

JRH DB2I2 for DB2™ OS/390 and UDB™/zOS

Reference Manual

Version 10.0

05/24/2011

JRH GoldenState Software Inc.

<http://www.jrh-inc.com/>

© Copyright JRH GoldenState Software Inc. 1997-2014. All rights reserved.
DB2 is a registered trademark of the International Business Machine Corporation

First Edition (05/2011)
Revised on 05/23/2014

JRH Golden State Software, Inc.
29011 Golden Meadow Drive
Rancho Palos Verdes, CA 90275
U.S.A
1-310-544-1497

<http://www.jrh-inc.com/>

DB2 is a registered trademark of the International Business Machine Corporation

CONTENTS

CONTENTS	3
CHAPTER 1. INTRODUCTION.....	7
<i>Product Overview</i>	7
<i>DB2I2 Function Overview</i>	11
<i>DB2I2 Object Oriented Approach and Reusability Architect</i>	12
CHAPTER 2. USER SETUP.....	14
<i>How to invoke DB2I2</i>	14
<i>User Setup</i>	14
CHAPTER 3. DB2I2 COMMAND DETAIL DESCRIPTION	18
<i>How DB2I2 workbench works</i>	18
What is Line Objects	18
How to select line object	19
Line Objects Description	20
Global Line Object option	21
Execution Result from DB2I2 command	21
Global command options	22
Global Variable and Host Variable Usage	25
<i>DB2I2 Command Description Summary</i>	27
<i>DB2I2 Command Description Detail</i>	41
AL	46
ALTER	47
AUTH	49
BATCH	50
BIND	56
BIND COPY	58
CANCEL [DDF] THRAED	60
CHECK	61
COAUTH	63
CONNECT(location) & CONNECT(RESET)	64
COPY	66
COPYAUTH	70
CPY2CPY DB2 V7 or above only.....	72
CREATE	75
CURSORD	76
DB	77
DBAUTH	78
DBDSIZE	79
DB2CMD	80
DCLGEN	82
DDL	84
DELETE	86
DISPLAY	87
DSADJ	89
DSCOPY	91
DSNJU004	95
DSNTEP2	97
DSNTIAD	98
DSNTIAUL	99
DSN1COPY	101
DSN1LOGP	105
DSN1PRNT	108
DT	109

ED	110
LOAD cache-name	111
INSERTB, INSERTA, INSERT	111
DELETE	112
CLINE	112
CPOS [#].....	112
REXX	112
DB2I2	113
LOOP & LOOP_END	113
GETG	113
SETG	113
SETG2	114
TOP or BOTTOM	114
POSTMIGR	114
DISTINCT	114
NEWJOB	114
FGET	115
FPUT	115
MODEBTCH.....	115
MODERSTR.....	115
RESIZEIT	115
MYMENU	119
EDIT obsoleted and replaced by ED command	124
EXEC	126
EXPLAIN	128
EXPLAINP	132
EXPLORE* A Priced Add-on	135
FETCH	136
FLASH	137
FREE	141
FTP** a price add-on with RXdb2i2	142
FU	146
GENVCAT	147
GRANT	149
HELP or ?.....	151
HELPLO	161
HMIGRATE	162
HRECALL	163
IMPACT	164
INFO	165
INSERT	166
IP	168
IPC	169
IS	170
ISC	171
ISP	172
ISPC	173
IX	174
IXC	175
JOBCARD	176
LISTC	177
LISTDEF DB2 v7 or above.....	180
LOAD	182
MAIL** a price add-on with RXdb2i2	187
MIGR	190
MODIFY	195

MS	197
MT	198
OI	199
OPTIONS	200
PACKIT	201
PARMUTIL	203
PG	205
PGAUTH	206
PL	207
PLAUTH	208
PM	209
QBUILD	210
QUIESCE	212
RBA	213
REBIND	215
REBUILD	217
RECOVER	219
REORG	223
REORGCHK	227
REPAIR	229
REPORT	231
RESETG	234
REVOKE	235
REXX	237
RSAUTH	239
RTAUTH	240
RUN	241
RUNSTATS	244
RXDB2I2** A pricing add-on for JRH-DB2I2	246
SDSF	250
SELECT	252
SELPATHU	253
SELPATHV	254
SETG	255
SETRBA	256
SHAUTH	257
SNAPSHOT	258
SPACE	260
SPACEADJ	264
SQ	269
SQAUTH	270
SSID(db2 sub-system ID)	271
START	272
STATS	273
STATUS	276
STOP	277
STOSPCE	278
SUPERC	279
SUPERCS	280
SUPPORT	281
SY	282
SYSIBM	283
TAG	284
TB	286
TBAUTH	287
TBC	288

