Contacting BMC Software

Several methods are available for contacting BMC Software.

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

**United States and Canada**

<table>
<thead>
<tr>
<th>Address</th>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC SOFTWARE INC</td>
<td>1 713 918 8800</td>
<td>1 713 918 8000</td>
</tr>
<tr>
<td>2101 CITYWEST BLVD</td>
<td>or</td>
<td></td>
</tr>
<tr>
<td>HOUSTON TX 77042-2827 USA</td>
<td>1 800 841 2031</td>
<td></td>
</tr>
</tbody>
</table>

**Outside United States and Canada**

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

© Copyright 2015 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 UNPUBLISHED—RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES.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](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](mailto:customer_support@bmc.com). (In the subject line, enter **SupID:**your**SupportContractID**, such as SupID:12345). Outside the United States and Canada, contact your local support center for assistance.

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

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

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

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

Chapter 1  Summary ........................................................................................................................................... 1-1
Chapter 2  Input: DD Statements, Syntax, Keywords, Parameters ................................................................. 2-1
  2.1 DD Statements ........................................................................................................................................ 2-1
  2.2 SYSIN Syntax Diagrams .......................................................................................................................... 2-3
    2.2.1 format option .................................................................................................................................. 2-4
    2.2.2 mode option .................................................................................................................................... 2-4
    2.2.3 select option ..................................................................................................................................... 2-4
    2.2.4 from table option ............................................................................................................................ 2-4
    2.2.5 order by option ................................................................................................................................ 2-4
    2.2.6 nullspec (Null specification) ........................................................................................................... 2-5
    2.2.7 fieldspec (Field specification) ......................................................................................................... 2-6
  2.3 Keywords ............................................................................................................................................... 2-7
  2.4 MODE option Keywords ....................................................................................................................... 2-13
    2.4.1 DB2SQL Restrictions ....................................................................................................................... 2-15
    2.4.2 General Limitations ......................................................................................................................... 2-15
    2.4.3 Data Compatibility Between MODE(DIRECT) And MODE(DB2SQL) ............................................ 2-16
    2.4.4 Page Encoding ............................................................................................................................... 2-16
  2.5 select option Keywords .......................................................................................................................... 2-18
  2.6 from table option Keywords .................................................................................................................. 2-18
  2.7 order by option Keywords ...................................................................................................................... 2-19
  2.8 Parameters.......................................................................................................................................... 2-20
    2.8.1 +BLANKPAD .................................................................................................................................... 2-20
    2.8.2 +CHARFLDDLML ........................................................................................................................... 2-21
    2.8.3 +CONNECTALL ............................................................................................................................... 2-22
    2.8.4 +DATEFMT ....................................................................................................................................... 2-23
    2.8.5 +DEGREE ....................................................................................................................................... 2-24
    2.8.6 +FIELDSEP ..................................................................................................................................... 2-26
    2.8.7 +MINROWS ..................................................................................................................................... 2-27
    2.8.8 +NUMFLDDLML ............................................................................................................................. 2-28
    2.8.9 +PASSWORD .................................................................................................................................... 2-29
    2.8.10 +TIMEFMT .................................................................................................................................... 2-30
Chapter 3  Output File Formats .................................................................................................................... 3-1
  3.1 Generated LOAD DATA Statements ..................................................................................................... 3-1
Table Of Contents

3.2 Unload Output File Formats .................................................................................. 3-1
3.2.1 DSNTIAUL Format ......................................................................................... 3-2
3.2.2 UNLOAD Format ............................................................................................. 3-4
3.2.3 ARCHIVE Format ........................................................................................... 3-6
3.2.4 DELIMITED Format ......................................................................................... 3-6
3.2.5 ASCIIDEL Format ............................................................................................ 3-7
3.2.6 REPORT Format ............................................................................................... 3-8
3.2.7 EXTERNAL Format ......................................................................................... 3-9

Chapter 4 Examples ...................................................................................................... 4-1
4.1 Unload using a table lookup .................................................................................. 4-1
4.2 Unload a specific number of rows ......................................................................... 4-1

Chapter 5 Relevant Automation .................................................................................... 5-1
5.1 General Utility Automation Control Points ............................................................ 5-1

Chapter 6 Technical Details ......................................................................................... 6-1
6.1 Sizing The Work Dataset (WK00001) .................................................................. 6-1
6.2 Selecting Data To Unload .................................................................................... 6-1
6.2.1 Selecting Columns ......................................................................................... 6-1
6.2.2 Selecting Rows ............................................................................................... 6-2
6.2.3 Partition Filtering ............................................................................................ 6-2
6.3 Unloading From Other Sources ............................................................................ 6-6
6.4 Unloading From Dropped Tables ......................................................................... 6-7
6.5 Saving Space with the DSNTIAUL Format ............................................................ 6-7
6.6 Customizing the Generated LOAD DATA Statement .......................................... 6-8
6.6.1 The STMTDDN Keyword ............................................................................... 6-8
6.7 Server use by Unload ........................................................................................... 6-8

Chapter 7 ROWEXIT Interface ..................................................................................... 7-1
7.1 Invoking The Process ............................................................................................ 7-2
7.2 Functional Description ........................................................................................ 7-2
7.2.1 Initialization Function Call ............................................................................. 7-3
7.2.2 Processing Function Call ............................................................................... 7-4
7.2.3 Termination Function Call .............................................................................. 7-6
7.2.4 Detailed Design ............................................................................................... 7-7
7.2.5 Parameter Structure ....................................................................................... 7-7
7.2.6 Assembler DSECT .......................................................................................... 7-8
7.2.7 Table Row Header .......................................................................................... 7-8
7.2.8 Row Descriptor Block ..................................................................................... 7-9

Index ............................................................................................................................ X-1
BMC New Generation Technology Unload (NGT Unload) is a utility for unloading database objects to sequential datasets.

<table>
<thead>
<tr>
<th>When To Use It</th>
<th>When you need to unload database objects to sequential files, such as for a test environment.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorizations Required</td>
<td>To run NGT Unload, you must use a privilege set that includes one of the following authorities:</td>
</tr>
<tr>
<td></td>
<td>- Ownership of the tables</td>
</tr>
<tr>
<td></td>
<td>- SELECT privilege on the tables</td>
</tr>
<tr>
<td></td>
<td>- DBADM authority for the database</td>
</tr>
<tr>
<td></td>
<td>- SYSADM authority</td>
</tr>
<tr>
<td></td>
<td>- SYSCTRL authority (catalog tables only)</td>
</tr>
<tr>
<td></td>
<td>- COPY authority for the database</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Restrictions</th>
<th>Hashed Tables. NGT Unload does not process hashed tables.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cloned tables. NGT Unload does not process tables with clones.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statuses</th>
<th>DB2 States</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Table space: RW, non-restricted states only</td>
</tr>
</tbody>
</table>
Chapter 2

Input: DD Statements, Syntax, Keywords, Parameters

This chapter covers the DD statements, SYSIN input (statement syntax), keywords and parameters for this utility.

2.1 DD Statements

SYSIN
The only required DD statement for use with this utility is SYSIN. 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.

ULDPARMS
ULDPARMS is only required if you wish to override the program's parameters for NGT Unload or NGT Unload specified at installation time. They may be specified in a dataset or in the JCL, as in:

//ULDPARMS DD *

or

//ULDPARMS DD DISP=SHR,DSN=NGT.UNLOAD.PARMS

UTLPARMS
UTLPARMS is only required if you wish to override the BMC Next Generation Utilities (NGT Utilities) global parameters specified at installation time. They may be specified in a dataset or in the JCL, as in:

//UTLPARMS DD *

or

//UTLPARMS DD DISP=SHR,DSN=NGT.UTIL.PARMS

For more information on global parameters, please see the BMC Next Generation Technology General User Guide.
SYSREC
SYSREC names the input file. This DD name can be overridden via the OUTDDN keyword on the LOAD DATA statement. The default is SYSREC00. The data in this file must match the format described by the LOAD DATA statement.

```bash
//SYSREC00 DD DISP=SHR,DSN=NEW.LOAD.INPUT.FILE
```

SYSPUNCH
SYSPUNCH is the dataset where the corresponding LOAD DATA statement is written. The statement may be written either to a sequential dataset, or to a SYSOUT class, as in:

```bash
//SYSPUNCH DD SYSOUT=* or 
//SYSPUNCH DD DISP=SHR,DSN=NGT.OUTPUT.LOAD.STMT
```

CDBEXEC
If you wish to use this utility with automation exits, 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:

```bash
//CDBEXEC DD DISP=SHR,DSN=NGT.AUTOMATN.EXITS
```

For more information on automation exits, please see the BMC Next Generation Technology Automation Reference Guide.
NOTE

NGT Unload does not support the following data types: VARBINARY, XML.
2.2.1 format option

```
FORMAT
  | DSN | IBM
  | NULLCHAR | nullspec
  | EXTERNAL | nullspec
  | REPORT | REP
  | UNLOAD | ULD
  | ARCHIVE | ARC
  | DELIMITED | COLDEL ',' | CHARDEL '"' | DECPT '.'
  | ASCIIDEL | ASC
```

2.2.2 mode option

```
MODE(
  | DIRECT
  | AUTO
  | DB2SQL
  | MRF(
    | integer
  | )
)
```

2.2.3 select option

```
SELECT
  * 
  , 
  columnname AS expression
  , 
fieldname
```

2.2.4 from table option

```
FROM
  | TABLE
  | tbcreator.
tbname
  | dbname
  | synonym
  | alias
```

2.2.5 order by option
2.2.6 **nullspec (Null specification)**

The Null specification syntax applies to the FORMAT specification for the DSNTIAUL and EXTERNAL formats.
When a nullable field is downloaded from the table a marker is placed in the unloaded data either trailing the field, leading it, or in the front of the unloaded record. The marker may have from one to four characters.

UPFRONT: All nullable columns’ markers in the entire record are placed at the beginning of the record in an array format. For example, assuming there are four nullable columns in the record; the 1st and 3rd are nulls but 2nd and 4th are not, also assume that the null marker is ‘?’. Therefore at the beginning of the record there will be four chars corresponding to the four nullable columns and will have the values in hex x’6F006F00’. X’6F’ is ‘?’. This is used when converting from “CA” unload. The generated load statement will have a NULLIF clause with correct position values in the null markers array.

The NULL specification must be coded immediately after the keywords "FORMAT DSN", "FORMAT IBM", "FORMAT EXT", "FORMAT EXTERNAL" or "FORMAT DSNTIAUL".

The default for ‘string’ is a question mark (“?”). The default position is TRAILING.

The content of the field when its value is null is binary zeroes. Conversely, when the nullable field is not null, the null marker is set to binary zeroes.

### 2.2.7 fieldspec (Field specification)

The Field Specification applies to the ASCIIDEL format. It is optional, but if coded it must follow the FORMAT ASCIIDEL keywords.

- **FIELDSEP**
  This parameter overrides the ULDPARMS +FIELDSEP parameter for an individual UNLD statement. The specification allows any single character.

- **CHARFLDDLM**
  This parameter overrides the ULDPARMS +CHARFLDDLM parameter for an individual UNLD statement. The specification allows a single character to the left of the field and a single character to the right of the field. If the AND is omitted then the same character is used for left and right.

- **NULLSTRING**
  Sets a value for a nullable field that is NULL. The maximum length of the null string is 4 and it defaults to none, e.g. a string of length 0.
2.3 **Keywords**

**DATA**
Optional keyword.

**OBID**
This keyword allows you to have NGT Unload look for a different table OBID value when selecting table rows. Without this parameter, NGT Unload assumes the input data has the same OBID as the one defined in SYSIBM.SYSTABLES for the table named in the FROM TABLE clause. If this is not what you want, use the OBID keyword to specify a different one. The OBID value supplied is not validated. Supply the value in integer format, not hexadecimal format. The parentheses are optional.

**NOTE**
If you use a value of 0, all OBIDs in the input records will be ignored. The value 0 is intended for use with single-table tablespaces only. Using a value of 0 with a multi-table tablespaces may produce unpredictable results or result in abends.

**SHRLEVEL CHANGE**
This keyword pair, if specified, will create an unload file while concurrent changes are taking place. However the unload will not be a consistent point in time (PIT). A NGT Unload without specifying SHRLEVEL CHANGE creates a consistent PIT online unload with concurrent changes. Specify SHRLEVEL CHANGE only when obtaining a DRAIN(WTR) for the object is an issue.

**BLANKPAD**
This keyword causes data in varchar columns to be padded with blanks rather than low values. There is also a +BLANKPAD Unload parameter for the same purpose. This keyword will override that value.

See the PAD NGT Unload keyword for controlling whether varchar fields are padded or not.

**ZEROPAD**
This keyword causes data in integer, small integer and decimal columns to be formatted with a preceding "+" or "+" sign, and left-padded with zeroes for all blank positions. This option applies only to the EXTERNAL and REPORT output formats.

**OUTDDN**
This keyword is optional. It names the DD statement that contains the unloaded data. If not specified and dynamic allocation is not used, NGT Unload will attempt to open and write to the default DD name of SYSREC00. The ddname specified must be allocated unless automation exit XULDDYNM is used for dynamic allocation.

**NOTE**
If the OUTDDN keyword is specified with a DD and dataset (e.g., SYSREC DD points to a named sequential dataset) then the unload will be performed in the master job rather than in any server job.
FORMAT
This keyword, if specified, identifies to NGT Unload the desired format of the output file.
The valid formats are listed below along with valid abbreviations.

<table>
<thead>
<tr>
<th>Format</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSN/IBM</td>
<td>DSNTIAUL</td>
</tr>
<tr>
<td>EXT</td>
<td>EXTERNAL</td>
</tr>
<tr>
<td>UNL</td>
<td>UNLOAD</td>
</tr>
<tr>
<td>ARC</td>
<td>ARCHIVE</td>
</tr>
<tr>
<td>DEL</td>
<td>DELIMITED</td>
</tr>
<tr>
<td>ASC</td>
<td>ASCIIDEL</td>
</tr>
<tr>
<td>REP</td>
<td>REPORT</td>
</tr>
</tbody>
</table>

STMTDDN
This option names the DD statement to which the generated LOAD statement will be written. If the ddname represents a sequential dataset, then records will be added to the end of the dataset. If it represents a member of a PDS and the member exists, it will be overwritten. The LOAD statement is not generated for FORMAT ASCIIDEL.

