1 Detail analysis on IBM DB2 z/OS's DSNTIAUL sample program

DSNTIAUL[1] is a IBM DB2 sample unload program. This program, which is written in assembler language, is a simple alternative to the UNLOAD utility. It unloads some or all rows from up to 100 DB2 tables. With DSNTIAUL, you can unload data of any DB2 built-in data type or distinct type. DSNTIAUL unloads the rows in a sequential file without row or column delimiters, which is compatible with the LOAD utility. It also generates utility control statements for the LOAD. DSNTIAUL also lets you execute any SQL non-SELECT statement that can be executed dynamically.

Other useful sample programs are ...

- DSNTIAD: A sample dynamic SQL program that is written in assembler language. With this program, you can execute any SQL statement that can be executed dynamically, except a SELECT statement.
- DSNTEP2: A sample dynamic SQL program that is written in the PL/I language. With this program, you can execute any SQL statement that can be executed dynamically. You can use the source version of DSNTEP2 and modify it to meet your needs, or, if you do not have a PL/I compiler at your installation, you can use the object code version of DSNTEP2.
- DSNTEP4: A sample dynamic SQL program that is written in the PL/I language. This program is identical to DSNTEP2 except DSNTEP4 uses multi-row fetch for increased performance. You can use the source version of DSNTEP4 and modify it to meet your needs, or, if you do not have a PL/I compiler at your installation, you can use the object code version of DSNTEP4.

Table of Contents:

[1] This description of IBM's DB2 sample program DSNTIAUL on DB2 for z/OS Version 10. These program, DB2 and z/OS are property of the IBM Corporation. Information can be found in the appropriate program product documentation.

1.1 How to build DSNTIAUL

DSNTIAUL is shipped only as source code, so you must precompile, assemble, link, and bind them before you can use them. An installation job is in data set DSN910.SDSNSAMP member DSNTEJ2A.

Default load library: DSNvv0.RUNLIB.LOAD(DSNTIAUL)
Default plan name: PLAN(DSNTIBvv) MEM(DSNTIAUL) (vv=version 81, 91, ...)

SQL code - 805 problems occur when running DSNTIAUL (or DSNTEP2 or DSNTEP4), if you omit binding the packages to any remote sites.

For importance of BIND parameter DBPROTOCOL, see section “Remote access using DSNTIAUL” on page 5.

1.2 Running DSNTIAUL

Sample JCL:

```
//UNLOAD  EXEC PGM=IKJEFT01
//*STEPLIB DD  DSN=DSNvv0.RUNLIB.LOAD,DISP=SHH    * OR USE LIB IN RUN
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
 DSN SYSTEM(ssid)
 RUN  PROGRAM(DSNTIAUL) PLAN(DSNTIBvv) -
       LIB('DSNvv0.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD DUMMY
//SYSREC00 DD DSN=MYUSERID.SYSREC00,
//            UNIT=SYSDA,SPACE=(TRK,(100,50),RLSE),DISP=(,CATLG,DELETE)
//SYSPUNCH DD DSN=MYUSERID.SYSPUNCH,
//            UNIT=SYSDA,SPACE=(TRK,(1,1)),DISP=(,CATLG,DELETE),
//            RECFM=FB,LRECL=80,BLKSIZE=0
//SYSIN    DD *
.PROJ WHERE DEPTNO='D01'
/*
```

DD-Names and datasets used:

- **SYSIN**: Input data set. You cannot enter comments in DSNTIAUL input. The record length for the input data set must be at least 72 bytes. DSNTIAUL reads only the first 72 bytes of each record.
- **SYSPRINT**: Output data set. DSNTIAUL writes informational and error messages in this data set. The record length for the SYSPRINT data set is 121 bytes.
- **SYSPUNCH**: Output data set. DSNTIAUL writes the LOAD utility control statements in this data set.
- **SYSRECnn**: Output data sets. The value nn ranges from 00 to 99. You can have a maximum of 100 output data sets for a single execution of DSNTIAUL. Each data set contains the data that is unloaded when DSNTIAUL processes a SELECT statement from the input data set. Therefore, the number of output data sets must match the number of SELECT statements (if you specify parameter SQL) or table specifications in your input data set.