TERM	291
TOKENSCN	292
TP	294
TPC	295
TR	296
TRACK** a price added on with RXDB2I2	297
TRAUTH	302
TS	303
TSC	304
TSIX	305
TSO	306
FCPY	306
FCPY2	306
TWAIT	306
CAPTBUFR	307
READBUFR	307
DB2I2LOG	307
DB2I2TRK	308
P000710	308
JOBGEN	309
TRANSFRM	310
TO72	315
TSSET	318
UCASE	319
UNLOAD	320
UPDATE	324
USAUTH	326
VIEWG	327
VIRTUAL	328
VTOC	330
ZPARM	332
<i>User Defined Function (UDF)</i>	334
ACCCOMP	334
ACCCOMPR	336
GENURLD	337
OBJCOMP	339
REMOVEID	341
TUNEPG	342
TUNETB	345
<i>Sample Script</i>	347
COPYSTAT	347
DSCOPYB, DSCOPYBS	349
DSCOPY2, DSCOPY2S	352
GENSCCMD	355
MIGRTB	356
MONBUFR	358
UNLDRELD	359
<i>USER DEFINED QUERY</i>	361
GENAI	361
GENAT	362
GENSC	363
GENUNLDQ	364
GENUNLD1	365
GENXC	366
OBIDQ	367
<i>REAL TIME STATISTICS RELATED QUERIES (V8+)</i>	370

COPYFIX, COPYFIX9 and COPYFIXS	370
COPYFTS, COPYFTS9 and COPYFTSS	373
COPYITS, COPYITS9 and COPYITSS	376
REORGIX, REORGIX9 and REORGIXS	378
REORGTS, REORGTS9 and REORGTS	382
RUNSIX , RUNSIX9 and RUNSIXS	385
RUNSTS, RUNSTS9 and RUNSTSS	388
<i>Special Case Study</i>	391
Case Study 1: Database Migration Script Generation	391
Case Study 2: DB2 Dataset Extents Removal	392
Case Study 3: Work Load Balancing	393
Case Study 4: DB2 Table Migration	394
Case Study 5: DB2 Package Tuning	397
Case Study 6: DB2 Table Tuning	400
Case Study 7: DB2 Object Comparison	403
Case Study 8: DB2 Bufferpool Capturing and Reporting	405
Case Study 9: Access Path Comparison	406
Case Study 10: User ID or System Admin ID removal	408
Case Study 11: Copy DB2 Catalog Statistics	409
Case Study 12: Callable Interface - MCCLI	411
Case Study 13: Callable Interface - UTILCLI	412
Case Study 14: Previous Point of Time Recovery	415
Case Study 15: Disaster Recovery	430
Case Study 16: Unload and Reload	432
Case Study 17: CODEGEN with ED macro	437
<i>DB2I2 Line Object to DB2I2 Command Cross Reference</i>	447
<i>DB2I2 Command to Line Objects Cross Reference</i>	453
INDEX	458

Chapter 1. Introduction

Product Overview

The JRH DB2I2 for OS/390 & zOS (DB2I2) is an integrated DB2 productivity development/administration tool, which provides DB2 application developer as well as DB2 database administrator with a quick and easy interface to the DB2 catalog.

With its unique reusable architect, you can easily use and reuse the information retrieved from DB2 catalog to help you to either solve your DB2/OS390 related problems or help you to ease your day to day repetitive and routine DB2 related tasks.

The following list briefly describes the functions DB2I2 provides:

- Copy and Backup management
- Recovery management
- Reorganization management
- DB2 Package/Plan management

DB2I2 Reference Manual

- DBRM to Package migration management
- DBRM or Package consistent token check
- Package Version control management
- Security and Authorization management
- DASD Space management
- Migration management
- Data Move management

The following is the highlight of the product:

- DB2I2 provides simple keystroke navigation through DB2 catalog, so that you, as a DB2 application developer or administrator, do not need to memorize where the information is and how to get them.
- DB2I2 utilizes fully ISPF edit session interface for information requested as well as for the results returned. This feature allows you to get most of your work done in one place, your DB2I2 workbench edit session.
- DB2I2 supports remote connect and remote access, which allows you to manage your DB2 objects from remote locations. For example, you can use DB2I2 DSCOPY command to copy a DB2 data set from one DB2 location to another DB2 location.
- The DB2I2 workbench contains numerous performance enhancement functions, which can help you to identify potential problems of your application or DB2 environment. For example,
 - You can use DB2I2 STATS command to display DB2 catalog statistics for selected DB2 object. The results from the STATS command contain wealthy information, which include whether the selected table space ever has a image copy, whether you have ever gather the statistics for the object, as well as CLUSTER RATIO, FAROFF, NEAROFF and LEAFDIST information.
 - You can use DB2I2 LISTC command to display the underlined physical VASM data set statistics information, such as volume information, extent information, HI-ALLOCATION, as well as total space usage. By Specifying EXT command option, you can limit the LISTC output to only those data set which has extents greater than the EXT specified. The output from LISTC can then be used as input to DB2I2 DSADJ command to generate necessary steps, such as ALTER TABLESPACE or ALTER INDEX to adjust data set allocation to remove those extents.
 - To help you to analysis the space requirement, DB2I2 SPACE command provides you with an estimation screen, which allows you to change space allocation parameters and recalculate space requirement for different parameter changes. For multiple Table and indexes selection, a SPACE summary report display total space requirement.
 - To further help you to manage the space requirement changes, Db2I2 SPACEADJ command can be used to produce line objects, which can be used with DB2I2 DSADJ command to produce DB2 ALTER and IDCAMS steps to carry out the space requirement changes.
 - You can use RCHK command option together with REORG command to invoke REORG CHECK function to decide whether the selected DB2 line objects require reorganization.
- As for the functions related to application developer, DB2I2 workbench provides numerous application development assists, which help application developer to get their day to day routine done faster and easier:
 - You can use DB2I2 SELECT, INSERT, UPDATE, DELETE, CURSORD or FETCH commands to generate dynamic or statistic SQL statements. These SQL statements can be embedded in your COBOL program or being executed with a DB2I2 RUN or EXEC command dynamically.

