBJECT REGISTRATION FILE.</td>
</tr>
<tr>
<td>E49</td>
<td>NGTY566</td>
<td>TCP/IP UNABLE TO ACCESS PRODUCT FILE.</td>
</tr>
<tr>
<td>E50</td>
<td>NGTY567</td>
<td>TCP/IP REQUESTED ENTRIES NOT FOUND IN PRODUCT FILE.</td>
</tr>
<tr>
<td>E51</td>
<td>NGTR216</td>
<td>UID $$$$$$$$$$$$$$$$$$ DOES NOT EXIST IN CONTROL MEMBER.</td>
</tr>
<tr>
<td>E52</td>
<td>NGTR155</td>
<td>ERROR OPENING SYSTEM PDS, DDNAME=SYSPDS.</td>
</tr>
</tbody>
</table>
9.2.5 **CDBI – Display Utility warning messages**

In the table below, the first column is the warning code displayed in the Display Utility panel, the second column is its description.

<table>
<thead>
<tr>
<th>Panel</th>
<th>Non-Error (Warning) Message description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W80</td>
<td>TCP/IP OUTPUT FILE ($*44) DOES NOT EXIST.</td>
</tr>
<tr>
<td>W81</td>
<td>DB2 SSID ($$$) WITH PLAN NAME ($$$$$$$) IS NOT AVAILABLE.</td>
</tr>
</tbody>
</table>

9.3 **MESSAGES**

This option initially shows the messages in order. From here you can find "F" or locate "L" a particular message ID or partial message ID. For example: L NGTR114. Then select "S" on a line to see all the message text.

![Message List](image)

Sel: **S** Display detail. Commands: Locate Message Id; Message Id.
9.4 **Display RTS**

The initial display below allows for the maintenance of the Utility Manager Criteria table, Exceptions table, or the Schedule table.

From here you can Select ‘S’ or Insert ‘I’ each type of record. By selecting a record first, those values will be presented in the Insert to save you typing and to make it easy to insert a record like an existing record. Below are overview of each.

9.4.1 **Criteria Table**

<table>
<thead>
<tr>
<th>SEL</th>
<th>Type</th>
<th>Name</th>
<th>Part</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>IX</td>
<td>% %</td>
<td>0</td>
<td>DEFAULT IX CRITERIA</td>
</tr>
<tr>
<td>_</td>
<td>TS</td>
<td>% %</td>
<td>0</td>
<td>BASE TS CRITERIA</td>
</tr>
</tbody>
</table>

From this panel you can Select, Update, or Delete individual rows. You can also insert a new row using the INS primary command. Note; if you Select a row first the results from that select will be used for the following Update or INSERT request. This not only saves typing but provides an easy way to insert a new row like an existing row.

9.4.2 **Exceptions Table**

<table>
<thead>
<tr>
<th>SEL</th>
<th>Type</th>
<th>Name</th>
<th>Part</th>
<th>Reorg</th>
<th>Copy</th>
<th>Stats</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>TS</td>
<td>BI12DWA. %</td>
<td>0</td>
<td>F</td>
<td>I</td>
<td>I</td>
<td>TEST RECORD</td>
</tr>
</tbody>
</table>

From this panel you can Select, Update, or Delete individual rows. You can also insert a new row using the INS primary command. Note; if you Select a row first the results from that select will be used for the following Update or INSERT request. This not only saves typing but provides an easy way to insert a new row like an existing row.
9.4.3 Schedule Table

<table>
<thead>
<tr>
<th>SEL</th>
<th>Type</th>
<th>Name</th>
<th>Part</th>
<th>Utility ID</th>
<th>Reorg</th>
<th>Copy</th>
<th>Stats</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TS</td>
<td>BI12DWNA.PART0041</td>
<td>1</td>
<td>RRGSCDH</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>TS</td>
<td>BI12DWNA.PART0040</td>
<td>0</td>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

From this panel you can Select, Update, or Delete individual rows. You can also insert a new row using the INS primary command. Note; if you Select a row first the results from that select will be used for the following Update or INSert request. This not only saves typing but provides an easy way to insert a new row like an existing row.
9.5 **Display AMS**

Few users need to use this feature. If you have shop standards for the VSAM Define statement which NGT Utilities use to define the DB2 datasets you can override the default model. CDBI selection allows you to display any global or local override that is in place for this AMS model.

Call BMC Customer Support for guidance with confirming the use of AMS overrides and migrating away from the SYSPDS dataset.

✔ **NOTE**

The location of the override and the procedure for providing an override changed with NGT V11.2 (formerly CDB V940). This CDBI feature is for the new method. Prior to V11.2 the AMS override was stored in the SYSPDS. Now the override is stored in the CDBCKPT and the SYSPDS is no longer used.
CDBDisp

Chapter 10  

CDBDisp

CDBDISP is an utility that displays the current status of NGT utilities.

☑️ NOTE  
This utility runs only in the master address space.

CDBDISP produces a report in the SUMMARY DD statement, QUICKEXIT JCL for selected utilities, and DELETE statements for selected datasets.
10.1 **Input: DD Statements, Syntax, Keywords**
This chapter covers the DD statements, SYSIN input (statement syntax), keywords, and parameters for this utility.