Arguments to DSNTIAUL using PARM(‘parm,parm, ...’):

- **SQL**: Specify SQL to indicate that your input data set contains one or more complete SQL statements, each of which ends with a semicolon. You can include any SQL statement that can be executed dynamically in your input data set. In addition, you can include the static SQL statements CONNECT, SET CONNECTION, or RELEASE. DSNTIAUL uses the SELECT statements to determine which tables to unload and dynamically executes all other statements except CONNECT, SET CONNECTION, and RELEASE. DSNTIAUL executes CONNECT, SET CONNECTION, and RELEASE statically to connect to remote locations.
- ***number of rows per fetch*:** Specify a number from 1 to 32767 to specify the number of rows per fetch that DSNTIAUL retrieves. If you do not specify this number, DSNTIAUL retrieves 100 rows per fetch. This parameter can be specified with the SQL parameter.

Specify 1 to retrieve data from a remote site when DSNTIAUL is bound with the DBPROTOCOL(PRIVATE) option.

- **TOLWARN** Specify NO (the default) or YES to indicate whether DSNTIAUL continues to retrieve rows after receiving an SQL warning:
  - NO: If a warning occurs when DSNTIAUL executes an OPEN or FETCH to retrieve rows, DSNTIAUL stops retrieving rows. If the SQLWARN1, SQLWARN2, SQLWARN6, or SQLWARN7 flag is set when DSNTIAUL executes a FETCH to retrieve rows, DSNTIAUL continues to retrieve rows.
  - YES: If a warning occurs when DSNTIAUL executes an OPEN or FETCH to retrieve rows, DSNTIAUL continues to retrieve rows.
- **LOBFILE**(prefix): Specify LOBFILE to indicate that you want DSNTIAUL to dynamically allocate data sets, each to receive the full content of a LOB cell. (A LOB cell is the intersection of a row and a LOB column.) If you do not specify the LOBFILE option, you can unload up to only 32 KB of data from a LOB column.
  <prefix>.Q<i>.C<j>.R<k>, where:
  - <prefix> is the data set name prefix. <prefix> cannot exceed 26 characters.
  - Q<i> is the (<i>-1)th query processed by the current DSNTIAUL session, where <i> is in the range from 00 to 99.
  - C<j> is the (<j>-1)th column in the current SELECT statement, where <j> is in the range from 000 to 999.
  - R<k> is the (<k>-1)th row fetched for the current SELECT statement, where <k> is in the range from 0000000 to 9999999.

  See also section “New to DSNTIAUL” on page 4.

1.2.1 Input format

There are two possible input formats:

1) If no parameter value is specified on invocation of DSNTIAUL, a table to be unloaded is specified by entering one 72-byte line of the form ...
   `<table-name> where <predicate>.`
2) If the 'SQL' parameter value is specified on invocation of DSNTIAUL, input is in the form of complete SQL statements:
   - non-SELECT statements will be prepared and executed.
   - SELECT statements will be used to determine the tables and rows to be unloaded. If the select statement is not a full SELECT, the generated load statement will contain the string 'tblname' for the table name. if the statement is a full SELECT which contains a table join, only the first table name will appear in the load statement.
     In either case, the user will need to modify the LOAD statement before using it.

1.2.2 Output Format

The output format is a sequential file without row or column delimiters. This file consists of an unbroken sequence of fixed-length records. Record length is limited to 32760 bytes.

1.2.3 Performance

The default value for DSNTIAUL retrieves 100 rows per fetch. When you retrieve 1000 or 10000 rows, the CPU time stays almost the same. In most cases the value of 32767 will show the best performance. However, consumption depends on number of columns and type of selection.

For more information on how to improve performance, see section “Improving performance and functionality” on page 7.