- You can use DB2I2 EXPLAIN command to dynamically explain SQL statements, or a DECLARE CURSOR statement from drill down of a DB2 PLAN or PACKAGE.
- DB2I2 EXPLAINP command can be used to display information stored in your PLAN_TABLE. By specifying GN-generation option, you can display from your PLAN_TABLE multiple generations of the same program. The output from two different generations can then be compared through DB2I2 SUPERC interface.
- You can use RUN DB2I2 command to dynamically execute a set of SELECT SQL statements directly from your workbench edit session. To help you reuse the query you created, DB2I2 supports host variable substitution for the RUN command. You can specify host variables in your SQL SELECT statements, and dynamically substitute those host variables during the execution of RUN command. By doing so, the query you created and saved can be reused.
- DB2I2 EXEC command allows you to dynamically execute DB2 commands; IDCAM commands, as well as NON-SELECT SQL statements, such as INSERT, UPDATE, DELETE, CREATE, DROP, GRANT and REVOKE directly from your workbench edit session.
- The DB2I2 BATCH command generates BATCH JCL, which allows you to execute most of the DB2I2 commands in batch mode (Check detail description of each DB2I2 command to see if it is supported in batch mode). This feature together with IDD-input DDNAME, IDSN-input DSNAME, ODSN-output DSNAME, RUN exit, as well as EDIT and END_EDIT ISPF edit exit, REXX exit, and TSO exit, gives you the ability to prepare the input for a DB2I2 command and manipulate the output from a DB2I2 command. All within one DB2I2 Batch execution.
- The DB2I2 utility interface, such as REORG, COPY, RECOVER, REBUILD, REPAIR, RUNSTATS, REPORT, MODIFY, gives you the abilities to create related DB2 utility JCL with all the work space allocation calculated. With DB2I2 PARMUTIL command, you can prepare these utility parameters in advance, invoke and reuse these parameters in batch mode to create these related DB2 utility jobs. This interface allows you as a database administrator to get your utility job prepared with minimum effort.
- DB2I2 supports global variable substitution. You use DB2I2 SETG – set global, command to define a set of global variables. Once they have been defined, you then can use these global variables within your DB2I2 command. Global variable substitution allows you to reuse pre-defined commands.
- DB2I2 supports User Defined Function (UDF) in batch mode. There are two types of UDF: System UDF and user UDF. UDF is a set of DB2I2 commands grouped together to serve as one function. The following is a list of system UDF shipped with DB2I2:
 - ACCCOMP: A set of DB2I2 commands to compare the content of two different generations of the same program from the same DB2 PLAN_TABLE.
 - ACCCOMPR: A set of DB2I2 commands to compare the content of current generation of the same program from two different PLAN_TABLE of different locations.
 - OBJCOMP: A set of DB2I2 commands to compare DB2 objects.
 - REMOVEID: A set of DB2I2 commands to remove a user ID from your DB2 sub-system.
 - TUNEPG: A set of DB2I2 commands to tune DB2 packages.
 - TUNETB: A set of DB2I2 commands to tune DB2 tables.

To invoke these system UDF, you use DB2I2 BATCH command together with ICMD=*udfname command option. Host variables, if exists in UDF, can be specified here to substitute those host variable with desired information.

To create and access your own UDF, you specify and save a set of DB2I2 commands in your own file. Once you have done that, you can invoke it with DB2I2 BATCH command together with ICMD='your.save.udf.dsname' command option.

- DB2I2 provides sample scripts, which can be accessed just like UDF except there are no host variable substitute. To invoke these scripts, you used DB2I2 BATCH command together with ICMD=*scriptnm

command option:

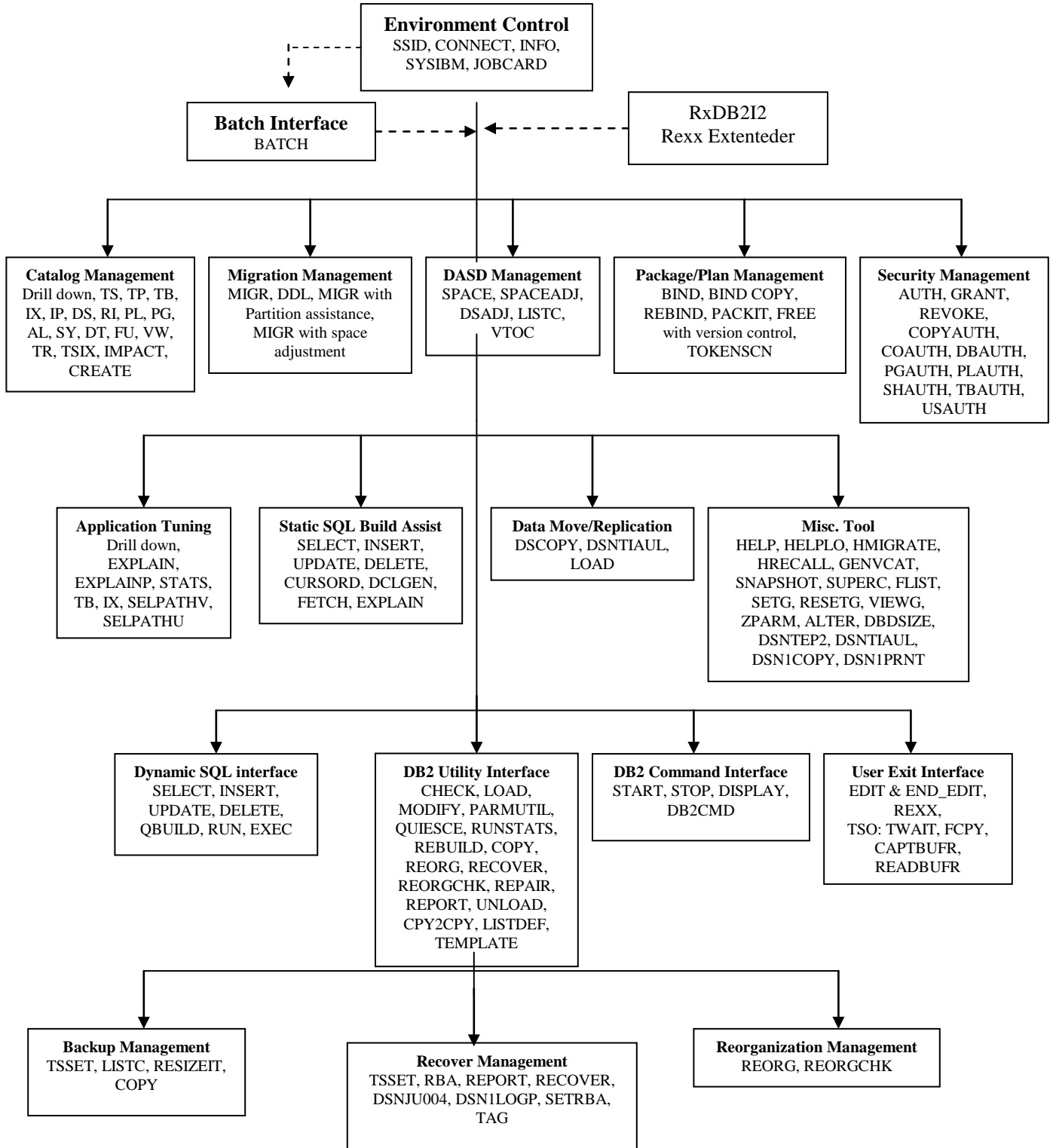
- **COPYSTATS:** A set of DB2I2 commands to copy DB2 catalog table statistics from one DB2 location to Another DB2 location.
 - **MIGRTB:** A set of DB2I2 commands to help you to migrate DB2 tables.
 - **MONBUFR:** A set of DB2I2 commands which capture DB2 Buffer Pool information and report them.
-
- DB2I2 supports User Defined Query (UDQ). UDQ is a set SELECT SQL statements, prepared in advance, and saved in a file to be reused at later time. There are two types of UDQ, system UDQ and your own UDQ. The following is a list of system UDQ shipped with DB2I2:
 - **GENAI:** A set SQL used with DB2I2 RUN command to generate AI lines – adjust index part line. The generated AI lines can be used as input to DB2I2 SPACEADJ command.
 - **GENAT:** A set SQL used with DB2I2 RUN command to generate AT lines - adjust table part line. The generated AT lines can be used as input to DB2I2 SPACEADJ command.

To invoke these system UDQ, such as GENAI or GENAT listed above, you specify

```
RUN IDSN=*GENAI &IXC='ixcreator' &IXN='ixname' or  
RUN IDSN=*GENAT &DB='dbname' &TS='tsname' &COMPRATIO=%compressed
```

You can create your own UDQ by specify a set of SELECT SQL statements, save in a file, and invoke it with IDSN='your.save.udq.dsname'.