NOTE
STMTDDN will be ignored if FORMAT UNLOAD is specified.

LOADSTMT
Specifies a LOADSTMT to customize the LOAD statement generated by NGT Unload. This is a character string that may not exceed 77 characters in length. However, multiple strings (delimited by single quotation marks) may be used for groups of operands longer than 77 characters (see example below).

All operands will be inserted, without any changes, between the LOAD DATA statement and the INTO TABLE statement.

Multiple keywords may be coded in this statement as in, for example:

```
LOADSTMT 'UNICODE CCSID(00367,01208,01200) ENFORCE NOCOPYDDN(ICOPYD)' 
  'STATISTICS TABLE(ALL) INDEX(ALL) UPDATE ALL'
```

Note that one set of operands begins with UNICODE and a second set begins with STATISTICS. In both cases, each group of operands is delimited with single quotation marks.

All of the keywords shown in the above example would produce a SYSPUNCH that would look like the following:

```
LOAD DATA INDDN(SYSREC00) 
UNICODE CCSID(00367,01208,01200) ENFORCE NOCOPYDDN(ICOPYD) 
STATISTICS TABLE(ALL) INDEX(ALL) UPDATE ALL 
INTO TABLE "dbname"."some_table_name" 
   ( 
     "COL01" POSITION(1:11) INTEGER EXTERNAL(11) 
   ,"COL02" POSITION(12:20) CHAR(9) 
   )
```
NOTE: Through the use of the LOADSTMT keyword, the LOAD DATA statement can now be produced for all formats except FORMAT ASCIIDEL. In addition, the customized load statement records can be specified as one or more data strings following the LOADSTMT keyword in the NGT UNLD command.

Notice that to group multiple statements, you DO NOT code another LOADSTMT keyword. This would cause a syntax failure.

COPYDSN
Specifies the name of an image copy dataset of a table space containing table data. The dataset name cannot be a PDS or member of a PDS. Relative GDG members can be used in the dataset name as in (0) or (-1). The dataset can be on tape or disk but must be cataloged. Its organization must either be sequential or VSAM linear dataset. During the unload process the dataset will be dynamically allocated with a DISP=SHR if sequential or DISP=OLD if VSAM.

NOTE: The PARTS option is not supported when using the COPYDSN option.

COPYDDN
Specifies the name of a DD statement that points to a dataset that contains table data to be unloaded. The rules regarding dataset organization are the same as with COPYDSN. However, with COPYDDN you can concatenate datasets having the same organization.

VSAM datasets can be concatenated and will be dynamically re-allocated with DISP=OLD unless VOL=PRIVATE is coded on the DD statement.

NOTE: The PARTS option is not supported when using the COPYDDN option.

SYSCOPY
If unloading from an image copy, this option instructs UNLD to obtain the image copy dataset name from SYSIBM.SYSCOPY. If a single part number is coded, then the search is limited to the coded part number. Otherwise the search is done with DSNUM=0. Three sub-options are provided:

FULL Obtain the full image copy dataset having the highest START_RBA.
INCR Obtain the incremental image copy dataset having the highest START_RBA.
LAST Obtain the image copy dataset having the highest START_RBA. This dataset can be incremental or full.

NOTE: If SYSCOPY is specified, neither COPYDDN nor COPYDSN may be specified. If PART is used with the SYSCOPY keyword, NGT Unload will attempt to find a copy of the specified partition in SYSIBM.SYSCOPY.

MAXROWS
MAXROWS limits the number of rows unloaded from a table. When integer is reached NGT Unload terminates processing. If you specify a value in MAXROWS that is greater than the number of rows in the table, all table rows are unloaded. To unload all rows without limit specify MAXROWS 0 or omit the keyword.
The FILSZ value passed to SORT will come from the MAXROWS keyword value if it is used. Otherwise, a value of 100,000 will be used.

**APPEND**

Instructs UNLD to append unload data to the end of the UNLOAD dataset. You can achieve the same results by coding DISP=MOD on the UNLD DD statement.

**MAXERR**

Specifies the maximum number of records in error that are to be allowed. If this limit is reached the Unload will stop and issue message NGTN071 and/or NGTN072.

**MAXLRECL**

Specifies a number not exceeding 32756 for the maximum logical record length of the UNLD dataset. The default is based on the FORMAT and the table’s column structure. When this parameter is coded a RECFM of VB will be set for the datasets as long as it does not conflict with existing attributes. If you are planning to APPEND data to the UNLD dataset, then specify a number large enough to accommodate future unloads.

**DATEFMT**

This parameter overrides the ULDPARMS +DATEFMT parameter for this UNLD statement only. It allows for the generation of a variety of DATE formats other than the standard formats currently supported by DB2.

Specify a string of up to 40 characters as an argument for DATEFMT. The following character strings are replaced by NGT Unload in order to generate the actual date:

- YYYY is replaced by a 4-digit year.
- YY is replaced by a 2-digit year.
- MM is replaced by a 2-digit month.
- DD is replaced by a 2-digit day of the month.

All other characters are left unchanged.

Thus, if today is the 17 of July of the year 1998, the string “YYYY.MM.DD” would result in the date format “1998.07.17”

A maximum of a single instance of a year, day or month element can be coded in a DATEFMT string. For example, the following is not valid because the year is repeated: ‘YYYY/YY/MM/DD’

**TIMEFMT**

This parameter overrides the ULDPARMS +TIMEFMT parameter for this UNLD statement only. It also allows for the generation of a variety of TIME formats other than the standard formats supported by DB2.

Specify a string of up to 40 characters as an argument for TIMEFMT. The following character strings are replaced by NGT Unload in order to generate the actual time:

- HH is replaced by the current hour, either in a 12-hour clock on in a 24 hour clock (See XM below).
- MM is replaced by the minutes elapsed from the top of the hour.
- SS is replaced by the number of seconds elapsed from the start of the current minute.
- XM is replaced by either AM or PM depending on whether the current time is before noon or after. If XM is present, HH is replaced by an hour from a 12 hour clock (00-12), while if it is absent it is replaced by an hour from a 24 hour clock (00-24).
Element in a coded message can occur only once and the entire string must contain at least one TIME element.

For example, if the time now is 3:14 in the afternoon, the format string 'HH-MM (XM)' results in the following time: 3-14 (PM).

**TSFMT**

The TIMESTAMP format overrides the standard DB2 TIMESTAMP format with a variety of new formats. (Note that here is no counterpart in the ULDPARMS for this parameter; there is no "+TSFMT" parameter.)

Specify a string of up to 40 characters as an argument for TSFMT. The following character strings are replaced by NGT Unload in order to generate the actual timestamp:

- **YYYY** is replaced by a 4-digit year.
- **YY** is replaced by a 2-digit year.
- **MM** is replaced by a 2-digit month.
- **DD** is replaced by a 2-digit day of the month.
- **HH** is replaced by the current hour, either in a 12-hour clock or in a 24-hour clock (See XM below).
- **MI** is replaced by the minutes elapsed from the top of the hour.
- **SS** is replaced by the number of seconds elapsed from the start of the current minute.
- **XM** is replaced by either AM or PM depending on whether the current time is before noon or after. If XM is present, HH is replaced by an hour from a 12-hour clock (00-12), while if it is absent it is replaced by an hour from a 24-hour clock (00-24).
- **NN** is replaced by a two digit number representing the hundreds of a second.
- **NNNN** is replaced by a four digit number representing tenths of a millisecond.
- **NNNNNN** is replaced by a four digit number representing the microseconds.

**NOTE:**
A maximum of three NN pairs can occur in a timestamp, or a maximum of one NNNN and one NN can or a maximum of one NNNNNN. All other characters are left unchanged.

**PAD/NOPAD**

The PAD option instructs UNLD to PAD a VARCHAR column to its full length, while NOPAD unloads the column as is without any padding.

PAD cannot be specified with the ASCIIDEL or ARCHIVE formats. NOPAD cannot be specified with the REPORT format.

If neither PAD nor NOPAD is specified the default is as follows:

For the DSN TIAUL and EXTERNAL formats the default is PAD unless RECFM=V is coded on the OUTDDN DD statement.

For the UNLOAD and REPORT formats the default is PAD.

**ROWEXIT**

Specifies the name of a load module that is called every time a row is unloaded from the DB2 table. This load module may change the data of each row, delete it or perform other processing. For more information, please see Appendix A.

**DIGITS**

Enforces compatibility between certain DB2 SQL functions and NGT SQL functions.

**Encoding Format**
Input: DD Statements, Syntax, Keywords, Parameters

Used to specify how unloaded data code pages will be encoded. Takes one of the values CCSID, ASCII, UNICODE or EBCDIC. See Section 2.4.4, "Page Encoding" for details.

**NULLCHAR**
Identifies a NULL character string that can be a maximum length of 4 bytes. It also identifies the location of the NULL bytes (leading or trailing) in relation to the unloaded column data.

**MINROWS**
This keyword accepts a number greater than or equal to 0 and causes the unload issue a warning (RC=4) if the number of unloaded rows is less than the specified number. If the coded number is 0, then the global +MINROWS value coded in //ULDPARMS DD statement will take effect if any.

**BYPART**
Applies only when the target table to be unloaded is partitioned and dynamic allocation is in use. Otherwise parameter is ignored. When the PARTS keyword is not specified, the total number of parts in the TS will be unloaded individually to separate files.

BYPART is mutually exclusive (cannot be used in combination) with:

1. COPYDDN
2. COPYDSN
3. LOADINDD
4. SYSCOPY
5. MODE(AUTO)
6. MODE(DB2SQL)
7. PART

**Example**

```
UNLD BYPART PARTS 1,5-7,20 FROM TABLE PARTS
```

is similar to

```
UNLD PART 1 FROM TABLE PARTS
UNLD PART 5 FROM TABLE PARTS
UNLD PART 6 FROM TABLE PARTS
UNLD PART 7 FROM TABLE PARTS
UNLD PART 20 FROM TABLE PARTS
```

However if the original UNLD BYPART has a WHERE clause that causes certain parts to be filtered, then only the applicable parts will be individually unloaded.

**PARTS**
Specifies a list of physical partitions. All partitions listed will be unloaded. Partition lists consist in numbers representing a single partition number separated by commas or ranges representing a range of numbers separated by a colon. For example:

```
1,5,7:9,20:50
```

will unload partitions 1, 5, 8, 9 and all partitions between 20 and 50.

If a partition list includes a partition more than once an error occurs. For example, the partition list: 1,5,2:9 is invalid because partition 5 is specified twice, once explicitly and a second time as part of a range.
If you want to include blanks in a partition list then the entire partition must be enclosed in parenthesis.

**WHERE clause**
Specifies a SQL-compatible WHERE logical condition.

**NOTES:**
The PARTS keyword must not be specified if the COPYDDN, COPYDSN and SYSCOPY options are specified.

When using the PARTS keyword and the WHERE clause in the same UNLOAD statement, the selection of parts is subject to part analysis. The resulting unloaded data will be the intersection of the results from the specified PARTS parameter and the results from the WHERE clause. If the intersection results in no parts being selected, then the +MINROWS parameter specification will be analyzed. If the +MINROWS criteria is satisfied, the unload will continue. If the +MINROWS criteria is not met, a warning is issued and the unload job will end with a RC=4. For more information, please refer to the NGT Unload and NGT Unload Reference concerning the WHERE Clause.

### 2.4 MODE option Keywords
NGT Unload uses the MODE keyword to trigger the execution of the SELECT clause under DB2's Full-Function SQL.

The keyword **MODE** is placed after the UNLD DATA keywords, but before the SELECT clause. The MODE keyword receives one of three operands: DIRECT, DB2SQL and AUTO.

**NOTE:** The MODE keyword requires the SELECT keyword. This applies to ALL of the three modes listed below, including MODE(DIRECT). If MODE is not specified, then the SELECT keyword is not required.

#### DIRECT

**MODE(DIRECT)** instructs UNLD to execute under the NGT Direct method and it is the default mode of execution.

Note that in this mode, the SELECT/WHERE/FROM/ORDER BY clauses may be coded anywhere and in any order within the scope of the Unload utility statement. For example, you could code your statement like this:

```plaintext
UNLD ORDER BY 1 FORMAT(DSN)
  WHERE C1=C2
  SELECT * PAD FROM TBL1 MODE(DIRECT)
```

You may, however, wish to standardize how you code such statements for readability and to minimize confusion from job step to job step.

#### DB2SQL

**MODE(DB2SQL)** instructs the Unload utility to extract the SELECT clause from the Unload utility statement and process it using IBM's DB2 SQL processor.

The output is then formatted to any of the Unload utility's popular formats including DSNTIAUL, EXTERNAL, ASCIIDEL, and REPORT.
All data types are supported including LOBs, however LOB columns may be truncated to fit the maximum record length of 32752. If more than one LOB column is present, then the truncation will occur evenly across all LOB columns. LONG VARCHAR columns are also subject to truncations to fit the maximum record length of 32752.

When using the Unload utility's Dynamic Allocation under MODE(DB2SQL), the MAXROWS value will be used to estimate the output dataset primary space allocation. If MAXROWS is not specified, the value of +FILESZ parameter will be used accordingly.

For all applicable formats, DATE, TIME, and TIMESTAMP fields will be reformatted based on the specification of the DATEFMT, TIMEFMT and TSFMT keywords respectively. It should be noted that global formatting options specified in the //ULDPARMS DD statement will NOT be used to reformat DATE/TIME fields.

See Sec. 2.4.1, "DB2SQL Restrictions", for a list of restrictions on the use of DB2SQL.