1.3 Limits of DSNTIAUL

- Input SQL statment may have 2 MB in size.
- Output files record lengths:
  If the record length is not specified or invalid, DSNTIAUL will use the record length that it calculated.
  - SYSPUNCH: 80 <= LRECL <= 255
  - SYSRECxx: DSNTIAUL calculated size <= LRECL <= 32760

- A maximum of 100 tables can be unloaded. Only error information or "SUCCESSFUL" is output, with the number of rows unloaded.
- Data records are limited to 32760 bytes, including data, lengths for VARCHAR data, and space for null indicators.
- DSNTIAUL depends on the LONG VARCHAR and LONG VARGRAPHIC data types to identify candidates for output truncation. However, these data types will no longer be returned by DESCRIBE in DB2 Version 9.1 for z/OS after application of the fix for APAR PK44943. This could cause DSNTIAUL to end with return code 8 and the message: DSNT507I RECORD LENGTH NEEDED IS GREATER THAN THE MAXIMUM ALLOWED FOR TABLE table-name* when running on DB2 V9.
  See also section “New to DSNTIAUL” on page 4.

1.4 New to DSNTIAUL

1.4.1 ... with version 8

With DB2 V8, it has been enhanced to:
- Handle SQL statements up to 2 MB in size
- Use multi-row FETCH

You can specify an additional invocation parameter called "number of rows per fetch". It indicates the number of rows per fetch that DSNTIAUL is to retrieve. You can specify a number from 1 to 32,767.

With DB Version 9 DSNTIAUL have been modified to support the following new data types:
- BIGINT
- BINARY
- VARBINARY
- DECFLOAT
- XML

1.4.2 ... with version 9

Starting with DB2Version 9 DSNTIAUL also permits to use LOB file reference variables to extract a full LOB value to an external data set. The output data sets for unloaded LOB data are allocated dynamically. This is done using the LOB file reference SQL_FILE_CREATE option and are associated with a DD statement.
The data set that was created by DSNTIAUL is named with the convention

The new DSNTIAUL option LOBFILE is used for this:

```
RUN PROGRAM(DSNTIAUL) PLAN(DSNTIB91)
PARMS('SQL,2,LOBFILE(DSN910.DSNTIAUL)') -
LIB('DSN910.RUNLIB.LOAD')
```

If the DSNTIAUL option LOBFILE is not specified, then LOB data is handled as in previous releases. You can unload up to 32 KB of data from an LOB column.

The parameters in above example create LOB unload data sets with names that starting with:
`DSN910.DSNTIAUL.Q0000000.C0000000.R0000000`
The names end with:
`DSN910.DSNTIAUL.Q0000099.C0000999.R9999999`
The generated LOAD statement from the execution of DSNTIAUL contains the LOB file reference variables that can be used to load data from these dynamically allocated data sets.

1.4.3 ... with version 10

New restrictions: DSNTIAUL can no longer be used to process CREATE FUNCTION (SQL scalar) statements that would result in a package or CREATE TRIGGER statements. DSNTIAUL also cannot be used to process any other statement that contains SQL-routine-body. These statements are CREATE PROCEDURE (SQL external), CREATE PROCEDURE (SQL native), CREATE FUNCTION (SQL table), ALTER PROCEDURE (SQL native) with an ADD or REPLACE clause, and ALTER FUNCTION (SQL scalar) with an ADD or REPLACE clause.

1.5 Remote access using DSNTIAUL

Because these four programs also accept the static SQL statements CONNECT, SET CONNECTION, and RELEASE, you can use the programs to access DB2 tables at remote locations.

To retrieve data from a remote site by using the multi-row fetch capability for enhanced performance, bind DSNTIAUL with the DBPROTOCOL(DRDA) option. To run DSNTIAUL remotely when it is bound with the DBPROTOCOL(PRIVATE) option, switch DSNTIAUL to single-row fetch mode by specifying 1 for the number of rows per fetch parameter.

To access data on the remote server with an implicit connection you need to use 3-part names to reference tables on the remote server. This can also be achieved by the definition of an ALIAS pointing to 3 3-part name.

SQL code - 805 problems occur when running DSNTIAUL (or DSNTEP2 or DSNTEP4), if you omit binding the packages to any remote sites.

1.6 Using DSNTIAUL to extract DB2 data from non-OS/390 and z/OS platforms

DSNTIAUL is officially supported for use on DB2 for z/OS only. The restriction exists because DSNTIAUL is dependent on the DB2 for z/OS parser to locate the terminating semicolon of each SQL statement in the SYSIN DD input stream. When the DSNTIAUL connection changes to DB2 on an alternate platform, parsers there flag the terminating semicolon as an error (SQLCODE -104) and reject the SQL statement.