DB2I2 Function Overview

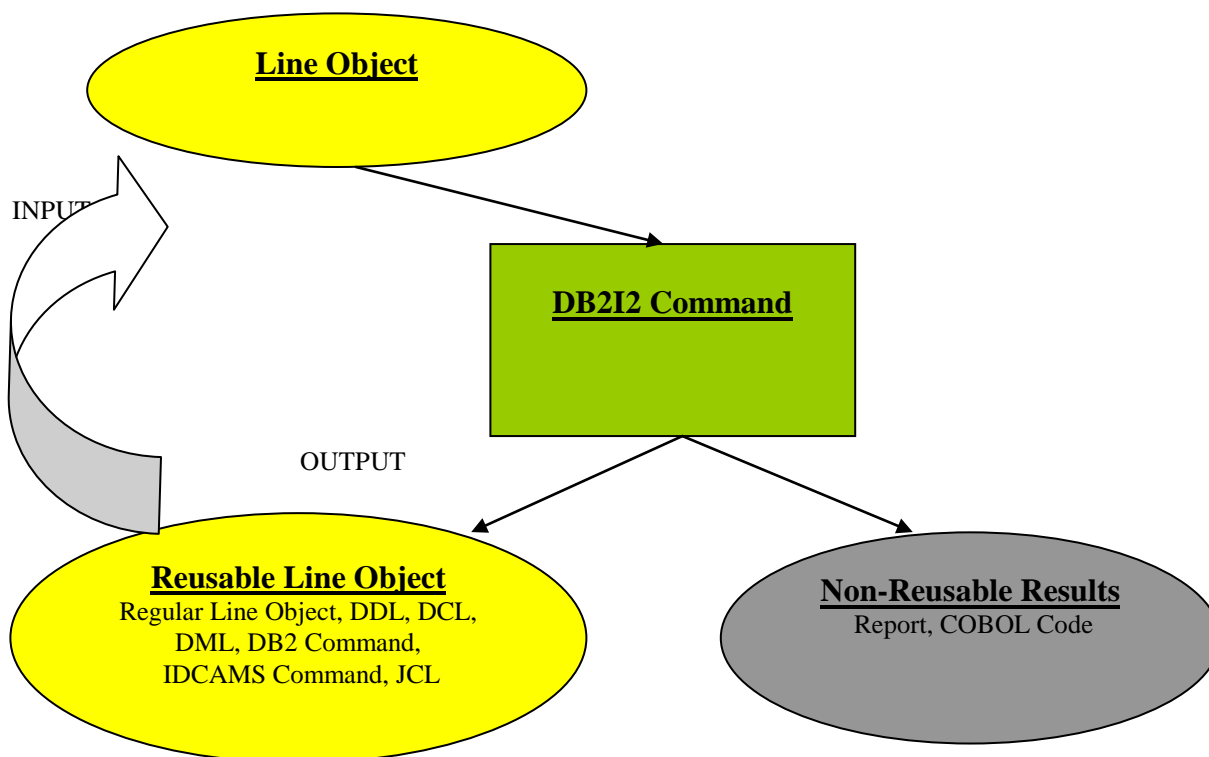


DB2I2 Object Oriented Approach and Reusability Architect

DB2I2 is built based on the concept of **objects oriented approach**. The DB2 object you ask DB2I2 to act on is defined as Line Object, and the actions you want DB2I2 to perform is defined as DB2I2 Command. By specifying a line object (a DB2 object) and a DB2I2 command, you ask DB2I2 to perform specific DB2I2 command against the selected DB2 objects.

For example, you can use LOAD DB2I2 command and a TB table line object to ask DB2 to generate DB2 utility JCL to load a specific DB2 table.

DB2I2 is built based on the concept of **Reusability**. With IDSN and ODSN command options, the information produced from one DB2I2 command can be easily Reused as input to the next DB2I2 command.



¹. Only Line object Output Type can be Reusable

Not all line objects produced from DB2I2 command are reusable. The following is a list of reusable and non-reusable line objects:

Reusable Line Object

Regular Line Object: The first 2 position of a line with line object abbreviation followed by object name (For example TB SYSIBM.SYSTABLE)

DDL: Data Definition Language. (For example. ALTER, DROP)

DCL: Data Control Language. (For example. GRANT, REVOKE)

DB2 Command: DB2 command prefixed with DB2CMD on the first 6 position (For example. BIND, REBIND, FREE, DCLGEN)

IDCAMS command: IDCAMS command prefixed with IDCAMS on the first 6 position.

SQL: Line(s) contain INSERT, UPDATE, DELETE or SELECT SQL spanned multiple lines.
Use DB2I2 EXEC command to process INSERT, UPDATE and DELETE SQL lines.
Use DB2I2 RUN command to process SELECT SQL lines.

JCL: JCL output produced by various DB2I2 commands to run DB2 Utilities.

Non-Reusable Line Object

Report: Report or screen output produced from various DB2I2 commands.

Cobol source: Generated Cobol source from various DB2I2 commands. (CURSOR, SELECT, INSERT, UPDATE, DELETE) This Cobol source can be modified and copied back to your COBOL source program.

Chapter 2. User Setup

How to invoke DB2I2

Issue the following TSO command to start your DB2I2 online session:

```
TSO EX 'db2i2.clist.library*(DB2I2STA)'
```

*Check with your DB2I2 system administration personnel to find out the name of the db2i2.clist.library.

User Setup

After you invoke the startup routine, DB2I2 displays the following logon screen so that you can enter DB2I2 workbench data set information. The data set name you entered here is used to hold all the works for your DB2I2 session. This is where you can group your works by specifying each data set for specific project or functions. If the data set name you entered here does not exist, DB2I2 creates it with the name you specified. The default data set name is DB2I2.WKBENCH. You can create as many as DB2I2 workbench data sets as you like to fit your own tracking and usage purpose. For example, you can create one for Payroll database and call it WORK.BENCH.PAYROLL and one for AP database call WORK.BENCH.AP.

```
#LOGON

          D B 2 I 2
for DB2/OS390 and UDB/zOS
  L O G O N   S C R E E N
          Version 10.0
Release Date: 02/01/2011

Licensed to EVALUATION COPY

JRH GoldenState Software  Inc.
Web Site:  www.db2i2.com
(c) CopyRighted 1997-2011

----- Please enter your DB2I2 Work Bench Dataset Name below -----
Dataset:   DB2I2.MENU_____
Your Email: jdoong@jrh-inc.com_____

Enter=Start DB2I2   F3=Exit
```