**AUTO**

MODE(AUTO) instructs the Unload utility to conditionally process the SELECT clause under DB2's Full-Function SQL when it encounters either a syntax or a statement validation error. Such errors exist when:

a. SELECT/WHERE clause(s) contain unsupported syntax.
b. SELECT clause results in an incompatible data type or nullability attribute with mode(DB2SQL)
c. A selected column is a LOB.
d. A complex view definition exists.
e. An output row size exceeding 32756 bytes exists.
f. An ORDER BY key length exceeding 4090 bytes exists.

If a statement passes any one of the above tests, then the mode is switched to MODE(DB2SQL), otherwise the statement is processed using the NGT Direct Method.

When the Direct Method is chosen under MODE(AUTO) certain processing options are changed to maintain compatibility with DB2SQL mode. These are:

a. +DIGITS(IBM) is enforced. This forces the NGT SQL functions DIGITS and CHAR to produce output compatible with DB2's SQL functions DIGITS and CHAR.
b. A selected ROWID column is always included in the data whether defined with the GENERATED ALWAYS attribute or by default.

Whenever UNLOAD is going to be performed under DB2SQL, the following message is issued:

**NGTN049 UNLOAD VIA DB2SQL**

For both modes, the SELECT clause must NOT be placed in parentheses, and there should be no Unload utility keywords that follow it. In other words, all specified Unload utility keywords must precede the SELECT keyword.

See Sec. 2.4.2, "General Limitations", for more information on the use of AUTO.

**MRF**

Directs processing to use multi-row fetch for a specified number of rows, and can significantly improve performance by reducing CPU time. This keyword may only be used when MODE(DB2SQL) or MODE(AUTO) is specified. An integer between 0 and 32767 may be specified; the default is 0.
2.4.1 **DB2SQL Restrictions**

When using MODE(DB2SQL), Unload statements must take the following form:

```sql
UNLD DATA FORMAT(format) OUTDDN(ddname) SELECT select-clause FROM TABLE table
```

as in, for example,

```sql
UNLD DATA FORMAT(EXTERNAL) OUTDDN(OUTDSET) SELECT CHAR(DT_EFF, USA),
CHAR(DT_STAT, ISO), CHAR(DT_TERM, EUR) FROM TABLE CORP.AR_RECORDS
```

If you have used previous releases of NGT Unload, you may be accustomed to coding something similar to:

```sql
UNLD DATA FORMAT(EXTERNAL) OUTDDN(OUTDSET) FROM TABLE CORP.AR_RECORDS
```

where the "FROM TABLE" clause immediately follows the UNLD DATA keyword combination. To select specific columns or format them, you may be tempted to merely append a SQL Select statement on to the end, as in:

```sql
UNLD DATA FORMAT(EXTERNAL) OUTDDN(OUTDSET) FROM TABLE CORP.AR_RECORDS SELECT
CHAR(DT_EFF, USA), CHAR(DT_STAT, ISO), CHAR(DT_TERM, EUR)
```

This is **not** a valid statement if you are using MODE(DB2SQL). (If you are using any of the other MODE options, it is.) When using MODE(DB2SQL) your statement must follow this keyword order:

```sql
UNLD DATA FORMAT(format) OUTDDN(outddname) ... SELECT ... FROM TABLE ...
```

2.4.2 **General Limitations**

MODE(DB2SQL) and MODE(AUTO) may not be used with any of the following Unload utility keywords:

1. COPYDDN
2. COPYDSN
3. LOADINDD
4. SYSCOPY
5. PART
6. PARTS
7. CCSID
8. ASCII
9. FORMAT(UNLOAD)
10. FORMAT(ARCHIVE)
11. FORMAT(UNL)
12. FORMAT(ARC)
13. UNICODE
14. EBCDIC
15. BYPART
16. DIGITS(CDB)

Attempts to do so will result in an error message.
### 2.4.3 Data Compatibility Between MODE(DIRECT) And MODE(DB2SQL)

A character string constant in MODE(DB2SQL) is treated as VARCHAR while in Direct Method Mode it will result in a fixed-length character string. For example:

```
UNLD MODE(AUTO) SELECT 'ABC' AS FIELD1 FROM SYSIBM.SYSDUMMY1
```

will result in the following field spec:

```
(FIELD1 POSITION(1) VARCHAR)
```

while:

```
UNLD SELECT 'ABC' AS FIELD1 FROM SYSIBM.SYSDUMMY1
```

will result in the following, different field spec:

```
(FIELD1 POSITION(1) CHAR(3))
```

Now, in order to make MODE(DB2SQL) compatible with Direct Method mode, specify:

```
UNLD MODE(AUTO) SELECT CHAR('ABC',3) AS FIELD1 FROM SYSIBM.SYSDUMMY1
```

Nullability rules are the same as DB2SQL except when performing arithmetic on two non-nullable (NOT NULL) fields. In DB2SQL mode, the result will be nullable, however, in Direct Method mode, it will result in a NOT NULL column. For example, assuming F1 and R2 are defined as NOT NULL integers:

```
UNLD MODE(AUTO) SELECT F1 + F2 AS FIELD1 FROM TBL1
```

Will result in the following field spec:

```
(FIELD1 POSITION(1) INTEGER NULLIF(5) = '?' )
```

While:

```
UNLD SELECT F1 + F2 AS FIELD1 FROM TBL1
```

will result in the following field spec:

```
(FIELD1 POSITION(1) INTEGER)
```

Now in order to make MODE(DB2SQL) compatible with Direct Method mode specify:

```
UNLD MODE(DB2SQL) SELECT IFNULL(F1+F2,0)
```

or

```
UNLD MODE(AUTO) SELECT IFNULL(CASE WHEN INT(1) = 1 THEN
  F1 + F2 ELSE NULL END,0)
```

### 2.4.4 Page Encoding

On the Unload (UNLD) Utility statement, you may encode the unloaded data using a variety of code pages. Encoding data applies only to Single Byte Character Strings (SBCS) in fields defined as CHAR or VARCHAR. External numeric and date/time fields are also subject to the encoding process. Character fields defined as BIT data are excluded. Double Byte Character Strings
(DBCS) are also excluded. Mixed-character strings result in an error message if the source table has an encoding scheme of ASCII or UNICODE.

A description of the four keywords that control the encoding process follows.

**CCSID number**

Specifies the Coded Character Set ID (CCSID) to encode the unloaded data. The source CCSID will be extracted from the table definition in the DB2 catalog. SYSIBM.SYSSTRINGS will be searched for an appropriate translate table to encode the data using the source and target CCSID. If no such entry exists, an error will occur and the UNLD operation will fail.

**ASCII**

Specifies the Coded Character Set ID (CCSID) used to encode the unloaded data is to be extracted from DB2's default ASCII CCSID, defined in the DSNHDECP configuration module. The number is then used as if a CCSID number was coded.

**UNICODE**

Specifies the Coded Character Set ID (CCSID) used to encode the unloaded data is to be extracted from DB2's default UNICODE CCSID defined in DSNHDECP configuration module. The number is then used as if a CCSID number was coded.

**EBCDIC**

Specifies the Coded Character Set ID (CCSID) used to encode the unloaded data is to be extracted from DB2's default EBCDIC CCSID defined in DSNHDECP configuration module. The number is then used as if CCSID number was coded.

When none of ASCII, UNICODE or EBCDIC is specified on the UNLD stmt, the default is based on the encoding scheme of the unloaded table as follows:

1. ASCII will encode data using EBCDIC CCSID defined to DB2,
2. UNICODE will encode data using EBCDIC CCSID defined to DB2, and
3. EBCDIC will NOT encode data in MODE(DIRECT) but WILL encode data using EBCDIC CCSID Defined to DB2 if MODE(AUTO).

### 2.4.4.1 Format Delimiters

Format delimiters will not be subject to encoding and will be used as is without any changes. You may override defaults with appropriate encoded delimiters. Format delimiters include:

1. NULLCHAR
2. FIELDSEP
3. CHARFLDDLDM
4. NULLSTRING
Behaviors for various data objects are summarized in the table below.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Encoded?</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE/TIME/TIMESTAMP Formats</td>
<td>YES</td>
<td>All DATEFMT, TIMEFMT and TSFMT skeletons will be subject to encoding from a CCSID of 37 to the desired CCSID.</td>
</tr>
<tr>
<td>HEX constants</td>
<td>NO</td>
<td>Hex constants in a SELECT statement will not be encoded and will be treated as BIT data such as &quot;SELECT x'c1c2' from B&quot;.</td>
</tr>
<tr>
<td>ZONED function</td>
<td>NO</td>
<td>The output of the NGT SQL ZONED function will not be encoded and will be treated as BIT data.</td>
</tr>
<tr>
<td>LOWER/UPPER/HEX functions</td>
<td>YES if not EBCDIC</td>
<td>LOWER, UPPER and HEX functions will result in an error if the encoding scheme of the table is not EBCDIC.</td>
</tr>
<tr>
<td>Padding</td>
<td>YES</td>
<td>Pad characters will be subject to encoding. In FORMAT(REPORT), VARCHAR fields are padded with EBCDIC blanks in EBCDIC and ASCII or UNICODE blanks in ASCII or UNICODE respectively.</td>
</tr>
</tbody>
</table>

2.5 **select option Keywords**

* Indicates all columns from the specified table are to be selected.

columnname Indicates one or more columns with the specified names is to be selected.

expression Indicates a subset of the columns in the unload table, or an order that differs from the natural order in which the columns are arranged, is to be selected. An expression may be used to refer to an operation on one or more columns.

2.6 **from table option Keywords**

FROM (TABLE) Specifies the table from which data records are to be unloaded. The keyword TABLE is optional. You may specify a table name, view name, synonym, or alias.

AS (TABLE) Specifies an unloaded table's new name. The keyword TABLE is optional. Unlike with the FROM keyword, you may only specify a table name.
2.7 order by option Keywords

ORDER BY
Indicates unloaded rows will be ordered.

*columnname ASC, columnname DESC*
Indicates one or more columns on which data shall be ordered before unloading. ASC means data for that column will be in ascending order; DESC means data for that column will be in descending order.

CLUSTER
Indicates processing is to unload rows in clustering index order (or physically sorted order).

*SQL-expression*
A valid SQL expression (constants not permitted) that will select only rows for unload meeting the criteria specified by the expression. ASC means data meeting this criteria will be in ascending order; DESC means data meeting this criteria will be in descending order.
2.8 Parameters

In this section we cover the parameters you can specify for running this utility. These passwords are specified after the ULDPARMS DD statement in your NGT job stream.

2.8.1 +BLANKPAD

Pad varchar columns with blanks

Syntax

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

Description

+BLANKPAD is used to pad varchar columns with blanks rather than low values. This parameter can be overridden by the BLANKPAD NGT Unload keyword.

See the PAD keyword for controlling whether varchar fields are padded or not.

Operands

- **YES**  
  Pad varchar columns with blanks.

- **NO**  
  Don't pad varchar columns with blanks.

Example

```
+BLANKPAD(YES)
```

Varchar columns will be padded with blanks.
2.8.2  +CHARFLDDLM
Character Field Delimiter

Syntax

\[ +\text{CHARFLDDLM}( \text{delim} ) \]

Description

+CHARFLDDLM is used to override the default field delimiter characters used for character data fields when unloading using FORMAT(ASCIIDEL). The default delimiter for character data is the double quote ("").

You can use most single characters as a delimiter except a plus sign, comma or blank. You may also use a null character (no character at all; code as +CHARFLDDLM() with no operand).

Operands

| delim    | Single character representing the character field delimiter. May also be null. |

Example

| +CHARFLDDLM( ) | Character strings are not delimited. |
| +CHARFLDDLM( . ) | Character strings are delimited with a period. |
2.8.3  **+CONNECTALL**

Connect all partitions of a table space to the NGT Subsystem when unloading select partitions.

**Syntax**

```
+CONNECTALL( YES NO )
```

**Description**

NGT Unload is an online utility so the NGT Subsystem (CDBSS) tracks changes during the Unload. When an Unload is run on a subset of partitions only those partitions have changes tracked.

It is most efficient to only track changes for the scope of partitions being unloaded, as with `+CONNECTALL(N)`; however, this prevents a separate Unload job on this same table from unloading separate or overlapping partitions concurrently.

Use `+CONNECTALL(Y)` when you expect to run multiple Unload jobs on the same partitioned table simultaneously and not all partitions are being unloaded.

**Operands**

- **YES**
  
  Connect all partitions even though WHERE clause limits the partitions being unloaded. Abbreviation Y accepted.

- **NO**
  
  Default. Only connect partitions selected by WHERE clause for unloading. Abbreviation N accepted.

**Example**

```
+CONNECTALL(Y)
```

Track changes for all partitions in preparation for concurrent Unloads on this table.

```
+CONNECTALL(N)
```

Only track changes for the specific partitions being unloaded.
2.8.4  +DATEFMT
Date Format

Syntax

```
+DATEFMT ( ISO | EUR | JIS | USA | LCL )
```

Description

+DATEFMT is used to set or override the default date format. The distributed default is ISO. This default only affects the external representation of date values when using formats DSNTIAUL, ASCIIDEL and REPORT.

Operands

- **ISO**: Use the ISO date format (yyyy-mm-dd).
- **EUR**: Use the European date format (dd/mm/yyyy).
- **JIS**: Use the Julian date format (yy.ddd).
- **USA**: Use the USA date format (mm/dd/yyyy).
- **LCL**: Use the local date format.

Example

```
+DATEFMT(EUR)
```
Sets the default format to European.
2.8.5  **+DEGREE**

Process successive UNLD statements for the same table concurrently

**Syntax**

```
+DEGREE( n )
```

**Description**

+DEGREE is used to specify the number of successive UNLD statements of the same table that can be processed concurrently with one read of the table and parallel writing of the various unloads to the separate unload datasets.

**Operands**

- **n**  
  Between 0 and 63

**Example**

```
+DEGREE(4)
```

Process up to 4 successive UNLD statements in parallel with one read of the table.