10.1.1 **DD Statements**

**SYSIN**
SYSIN is the first required DD statement for use with this utility. Specify utility statements as follows:

```
//SYSIN DD *
 (statement)
 (statement)
 (statement)
```

Statement syntax is covered in Sec. 2.2, "SYSIN Syntax Diagram", on the next page.

**SUMMARY**
SUMMARY is the second required DD statement for use with this utility. Specify this statement as follows:

```
//SUMMARY DD *
```

Output may be sent to the job stream, or to an output dataset.

**CDBEXEC**
If you wish to use this utility with automation control points, specify a CDBEXEC DD statement. This statement should reference a partitioned dataset containing the automation exits you wish to use. An example of how this DD statement might appear in your JCL is shown here:

```
//CDBEXEC DD DISP=SHR,DSN=NGT.AUTOMATN.CTRL.PTS
```

For more information on automation control points, please see the manual *BMC Next Generation Technology Automation Reference Manual*. 
10.1.2 SYSIN Syntax Diagrams

10.1.3 Keywords

10.1.3.1 Display Utility keywords

CDBDisp

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 datasets, 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. See Chapter 5, "Relevant Automation", for details on XSUTDISP.

GENQE
CDBDisp

Causes the automation control point XSUTDISP to be invoked, and to generate a QUICKEXIT. See Chapter 5, "Relevant Automation", for details on XSUTDISP.

GENDELETES

Causes the automation control point XSUTDISP to be invoked, and to generate DELETE statements. See Chapter 5, "Relevant Automation", for details on XSUTDISP.

ORDER BY

Causes the report to be sorted by the specified field. The fields which can be sorted are: UID, STATUS, STARTED, ENDED, or JOBNAME.
10.1.4 **Display Matrix**

**CDBDISP MATRIX**
Displays all entries in the matrix. Use to show any “NGT locks” held on objects by NGT utility IDs.

10.1.5 **Display Passwords**

**CDBDISP PASSWORDS**
Produces a report of all installed products with their passwords, expiration dates, and the number of days until those expiration dates.
## 10.2 Report Columns

### 10.2.1 Display Utility report

<table>
<thead>
<tr>
<th>UTILITY-ID</th>
<th>UTILITY STATUS</th>
<th>STARTED</th>
<th>ENDED</th>
<th>JOBNAME</th>
<th>JOB #</th>
<th>STEPNAMES</th>
<th>CODE QUALIFIER</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDBCOPYXL2</td>
<td>RESTARTABLE</td>
<td>2012-07-01 14:15:47</td>
<td>2012-07-02 14:16:00</td>
<td>CDBRRX</td>
<td>8</td>
<td>MJPEFOCK.PMKANLIE</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>CDBCOPYXL3</td>
<td>RESTARTABLE</td>
<td>2012-08-01 09:03:45</td>
<td>2012-08-02 09:03:58</td>
<td>CDBRRX</td>
<td>8</td>
<td>MJPEFOCK.PMKANLIE</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>CDBRRIX</td>
<td>RESTARTABLE</td>
<td>2012-08-03 14:15:47</td>
<td>2012-08-03 14:16:00</td>
<td>CDBRRX</td>
<td>8</td>
<td>MJPEFOCK.PMKANLIE</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>JOBP701</td>
<td>COMPLETED OK</td>
<td>2012-09-23 20:07:08</td>
<td>2012-09-24 20:07:28</td>
<td>JOB25349</td>
<td>0</td>
<td>MJPDGFKA.EIPEALIE</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>JOBL1006</td>
<td>COMPLETED OK</td>
<td>2012-09-25 09:50:47</td>
<td>2012-09-25 09:51:08</td>
<td>JOB25430</td>
<td>0</td>
<td>MJPHCHKE.GBPBGAAE</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>JOBL1010</td>
<td>NO RESTART</td>
<td>2012-09-29 16:38:57</td>
<td>2012-09-29 16:39:26</td>
<td>JOBL1010</td>
<td>8</td>
<td>MJDDJCMX.BOLLAPIA</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>CDBRXX</td>
<td>RESTARTABLE</td>
<td>2012-09-25 10:30:44</td>
<td>2012-09-25 10:31:04</td>
<td>CDBRXX</td>
<td>8</td>
<td>MJDDJCMX.BOLLAPIA</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

This is a sample report produced by CDBDISP. The columns in a CDBDISP 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 Figure 3.2) – this signifies the utility has completed. Utility ID values without an asterisk (such as for the last row in the sample report in Figure 3.2) were not fully completed at the time the report was compiled.

**UTILITY STATUS**

The status of the associated utility ID.

<table>
<thead>
<tr>
<th>UTILITY 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 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>NO RESTART</td>
<td>Utility ID 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>
<tr>
<td>JOB RUNNING</td>
<td>Utility ID active when CDBDISP 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 dataset.
### 10.2.2 Matrix report