If this is the first time you logon to DB2I2, The following setup screen is displayed, which assists you to setup your user environment. You must follow onscreen instruction and complete this three-step setup before you can use DB2I2 properly.

```
Command ==> command [command options]2                               Scroll ==> CSR
***** ***** Top of Data *****
==MSG> /* ----- DB2I2 For DB2/OS390 & UDB/zOS ----- */
==MSG> /*   By JRH GoldenState Software, Inc. (c) Copyrighted 1997-2011   */
==MSG> /*                               Licensed to EVALUATION COPY                               */
==MSG> /*   SSID(DB8G) CONNECT( ) SYSIBM(SYSIBM) UCASE(ON)                               */
==MSG> /* -----*/
=NOTE= ** Please use 1. ISPF KEYS command to setup Hotkey for DB2I2
=NOTE= **           F4 key is a good candidate to assign to DB2I2
=NOTE= **           2. DB2I2 command SSID to setup DB2 Sub-System ID
=NOTE= **           3. DB2I2 command JOBCARD to setup Jobcard
=NOTE= **** Use DB2I2 command HELP to display any DB2I2 command Syntax
..... line object3
```

² Command is where you can enter online DB2I2 command. The online DB2I2 commands are add-in edit macros, which you can use together with all your other ISPF edit commands, as long as you are in the DB2I2 workbench environment.

³ Line object is where you enter the line objects. The completed list of all the line objects listed in the next chapter. The combination of the command and the line object direct DB2I2 what the action it will take.

Follow these setup instructions to complete your user session setup:

1. Use ISPF **KEYS** command to setup a Hotkey PF key for DB2I2. The F4 is a good candidate key.

```
----- Keylist Utility -----
File
-----
Private          ISR Keylist ISRSPEC Change          Row 1 to 12 of 24
Command ==>>>          Scroll ==>> PAGE
Make changes and then select File action bar.

Keylist Help Panel Name . . . ISRSPECH

Key      Definition          Format  Label
F1 . . . HELP              SHORT  Help
F2 . . . SPLIT             LONG   Split
F3 . . . EXIT              SHORT  Exit
F4 . . . DB2I2             LONG   DB2I2
F5 . . . RFIND             SHORT  Rfind
F6 . . . RCHANGE           SHORT  Rchange
F7 . . . UP                LONG   Up
F8 . . . DOWN             LONG   Down
F9 . . . SWAP             LONG   Swap
F10 . . LEFT              LONG   Left
F11 . . RIGHT            LONG   Right
F12 . . retrieve         SHORT  retrieve
```

Once you have set up your Hotkey to be used for DB2I2 command, you can simply press Hotkey to execute any DB2I2 command listed in Chapter 3.

2. Issues DB2I2 **SSID**(db2 sub-system id) command to attach your workbench to the specified DB2 sub-system. The db2i2 sub-system id specified here must be previous defined in db2i2.clist.library*(SSID) by your DB2I2 system administrator.
3. Issues DB2I2 **JOB CARD** command to setup Job cards information. These job cards information are used for any JCL generated from DB2I2 workbench.

Once you have done this 3-step user setup, you are ready to experience the exciting DB2I2 workbench for DB2/OS390.

Chapter 3. DB2I2 Command Detail Description

How DB2I2 workbench works

DB2I2 workbench provides both online interactive mode and batch mode access which allows you to access DB2 catalog easily and provide you with assistance to help you manage your DB2 application and DB2 system environment.

In online mode, DB2I2 workbench is a superset of ISPF edit macro command. It works just like any of the other ISPF command. To direct DB2I2 to perform a task, you need to do the following in sequence:

- Enter line object(s) in the edit lines
- Enter DB2I2 command on command line
- Select line object(s) and press HOTKEY (Use ISPF KEYS command to define one of the FKEY to DB2I2 as HOTKEY) to execute the DB2I2 command.

```

EDIT          JD.DB2I2.WKBENCH
Command ==>  DB2I2 command [command options]
=COLS> -----1-----2-----3---
SS' ' ' ' ' line object
SS' ' ' ' ' line object

```

To process DB2I2 command in batch mode, you can use DB2I2 BATCH command to produce a set of JCL, which allows you to run the multiple DB2I2 commands in sequence without tie up your work station.

The major difference between the DB2I2 command running in online mode verse running in batch mode are:

- In online mode, you can only execute **one DB2I2 command at a time**. The results of the online execution is usually returned within the workbench, so that you can view the result and make your next request.
- In batch mode, you can stream line **many DB2I2 commands in one batch execution**. The output result from one command is usually saved with an ODSN command option and used it as input to the next DB2I2 command in sequence. This feature allows you to streamline many DB2I2 commands in sequence in one batch execution run. To document your DB2I2 command, you can use ‘—’ on column 1 and 2. These command lines are treated as comment lines and are ignored by DB2I2 batch process engine.

What is Line Objects

- Line object consists of two parts: **line object abbreviation** and **line object detail** separated by a blank character.
- Once you have logon to DB2I2, you can enter line objects anywhere on your edit line field. However, to be recognized by DB2I2 as a valid line object line, **you can not enter anything before line object abbreviation**.
- Line object detail can be a fully qualified name, for example Q.PROFILES, or a name with wild card (%) for example, Q.%. DB2I2 supports wild card name, which allows you to process a group of line objects by specifying single line object detail specification. For example, a TB Q.% represents all DB2 table with Q as table creator.
- ‘—’ On column 1 and 2 represents a comment line.
- ‘—’ On any position in SQL block for DB2I2 RUN command indicates the rest of the line from ‘—’ position is treated as comment.