**Qualifications**

For the UNLD statements to be processed in parallel the following conditions must be met:

1. +DEGREE > 1
2. No ORDERBY
3. No specific part numbers in UNLD statements.
4. No BYPART keyword to unload by partition.
5. No Unload from ImageCopy.
6. The UNLD keyword MODE must not use DB2SQL.
7. Each UNLD statement must be to a different unload dataset.
8. No mix of SHRLEVEL CHANGE (dirty unload) and CONCURRENT (online unload).
9. All unloads must be to statically allocated unload datasets, or all to dynamically allocated datasets; no mix of the two.
10. The size of all merged UNLD statements to be processed in parallel can not exceed 63K.
If they do, the first 63K will be processed in parallel then the remaining will be processed in parallel.

11. Disk or VTS unload files recommended. Specify MEDIA=D and copy to Disk or VTS. An attempt to stack datasets on VTS or Tape will result in an error. If you must unload to Tape, specify FSEQ=1 in XULDDYNM to prevent stacking and specify DDNCOUNT(n) in CDBTAPE to allocate n tape drives to match +DEGREE(n).

Additional Information

The UNLD reader will read a max of 32 partitions concurrently and process up to 63 output datasets in parallel.

Successive statements that satisfy the above criteria form a parallel output group. When a statement does not satisfy the criteria, the existing group is ended and another group is started. Multiple groups can be dispatched at the same time depending on +MAXTASKS.

Advantage of grouping:

1. Reduces CPU/elapsed and connect time immensely.
2. Reduces memory utilization.
3. Reduces EXCP counts.

Disadvantage of grouping:

1. No two Output datasets in a single group can be stacked on the same tape.

+DEGREE related messages in output.

When two statements are combined: the following message is issued (during parse):
NGTN069 UNLD STMT 1 has been merged with STMT 2. Queue Depth=2

It will issue the following message before starting the work:
NGTN070 Dispatching 10 concurrent reader(s) using 22 buffers to 5 parallel writer(s)
2.8.6: FIELDSEP

Field Separator

Syntax

```
+FIELDSEP( char )
```

Description

+FIELDSEP is used to override the default field separator character (comma ",") between fields in ASCIIDEKL output files. You can use any single characters as separator characters except a plus sign. You may use a blank as a separator.

Operands

<table>
<thead>
<tr>
<th>char</th>
<th>A single character representing the field separator.</th>
</tr>
</thead>
</table>

Example

```
+FIELDSEP(-)
All fields are separated with a hyphen.

+FIELDSEP()
A blank is placed between all fields.

+FIELDSEP()
Same as above; a blank is placed between all fields.
```
2.8.7 **+MINROWS**

Minimum Number of Rows To Retrieve

**Syntax**

```
+MINROWS ( integer )
```

**Description**

If the +MINROWS keyword is specified and the number of rows unloaded is less than MINROWS, one will receive a message stating the number of rows unloaded, the value of MINROWS, and the unload will end with a RC=4.

**Operands**

- **0**  
  Default. Processing will retrieve all rows.
- **n**  
  An integer representing the minimum number of rows to retrieve.

**Example**

```
+MINROWS(200)
```

Processing must unload at least 200 rows or the job will end with a warning.
2.8.8  

**+NUMFLDDLM**  
Numeric Field Delimiter  

**Syntax**  

```
+NUMFLDDLM( delim )
```

**Description**  

`+NUMFLDDLM` is used to override the default field delimiter character used for numeric data fields when unloading using the ASCIIDEL format. The default action for placing a delimiter around numeric fields is for NGT Unload to not delimit numeric fields. You can use most single characters as a delimiter except the plus sign, a comma or a blank. You may also use no character as a delimiter.

**Operands**  

- **null**  
  Default. (Represented by upper syntax path in diagram above.)
- **delim**  
  A single character representing the numeric field delimiter. May also be null.

**Examples**  

```
+NUMFLDDLM()
```

Numeric strings are not delimited.

```
+NUMFLDDLM($)
```

Numeric strings are delimited with the USA dollar currency sign.
2.8.9  +PASSWORD  
License Expiration Password

Syntax

+PASSWORD(string)

Description

Obtain the password string value from BMC Customer Support. The password may be supplied to extend the no-obligation trial or to set the utility as non-expiring after license. +PASSWORD is usually specified at installation time, eliminating the need to specify it in each job.

Operands

string A character string representing the password for NGT Unload.
2.8.10  

+TIMEFMT

Default Time Format

Syntax

+TIMEFMT ( ISO  
  EUR  
  JIS  
  USA  
  ST )

Description

+TIMEFMT is used to set or override the default time format. The distributed default is ISO. This default only affects the external representation of time values when using formats DSNTIAUL, ASCIIDEL and REPORT.

Operands

ISO  Use the ISO time format.
EUR  Use the European time format.
JIS  Use the Julian time format.
USA  Use the USA time format.
ST   Use the standard time format.

Example

+TIMEFMT(ISO)
Sets the default format to ISO.
Chapter 3 Output File Formats
This chapter covers the outputs produced by this utility.

3.1 Generated LOAD DATA Statements
NGT Unload can automatically produce LOAD DATA statements (to a defined SYSPUNCH DD card) when the REPORT, DSNTIAUL or EXTERNAL format is specified. The resulting statement can then be used to reload data in a successive job. The ARCHIVE, ASCIIDEL and UNLOAD formats will not produce a LOAD statement. Refer to the table below for quick reference, or to each section for more details.

<table>
<thead>
<tr>
<th>Unload Format</th>
<th>Produces LOAD Statement in SYSPUNCH DD?</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNTIAUL</td>
<td>YES</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>no</td>
</tr>
<tr>
<td>ARCHIVE</td>
<td>no</td>
</tr>
<tr>
<td>ASCIIDEL</td>
<td>no</td>
</tr>
<tr>
<td>REPORT</td>
<td>YES</td>
</tr>
<tr>
<td>EXTERNAL</td>
<td>YES</td>
</tr>
</tbody>
</table>

If you require the LOAD DATA statement to be generated along with the unloading of the data from the table for a particular output format, allocate the DD name //SYSPUNCH to a sequential file, as in:

```
//SYSPUNCH DD SYSOUT=*  
```

or

```
//SYSPUNCH DD DISP=SHR,DSN=NGT.UNLOAD.AUTO.GENERATE  
```

You may also override this DD name via the STMTDDN keyword of the UNLD statement.

If this format is chosen and //SYSPUNCH is not allocated and you have not overridden the DD name via STMTDDN or the DD name you specified after STMTDDN is not allocated, the LOAD DATA statement will not be produced and no error message will be issued (it will be assumed that you wish to bypass generation of the LOAD DATA statement).

3.2 Unload Output File Formats
This section describes the various output data formats supported by NGT Unload. There are several standard output formats named in the FORMAT keyword. Currently available operands for the FORMAT keyword are:

- DSNTIAUL
- UNLOAD
- ARCHIVE
Output File Formats

- DELIMITED
- ASCIIDEL
- REPORT
- EXTERNAL

With all output formats you have the option of unloading all columns or a subset of the columns.

**Case Study**

To better illustrate, we will be using the following table that contains three rows of data. The DDL representing this object is shown below.

```
-- HERE'S THE TABLE DEFINITION:

CREATE TABLE NGT.TABLE1
  (  NAME      CHAR(6) NOT NULL ,
    AGE       INTEGER NOT NULL ,
    SALARY    DECIMAL (6,2 ) ,
    COMMENT   VARCHAR (20) ) IN DATABASE NGT ;

-- HERE ARE THE THREE ROWS OF DATA:

INSERT INTO NGT.TABLE1 VALUES ( 'TODD' ,  16  ,  123.45 ,  'USE UNLOAD!!' ) ;
INSERT INTO NGT.TABLE1 VALUES ( 'MATTEO' ,  32  ,  9500.50 ,  NULL ) ;
INSERT INTO NGT.TABLE1 VALUES ( 'IRINA' ,  48  ,  NULL ,  ''  ) ;
```

### 3.2.1 DSNTIAUL Format

**Automatically Generates LOAD DATA Statement To SYSPUNCH: YES**

NGT DSNTIAUL format is the same as IBM's DSNTIAUL. By default variable length fields are padded to their maximum length with binary zeros but you may override this. Date, Time and Timestamp fields are converted to character strings.

If a column is nullable the column value in the output file is immediately preceded or immediately followed by the NULL marker. This NULL marker may consist of a single character or of multiple characters. The maximum number of characters allowed is four. If the NULL marker is binary zeros (X'00'), the column value was not null when unloaded. If the NULL marker is not binary zeros, the column value was null when unloaded. In this case, the area in the output row represented by that column contains binary zeros.

Numeric values are converted from internal DB2 format to S/390 data type formats. Each variable-length column is prefixed by a two byte number representing the length of the column excluding both the prefix itself and the null indicator field if present. Depending upon the NULL field specifications the null indicator byte will either immediately precede or immediately follow the nullable field.

The RECFM of the output file when using the DSNTIAUL format can be set via two methods. The first way is to specify the PAD or NOPAD keyword in the unload command. The second way is to override the RECFM value in the JCL for the unload dataset.
If the RECFM parameter is not overridden in the JCL then specifying the PAD keyword sets the RECFM to FB. This keyword sets the unused area to the right of the data value in a varying field to binary zeroes.

If NOPAD is specified then the RECFM of the output file is set to VB. This keyword does not pad the area to the right of the data value in a varying field to binary zeroes.

You can also set the record format to VB by specifying DCB=RECFM=VB in the DD statement for the unload dataset. If you specify this the unloaded variable fields will not be padded with binary zeroes. This is incompatible with the specification of the PAD keyword.

If you wish to unload table NGT.TABLE1 with NGT Unload using the DSNTIAUL format to a file with record format FB you can use either of these statements:

```
UNLD FROM TABLE NGT.TABLE1
UNLD DATA FORMAT(DSNTIAUL) FROM TABLE NGT.TABLE1
```

This would result in the following:

```
ROW 1:  TODD ..........USE UNLOAD!!..........
       (EDCC4400010135000EEC4EDDCC55000000000)
       364400000024C00CE250453614AA000000000
ROW 2:  MATTEO...............................?
       (DCEEC000205000000000000000000000006)
       4133560000905C0000000000000000000C
ROW 3:  IRINA ........?.......................?
       (CDCDC40003000060000000000000000000)
       99951000000000C0000000000000000000
```

**NOTE:**
In this figure and others like it in this chapter the row labeled “COLS/NULLS” shows the start of each column with a vertical bar (|). Null indicators are marked with an asterisk (*).

The ROW 1 columns and values are:

- **NAME**: "TODD ", 6 bytes.
- **AGE**: X'00000010', 4 bytes.
- **SALARY**: X'0012345C', 4 packed decimal bytes, followed by X'00', a one-byte null flag, indicating NOT NULL.
- **COMMENT**: X'000C', a two byte length field, followed by USE UNLOAD!! , 12 bytes of characters, followed by X'0000000000000000' , 8 bytes binary zero fill bytes, followed by X'00', a 1-byte null flag.

Other interesting fields are:

**ROW 2/Column COMMENT**
Always at offset 15 in the row, this column is varying length. The length in Row 2 shows a zero length. The null indicator, always at offset 37 in the row, shows a ?. Therefore, this field is null.
Output File Formats

ROW 3/Column SALARY

Similar to Row 2, column COMMENT, Row 3 column SALARY is null. SALARY is at offset 10 through 13. Offset 14 is the null indicator and it shows to be null also.

ROW 3/Column COMMENT

The entire field is binary zeros, including the length and the null indicator. This means that the value is NOT NULL, and it is a varying length string of length 0.

3.2.2 UNLOAD Format

Automatically Generates LOAD DATA Statement To SYSPUNCH: NO

This format is the same format used by IBM REORG with the UNLOAD PAUSE option. It is based on the following rules:

- RECFM is always VB.
- The expansion of varying fields to their full length is based on the use of the PAD and NOPAD keywords.
- Variable-length field length bytes indicate the actual length of the data, not the maximum allowed length.
- Variable-length fields are prefixed with a 2-byte field containing the length of the field exclusive of the prefix itself.
- Column values are left in DB2 internal format.
- Each row contains a 6 byte record prefix that contains a 1 byte row flag, a 2 byte length (the same for every row), a 2 byte OBID for the owning table and a 1 byte MAP ID reference.
- Null indicators precede nullable columns and are copied without changes from the values found in the table being unloaded.
- You may override the blocksize chosen by NGT Unload by coding the blocksize keyword in the JCL.

The following unload statement produces the output file shown on the next page:

```
UNLD DATA FORMAT(UNLOAD) FROM TABLE NGT.TABLE1
```
Output File Formats

This will result in the following unloaded data:

<table>
<thead>
<tr>
<th>1</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**COLS/NULLS:**

**ROW 1:**

```
002000EDCC44000100135000EEC4EDDDCC5500000000
00B03136440000000024C0D0E250453614AA00000000
```

**ROW 2:**

```
002000DCEECDO00020050000F00000000000000000000
00B0324133560000905C01F00000000000000000000
```

**ROW 3:**

```
002000CDCDC40003F000000000000000000000000000000
00B03399951000001F000000000000000000000000000000
```

The ROW 1 columns and values are:

- **Field 1, Record ID**: X'00', 1 byte.
- **Field 2, Record Length**: X'002B', 2 bytes long. This is the total length of the record.
- **Field 3, Record OBID**: X'0003', 2 bytes.
- **Field 4, MAP ID Reference**: X'01', 1 byte.
- **NAME**: TODD, 6 bytes
- **AGE**: X'00000010', 4 bytes
- **SALARY**: X'00', the 1 byte null indicator, followed by X'0012345C', 4 packed decimal bytes. The field is NOT NULL.
- **COMMENT**: X'000D', a two byte length field, that includes the length of the null indicator, followed by X'00', a 1 byte null flag, followed by USE UNLOAD!!, 12 bytes of characters, followed by X'0000000000000000', 8 bytes binary zero fill bytes.