**10/05/12 06:18:57**

<table>
<thead>
<tr>
<th>DATABASE OBJECT</th>
<th>OBJECT LOCK</th>
<th>LOCK ID</th>
<th>NAME</th>
<th>PART</th>
<th>TYPE</th>
<th>TYPE</th>
<th>STATUS</th>
<th>UTILITY-ID</th>
<th>JOBNAME</th>
<th>JOB #</th>
<th>TIMESTAMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDB1DSU2 TSRSPTV2</td>
<td>TS</td>
<td>EXCLUSIVE</td>
<td>HOLDING</td>
<td>JOBF6244</td>
<td>JOBF6244</td>
<td>JOBF6244</td>
<td>JOBF6244</td>
<td>2012-09-25-16.44.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDB1DSU2 IRSPTV20</td>
<td>IX</td>
<td>EXCLUSIVE</td>
<td>HOLDING</td>
<td>JOBF6244</td>
<td>JOBF6244</td>
<td>JOBF6244</td>
<td>JOBF6244</td>
<td>2012-09-25-16.44.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDB1DSU2 IRSPTV21</td>
<td>IX</td>
<td>EXCLUSIVE</td>
<td>HOLDING</td>
<td>JOBF6244</td>
<td>JOBF6244</td>
<td>JOBF6244</td>
<td>JOBF6244</td>
<td>2012-09-25-16.44.16</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is a sample SUMMARY report from a CDBDISP 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. The NGT Matrix is used to prevent two NGT utilities from processing the same object at the same time. This NGT Lock has no effect on DB2 or other non-NGT processing.

### 10.2.3 Passwords report

**10/05/12 11:00:08**

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>PASSWORD</th>
<th>LOCAL PASSWORD</th>
<th>EXPIRE DATE</th>
<th>EXPIRE DAYS</th>
<th>ERROR MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Index</td>
<td>A52EBF69B4811860</td>
<td>10/15/2012</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copy</td>
<td>26Z119F194AB81CA</td>
<td>10/15/2012</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dlad</td>
<td>251AA93B09F60A1E</td>
<td>10/15/2012</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load</td>
<td>65AC7702B5A5B30F</td>
<td>10/15/2012</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rebuild Index</td>
<td>B57192648A57B50F</td>
<td>10/15/2012</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restore</td>
<td>7B5112B8B3D301A</td>
<td>10/15/2012</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Report</td>
<td>8EB3B7C66C9751F</td>
<td>10/15/2012</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reorg</td>
<td>98B3B696B02SS1S</td>
<td>10/15/2012</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rts</td>
<td>267B0636D43C6A5</td>
<td>10/15/2012</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stats</td>
<td>9FF7F41163B5C9F</td>
<td>10/15/2012</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unload</td>
<td>7F48B7113F02BA51</td>
<td>10/15/2012</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is a sample SUMMARY report from a CDBDISP PASSWORDS. It shows the password for each NGT Utility as well as when and how long before it expires.

### 10.2.4 Restore report

**10/05/12 09:41:15**

<table>
<thead>
<tr>
<th>DATABASE OBJECT</th>
<th># PARTS</th>
<th>NPIS FIRST</th>
<th>PARTS DONE</th>
<th>NPIS DONE</th>
<th>NAME</th>
<th>QUALIFIER</th>
<th>DATASET</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDB/UTILITIES NGT</td>
<td>12</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>RESTORP1</td>
<td>USER.B.</td>
<td>10/02/12 8:37:05 RESTORP3 10/02/12 8:37:09 WORK.PREFIX.MSSM.getAbsolutePath().</td>
</tr>
</tbody>
</table>

This is a sample SUMMARY report from a CDBDISP RESTORE DB(USER%). It shows a 12 part table space where 3 parts have been restored. Job RESTORP1 was the initial job used to restore this TS and RESTORP3 was the latest job to run.
10.3 **Examples**

This chapter covers the syntax for specific examples of CDBDISP for displaying utilities. The other reports available with CDBDISP have few keywords and are self-explanatory.

10.3.1 **Show the Status of All Utilities**

To produce a report showing the status of all utilities, simply specify the CDBDISP keyword without any other keywords following it.

```sql
//SYSIN DD *
CDBDISP
```

10.3.2 **Show 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.

```sql
//SYSIN DD *
  CDBDISP UTIL(BANKSTATMNTS)
  CDBDISP UTIL(WILMINGTON49)
  CDBDISP UTIL(CME18)
```

10.3.3 **Show 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.

```sql
//SYSIN DD *
  CDBDISP UTIL(SWREGION%)
  CDBDISP UTIL(*A320)
  CDBDISP UTIL(%STREORG%)
```

10.3.4 **Show the Status of Only Completed Utilities**

To produce a report showing the status of only completed utilities, specify the COMPLETED keyword (RC = 0)

```sql
//SYSIN DD *
  CDBDISP COMPLETED
```

You can combine this keyword with the UTIL keyword to check for completed utilities with a common naming pattern, as in:

```sql
//SYSIN DD *
  CDBDISP UTIL(AMBACKUPS%) COMPLETED
```

10.3.5 **Show 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 *
  CDBDISP 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 *
  CDBDISP UTIL(COPIES909%) ERROR

## 10.3.6 Show the Status of Only Restartable Utilities

To produce a report showing the status of only restartable utilities, use the RESTARTABLE keyword.

//SYSIN DD *
  CDBDISP 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 *
  CDBDISP UTIL(TXREGION%) RESTARTABLE

## 10.3.7 Show the Status of Utilities Currently Running

CDBDISP 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 *
  CDBDISP ACTIVE

You can combine this keyword with the UTIL keyword to check for active utilities with a common naming pattern, as in:

//SYSIN DD *
  CDBDISP UTIL(COPYPACIF%) ACTIVE

You can also use the keyword RUNNING if you prefer, to display all running utilities, as in:

//SYSIN DD *
  CDBDISP RUNNING

Or, for a list of all running utilities meeting a specific naming pattern:

//SYSIN DD *
  CDBDISP UTIL(%PETRO55) RUNNING

The ACTIVE and RUNNING keywords are identical in meaning.