Nevertheless, there is an unofficial workaround that allows you to use DSNTIAUL to unload data from DB2s running on alternate platforms. Though it lacks the capability to process SQL, it is effective for simple unloads. Requirements vary according to whether you are using the DSNTIAUL provided in a release prior to DB2 UDB for z/OS Version 8:

(1) Your environment must be configured to connect from DB2 for z/OS to a remote server on an alternate platform. This document presumes that you can already do so.
(2) To bind a package for the DSNTIAUL provided with DB2 V8 or a subsequent release on a non-z/OS DB2, you need to add the SQLERROR(CONTINUE) option.
(3) You need to use 3-part names to reference tables on the remote server. This permits DB2 to access data on the remote server with an implicit connection.
(4) You need to run DSNTIAUL in so-called "non-SQL" mode. DSNTIAUL has two modes: SQL and non-SQL. SQL mode permits DSNTIAUL to interpret and process full SQL requests but, as indicated above, requires a connection to DB2 for z/OS. Non-SQL mode accepts a 1-, 2-, or 3-part name for a table to be unloaded and allows a WHERE clause for filtering. However: - When using the DSNTIAUL provided with DB2 V7 and earlier releases, the entire request must fit on a single 72-byte input record. - When using the DSNTIAUL provided with DB2 V8 or a subsequent release, the table name can wrap onto subsequent records but additional terms must fit onto the same record that the table name ends on. Regardless of the DSNTIAUL version, you cannot request table joins or specific column names in non-SQL mode. .
(5) To run the DSNTIAUL provided with DB2 V8 or a subsequent release, you also need to disable multi-row FETCH mode. Beginning in DB2 V8, DSNTIAUL has two cursors: The traditional one for single row FETCH and a new one for multi-row FETCH, which offers enhanced performance but which is not available on all DB2 platforms. Beginning with V8, in order to use DSNTIAUL on a platform that does not support multi- row FETCH compatible with DB2 for z/OS, you pass a rowfetch size of 1 via to DSN PARMS parameter, for example:

```
RUN PROGRAM(DSNTIAUL) PARMS('1')
```

Suppose you want to use DSNTIAUL to extract data from a DB2 server on Windows called MYDB2WIN (this procedure applies to other platforms as well as Windows). Bind the package and plan on DB2 for z/OS as follows:

```
DSN SYSTEM(V81A)
```

```
BIND PACKAGE(MYDB2WIN.DSNTIB81) MEM(DSNTIAUL) ACTION(REPLACE) -
      DBPROTOCOL(DRDA) ISOLATION(CS) VALIDATE(BIND) -
      SQLERROR(CONTINUE) **
BIND PACKAGE(DSNTIB81) MEM(DSNTIAUL) ACTION(REPLACE) -
      DBPROTOCOL(DRDA) ISOLATION(CS) VALIDATE(BIND)
BIND PLAN(DSNTIB81) PKLIST(*.DSNTIB81.DSNTIAUL) -
      ACTION(REPLACE) ISOLATION(CS) VALIDATE(BIND)
```

** The SQLERROR(CONTINUE) clause on the remote BIND PACKAGE statement is required only when binding the DSNTIAUL provided with DB2 V8 or a subsequent release.

You can then use three part names to unload from a remote system and using the non-SQL mode of DSNTIAUL. Here's the difference between SQL mode and non-SQL mode: SQL mode (works only when connected to a DB2 for z/OS server):

```
//UNLOAD EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
      DSN SYSTEM(V81A)
      RUN PROGRAM(DSNTIAUL) PLAN(DSNTIB81) PARMS('SQL') -
      LIB('USER.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSREC00 DD DSN=USRT003.G.DSN8UNLD.SYSREC00,
//SYSREC..
//SYSIN DD *
LOCK TABLE DSN8810.DEPT IN SHARE MODE;
SELECT * FROM DSN8810.DEPT WHERE DEPTNO = 'ABC';
SELECT * FROM DSN8810.VPHONE;
//*
```