Other interesting fields are:

- **ROW 2/Column COMMENT**
  Always at offset 21 in the row, this column is varying length. The length in Row 2 shows a length of 1. Of this length, the null indicator is 1 byte and the data is the rest. This field is null, so there is no data.

- **ROW 3/Column SALARY**
  Similar to Row 2, column COMMENT, Row 3 column SALARY is null.

- **ROW 3/Column COMMENT**
  The field is length 1, which includes only the null indicator. In this case, there is a varying length string of length 0.
### 3.2.3 ARCHIVE Format

**Automatically Generates LOAD DATA Statement To SYSPUNCH: NO**

The ARCHIVE format is a variation of the UNLOAD format. Whereas the DB2 UNLOAD format expands all variable-length records to their maximum length and adds a 6 byte record header, format ARCHIVE does not. In essence, NGT Unload simply copies the record from the table to the output file without changes except for the stripping of the six-byte record prefix.

The ARCHIVE format is also supported by BMC Next Generation Technology Load (NGT Load) and BMC Next Generation Technology Reorg (NGT Reorg) and is the recommended format when data needs to be exchanged among these products and NGTUnload.

The following unload statement produces the output file shown just below:

```
UNLD DATA FORMAT(ARCHIVE) FROM TABLE NGT.TABLE1
```

This would result in the following:

<table>
<thead>
<tr>
<th>Row</th>
<th>Columns</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NAME: TODD............USE UNLOAD!! (EDC40000100135000EEC4EDDC55)</td>
<td>36440000000024C0D0E250453614AA</td>
</tr>
<tr>
<td>2</td>
<td>NAME: MATTEO........... (DCEEC000200500000F)</td>
<td>41335600000905C01F</td>
</tr>
<tr>
<td>3</td>
<td>NAME: IRINA............ (CD0C40003F00000000)</td>
<td>992251000000080010</td>
</tr>
</tbody>
</table>

The ROW 1 columns and values are:

- **NAME**: "TODD ", 6 bytes
- **AGE**: X'000000010', 4 bytes
- **SALARY**: X'00', the 1 byte null indicator, followed by X'0012345C', 4 packed decimal bytes. The field is NOT NULL.
- **COMMENT**: X'000D' , a two byte length field, that includes the length of the null indicator, followed by X'00', a 1 byte null flag, followed by USE UNLOAD!! , 12 bytes of data.

If you have a FORMAT(ARCHIVE) file, you can run NGT Utilities with the FORMAT(ARCHIVE) file as input. In turn, you can unload it into any format you choose.

### 3.2.4 DELIMITED Format

Specifies that the unload data file is in a delimited format. When data is in a delimited format, all fields in the input data set are character strings or external numeric values. In addition, each column in a delimited file is separated from the next column by a column delimiter character.

For each of the delimiter types that you can specify, you must ensure that the delimiter character is specified in the code page of the source data. The delimiter character can be specified as either a character or hexadecimal constant. For example, to specify '#' as the delimiter, you can specify either COLDEL '#' or COLDEL X'23'. For Unicode and ASCII encodings, you should...
specify the delimiter character in the utility statement as a hexadecimal constant, or the result can be unpredictable.

You cannot specify the same character for more than one type of delimiter (COLDEL, CHARDEL, and DECPT).

**COLDEL coldel**
Specifies the column delimiter that is used in the unload file. The default value is a comma (,). For ASCII and UTF-8 data this is X'2C', and for EBCDIC data it is X'6B'.

**CHARDEL chardel**
Specifies the character string delimiter that is used in the unload file. The default value is a double quotation mark ("). For ASCII and UTF-8 data this is X'22', and for EBCDIC data it is X'3F'. Character strings that contain the character string delimiter will repeat the character string delimiter where it is used in the character string. LOAD interprets any pair of character delimiters that are found between the enclosing character delimiters as a single character. For example, the phrase "FRED""S JOB" is interpreted as "FRED'S JOB". The LOAD utility recognizes these character delimiter pairs for only CHAR, VARCHAR, and CLOB fields. Character string delimiters are required only when the string contains the CHARDEL character. However, you can put the character string delimiters around other character strings.

**DECPT decpt**
Specifies the decimal point character that is used in the unload file. The default value is a period (.). The default decimal point character is a period in a delimited file, X'2E' in an ASCII or Unicode UTF-8 file. The only allowed delimiters for DECPT is '.' and ',' or their equivalents in ASCII /UNICODE.

### 3.2.5 ASCIIDEL Format
** Automatically Generates LOAD DATA Statement To SYSPUNCH: NO **

Format ASCIIDEL unloads all fields to character format. By default, it delimits character fields with double quotes, does not delimit numeric fields and separates each field with a comma. This unload statement:

```sql
UNLD DATA FORMAT(ASCIIDEL) FROM TABLE NGT.TABLE1
```

results in this file output:

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>ROW 1:</td>
<td>&quot;TODD&quot;,16,123.45,&quot;USE UNLOAD!!&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROW 2:</td>
<td>&quot;MATTEO&quot;,32,9500.00,&quot;&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROW 3:</td>
<td>&quot;IRINA&quot;,48,&quot;&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

By default, null character data is represented via the empty string, or two double quotes and null numeric fields receive no data, as is shown in ROW 3.

You have the ability to override certain formatting characteristics via NGT Unload parameters. To change the field delimiters for numeric or character data, use the +NUMFLDDLM or +CHARFLDDLM parameters respectively.

To change the character used to separate fields from the default of comma (,) to any other character, use the +FIELDSEP parameter.
Output File Formats

These parameters affect all the UNLD statements in the job. You may also change them for a specific UNLD statement by using the UNLD keywords FIELDSEP, CHARFLDDLM and NULLSTRING. See Field Specification (fieldspec) for more details.

3.2.6 REPORT Format
Automatically Generates LOAD DATA Statement To SYSPUNCH: YES

The REPORT format unloads all fields to character format. Fields are not delimited, nor are fields separated. The file produced is a fixed length file. If a SYSPUNCH DD is allocated, a LOAD DATA statement is produced.

NOTE

The Load statement generated to SYSPUNCH for Format REPORT is only valid if there are no nullable columns in the table being unloaded. Null fields will be converted to blanks and there are no null markers so it will not be valid for reloading the data if nullable columns are unloaded.

Character and graphic fields are unloaded without changes or conversions of any type. Variable-length fields are expanded to their maximum length and padded with blanks. In the LOAD DATA statement they are assigned a type of CHAR or GRAPHIC accordingly.

Date, Time and Timestamp fields are edited to contain appropriate separator characters. You may choose the formats for Date or Time columns.

Numeric fields are converted to character strings, extended to a predefined length and right-justified. These predefined lengths are:

- INTEGER 11 bytes
- SMALLINT 6 bytes
- FLOAT 23 bytes
- DECIMAL The precision plus 2 bytes
- BIGINT 20 bytes
- DECFLOAT(16) 23 bytes
- DECFLOAT(34) 42 bytes

A SYSPUNCH file, if allocated, will contain a LOAD DATA statement for the generated file. If a field is null, and non-numeric, it contains blanks. If numeric and null, it contains a 0. Both of these default null placeholders can be overridden with an IFNULL clause.

The following unload statement:

```
UNLD DATA FORMAT(REPORT) FROM TABLE NGT.TABLE1
```

will produce this unload output file:

```
1 5 10 15 20 25 30 35 40 45
+--------------------------+
```
### Output File Formats

<table>
<thead>
<tr>
<th>Row</th>
<th>Name</th>
<th>Age</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TODD</td>
<td>16</td>
<td>123.45</td>
</tr>
<tr>
<td>2</td>
<td>MATTEO</td>
<td>32</td>
<td>9500.00</td>
</tr>
<tr>
<td>3</td>
<td>IRINA</td>
<td>48</td>
<td>0</td>
</tr>
</tbody>
</table>

**NOTE:**
VARG and LONGVARG columns are unloaded as GRAPHIC. Depending on the length of the variable-length column this may produce a LOAD DATA statement that will not parse correctly with the LOAD utility. However, the LOAD DATA statement can still be used as a layout reference for the output file.

### 3.2.7  EXTERNAL Format

**Automatically Generates LOAD DATA Statement To SYSPUNCH: YES**

This is similar to format DSN, except that numeric fields are unloaded in right-justified external format.

If a column is nullable, the column value in the output file will be immediately preceded or immediately followed by the NULL marker which may consist of a single character or a string of up to four characters. If the NULL marker is binary zeros (X'00'), the column value was not null when unloaded. If the NULL marker is not binary zeros, the column value was null when unloaded. In this case, the area in the output row represented by that column will contain binary zeros.

The following table shows maximum length for the different numeric data types.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Maximum Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>6</td>
</tr>
<tr>
<td>INTEGER</td>
<td>11</td>
</tr>
<tr>
<td>DECIMAL(p,s)</td>
<td>P+2</td>
</tr>
<tr>
<td>FLOAT(21)</td>
<td>15</td>
</tr>
<tr>
<td>FLOAT(53)</td>
<td>23</td>
</tr>
<tr>
<td>BIGINT</td>
<td>20</td>
</tr>
<tr>
<td>DECFLOAT(16)</td>
<td>23</td>
</tr>
<tr>
<td>DECFLOAT(34)</td>
<td>42</td>
</tr>
</tbody>
</table>
Chapter 4  Examples

This chapter covers the syntax for specific examples of NGT Unload.

4.1 Unload using a table lookup

It is sometimes useful to select the rows to unload by determining if the rows exist in another table. For example, you may have a production application that identifies rows to be unloaded over the weekend for later transfer to a non-DB2 application. The following UNLD statement accomplishes this:

```plaintext
//SYSIN DD *
UNLD DATA FORMAT(DSNTIAUL) OUTDDN(SYSREC)
FROM TABLE NGT.PRODUCTION_TBL
WHERE RECID IN (SELECT RECID FROM ROWS2UNLD)
```

4.2 Unload a specific number of rows

Use the MAXROWS keyword to set a limit for the number of rows to unload from the specified table. NGT Unload will stop when it reaches this limit, or when it reaches end of file, whichever comes first. This example stops unloading at 25,000 rows:

```plaintext
//SYSIN DD *
UNLD DATA FORMAT(UNLOAD) OUTDDN(SYSREC)
FROM TABLE DBX14.RFID_CODES
MAXROWS 25000
```

Notes

1. You are not required to unload all columns from every row. If the table defined in DB2 has an OBID of 64, and the following columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAB1.ACOL1</td>
<td>CHAR (8)</td>
</tr>
<tr>
<td>TAB1.ACOL2</td>
<td>INTEGER NOT NULL</td>
</tr>
<tr>
<td>TAB1.ACOL3</td>
<td>SMALLINT NOT NULL</td>
</tr>
<tr>
<td>TAB1.ACOL4</td>
<td>FLOAT</td>
</tr>
<tr>
<td>TAB1.ACOL5</td>
<td>VARCHAR (100)</td>
</tr>
<tr>
<td>TAB1.ACOL6</td>
<td>CHAR (4) NOT NULL</td>
</tr>
<tr>
<td>TAB1.ACOL7</td>
<td>INTEGER</td>
</tr>
</tbody>
</table>

and the table in the COPY (see note 3) has an OBID of 200 and the following columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAB2.BCOL1</td>
<td>CHAR (8)</td>
</tr>
<tr>
<td>TAB2.BCOL2</td>
<td>INTEGER NOT NULL</td>
</tr>
<tr>
<td>TAB2.BCOL3</td>
<td>SMALLINT NOT NULL</td>
</tr>
</tbody>
</table>

you can still unload the rows from the COPY if you code:

```plaintext
UNLD OBID 200 COPYDDN(SYSCOPY)
```
Examples

### Examples

```sql
SELECT ACOL1, ACOL2, ACOL3 FROM TABLE TAB1
```

2. **You can also select a subset of the columns.** In this case, only the table columns that will be unloaded have to match the table defined in DB2. Here is an example: Suppose that TAB1 (shown in note 1) is defined in DB2. However, the COPY you want to unload has a table in it, TAB3, but with the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAB3.CCOL1</td>
<td>INTEGER</td>
</tr>
<tr>
<td>TAB3.CCOL2</td>
<td>CHAR (8) NOT NULL</td>
</tr>
<tr>
<td>TAB3.CCOL3</td>
<td>SMALLINT NOT NULL</td>
</tr>
<tr>
<td>TAB3.CCOL4</td>
<td>CHAR (2) NOT NULL</td>
</tr>
<tr>
<td>TAB3.CCOL5</td>
<td>CHAR (2) NOT NULL</td>
</tr>
<tr>
<td>TAB3.CCOL6</td>
<td>VARCHAR (100)</td>
</tr>
<tr>
<td>TAB3.CCOL7</td>
<td>CHAR (1) NOT NULL</td>
</tr>
<tr>
<td>TAB3.CCOL8</td>
<td>CHAR (3) NOT NULL</td>
</tr>
<tr>
<td>TAB3.CCOL9</td>
<td>INTEGER NOT NULL</td>
</tr>
</tbody>
</table>
Examples

The column characteristics are summarized in the table below.

<table>
<thead>
<tr>
<th>TAB1</th>
<th></th>
<th>TAB3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Len</td>
<td>Col</td>
<td>Type</td>
</tr>
<tr>
<td>9</td>
<td>ACOL1</td>
<td>CHAR</td>
</tr>
<tr>
<td>4</td>
<td>ACOL2</td>
<td>INT</td>
</tr>
<tr>
<td>2</td>
<td>ACOL3</td>
<td>SINT</td>
</tr>
<tr>
<td>5</td>
<td>ACOL4</td>
<td>FLT</td>
</tr>
<tr>
<td>?</td>
<td>ACOL5</td>
<td>VCHAR</td>
</tr>
<tr>
<td>4</td>
<td>ACOL6</td>
<td>CHAR</td>
</tr>
<tr>
<td>5</td>
<td>ACOL7</td>
<td>INT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The columns you can successfully unload from TAB3 using table TAB1 as a model are:

- `TAB3.CCOL3` Because offsets, lengths and types match.
- `TAB3.CCOL6` Because of the same reasons. Note that this column is a varying length column.
- `TAB3.CCOL7` Could be unloaded as one data field, even of the lengths of the varying data in the preceding column is different (See note 5)
- `TAB3.CCOL8` Same as TAB3.CCOL7.
- `TAB3.CCOL9` This column, in both tables, is physically +4 after the varying length column. Positions are “reset” after varying field.