## 10.3.8 Show 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.
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 *
CDBDISP UTIL(*)BATCH*) QE
```

### 10.3.9 Show 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 *
CDBDISP 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 *
CDBDISP UTIL(***RGG%) QEFAILED
```

### 10.3.10 Show 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 *
CDBDISP 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 *
CDBDISP UTIL(HVY%) FORCEID
```

### 10.3.11 Show 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 *
CDBDISP AGE(7)
```

To check utilities that are at least a two weeks old, or 14 days old, you would type:

```
//SYSIN DD *
```
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:

```bash
//SYSIN DD *
  CDBDISP UTIL(STM%) AGE(3)
  CDBDISP UTIL(EEH%) AGE(5)
  CDBDISP UTIL(DELIV%) AGE(21)
```

10.3.12 **Show the Status of Utilities with Multiple Criteria**

CDBDISP 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.

```bash
//SYSIN DD *
  CDBDISP UTIL(R55%) QEFAILED AGE(7)
```

This second example displays all utility IDs that are currently running, restartable, or completed.

```bash
//SYSIN DD *
  CDBDISP RUNNING RESTARTABLE COMPLETED
```

This same example with a naming pattern would look like this:

```bash
//SYSIN DD *
  CDBDISP UTIL(EASTC%) RUNNING RESTARTABLE COMPLETED
```
10.4 Relevant Automation

NGT automation control points are integrated into all NGTUtilities and can greatly enhance and expand your processing options – you are limited only by your imagination. In this chapter we present automation exits we recommend incorporating into your processing. (Automation control points are not required, however, for you to use this product.)

The presence of the CDBEXEC DD statement triggers automation control point processing, as in this example:

```
//CDBEXEC DD DISP=SHR,DSN=NGT.AUTO.CTRL.PTS
```

The automation control point dataset 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:

- **XSUTDISP** Can be used to create QUICKEXIT and DELETE statements. This automation control point is detailed below.
- **XSUTGLOB** Can be used to set global variables that are then used by other automation points.
- **XSUT0000** Called before any processing starts. Can be used to abort a run before it starts.
- **XSUTSYIN** Called before the parser reads SYSIN. It has full access to the SYSIN and can modify it. It may be used to override or disallow some input parameters.
- **XSUTTERM** Called after processing has finished. It may be used, for example, to insert run statistics into a DB2 table or to e-mail an audit report at the completion of processing, automatically.
10.4.1 XSUTDISP
CDBDISP Report Automation Control Point

A sample XSUTDISP is supplied in the NGT CNTL library. This XSUTDISP is required to use
CDBDISP to generate QUICEXIT or DELETE jobs.

Profile

When Called When the keywords GENQE or GENDELETEs are encountered in
the SYSIN for any utility job.

Purpose Use to trigger QUICKEXITS, DELETEs, or other actions in processing.

Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Contains</th>
<th>Possible Values</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Code to trigger either the generation of QUICKEXITS or DELETEs.</td>
<td>GEND, GENQ</td>
<td></td>
</tr>
<tr>
<td>uid</td>
<td>Utility ID.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jobname</td>
<td>Utility job name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stepname</td>
<td>Utility step name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jobid</td>
<td>Utility job ID.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>utility_rc</td>
<td>Utility return code.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>Status value.</td>
<td>RESTARTABLE, QUICKEXIT, JOB RUNNING, COMPLETED OK, FORCEID RUN, QEXIT ERROR, ERROR, NO RESTART</td>
<td></td>
</tr>
<tr>
<td>work_prefix</td>
<td>Utility work dataset prefix.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>start_date</td>
<td>Utility start date.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>start_time</td>
<td>Utility start time.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end_date</td>
<td>Utility end date.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end_time</td>
<td>Utility end time.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE: All of the variables above are issued in stems. For details on stem
variables, see the IBM TSO/E REXX Reference.

For more details on the use of automation in your processing, please see the BMC Next
CDB1DBUT

Program CDB1DBUT is used to copy the NGT VSAM work files .OUT and .INFO to sequential datasets. This sequential dataset can then be viewed or sent to BMC Customer Support for further analysis.

There are two SYSIN statements supported, one to dump both the .INFO and .OUT work files, and one to EXTRACT either or both separately.

Syntax

Syntax diagram for the DBUT DUMP utility command.

```
>> 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.
- `outDD` Optional. A 1- to 8-byte DD name in the JCL. The default ddname is OUTFILE.

Syntax diagram for the DBUT EXTRACT utility command.