Non-SQL mode (can be used on other DB2 platforms):

```
//UNLOAD EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(V81A)
      RUN PROGRAM(DSNTIAUL) PLAN(DSNTIB81) PARMS('1') -
      LIB('USER.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=* //SYSUDUMP DD SYSOUT=*
//SYSREC00 DD DSN=USRT003.G.DSN8UNLD.SYSREC00,
//SYSREC..
//SYSIN DD *
MYDB2WIN.SAMPLE.DEPT WHERE DEPTNO = 'ABC'
MYDB2WIN.SAMPLE.VPHONE
//*
```

Note that:

- The PARMS setting specifies '1' -> Do not use PARMS('1') with the DSNTIAUL provided with DB2 V7 and earlier releases
- The SELECT syntax is removed
- There are no terminating semicolons
- You cannot request particular columns, do joins, etc.
- You can specify a WHERE clause but:
  - For DSNTIAUL prior to V8, the entire "statement" has to fit on a single input record, between columns 1-72
  - For V8 DSNTIAUL and subsequent releases, the 3-part table name can wrap onto additional lines but additional terms must fit onto the same record that the table name ends on.
- You can't process the LOCK statement

Refer to the section entitled "Running DSNTIAUL" in Appendix D (Running DSNTIAUL, DSNTIAD, and DSNTEP2) of the DB2 for z/OS Utilities Guide and Reference for information about how to use DSNTIAUL without SQL mode.

1.7 Improving performance and functionality

The original DSNTIAUL BIND statement is supplied by IBM in installation job DSNTEJ2A on library SDSNSAMP.

```
BIND PLAN(DSNTIBÜÜ) MEM(DSNTIAUL) +
     CURRENTDATA(NO) ACT(REP) ISO(CS) ENCODING(EBCDIC) +
     LIB('DSNÜÜ0.DBRMLIB.DATA')
```

... and – after some work on customization – yields in ...

```
BIND PACKAGE(DSNTIAUL)       MEMBER(DSNTIAUL)+
     LIBRARY(lib-name')+
     OWNER(id)               QUALIFIER(id)          SQLERROR(NOPACKAGE)+
     DEFER(PREPARE)          VALIDATE(RUN)          ISOLATION(CS)+
     CURRENTDATA(NO)         DEGREE(1)              RELEASE(COMMIT)+
     EXPLAIN(NO)             DYNAMICRULES(RUN)      REOPT(ONCE)+
     KEEPDYNAMIC(NO)         DBPROTOCOL(DRDA)       IMMEDWRITE(NO)+
     ENCODING(273)           ROUNDING(HALFEVEN)+
     ENABLE(*)+
     ACTION(ADD|REPLACE)    [REPLVER(vvrr.PMnnnnn)]
END
BIND PLAN(DSNTIAUL)          OWNER(id)              QUALIFIER(id)+
     NODEFER(PREPARE)        VALIDATE(RUN)          ISOLATION(CS)+
     CACHESIZE(3072)         CURRENTDATA(NO)        DEGREE(1)+
     SQLRULES(DB2)           ACQUIRE(USE)           RELEASE(COMMIT)+
     EXPLAIN(NO)             REOPT(NONE)            KEEPDYNAMIC(NO)+
     IMMEDWRITE(NO)          DBPROTOCOL(DRDA)       ENCODING(273)+
     ROUNDING(HALFEVEN)      DISCONNECT(EXPLICIT)+
     PKLIST(DSNTIAUL.*)+
     ENABLE(*)+
     ACTION(ADD|REPLACE)    [RETAIN]
END
```

To improve access path for dynamic SQL statements you may want to apply the **REOPT** BIND option. Read how it works:

> The REOPT(ONCE) (V8) bind option tries to combine the benefits of REOPT(ALWAYS) and dynamic statement caching. The idea of REOPT(ONCE) is to re-optimize the access path only once (using the first set of input variable values) no matter how many times the same statement is executed.
>
> If you specify the new REOPT(AUTO) (V9) bind option, DB2 automatically determines whether a new access path is required to further optimize the performance of a statement for each execution. REOPT(AUTO) or REOPT(ONCE) only applies to dynamic statements that can be cached. If dynamic statement caching is turned off and DB2 executes a statement that is bound with REOPT(AUTO) or REOPT(ONCE), no reoptimization occurs.