These can be unloaded because their physical relative position in the rows of both tables is the same.

Any attempt to unload the other columns would generate unpredictable results.

In this scenario your UNLD statement might be:

```
UNLD OBID 80 COPYDSN(MY.DATASET)
   SELECT ACOL3, ACOL5, ACOL3 FROM TABLE TAB1
```

3. The term "COPY" is used but you can also assign an OBID override to a real DB2 table when not using the unload-from-copy feature.

4. You can use any FORMAT with any of these features.

5. The variable length fields in each table do not have to match. However, NGT Unload will calculate the LRECL of the output dataset based on the maximum length of the DB2 defined columns. If the COPY dataset contains rows that cause an output record to be greater than the predicted LRECL, you will get an ABEND.
Chapter 5

**Relevant Automation**

NGT Automation control points are integrated into all NGT Utilities and can greatly enhance and expand your processing options — you are limited only by your imagination. In this chapter we present automation control points 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:

```plaintext
//CDBEXEC DD DISP=SHR,DSN=NGT.AUTO CTRL.POINTS
```

The automation control point dataset will contain one member with each of the control points you choose to use.

In listing the automation control points below, be aware that these are not the only ones 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 points as desired for whatever additional uses required. Furthermore, where this section lists more than one automation control point, there is no constraint to using all such listed control points as a set; control points may be excluded as needed.

The following standard NGT Automation control points can be used with this utility.

### 5.1 General Utility Automation Control Points

- **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.

- **XSVRXERR** Called if processing ends with an error condition. It may be used to alert a user or to raise an error flag.

For further information and details on the use of all of these automation control points, please see *BMC Next Generation Technology Automation Reference Guide*. 
Chapter 6  Technical Details

6.1  Sizing The Work Dataset (WK00001)
During processing, the size required for the WK00001 is double the size of the SYSREC file. This disk space is used by processing for internal sort and data staging/de-staging for multiple server processes.

For further information on the allocation of the WK00001 work dataset, see the reference BMC Next Generation Technology Automation Reference Guide, in the section on the XSUTDBMG automation control point.

6.2  Selecting Data To Unload
In this section we present some basic information on how to select data for unload.

6.2.1  Selecting Columns
To select columns in NGT Unload, use standard SQL SELECT statements. This is a simple example of a SELECT clause in NGT Unload:

```
UNLD DATA FORMAT(ASCIIDEL)
SELECT COL01, COL02, 'CONSTANT', INTEGER(123)
WHERE COL02 = '20'
ORDER BY COL01
FROM TABLE NGT.UNLOAD_TEST
```

In the list of selected expressions you can specify column names, functions, expressions and constants, just as you would with SQL. In addition you can also use the SQL CASE construct to apply logic to data and substitute values.

An example using two CASE constructs follows:

```
UNLD DATA FORMAT(DSNTIAUL)
SELECT COL01,
    CASE COL02
        WHEN 'TX' THEN 'TEXAS'
        WHEN 'CA' THEN 'CALIFORNIA'
        WHEN 'MO' THEN 'MISSOURI'
        WHEN 'AK' THEN 'ALASKA'
        ELSE CASE
            WHEN COL02 BETWEEN '01' AND '50' THEN 'USA'
            WHEN COL02 = '51' THEN 'ITALY'
            WHEN COL02 = '52' THEN 'CANADA'
            ELSE NULL
        END
    END,
    'NGT',
    INTEGER(123)
FROM TABLE NGT.UNLOAD_TEST
```
Technical Details

Concatenation, substring operations, functions and special registers can be used as well. See the BMC Next Generation Technology General User Guide for full details on this support.

NGT Unload also supports SUBSELECT clauses used within the CASE construct:

```sql
UNLD DATA FORMAT(UNLOAD)
SELECT  COL01,
CASE  COL02
  WHEN 'TX' THEN 'TEXAS'
  WHEN 'CA' THEN 'CALIFORNIA'
  WHEN 'MO' THEN 'MISSOURI'
  WHEN 'AK' THEN 'ALASKA'
ELSE CASE
  WHEN COL02 = '01' THEN 'USA'
  WHEN COL02 = '51' THEN 'ITALY'
  WHEN COL02 = '52' THEN 'CANADA'
  WHEN COL02 IN
    (SELECT EXCEPTION_CODE
      FROM NGT.CODES WHERE
      EXCEPTION_CODE > '53' ) THEN 'EARTH'
ELSE NULL
END
END ,
'NGT',
INTEGER(123)
FROM TABLE NGT.UNLOAD_TEST
```

6.2.2 Selecting Rows

By default, all table rows are unloaded by NGT Unload. If you want to filter the rows to be unloaded, you can do this in several ways:

- Adding a WHERE clause
- Using the MAXROWS keyword
- Selecting with an alternate OBID

More than one of these methods can be used at the same time. If you specify multiple methods, they are applied in the order shown above.

SQL WHERE clauses that contain subselects may only be used with NGT Unload. Subselects may not be used with NGT Unload. The NGT Unload WHERE clause is discussed in detail in the BMC Next Generation Technology General User Guide

6.2.3 Partition Filtering

An SQL WHERE clause can be used to select which rows from the table are to be unloaded. See the BMC Next Generation Technology General User Guide for details on the WHERE clause specification. In addition, NGT Unload automatically skips partitions that do not satisfy the request based on the provided WHERE clause criteria. This may reduce the number of partitions to be read by NGT Unload.

For an Unload to be a candidate for this performance improvement the following criteria must be met:

1. The PART keyword must not be specified on the UNLD utility statement.
2. The COPYDSN or COPYDDN keyword must not be specified on the UNLD utility statement.

3. The “NOFILTER” keyword is not specified.

4. The unloaded table must be a partitioned table space.

5. The UNLD statement must have a WHERE clause that has at least one predicate referencing the first key column of a LIMITKEY.

The NGT Unload output will show the total number of partitions and the partition numbers that satisfy the search condition.

WHERE predicates are used to determine if a partition will be skipped or not. Three types of predicates are supported for this purpose:

- **Basic predicate**
  For a basic predicate to be analyzed, it must be in one of the following forms:

  Column_name RELOP Constant
  Constant RELOP Column_name

  Where:

  Column_name is a key column of the partitioning index.
  RELOP is the relational operator. All operators are analyzed except “^=” or “<>”
  Constant is a Constant or a Special Register that is free of any expressions.

- **IN predicate**
  For an “IN” predicate to be analyzed it has to be in the following form:

  Column_name IN (Constant, Constant, …etc)

- **BETWEEN predicate**
  For the “BETWEEN” predicate to be analyzed, it must be in one of the following forms:

  Column_name BETWEEN Constant AND Constant
  Or
  Column_name NOT BETWEEN Constant AND Constant

Furthermore, the use of search condition operators such as “AND” / “OR” is analyzed to reduce/expand the partition limit range. ANDing predicates may reduce the number of partitions. ORing predicates may increase the selected partition range.

If a predicate does not conform to the above rules, then all parts are selected.

Conditions are logically ANDed with first key column conditions.

The partition selection analysis is exact when the partitioning index has a single-column key. For multiple-column key indexes, the analysis may result in an additional partition per range.
Technical Details

6.2.3.1 A. Single column key
Consider table (T1) defined in a large partitioned TS with 9 partitions. T1 has two columns C1 and C2 and both of type integer. A partitioning index (CX1) indexes C1 in ascending order and has the following limitkeys:

<table>
<thead>
<tr>
<th>Part</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>2</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>3000</td>
</tr>
<tr>
<td>4</td>
<td>4000</td>
</tr>
<tr>
<td>5</td>
<td>5000</td>
</tr>
<tr>
<td>6</td>
<td>6000</td>
</tr>
<tr>
<td>7</td>
<td>7000</td>
</tr>
<tr>
<td>8</td>
<td>8000</td>
</tr>
<tr>
<td>9</td>
<td>9000</td>
</tr>
</tbody>
</table>

6.2.3.1.1 Example A1
UNLD FROM T1 WHERE (C1 > 1000 AND C1 <= 5000 AND C2+C1 = 6000)
This results in the following messages:

4 PARTS SELECTED.
PARTS 2-5

This means that NGT Unload will read partitions 2 through 5 only and then evaluate the search criteria to select the applicable rows.

6.2.3.1.2 Example A2
UNLD FROM T1 WHERE (C1 > 1000 AND C1 < 5000 OR C2+C1=6000)
This will result in the following messages:

9 PARTS SELECTED.

All parts are selected because the predicate “C1+C2=6000” does not conform to the basic predicate rules and the condition is ORed with the analyzed predicates.

6.2.3.1.3 Example A3
UNLD FROM T1 WHERE (C1 > 9000)
This will result in the following message:

ALL PARTS EXCLUDED.

The reason that no parts were selected is because the TS is LARGE and the last partition does not address keys > 9000. If you attempt to insert into this table a key value greater than 9000, the SQL will result in a DB2 error.

6.2.3.1.4 Example A4
UNLD FROM T1 WHERE (C1 > 1000 AND C1 < 5000 OR C1 >= 8000 AND C1 < 9000)

This will result in the following messages:

6 PARTS SELECTED.
PARTS 2-5, 8-9

This means that NGT Unload will read parts 2-3,8-9 only, then will evaluate the search criteria after reading the row data to determine applicable rows.

6.2.3.1.5 Example A5

UNLD FROM T1 WHERE (C1 BETWEEN -100 AND 5000)

This will result in the following messages:

5 PARTS SELECTED.
PARTS 1-5

This means that UNLD will read the first 5 parts.

6.2.3.1.6 Example A6

UNLD FROM T1 WHERE (C1 NOT BETWEEN 3500 AND 7500)

This will result in the following messages:

6 PARTS SELECTED.
PARTS 1-4, 8-9

This means that NGT Unload will read parts 1,2,3,8,9.

6.2.3.1.7 Example A7

UNLD FROM T1 WHERE (C1 IN 1,5000,8000,10000))

This will result in the following messages:

3 PARTS SELECTED.
PARTS 1,5,8

This means that NGT Unload will read parts 1,5,8

6.2.3.2 B. Multi-column key

In this set of examples we shall show how the result differs from the previous examples when the clustering index indexes columns C1 and C2 with the same limitkey values as above.

Part 1 values(1000,0)
Part 2 values(2000,1)
Part 3 values(3000,2)
6.2.3.2.1 Example B1

UNLD FROM T1 WHERE (C1 > 1000 AND C1 <= 5000 AND C2+C1=6000

This will result in the following messages:

5 PARTS SELECTED.
PARTS 2-6

This differs from Example A1 in that partition 6 has been selected.

6.2.3.2.2 Example B2

UNLD FROM T1 WHERE (C1 BETWEEN -100 AND 5000)

This results in the following messages:

6 PARTS SELECTED.
PARTS 1-6

This means that UNLD will read the first 6 parts. This differs from Example A2 in that partition 6 has been selected.

6.2.3.2.3 Example B3

UNLD FROM T1 WHERE (C1 IN (1,5000,8000,10000))

This will result in the following messages:

5 PARTS SELECTED.
PARTS 1,5-6,8-9

This means that UNLD will read parts 1,5,6,8,9. This differs from Example A5 in that partitions 6 and 9 have been selected.

6.3 Unloading From Other Sources

The procedure for unloading from a copy of a table space, either an image copy, incremental copy or a DSN1COPY, is roughly the same as unloading from a table space.

To invoke this feature, change the unload statement to add one of the following keywords: COPYDDN, COPYDSN or SYSCOPY.

If you use COPYDDN you must change the JCL to add the ddname supplied with the COPYDDN keyword and have it point to the input dataset.
If you do not want to change your JCL, use the COPYDSN keyword and supply the full dataset name. Quotes around the dataset name are optional.

If you want to unload from the last posted copy in the DB2 catalog, then simply code SYSCOPY. With this keyword you can also specify that you want the last FULL or the last INCREMENTAL copy.

The attributes of the input dataset that describe an image copy or DSN1COPY must be RECFM=F or RECFM=FB and LRECL=4096. The input dataset is assumed to have valid DB2 page layouts. When unloading from a copy, control pages (header pages and spacemap pages) are counted, but are not used. Therefore, as long as the input dataset contains valid DB2 pages, they can be in any order or sequence that you like.

NOTE:
The PART option is not supported when unloading from an image copy using the COPYDDN, COPYDSN or SYSCOPY options. To select a single partition use the Where Clause.

6.4 Unloading From Dropped Tables

NGT Unload allows you to override the DB2 defined OBID value when qualifying rows for unloading. By default, NGT Unload uses the DB2-assigned value. This capability can make it easier to recover data if a table is dropped and you must unload from a copy of the table space.

With the alternate OBID feature of NGT Unload you can override the OBID specification that NGT Unload retrieves from the DB2 catalog. This allows you, for example, unload data from image copies or DSN1COPYS after a table has been dropped.

To use the feature you must do the following:
1. Code the OBID keyword in the UNLD statement.
2. Supply the OBID value in integer format (no hex). No range validation of the supplied OBID value is done.
3. Ensure that the table name in the unload statement FROM TABLE clause exists (although it does not have to contain rows). All NGT Unload needs is the column specification for the table.
4. Ensure that the columns that are being unloaded, (see note 1), for the table in the FROM TABLE clause, must be compatible (see note 2) with the table in the input COPY dataset.
5. If the table in the COPY has an EDITPROC, make sure that the DB2 defined table must have the same EDITPROC defined.

NOTE

If you use a value of 0, all OBIDs in the input records will be ignored. The value 0 is intended for use with single-table table spaces only. Using a value of 0 with a multi-table table spaces may produce unpredictable results or result in abends.