```
>> DBUT EXTRACT -FILE - ( INFO
                       
                       JRNL
                       )
            
            INDD (INFILE)
            
            INDD(inDD)
            
            OUTDD (OUTFILE)
            
            OUTDD(outDD)
```

jobname Required. A 1- to 8-byte job name.
jesID Required. An 8-byte JES ID.
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 dataset name. The default ddname is INFILE.

OUTDD
Optional. A 1- to 8-byte DD name in the JCL. The default ddname is OUTFILE.

Usage Notes
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 multi-step 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 datasets, 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.

Sample JCL

```bash
//CDB1DBUT JOB . . .
//JOBLIB DD DISP=SHR,DSN=your.CDB.loadlib
//STEP01 EXEC PGM=CDB1DBUT,PARM='DSNA,,NORESTART'
//SYSPRINT DD SYSPRINT=* 
//OUTFILE DD DISP=(,CATLG),DSN=your.journal.datasetname,
//          UNIT=SYSALLDA,SPACE=(CYL,(5,50),RLSE)
//SYSSYN DD *
//        DBUT DUMP JOBNAME(jobname) JESID(JOB31569)
//

//CDB1DBUT JOB . . .
//JOBLIB DD DISP=SHR,DSN=your.CDB.loadlib
//CDB1DBUT EXEC PGM=CDB1DBUT,PARM='DSNA,,NORESTART'
//CDBPRINT DD SYSOUT=* 
//INFILE DD DISP=SHR,DSN=workprefix.MMBNLNJG.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)
//```
Index

"A/B" name switching, 5-21
"I/J" name switching, 5-21
"Must Complete" state, 2-7

&
&MJ, 3-4
&MJC, 3-4
&MU, 3-4
&N, 3-4
&SPEC, 3-4

/
//AGEREPRT, 6-1

- _Ispace( ), 1-37

+
+AGEREPORT, 6-1
+AGELIMIT, 5-9
+ALLO, 1-34
+CHECKDEFERRED, 5-10
+CLEANUP, 5-11
+CONTROLPREFIX, 5-12, 7-1
+COPYPENDING, 5-13
+DEADLINE, 5-14
+DELETEOUTPUT, 5-15
+DELETEWORK, 5-16
+DIGITS, 5-17
+DSCRL, 5-19
+EDPROCSUP, 5-20
+FASTSWITCH, 5-21
+FLASHCOPY, 5-22
+HRECALL, 5-23
+LARGEOBJSIZE, 5-26
+MAXCONNECT, 5-27
+MAXIDLE, 5-28
+MAXMEMORY, 5-29, 5-32
+MAXSERVERS, 5-30, 5-56
+MAXWAIT, 5-31
+MEMORYLIMIT, 5-32
+MONITORRATE, 5-33
+ONERROR, 5-34
+OUTPUTAGE, 5-9, 5-35
+PARMLIST, 5-36
+PLAN, 5-37
+PSWDWTO, 5-38
+QRETRY, 5-39
+RACFID, 5-40
+RERUN, 5-41
+REXXBUFR, 5-42
+REXXTIME, 5-43
+SQLID, 5-44
+SVRDISK, 3-1, 5-45
+SVRDISKJC, 5-46
+SVRJCLDISK, 5-47
+SVRJCLJOB, 5-48
+SVRJCLTAPE, 5-49
+SVRMODE, 5-50
+SVRPROCID, 5-52
+SVRPROCMMSGCLASS, 5-53
+SVRPROCMMSGLEVEL, 5-54
+SVRPROCT, 5-55
+SVRTAPE, 5-58
+SVRTAPEJC, 5-59
+SVRTYPE, 5-60
+SYSINERROR, 5-61
+SYSINORDER, 5-62
+TIMEEXEC, 5-63
+UIDHISTORY, 5-64
+VOLUMES, 1-35
+WAITTIMEOUT, 5-65
+WORKMODE, 5-66, 7-6
+WORKPREFIX, 2-4, 5-67, 7-1
+WORKEUNIT, 5-69
+RERUN, 5-41
+REXXBUFR, 5-42
+REXXTIME, 5-43
+SQLID, 5-44
+SVRDISK, 3-1, 5-45
+SVRDISKJC, 5-46
+SVRJCLDISK, 5-47
+SVRJCLJOB, 5-48
+SVRJCLTAPE, 5-49
+SVRMODE, 5-50
+SVRPROCID, 5-52
+SVRPROCMMSGCLASS, 5-53
+SVRPROCMMSGLEVEL, 5-54
+SVRPROCT, 5-55
+SVRTAPE, 5-58
+SVRTAPEJC, 5-59
+SVRTYPE, 5-60
+SYSINERROR, 5-61
+SYSINORDER, 5-62
+TIMEEXEC, 5-63
+UIDHISTORY, 5-64
+VOLUMES, 1-35
+WATTIMEOUT, 5-65
+WORKMODE, 5-66, 7-6
+WORPKPREFIX, 2-4, 5-67, 7-1
+WORKEUNIT, 5-69

0

0C3, 8-2

A

ABEND, 2-7, 5-34
abends, 8-2
Action Taken In Case Of Abend Or Error, 5-34
actions, 9-9
ACTIVE, 10-4, 10-11
active step, 10-8
AGE, 10-4, 10-12
AGELIMIT, 7-2
AGEREPORT, 2-6
ALL, 1-10
ALLO, 1-35
Allocation Strategies, 1-34
ALTER, 1-38
AMS Define, 1-39
APPGENR8, 1-10
applname, 1-5, 1-9, 1-10
ARCHIVE, 1-23
automation exit, 7-4
automation exit processing, 10-14
Index

B
backing up, 8-2
backup, 8-2
before access, 5-63
BFACC, 5-63
BLOB, 1-27
BYDATABASE, 1-5