How to select line object

DB2I2 provides various ways to select line objects to be processed by DB2I2 command. The following is a list of different ways in selection sequence:

1. Use **IDSN='line.object.input.dsname'** command option to read line objects from specified file.
2. Use **IDD='line.object.input.ddname'** command option to read line objects from specified DDNAME. This option can only be used in batch mode. To select line objects to process in batch mode, you must use IDSN or IDD command options.
3. In online mode, use a new ISPF line command **S** to select **single line object**.
4. In online mode, use a pair of new ISPF block line command **SS** to select **a block of line objects**.
5. In online mode, if there are no S or SS line command entered, **the current cursor positioned line object is selected** for you to confirm the selection.
6. In online mode, if it does not meet any of the criteria listed above, **the first displayed line object is selected** for you to confirm the selection.

Line Objects Description

The following is a list of all line objects abbreviation, format of line object detail, and their descriptions:

<u>Line Objects Abbreviation & Detail</u>	<u>Object Description</u>
AC	active log line object
AI	ixcreator.ixname partno row-count key-card (Adjust Index part line)
AL	creator.name (Alias)
AR	Archive log line object
AT	dbname.tsname partno tcreator.tbname row-count %compressed (Adjust Table part line)
BP	buffername (Bufferpool)
CI	Catalog table insert line (information influence optimizer selection path)
CL	collid (Collection)
CO	tcreator.tbname.colname (Column)
CP	System check point RBA line object
CU	Catalog table update line (information influence optimizer selection path)
DB	dbname (Database)
DM	planname.dbrm (DBRM)
DS	Dataset name (Dataset name)
DT	schema.name (Distinct Type DB2 V6 or above only)
FC	[ssid\loc.]dbname.tsname [ssid\loc.]dbname.tsname (Flash Copy Line)
FU	schema.name (Function name DB2 V6 or above only)
GV	&varname=value (Global Variable line)
IC	tcreator.tbname.column (Index Column)
IP	creator.name partno (Index Part)
IPC	creator.name partno (Index Part Clone)
IS	dbname.indexspace (Index Space)
ISC	dbname.indexspace (Index Space Clone)
ISP	dbname.indexspace partno (Index Space Part)
ISPC	dbname.indexspace partno (Index Space Part Clone)
IX	creator.name (Index)
IXC	creator.name (Index Clone)
JA	Jar-schema.Jar-Id (JAR)
JI	jobname.jobnum (Job Information)
MS	schema.name (Column Mask)
MT	creator.name (Material query Table)
OI	dbid[,obid].psid[,isobid] (object ID)
PG	location.collid.name.version (Package)
PL	planname (Plan)
PM	schema.name (Row Permission)
RI	Parent_creator.Parent_tbname.Child_creator.Child_tbname (Referential Integrity)
RL	archive log command RBA line object
RO	Role name (Role Name)
SC	[ssid\loc.]dbname.tsname [ssid\loc.]dbname.tsname [part#] (Tablespace Copy)
SG	stogroupname (StoGroup)
SH	schema (Schema name DB2 V6 or above only)
SP	schema.name (Store Procedure DB2 V6 or above only)
SQ	schema.name (SeQuence)
SY	creator.name (Synonyms)
TB	creator.name (Table)
TBC	creator.name (Table Clone)
TP	dbname.tsname partno (Table Partition)
TPC	dbname.tsname partno (Table Partition Clone)
TR	schema.name (Trigger name DB2 V6 or above only)
TS	dbname.tsname (Tablespace)
TSC	dbname.tsname (Tablespace Clone)
US	username (User)
VL	volume (Volume)
VW	creator.name (View)
XC	[ssid\loc.]ixcreator.ixname [ssid\loc.]ixcreator.ixname [part#] (Index Space Copy)

For certain DB2I2 command, the wildcard line objects are allowed. These commands include drill down, BIND, BIND COPY, FREE, IX, LISTC, PG, PL, RI, REBIND, SELPATHV, STATS, TB, TS, TSIX. Please refer to command detail to see whether the command you are interesting supports wild card line objects.

The symbol of % indicates it is a wildcard line object. For example, a TB Q.% represents all QMF tables.

Global Line Object option

DB2I2 provides global line object options, which allows you to **override the command options** specified on the command. This override only applied to the line where global line object option located. To specify global line object option, you enter one of the options listed below at the end of the line, which you want this override applied.

The following is a list of all global line object options:

<NEWJOB>	Use <NEWJOB> global line object option to generate additional job cards within a JCL. When you select multiple line objects which produces multiple job steps in a generated JCL. <NEWJOB> on the line signals the beginning of a new job so that you can produce multiple jobs with a single DB2I2 command.
%####	Use %#### global line object option to override the space allocation as a #### percent of the current allocation.
ALLOC=(alloc_type,primary,secondary)	Use ALLOC=(alloc_type,primary,secondary) global line object option to override the space allocation for the selected line. Alloc_type can be PAGE, CYL, or TRK. Primary and secondary must be numeric.