REOPT specifies whether DB2 determines access paths at bind time and again at execution time for statements that contain:

- Input host variables
- Parameter markers
- Special registers

Your input to DSNTIAUL, DSNTEP2 or DSNTEP4 or usually does not contain input host variable or parameter markers, but less or more often special registers like CURRENT DATE, CURREN TIME and others. And these predicates and statements may tremendously profit from re-optimized acces path!

Some minor drawbacks using this technique:

- DB2's DSC (dynamic statement cache) must be active
- a large number of small and short-running SQL may raise re-optimization overhead
- DSNTEP2 ends with return code 4 if UPDATE or DELETE statemen occurs without WHERE condition. The PREPWARN NO option has no effect on that .
- EXPLAIN/PLAN_TABLE possibly shows different access path as chosen after re-optimization a run-time.

DEFER(PREPARE) and distributed processing:

Specify the bind option DEFER(PREPARE) to improve performance, instead of NODEFER (PREPARE), when binding dynamic or static SQL for DB2 private protocol access and when binding dynamic SQL for DRDA access. DB2 does not prepare the dynamic SQL statement until that statement executes. (The exception to this situation is dynamic SELECT, which combines PREPARE and DESCRIBE, regardless of whether the DEFER(PREPARE) option is in effect.) When a dynamic SQL statement accesses remote data, the PREPARE and EXECUTE statements can be transmitted over the network together and processed at the remote location. Responses to both statements can be sent back to the local subsystem together. This reduces network traffic, which improves the performance of the dynamic SQL statement.

To acccess data on remote servers you first have to BIND the DSNTIAUL, DSNTEP2 or DSNTEP4 package at the remote server. Then you must add the package and location to the package list of the plan. But you also may BIND the plan „at all locations": Just use „*" to tell DB2 to use the required package at every location connected to.

The BIND of package and plan finally looks like ...

```
BIND PACKAGE(collection)     MEMBER(DSNTIAUL)+
     LIBRARY(lib-name')+
     OWNER(id)               QUALIFIER(id)          SQLERROR(NOPACKAGE)+
     DEFER(PREPARE)          VALIDATE(RUN)          ISOLATION(CS)+
     CURRENTDATA(NO)         DEGREE(1)              EXPLAIN(NO)+
     REOPT(ONCE)             KEEPDYNAMIC(NO)        DBPROTOCOL(DRDA)+
     IMMEDWRITE(NO)          ENCODING(273)          ROUNDING(HALFEVEN)+
     ENABLE(*)+
     ACTION(ADD)
END
BIND PLAN(MYTIAUL)           OWNER(id)              QUALIFIER(id)+
     DEFER(PREPARE)          VALIDATE(RUN)          ISOLATION(CS)+
     CACHESIZE(3072)         CURRENTDATA(NO)        DEGREE(1)+
     SQLRULES(DB2)           ACQUIRE(USE)           RELEASE(COMMIT)+
     EXPLAIN(NO)             REOPT(ONCE)            KEEPDYNAMIC(NO)+
     IMMEDWRITE(NO)          DBPROTOCOL(DRDA)       ENCODING(273)+
     ROUNDING(HALFEVEN)      DISCONNECT(EXPLICIT)+
     PKLIST(*.collection.*)+
     ENABLE(*)+
     ACTION(ADD)
END
```

1.8 Error Processing and Error Messages

- Normal end = no errors were found in the source, and no errors occurred during processing.
- Ended with error(s) = errors were found in the source, or occurred during processing. Return Code:
  - 0 – Successful completion
  - 4 – SQL warning level errors detected.
    If TOLWARN(YES) is specified, and the warning occurred on a FETCH or OPEN during the processing of a SELECT statement, DB2 performs the unload operation.
    Otherwise if the SQL statement was a SELECT statement, DB2 did not perform the associated unload operation.
    If DB2 returns a +394, which indicates that it is using optimization hints, or a +395, which indicates one or more invalid optimization hints, DB2 performs the unload operation.