C
CASE statement syntax, 1-22
CDB, 5-17
CDB Interactive, 9-1
CDB1DBMG, 8-1
CDB1DBUT, 1
CDBAMS DEFINE, 1-39
CDBAMS DELETE, 1-39
CDBAMS DISPLAY, 1-39
CDBAPPL, 1-2, 2
CDBBPG, 2-9
CDBPRINT, 2-5
CDBR858, 5-14
CDBRTS, 2-9
CDBRTS DEFINE, 5-44
CDBSS Connections, 5-27
CDBTAPE command, 1-13
CDBU056, 1-38
CDBUTDSP, 9-1
CDBWORK command, 7-3
CDBWORK Syntax, 7-4
Change Apply phase, 5-14
CHAR( ), 5-17
checkpoint, 8-2
Checkpoint Reorganization, 8-1
CLOB, 1-27, 1-28
command line prompt, 9-9
Comment in SYSIN, 2-8
Comparing constants, 1-26
COMPLETED, 10-4, 10-10
CONTINUE, 5-23, 5-34
Copy Pending Flag Control, 5-13
CPU seconds, 5-43
creator ID, 1-1
Criteria Table, 9-13

D
DASD volumes, 7-5
data columns, 9-4
DATACLAS, 7-5
dataset name prefix, 5-67
Dataset Name Prefix For Work Datasets, 5-67

DATASET QUALIFIER, 10-8
DATASIZE, 4-3, 4-5
date and time, 10-7
Date durations, 1-27
DAYOFMONTH function, 1-27
DAYOFWEEK function, 1-28
DAYOFYEAR function, 1-28
DB2 Compatibility, 1-34
DB2 I-J Name Switching Control, 5-21
DB2 operand, 1-35
DB2 Plan Name, 5-37
DB2 subsystem ID, 9-2
DB2 threads, 5-27
DB2-managed, 1-34
DBCLOB, 1-27
DBMG CHECKIX, 8-1
DBMG COMPACT, 8-1
DBMG OPTIMIZE, 8-1
DBMG REORG, 8-1
DBMG REPORT, 8-1
DD Name For Disk Server Job JCL, 5-47
DD Name For Tape Server Job JCL, 5-49
DD statements, 10-2
ddname, 5-47, 5-49
DDNAME, 1-14
DDNCOUNT, 1-14
Deadline Processing, 5-63
DECIMAL, 1-25, 1-31, 5-17
DECIMAL comparisons, 1-25
default parameters, 5-1
Default SQLID Control, 5-44
DEFINE, 1-5, 7-4
DEFINE command, 1-3
DEFINE datasets, 1-34
DEFINE keyword, 7-4
DEFINE Retry, 1-35
DEFINE statement, 1-37
Defining Server JCL, 3-1
Defining Servers, 3-1
DELETE, 1-38, 7-4, 10-1, 10-5, 10-14
DELETE command, 1-7
DELETEOUTPUT, 7-2
DELETEWORK, 7-2
details of a utility, 9-5
DFDSS, 8-2
DFSORT, 3-5
Diagram, 10-3
DIGITS Function, 1-29
DIGITS( ), 5-17
disk (or DASD) servers, 5-45
DISPLAY, 7-4
Display AMS, 9-15
DISPLAY command, 1-10
Display RTS, 9-13
Display SPACE Command Retry Limits, 5-19
double forward slash, 1-21
DRAIN(ALL), 5-14
<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAIN1, 5-18</td>
</tr>
<tr>
<td>DSERVJCJL, 5-47, 5-49</td>
</tr>
<tr>
<td>Dynamic Allocation, 1-37</td>
</tr>
</tbody>
</table>

**E**

| EDITPLIB, 2-6 |
| ENDED, 10-7 |
| environment, 9-2 |
| Environment, 9-9 |
| Environment Selection, 9-2 |
| ERROR, 10-4, 10-10 |
| ESCAPE clause, 1-22 |
| EURO Conversion Function, 1-30 |
| examples, 10-10 |
| examples combining more than two selection criteria, 10-13 |
| Exceptions Table, 9-13 |
| EXCLUDE, 1-9, 1-10 |
| EXCLUDES, 1-11 |
| EXEC, 3-5 |
| EXEC Statement, 2-2 |
| EXPDT, 1-14 |
| expression syntax, 1-22 |
| EXTERNAL, 5-50 |

**F**

| Fast Switch, 5-21 |
| FDR, 8-2 |
| Filter Results, 9-9 |
| FLOAT, 1-25, 1-31 |
| FORCE, 7-5 |
| forced to exit, 10-12 |
| FORCEID, 2-4, 8-3, 10-4, 10-12 |

**G**

| GEND, 10-15 |
| GENDELETES, 10-5, 10-15 |
| GENERATE, 1-10 |
| GENQ, 10-15 |
| GENQE, 10-4, 10-15 |
| GLOBAL, 1-39 |
| Global parameters, 5-3 |
| Global Utility Alphabetical Parameter Reference, 5-9 |

**H**

| home screen, 9-4, 9-11, 9-12 |
| HONOR, 5-62 |

**I**

| IBM, 5-17 |
| ICBackupSpec, 1-14 |
| IDCAMS, 1-37 |
| IN Predicate, 1-31 |
| INCLUDE, 1-9, 1-10 |
| index on expression, 1-21 |
| in-flight, 8-2 |
| INFO, 7-1 |
| INFO and OUTPUT File Allocation, 5-12 |
| INFO_PRIMARY, 7-5 |
| INFO_SECONDARY, 7-5 |
| input, 10-2 |
| INTEGER, 1-31, 5-17 |
| INTERNAL, 5-50 |
| IS, 1-22 |
| IS NOT, 1-22 |
| ISPF, 9-1 |
| ISPF Command Shell, 9-1 |