6.5 Saving Space with the DSNTIAUL Format

You can save output dataset space using the DSNTIAUL format by either using the NOPAD unload keyword or coding DCB=RECFM=VB in your JCL for the output dataset. This causes varying length fields to not be padded. The LOAD DATA statement produced, while
representative of the data in the file, may not be usable with other vendors’ LOAD utility products if null fields are present in the data and the nulls field(s) follow a varying length field.

6.6 Customizing the Generated LOAD DATA Statement

There is a facility for managing the creation of the LOAD DATA statement. It is accessed via the optional STMTDDN keyword in the UNLD statement. It applies to all output formats except ASCIIDEL.

6.6.1 The STMTDDN Keyword

When using the DSNTIAUL or REPORT formats the default ddname that receives the LOAD DATA statement is //SYSPUNCH. This can be changed with the STMTDDN() keyword of the unload statement. To change the DD name to, for example, //MYDD code the following unload statement:

```
//SYSIN DD *
   UNLD DATA STMTDDN(MYDD) FROM TABLE tablename
//MYDD DD SYSOUT=X
```

The //SYSPUNCH dataset can have a record format (RECFM) of F, FB, V, VB or U. LRECL must be at least 80 if fixed or undefined, or 84 if variable. The blocksize must fit the requirements of the RECFM/LRECL combination. If the blocksize that exists or that is coded in the JCL is invalid for the RECFM/LRECL combination, NGT Unload will override the blocksize and set a correct blocksize.

NOTE:
When multiple unloads are in the same SYSIN, either code individual //SYSPUNCH DD's and supply a different member name on each DD statement or code DISP=MOD or DISP=OLD. //SYSPUNCH may be allocated to DD DUMMY or SYSOUT, and it may be written to TAPE or DASD.

If the DD name specified is not allocated, no error message will be issued and the LOAD DATA statement will not be produced.

6.7 Server use by Unload

NGT Unload employs Servers to greatly increase parallelism. Depending on how Unload jobs are set up, the use of Servers can be restricted. For instance:

1. If unloading from image copy via COPYDDN, then the image copy dataset is allocated and read in the master job and is not eligible to be read by servers. This restriction does not exist when using COPYDSN.
2. Servers are used when there is no output DD statements defined in the master (subject to 1 above). In these cases NGT Automation dynamically names and allocates the unload dataset based on logic in CDBEXEC(XULDDYNM).
3. If the output DD is statement is present in the master and the UNLD statement has ORDERBY then the read phase can be done in server and the output phase is done in the master where the output dataset is allocated.
Chapter 7

ROWEXIT Interface

The ROWEXIT exit processor supports a feature in the NGT utilities that allows a specified user exit to be called from NGT Utilities. This exit processor is an assembler language API that makes available to a user exit each row that is processed by the NGT utility, on a row-by-row basis.

The exit processor supports the following calls: initialization, processing and termination. Each call has a specific set of parameters and returns specific values as outlined in this document. Upon returning from the exit processor call, two values are always examined: the return code value and a row disposition value, with the row disposition value being examined and acted upon first. The return code value indicates the success or failure of the API call. If the NGT exit processor call receives a return code value of 8 or higher, processing terminates after the handling of the current row. The row disposition value indicates the processing request to be performed by the API for the current row being handled.

The initialization call establishes a dialog session with the user exit and identifies to the user exit the characteristics of the rows being processed. The termination call ends a dialog session with the user exit. All processing calls handled between the initialization and termination calls belong to a single table in the current table being processed.

The process call passes the individual row to the user exit. The user exit examines the row and returns to the API its request for the disposition for that specific row. The returned request can be one of the following:

- Process the returned unmodified row, as the NGT utility would normally process the row for that specific utility.
- Process the returned modified row, as the NGT utility would normally process the row for that specific utility.
- Ignore or skip the row from the NGT Utility normal processing.

The user exit is allowed to alter the contents of the row, within the following limitations:

- The exit cannot change the row length if it is a fixed length row.
- The exit may change the length of a varying length row, but only the varying length fields/columns within that row, and again, cannot exceed maximum field/column lengths.
- The exit should use caution when altering field/column contents, as subsequent processing may or may not have data type expectations for the field/column value.

An 8-byte token area is defined in the parameter structure for each call. This token is reserved for use by the user exit. The token value is passed between the exit processor and the user exit. Once set, this value is not be modified by the exit processor. This provides the user exit with a modifiable storage area which remains unchanged across API calls.
7.1 Invoking The Process

The ROWEXIT module is constructed so that it could be easily called from NGT Unload Utility. This is done by specifying the ROWEXIT keyword on the NGT utility control statement. The ROWEXIT keyword value must specify the name of the user exit that is to be called from the utility. This user exit must exist in the load library concatenation referenced in the utility JCL. If the user exit specified for the ROWEXIT keyword in the utility control statement does not exist in the load library concatenation, the job fails with an error message ngtN038 indicating that the user exit is not available for processing.

An example of specifying the ROWEXIT keyword is shown below:

```
UNLD DATA FORMAT(DSNTIAUL) ROWEXIT(NODNAME)
FROM TABLE NGT.UNLOAD_TEST
```

In this sample unload command, the MODNAME represents the 8 character user exit module name found in the load library concatenation.

This user exit module receives the initialization, processing and termination calls from the NGT Utility's ROWEXIT module. The ROWEXIT module performs the necessary steps to pass the row information on to the user exit module. The user exit module performs its desired process for each call received from the NGT utility. For each call received, the user exit module also sets the desired return code value and disposition value. These values indicate to the ROWEXIT module what processing should be done once it receives control from the user exit routine. Upon returning from the user exit, these values are examined by the ROWEXIT module and are processed accordingly.

7.2 Functional Description

The exit is called authorized, in user key 8. An ESTAE is created prior to calling the exit for recovery purposes. Any ABEND within the user exit causes a termination of processing. In the event of an ABEND in the user exit, the row disposition code is not inspected. The user exit may establish its own recovery environment, but it should cancel any such recovery and/or retry routines prior to passing control back to the exit processor.

The ESTAE is established unconditionally for the initialization call. After the initialization call, the user exit has the ability to turn off the ESTAE processing through a field in the parameter structure. Turning off the ESTAE capability reduces the overhead of the processing for the table. However, in doing so, the ROWEXIT module is not able to intercept errors if the user exit abends. The user exit should be prepared to accept the parameter list and fields in storage that contain 31 bit addresses. There are no resource consumption restrictions (CPU limits) for the exit. The exit is allowed to connect to the DB2 SSID passed in the table row header block, and is responsible for properly maintaining and terminating its connection to DB2.

Register conventions on entry to the exit are:

- R0: Unpredictable
- R1: Pointer to a parameter list block
- R2 - R12: Unpredictable
- R13: Pointer to save area
- R14: Return address
- R15: Entry point address of user exit
On return from processing, the register conventions are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>R1</td>
<td>Pointer to parameter list block</td>
</tr>
<tr>
<td>R2 - R12</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>R13</td>
<td>Register save area</td>
</tr>
<tr>
<td>R14</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>R15</td>
<td>Return code as also found in CDBRETCODE in parameter list block</td>
</tr>
</tbody>
</table>

**NOTE:**
Those 8-byte fields are provided for the user token field and the pointer field (row description block or row pointer). This is being done to prepare for 64-bit architecture and 8-byte pointers. If a 31-bit environment is in use, only use the last 4 bytes of the field CDBROWPTR for addressability to either the table row header/row descriptor block information or the row pointer. The ROWEXIT processing supports 3 basic calls: initialization, processing, and termination. The input and output specifications are as follows:

### 7.2.1 Initialization Function Call

**Input Parameters**

Register 1 points to the initialization parameters defined as: (in the form of +offset(length))

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0(4)</td>
<td>Function code of 0 which identifies the initialization call.</td>
</tr>
<tr>
<td>+4(4)</td>
<td>Row disposition which is initialized to a value of -1.</td>
</tr>
<tr>
<td>+8(4)</td>
<td>Return code value which is initialized to a value of -1.</td>
</tr>
<tr>
<td>+12(4)</td>
<td>ESTAE request value which is initialized to a value of 1.</td>
</tr>
<tr>
<td>+16(8)</td>
<td>User token area, which can be set by the user exit.</td>
</tr>
<tr>
<td>+24(8)</td>
<td>Pointer which points to the structure identifying the format of the row as defined by the table being processed. This pointer refers to data that is mapped by the Table Row Header followed by the Row Descriptor Block structure, which is outlined in the Detail Design section of this document. This data structure identifies one and only one Table Row Header structure followed by multiple Row Descriptor Block structures. The number of Row Descriptor Block structures corresponds to the number of columns in the table being processed by the NGT Utility.</td>
</tr>
<tr>
<td>+32(82)</td>
<td>Message text area which is an 82-byte area that holds messages created by user exit.</td>
</tr>
<tr>
<td>+114(2)</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td>+116(4)</td>
<td>Reserved for future use. The high order byte of this value is set to X'80'.</td>
</tr>
</tbody>
</table>

**Return Values**

Register 1 points to the values upon return from the initialization call as defined below:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0(4)</td>
<td>Function code which is not inspected upon return.</td>
</tr>
<tr>
<td>+4(4)</td>
<td>Row disposition value. This value is not inspected upon return.</td>
</tr>
<tr>
<td>+8(4)</td>
<td>Return code value.</td>
</tr>
</tbody>
</table>
ROWEXIT Interface

0  Indicates a successful initialization process.
4  Indicates that utility processing should continue normally, without any further calls made to the user exit. The only call processed after this is the terminate call.
>4 Indicates an unsuccessful initialization call. Processing terminates for non-zero values. If the message text area contains a message, it is issued by the ROWEXIT module upon return from the user exit.

+12(4) ESTAE request value
0  Indicates that no ESTAE is established for subsequent ROWEXIT calls.
1  Indicates to continue establishing ESTAE for subsequent ROWEXIT calls.

+16(8) User token area may be set by the user exit.
+24(8) Pointer to the Table Row Header/Row Descriptor Block structure which is not inspected upon return.
+32(82) Message text area which is an 82 byte area that holds messages created by user exit. The first 2 bytes of this field represent the length of the message and does not include the 2 byte LL field. This message is automatically printed if the return code value is 8 or higher.

+114(2) Reserved for future use.
+116(4) Reserved for future use. The high order byte of this value is set to X'80'.

7.2.2 Processing Function Call

Input Parameters

Register 1 points to the processing parameters defined as:

+0(4) Function code of 4 which identifies the processing function call.
+4(4) Row disposition which is initialized to a value of -1.
+8(4) Return code value which is initialized to a value of -1.
+12(4) ESTAE request value, which is set as per previous ROWEXIT call.
+16(8) User token area which is set by the user exit and is not modified by the exit processor.
+24(8) Pointer to the current row being processed.
+32(82) Message text area which is an 82 byte area that holds messages created by user exit.
+114(2) Reserved for future use.
+116(4) Reserved for future use. The high order byte of this value is set to X'80'.

Return Values

Register 1 points to the values upon return from the process call as defined below:

+0(4) Function code of 4 which remains unchanged.
+4(4) Row disposition value.
### ROWEXIT Interface

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates that processing for the specific utility continues as normal. The unmodified row as returned from the user exit is processed by the NGT utility.</td>
</tr>
<tr>
<td>4</td>
<td>Indicates that the processing for the specific utility continues as normal, however, the exit has modified the row. This modified row is processed by the NGT utility as it normally would. The row that is presented to the user exit is the actual row in the output buffer. Upon return from the user exit, the NGT utility performs no validation on the value of the row. If the modifications made to the row by the user exit cause an overlay, results are unpredictable. Therefore, the user exit should modify the row with extreme caution.</td>
</tr>
<tr>
<td>8</td>
<td>Indicates that this row should be skipped and excluded from the utility's normal processing.</td>
</tr>
<tr>
<td>+8(4)</td>
<td>Return code value.</td>
</tr>
<tr>
<td>0</td>
<td>Indicates a successful processing call.</td>
</tr>
<tr>
<td>4</td>
<td>Indicates that utility processing should continue normally, without any further calls made to the user exit. The only call processed after this is the terminate call.</td>
</tr>
<tr>
<td>8</td>
<td>Indicates an unsuccessful processing call. Processing terminates for non-zero values. If the message text area contains a message, it is issued by the ROWEXIT module upon return from the user exit.</td>
</tr>
<tr>
<td>+12(4)</td>
<td>ESTAE request value</td>
</tr>
<tr>
<td>0</td>
<td>Indicates that no ESTAE is established for subsequent ROWEXIT calls.</td>
</tr>
<tr>
<td>1</td>
<td>Indicates to continue establishing ESTAE for subsequent ROWEXIT calls.</td>
</tr>
<tr>
<td>+16(8)</td>
<td>User token area which is set by the user exit and is not modified by the exit processor.</td>
</tr>
<tr>
<td>+24(8)</td>
<td>The address of the row that is currently being processed. If the row is to be modified, this value should be set to the address of the modified row. The first 2 bytes of this modified row must contain a length value that correctly reflects the modifications made to that row. The length of the modified row must not exceed the maximum row length, as identified in the row descriptor block referenced in the initialization call.</td>
</tr>
<tr>
<td>+32(82)</td>
<td>Message text area which is an 82 byte area that holds messages created by user exit. The first 2 bytes of this field represent the length of the message and does not include the 2 byte LL field. This message is automatically printed if the return code value is 8 or higher.</td>
</tr>
<tr>
<td>+114(2)</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td>+116(4)</td>
<td>Reserved for future use. The high order byte of this value is set to X'80'.</td>
</tr>
</tbody>
</table>
ROWEXIT Interface

7.2.3 Termination Function Call

Input Parameters

Register 1 points to the processing parameters defined as:

+0(4) Function code of 8 which identifies the termination function call.
+4(4) Row disposition which is a full word of zeros.
+8(4) Return code value which is a full word of zeros.
+12(4) ESTAE request value, which is set as per previous ROWEXIT call.
+16(8) User token area which is set by the user exit and is not modified by the exit processor.
+24(8) Value of the current row being processed which is set to a full word of zeros.
+32(82) Message text area which is an 82 byte area that holds messages created by user exit.
+114(2) Reserved for future use.
+116(4) Reserved for future use. The high order byte of this value is set to X'80'.