  - 8 – SQL error level errors detected.
    An SQL statement received an error code. If the SQL statement was a SELECT statement, DB2 did not perform the associated unload operation or did not complete it.
  - 12 - unable to open files.
    DSNTIAUL could not open a data set, an SQL statement returned a severe error code (-144, -302, -804, -805, -818, -902, -906, -911, -913, -922, -923, -924, or -927), or an error occurred in the SQL message formatting routine.

- DSNT490I SAMPLE DATA UNLOAD PROGRAM
  This is the header, indicating a normal start for this program.
  RETURN CODE = 0.
- DSNT491I TOO MANY TABLES ENTERED (MAXIMUM OF 100 TABLES)
  The maximum number of tables used by this program is 100. If you have more tables, split them into multiple runs.
  RETURN CODE = 8.
- DSNT492I SQL WARNING DURING SQL STATEMENT TYPE
  Type is PREPARE, OPEN, FETCH, or CLOSE
  RETURN CODE = 4.
- DSNT493I SQL ERROR DURING SQL STATEMENT TYPE
  Type is PREPARE, OPEN, FETCH, or CLOSE
  RETURN CODE = 8.
- DSNT494I ERROR DURING OPEN OF DDNAME SYSRECnn
  The DDNAME listed, where nn is filled in with a number from 00 to 99, could not be opened. See if you provided as many DD statements as table names.
  RETURN CODE = 12.
- DSNT495I SUCCESSFUL UNLOAD XXXXXXXX ROWS OF TABLE TTTTTTTT
  The unload was successful. xxxxxxxx is the number of rows unloaded. tttttttt is the name of the table or view from which it was unloaded.
  RETURN CODE = 0.
- DSNT496I UNRECOGNIZED DATA TYPE CODE OF NNNNN
  The prepare returned an invalid data type code. nnnnn is the code, printed in decimal. usually an error in this routine or a new data type.
  RETURN CODE = 12.
- DSNT497I RETURN CODE FROM MESSAGE ROUTINE DSNTIAR
  The message formatting routine detected an error. See that routine for return code information. Usually an error in this routine.
  RETURN CODE = 12.
- DSNT498I ERROR, NO VALID COLUMNS FOUND
  The prepare returned data which did not produce a valid output record. Usually an error in this routine.
  RETURN CODE = 12.
- DSNT499I ERROR IN DSNTIAUL INPUT PARAMETER LIST
  The parameter value was something other than 'SQL'.
  RETURN CODE = 12.
- DSNT502I ERROR IN DSNTIAUL SQL INPUT: statement
  An SQL error occurred when the input statement was prepared.
  RETURN CODE = 12.
- DSNT503I UNLOAD DATA SET ddname RECORD LENGTH SET TO nnnnn.
  Either an output data set record length was not specified, or the user-specified record length was not large enough for DSNTIAUL. DSNTIAUL set the record length to nnnnn.
  RETURN CODE = 0.
- DSNT504I UNLOAD DATA SET ddname BLOCKSIZE SET TO nnnnn.
  Either an output data blocksize length was not specified, or the user-specified blocksize was not large enough for DSNTIAUL. DSNTIAUL set the blocksize to nnnnn.
  RETURN CODE = 0.
- DSNT505I DSNTIAUL OPTIONS USED 'nnn'.
  You specified these options when you invoked dsntiaul. There is currently one valid option: 'SQL'.
  RETURN CODE = 0.
- DSNT506I INPUT STATEMENT WAS NOT SELECT * FROM A SINGLE TABLE. LOAD STATEMENT WILL NEED MODIFICATION.
  A SELECT statement used to unload a table contained either of the following: A SELECT clause

other than SELECT * FROM, a FROM clause that does not contain exactly one table name.
RETURN CODE = 4.

- DSNT507I DSNTIAUL RECORD LENGTH NEEDED IS GREATER THAN MAXIMUM ALLOWED FOR TABLE XXXXXXXX.
  The sum of the lengths of the fields into which columns of table xxxxxxxx will be unloaded is greater than the system maximum data set record length of 32760. The table cannot be unloaded.
  RETURN CODE = 12.
- ERROR MESSAGES FROM MODULE DSNTIAR
  When an error occurs, this module produces corresponding messages.

Sources: IBM Redbooks, official IBM DB2 for z/OS Guides, IBM Informational APAR II13592.