**J**

| JCL Overview, 2-1 |
| JCL procedures, 2-1 |
| JCLRXMDL, 1-37 |
| JOB #, 10-7 |
| Job Card JCL DD Name, 5-48 |
| Job Completion Codes, 2-7 |
| job number, 10-7 |
| JOB Statement, 2-2 |
| JOBNAME, 10-7 |
| journal, 7-5 |
| Journal, 1-2 |
| JOURNAL, 5-24 |
| JT, 3-4 |

**K**

| KEEPDB2, 1-35 |
| Keywords, 10-3 |

**L**

| labeled duration syntax, 1-23 |
| Labeled durations, 1-26 |
| level-consistent, 1-5 |
| Levels of Control, 1-34 |
| LIKE, 1-1 |
| LIKE clause, 1-22 |
| LIKE Predicate, 1-29 |
| line commands, 9-5 |
| Literals, 3-3 |
| LOGPDS, 2-6, 8-2 |

**M**

| MANAGEMENTCLASS, 1-36 |
| master address space., 10-1 |
| master job, 9-4 |
| MASTER JOB, 9-4 |
| Master Job Information, 9-5 |
### Index

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix, 9-9</td>
<td>maximum number of connections to the NGT Subsystem, 5-27</td>
</tr>
<tr>
<td>Maximum Objects To Allow To Wait, 5-30</td>
<td>maximum size, 7-5</td>
</tr>
<tr>
<td>MAXSIZE, 7-5</td>
<td>MAXTAPESERVERS, 1-14</td>
</tr>
<tr>
<td>MAXVOLS, 1-14</td>
<td>Memory Limit, 5-32</td>
</tr>
<tr>
<td>MGMTCLAS, 7-5</td>
<td>migrated VSAM datasets, 5-23</td>
</tr>
<tr>
<td>mini-dump, 8-2</td>
<td>MIXED, 5-50</td>
</tr>
<tr>
<td>MMSSTH, 5-19, 5-39</td>
<td>MOD function, 1-21</td>
</tr>
<tr>
<td>Model AMS DEFINE, 1-36</td>
<td>multiple criteria, 10-13</td>
</tr>
<tr>
<td>multitasking, 1-12</td>
<td>must complete, 10-4</td>
</tr>
<tr>
<td>MUSTCOMPLETE, 10-4</td>
<td>MVS high-level qualifier, 5-12</td>
</tr>
<tr>
<td>MXIG, 1-35</td>
<td>NNN, 5-19, 5-39</td>
</tr>
<tr>
<td>NAME, 7-4</td>
<td>non-zero return code, 10-4</td>
</tr>
<tr>
<td>naming pattern, 10-4, 10-10</td>
<td>NOREPORT, 10-4</td>
</tr>
<tr>
<td>NGT AMS Define, 8-3</td>
<td>NORESTART, 2-3, 10-4</td>
</tr>
<tr>
<td>NGT Automation, 10-14</td>
<td>NOT Operator, 1-31</td>
</tr>
<tr>
<td>NGT Reorg Change Apply Control, 5-14</td>
<td>NULL/NOT NULL predicate, 1-29</td>
</tr>
<tr>
<td>NGT Subsystem, 5-27</td>
<td>NUMBER, 7-4</td>
</tr>
<tr>
<td>NNN, 5-19, 5-39</td>
<td>Number Formatting Control, 5-17</td>
</tr>
<tr>
<td>non-zero return code, 10-4</td>
<td>Number Of Disk Units, 5-45</td>
</tr>
<tr>
<td>NOREPORT, 10-4</td>
<td>number of pages, 5-26</td>
</tr>
<tr>
<td>NORESTART, 2-3, 10-4</td>
<td>number of retries, 5-39</td>
</tr>
<tr>
<td>NOT Operator, 1-31</td>
<td>Number Of Servers, 3-1</td>
</tr>
<tr>
<td>NULL/NOT NULL predicate, 1-29</td>
<td>Number Of Tape Units, 5-58</td>
</tr>
<tr>
<td>NUMBER, 7-4</td>
<td>Numeric comparisons, 1-25</td>
</tr>
<tr>
<td>number of pages, 5-26</td>
<td>Numeric String Comparisons, 1-26</td>
</tr>
</tbody>
</table>

### Out_Secondary, 7-5

- Output Age, 5-9
- Output Deletion Control, 5-14
- Output Listings, 9-5
- OUTPUTAGE, 7-2

### P

- parallelism, 1-12
- Parameter List Print Control, 5-36
- Parameters, 1-34, 5-1
- parameters for a particular utility, 2-5
- Parameters Grouped By Function, 5-6

### QuickExit Command, 9-5

- QUIESCE Retry Limits, 5-39

### R

- RANDOM, 5-62
- RC, 9-4
- Refresh, 9-9
- REGION, 2-2
- RELEASE, 7-4
- REPRO, 8-2
- reserved words, 1-31
- RESPECIFY command, 1-8
- RESTART, 2-2, 10-4
- RESTARTABLE, 10-4, 10-11
- restoring datasets, 8-2
- RET CODE, 10-8
- RETPD, 1-14
- return code, 10-4, 10-8, 10-10
- Return Codes, 2-7
- REUSE, 1-35
- RGTS, 7-1
- RGTS work datasets, 7-5
- RGTS_PRIMARY, 7-5
- RGTS_SECONDARY, 7-5
- RPT, 10-4