Return Values

Register 1 points to the values upon return from the process call as defined below:

+0(4) Function code of 8 which remains unchanged.
+4(4) Row disposition value which is not be examined for termination calls.
+8(4) Return code value.

0 Indicates a successful termination call.

>0 Indicates an unsuccessful termination call. Processing terminates immediately for all non-zero values. If the message text area contains a message, it is issued by the ROWEXIT module upon return from the user exit.

+12(4) ESTAE request value.
+16(8) User token area which is set by the user exit and is not modified by the exit processor.
+24(8) The address of the row that is currently being processed. This value is not examined.
+32(82) Message text area which is an 82 byte area that holds messages created by user exit. The first 2 bytes of this field represent the length of the message and does not include the 2 byte LL field. This message is automatically printed if the return code value is 8 or higher.
+114(2) Reserved for future use.
+116(4) Reserved for future use. The high order byte of this value is set to X'80'.
7.2.4 **Detailed Design**
This section describes the control blocks and data structures passed from the NGT ROWEXIT module to the user exit. The parameter structure outlined below is the structure to which register 1 points for each function call.

7.2.5 **Parameter Structure**
The following is the parameter structure used on all function calls. This is the format of the data that is contained in register 1 for the initialize, processing and termination calls.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>FUNCTION</td>
<td>This full word identifies the function request.</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>Identifies the initialize call.</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Identifies the process call.</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Identifies the termination call.</td>
</tr>
<tr>
<td>+4</td>
<td>ROW DISPOSITION</td>
<td>This full word identifies the function to be performed for the current row being processed. This value is examined only for processing calls and ignored for the initialization and termination calls.</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>Informs the utility to process the unmodified row as it normally would.</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Informs the utility that the row has been modified and to process the updated row as it normally would.</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Informs the utility to skip the processing for this row from its normal processing.</td>
</tr>
<tr>
<td>+8</td>
<td>RETURN CODE</td>
<td>This full word identifies the return code value as set by the user exit.</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>Indicates that processing should continue after the current row is handled.</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Indicates that utility processing should continue as it normally would without any further calls to the user exit.</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Indicates that processing should be terminated after the current row is handled.</td>
</tr>
<tr>
<td>+12</td>
<td>ESTAE REQUEST</td>
<td>This full word identifies the request from the user exit to continue or to discontinue the establishment of the ESTAE for subsequent user exit calls.</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>Indicates that no ESTAE is to be established for subsequent user exit calls.</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>Indicates that an ESTAE should be established for subsequent user exit calls. This is the default value for this field which is set prior to the initialization call.</td>
</tr>
<tr>
<td>+16</td>
<td>USER TOKEN</td>
<td>This double word can be used by the user exit for its processing. This value can be populated by the user exit and is not changed by the exit processor.</td>
</tr>
<tr>
<td>+24</td>
<td>ROW VALUE</td>
<td>For initialization calls, this double word points to the table row header / row descriptor block information. For processing calls, this value points to the row value itself. Upon return from the processing call from the user exit,</td>
</tr>
</tbody>
</table>
this must point to the modified row value if the user exit chooses to modify the row value.

+32 MESSAGE TEXT
This 82 byte field holds the message that is created by the user exit. This message is printed only if the return code value is 8 or higher. The first 2 bytes of this field represent the length of the message and does not include the 2 byte LL field.

+114 RESERVED1
Unused 2 bytes of storage at this time.

+116 RESERVED2
Unused 4 bytes of storage at this time. High order bit set to X'80'.

7.2.6 **Assembler DSECT**

The above outlines parameter structure is represented internally by the following DSECT:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARMLIST_DSECT</td>
<td></td>
</tr>
<tr>
<td>CDBFCODE DS 0D,A</td>
<td>Function Code</td>
</tr>
<tr>
<td>CDBROWDISP DS A</td>
<td>Row Disposition Code</td>
</tr>
<tr>
<td>CDBRETCODE DS A</td>
<td>Exit Return Code</td>
</tr>
<tr>
<td>CDBESTAE DS A</td>
<td>ESTAE Request Value</td>
</tr>
<tr>
<td>CDBUSERTOK DS D</td>
<td>Optional User Token</td>
</tr>
<tr>
<td>CDBROWPTR DS D</td>
<td>Pointer to Row Structure or Row</td>
</tr>
<tr>
<td>CDBMESSAGE DS XL82</td>
<td>User exit message</td>
</tr>
<tr>
<td>FILLER 1 DS XL2</td>
<td>Reserved area of 2 bytes</td>
</tr>
<tr>
<td>FILLER 2 DS 4</td>
<td>Reserved full word</td>
</tr>
</tbody>
</table>

7.2.7 **Table Row Header**

This table information precedes the row descriptor block information on the initialization call. These are the attributes of the table. The “table” defined here may be a physical table or a logical table. For instance, in the case of using this exit with NGT Unload, the row descriptor defines the output row. Since NGT Unload is able to create columns as part of the unload process, there is not a physical table column. However, the table still has attributes, such as a name and some number of columns, etc.

+0(4) Eyecatcher value CDBX.
+4(4) Version number of structure passed on initialization call. The initial value of this field is V100.
+8(4) Creator ID Pointer; Pointer to the creator ID of the table, which can be a maximum of 128 characters in length.
+12(4) Name Pointer; Pointer to the table name, which can be a maximum of 128 characters in length.
+16(4) ColCount; Number of columns in the row being defined.
+20(4) MaxLen; Maximum output row length.
+24(1) VarFlag; Y if row is varying length, N if row is fixed length.
+25(1) Unused1 Unused
+26(8) SSID DB2 subsystem or group attach name already connected to from the invoking process.
ROWEXIT Interface

+34(2) RDBLen; Row descriptor block length. This is the length of each repeating section for each field/column in the row.

+36(4) Unused.

7.2.8 Row Descriptor Block
The following is a breakdown of the row descriptor block (array) that is passed as an input parameter on the initialization call. This information is populated by the exit processor prior to the initialization call and is maintained throughout the life of the exit processing. This structure repeats for each field/column in the table row and it immediately follows the Table Row Header Block on the initialization function call.

+0 COLUMN NAME POINTER
4 byte pointer which holds the address of the name of this column. POINTER This name field is character field with a varying size of up to 128 characters. For derived fields (expressions, etc.) the names are in the form of "EXP1", "EXP2", etc.

+4 COLUMN LENGTH
Length value of the column; 4 byte field. For decimal values, in the form PPSS where PP = 2 byte precision and SS = 2 byte scale. For varying columns, the max length allowed (not including the 2 byte LL field).

+8 COLUMN TYPE
This 8 byte field identifies the column type. Value values are:

- BLOB LOB field
- CHAR character field
- CLOB LOB field
- DATE date field
- DBCLOB LOB field
- DECIMAL decimal field
- DISTINCT distinct type values
  Note: This is not supported at this time
- FLOAT floating numeric field
- GRAPHIC graphic field
- INTEGER integer field
- LONGVARG long variable graphic field
- LONGVAR real numeric field
- SMALLINT small integer field
- TIME time field
- TIMESTMP timestamp field
- VARCHAR variable character field
- VARG variable graphic field
- ROWID identity column
**ROWEXIT Interface**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+16 NULLABLE</td>
<td>This 1 byte field indicates whether or not the field can be NULL. Y indicates that this field can be NULL. N indicates that this field can not be NULL.</td>
</tr>
<tr>
<td>+17 NULL CHARACTER LENGTH</td>
<td>This is the length of the null field indicator, if any, 1 byte long.</td>
</tr>
<tr>
<td>+18 NULL CHARACTER</td>
<td>This field identifies the NULL character value for this column, up to a max of 4 characters long.</td>
</tr>
<tr>
<td>+22 VARYING TYPE</td>
<td>This 1 byte field indicates if the field is varying (V) or fixed (F).</td>
</tr>
<tr>
<td>+23 Unused</td>
<td>5 free bytes reserved for future use.</td>
</tr>
</tbody>
</table>
Index

/ //SYSPUNCH, 3-1, 6-8

+ +CHARFLDDLML, 2-20, 2-21, 3-7 +CONNECTALL, 2-22 +DATEFMT, 2-10, 2-23 +DEGREE, 2-24 +FIELDSEP, 2-26, 3-7 +MINROWS, 2-27 +NUMFLDDLML, 2-28, 3-7 +PASSWORD, 2-29 +TIMEFMT, 2-30

A

abend code 4095, 2-27 alternate OBI, 6-2 APPEND, 2-10 ARCHIVE, 2-8, 3-1, 3-6 ASCII, 2-12, 2-17 ASCIIDEL, 2-8, 2-9, 2-21, 2-23, 2-26, 2-28, 3-1, 3-2, 3-7 Authorizations, 1-1 AUTO, 2-12 automation exit, 2-7 automation exit processing, 5-1

B

BLANKPAD, 2-7 BYPART, 2-12

C

CASE construct, 6-1 Case Study, 3-2 CCSID, 2-12, 2-17 CDBEXEC, 2-2, 5-1 CHAR, 3-8 character field delimiter, 2-21 Character Field Delimiter, 2-21 CHARDEL, 3-7 CHARFLDDLML, 2-17

D

Cloned tables, 1-1 Coded Character Set ID, 2-17 COLDEL, 3-7 COPYDDN, 2-9, 2-12, 6-6, 6-7 COPYDSN, 2-9, 2-12, 6-6, 6-7

E

DATA, 2-7, 2-18 Date Format, 2-23, 2-24 DATEFMT, 2-10 DB2, 2-11 DB2 States, 1-1 DB2SQL, 2-12, 2-13 DBCS, 2-17 DD name, 2-7 DD Statements, 2-1 DECIMAL, 3-8 DECEPT, 3-7 Default Time Format, 2-30 DELIMITED, 3-2, 3-6 Diagram, 2-3 DIGITS, 2-11 DIRECT, 2-13 DISP, 2-9, 2-10 Double Byte Character Strings, 2-16 DSN1COPY, 6-6 DSNHDECP, 2-17 DSNTIAUL, 2-8, 2-23, 3-1, 3-2, 6-7, 6-8 DSNUM, 2-9 DUMMY, 6-8

F

EUR, 2-23, 2-30 EXTERNAL, 2-8, 3-1, 3-2, 3-9

Field Separator, 2-26 Field Specification, 3-8 FIELDFSEP, 2-17 FILSZ, 2-10 FLOAT, 3-8 FORMATT, 2-21, 3-6
Index

Format Delimiters, 2-17
  from table option, 2-4
FULL, 2-9

G
  GDG, 2-9
  Generated LOAD DATA Statements, 3-1
  global parameters, 2-1
  GRAPHIC, 3-8

H
  Hashed tables, 1-1

I
  INCR, 2-9
  INDDN, 2-2
  input, 2-1
  INTEGER, 3-8
  ISO, 2-23, 2-30

J
  JIS, 2-23, 2-30

K
  Keywords, 2-7, 2-18, 2-19

L
  LAST, 2-9
  LOAD DATA, 3-8
  LOAD DATA statement, 2-2, 3-1
  LOADINDD, 2-12
  LOADSTMT, 2-8
  LRECL, 6-7

M
  MAXERR, 2-10
  MAXLRECL, 2-10
  MAXROWS, 2-9, 4-1, 6-2
  MINROWS, 2-12
  MODE, 2-12
  mode option, 2-4
  MRF, 2-14

N
  NO, 2-29
  NOPAD, 2-11, 3-3, 3-4
  null, 2-21
  NULL, 3-9
  NULLCHAR, 2-12, 2-17
  NULLSTRING, 2-17

O
  OBID, 3-4
  order by option, 2-4

P
  PAD, 2-11
  Page Encoding, 2-16
  parameters, 2-20
  PART, 2-12
  PARTS, 2-12
  PDS, 2-9
  PIT, 2-7
  Point In Time, 2-7

R
  RBA, 2-9
  RECFM, 2-10, 3-3, 3-4, 6-7
  REPORT, 2-8, 2-23, 2-24, 3-2, 3-8, 6-8
  ROWEXIT, 2-11
  RW, 1-1

S
  SBCS, 2-16
  select option, 2-4
  Selecting Columns, 6-1
  Selecting Data To Unload, 6-1
  Selecting Rows, 6-2
  sequential datasets, 1-1
  SHRLEVEL CHANGE, 2-7
  Single Byte Character Strings, 2-16
  SMALLINT, 3-8
  ST, 2-30
  Statuses, 1-1
  STMTDDN, 2-8, 3-1, 6-8
  Summary, 1-1
  Syntax, 2-1, 2-3
  SYSCOPY, 2-9, 2-12, 6-6, 6-7
  SYSDM.SYSCOPY, 2-9
  SYSIN, 2-1
  SYSOUT, 6-8
  SYSPRINT, 2-2, 3-1, 3-8
  SYSREC, 2-2
  SYSREC00, 2-7

T
  TIME, 2-10
  TIMEFMT, 2-10
  TIMESTAMP, 2-11
  TSFMT, 2-11
Index

U
ULDPARMS, 2-1, 2-10, 2-20
UNICODE, 2-12, 2-17
UNLD, 2-12
UNLOAD, 2-8, 3-1, 3-4
Unload a specific number of rows, 4-1
Unload using a table lookup, 4-1
Unloading From Other Sources, 6-6, 6-7
USA, 2-23, 2-30
UTLPARMS, 2-1

V
VSAM, 2-9

W
WHERE clause, 2-12, 6-2
WK00001, 6-1
Work Dataset, 6-1

X
XSUT0000, 5-1
XSUTGLOB, 5-1
XSUTSYIN, 5-1
XSUTTERM, 5-1
XSVRXERR, 5-1
XULDDYNM, 2-7

Z
ZEROPAD, 2-7