### X-6
Index

RTMAN tables, 5-44
RTS tables, 8-2
RTS Tables, 8-3
RUNNING, 10-4, 10-11
RUNSTATS, 2-5, 3-6

S
sample JCL skeleton, 3-5
Schedule Table, 9-14
Searched WHEN clause, 1-22
Secondary allocation, 7-5
Secondary quantity, 1-34
secqty, 1-34
SEL, 9-4
Selection, 9-4
Server Job Information, 9-5
Server Types, 3-1
Servers, 1-12
SET CURRENT SQLID, 5-44
set of work datasets, 7-4
Setting Default Parameters, 5-1
SHR, 5-18
Simple WHEN clause, 1-23
SMALLINT, 5-17
SMS, 1-35
SMS data class, 7-5
SMS management class, 7-5
SMS storage class, 7-5
SORT, 2-2
SOUNDEX Function, 1-29
Space-Specific Defines, 1-37
Special Server, 5-56
specified number of days, 10-4
SQL in SYSIN, 2-8
SQTY, 1-34
SSID, 2-2, 9-2, 9-4
-START, 5-19
start time, 9-4
START TIME, 9-4
STARTED, 10-7
STATUS, 9-4
status of a single utility ID (UID), 10-10
status of all active utilities, 10-11
status of all utilities, 10-10
status of only completed utilities, 10-10
status of only restartable utilities, 10-11

stem variables, 10-15
STEPLIB, 2-1, 2-4, 3-5
STEPNAME, 10-8
STOGROUP, 1-34
STOGROUPs, 1-37
STOGROUPS, 1-35
STOP, 5-34
-STOP, 5-19
STORCLAS, 1-15, 7-5
Subselects, 1-34

SUBSTR Function, 1-30
subtasks, 1-12
SUMMARY, 1-1, 2-5, 3-6, 10-1, 10-2, 10-4
Supervisor State, 5-20
SVC 8, 2-6
SVRDISK, 5-47
SVRJOBCD, 5-48
SVRSPECIAL, 5-56
SVRTAPE, 5-49
Syntax, 10-2, 10-3
Syntax Rules, 5-1
SYSERROR, 2-6, 2-7, 3-5
SYSIN, 2-4, 2-8, 5-63, 10-2, 10-15
SYSIN Comments, 2-8
SYSIN Order Processing, 5-62
SYSIN SQL, 2-8
SYSIN statements, 5-62
SYSIN Statements, 9-5
SYSIN USRCMD, 2-8
SYSOUT, 3-5
SYSPDS, 1-39
SYSPRINT, 1-1, 2-4, 3-5
SYSTRACE, 2-6, 3-6
SYSTSPRT, 2-5, 3-6

T
TAPE_LP_DDNCOUNT, 1-15
TAPE_LP_EXPRDT, 1-16
TAPE_LP_MAXVOLS, 1-15
TAPE_LP_RETPD, 1-15
TAPE_LP_STORCLAS, 1-16
TAPE_LP_UNITNAME, 1-15
TEMPORARY, 1-6, 1-10
temporary work datasets, 7-3
TESTUID, 2-3
THEREST, 1-5
Threshold, 5-26
TIME durations, 1-27
Time Limit For Automation Exits, 5-43
TIMESTAMP durations, 1-27
Trace Function, 1-33

U
UID, 2-2, 9-4
UNIT, 7-5
UNITNAME, 1-14
UNLOAD, 1-23
unmatched utility IDs, 10-4
UNMATCHEDUIDS, 10-4
UPDATE command, 1-9
updating the DB2 Catalog, 1-34
Updating The DB2 Catalog, 1-37
USED, 1-35
USER1, 1-15
USRCMD in SYSIN, 2-8
Index

User-Defined RACF ID, 5-40
user-managed, 1-34
UTIL, 10-4, 10-10, 10-11, 10-12, 10-13
Utility Age Report, 6-1
utility ID, 9-4, 10-4, 10-7
Utility ID History Control, 5-64
UTILITY STATUS, 10-7
UTILITY-ID, 10-7
UTLPARMS, 2-5, 5-3
UTPRINT, 3-6

V
VALUE Function, 1-29
value syntax, 1-23
variable buffer size, 5-42
VOLUMES, 7-5
VSAM allocation values, 1-34
VSAM Dataset Migration Control, 5-23
VSAM datasets, 8-2
VSAM output file, 5-35
VSAM Output File Retention, 5-35
VSAM work dataset, 10-8

W
WEEK function, 1-28
WHERE clause, 1-1, 1-21, 1-33
WHERE Clause Differences, 1-20
Wildcard Level Consistency Table, 1-6
wildcards, 1-1, 10-4, 10-10
WK, 7-1
Work Dataset Deletion Control, 5-16
Work Datasets, 7-1
WORK_PRIMARY, 7-5
WORK_SECONDARY, 7-5
WORKMODE, 7-4

X
XSUT0000, 10-14
XSUTALLO, 1-36
XSUTDBMG, 7-4
XSUTDISP, 10-5, 10-14, 10-15
XSUTGLOB, 10-14
XSUTSYIN, 10-14
XSUTTERM, 10-14
XSUTTIME, 5-63