Execution Result from DB2I2 command

If you run DB2I2 command in online interactive mode, most the command output results returned within your workbench ISPF edit session. With the exception of the following commands:

TSOID.DB2I2.LISTC.OUTPUT	for the output of LISTC command
TSOID.DB2I2.EXEC.OUTPUT	for the output of DSNTIAD, DSNTPE2 and EXEC command
TSOID.DB2I2.DB2CMD.OUTPUT	for the output of DB2CMD START, STOP, DISPLAY
TSOID.DB2I2.EXPLAIN.OUTPUT	for the output of EXPLAIN and EXPLAINP command
TSOID.DB2I2.DSCOPY.OUTPUT	for the output of DSCOPY command
TSOID.DB2I2.DSNJU004.OUTPUT	for the output of DSNJU004 command
TSOID.DB2I2.RUN.OUTPUT	for the output of RUN command
TSOID.DB2I2.OUTPUT	for all the other commands that do not return right back in you ISPF edit session

If you select to run DB2I2 in batch mode, a time suffix THHMMSS is added to the end of the file listed above. For example, the output of a batch LISTC run could be tsoid.DB2I2.LISTC.OUTPUT.T010101.

The above default output file name can be overridden by ODSN='output.dsname' command option. If you specify ODSN='output.dsname' as the command option together with your DB2I2 command, DB2I2 will direct the output to the 'output.dsname' you specified. For example, specify LISTC ODSN=my.outfile.name direct DB2I2 to write the LISTC output to my.file.name.

Global command options

Global command options are command options, which can be applied to most of the DB2I2 command. The following is a list of these options:

- **%=###** When space calculation is required for a DB2I2 command, specify %=### option to adjust space allocation. The adjustment is a percentage ### % of the current space allocation based on the statistics from ICF catalog and DB2 catalog.. This option can be used with DDL, MIGR, DSADJ, and GENVCAT. You can use CYL or TRK command option together with %=### to round up the space allocation to cylinder or track boundary. This option can also used with utility generation commands such as REORG, COPY, and LOAD..to adjust the spaces allocation for the work files required for those utilities.
- **APPEND** Specify APPEND command option to append the result to an existing sequential file. (no PDS allowed)
- **DSPRE=DatasetPrefix** Specify DSPRE option to set work file prefix for those generated DB2 utility JCL which requires work files. If no DSPRE= option specified, is default to your TSO ID.
- **DFLTSP=(pri,sec)** Specify DFLTSP option to assign primary and secondary quantity allocation in cylinders for the space allocation for all work files of object that does not have RUNSTATS information available for CHECK, DSCOPY, REBUILD, RECOVER and REORG db2i2 commands. The default if not specified is DFLTSP=(1,1).
- **ERROR(CONTINUE|SKIP #)** In batch mode, you can decide what to do after an error occur. ERROR(CONTINUE) command option allows you to continue process next DB2I2 command in sequence. ERROR(SKIP #) command option allows you to skip # commands. Specify ERROR(SKIP 0) is the same as specify ERROR(CONTINUE) which does not skip any command lines.
- **IDD='line-object.input.ddname'** In batch mode, specify IDD command option to set the line object input ddname. You should specify a ddname in your batch jcl which contains the line objects as input to the current DB2I2 command. If you do not specify this option, it defaults to LINEOBJ.
- **IDSN='line-object.input.dsname'** Specify IDSN option to use line objects from external source as input to the current command. This option also can be used to access a set of pre-defined UDF or UDQ. To access these system UDF or UDQ, you specify *memname on the 'line-object-input-dsname' to read the information from DB2I2 system library. For example, IDSN=*GENAT to read in predefined query from DB2I2 system library to generate AT line for a db2 tablespace.
- **JOB#=##** When a JCL is produced from DB2I2, if you specify #s on your first JOB CARD, those #s are replaced with job number started at 1. Specify JOB#=## to override 1 with different job number.(To use this command option, you must use DB2I2 JOBCARD command to specify job card information with ## defined in the job name field)
- **JOBNM=jobpref##** When a DB2 utility JCL is produced from DB2I2, if you specify #s on your first JOB CARD, those #s are replaced with job number started at 1. Specify JOBNM=jobpref## to override the job name from job card with the same number schema as from job card.
- **JOBCARD=N** When a JCL is produced from DB2I2, by default, job cards are always produced. However, when you prepare a serial of multi-step job, you do not want to generate job cards for all DB2I2 commands but the first command. Specify JOBCARD=N option to disable JOBCARD generation for any DB2I2 command which produces JCL.. Use this option with STEP# option to generate a multiple steps job.

- **MACRO(your.ed.macro)** Use this option to customize your output JCL generated from QUIESCE, RECOVER, REBUILD, COPY, REORG, LOAD, REPAIR, MODIFY, CHECK, REPORT, RUNSTATS, DSN1COPY, DSN1PRNT and DSCOPY. (For Db2 V7 or above only)
- **NEWJOB=###** When a batch job produced from DB2I2, it generates up to 255 job steps before generates next job. Specify NEWJOB=### option allows you to set the maximum number job steps for each generated job. Because DB2I2 can generate multiple jobs and multiple steps for each batch run, this option allows you to specify for each generated job contains no more than ### job step. The ### specified must be a positive number between 1 and 255. The default value for ###, if not specified, is 255. One way to use this option is to balance the work load. For example, if you have 100 tablespaces you want to replicate from production environment to test environment with DSCOPY command using image copy, for each tablespace, DSCOPY generates 3-4 job steps. Specify NEWJOB=40 roughly break the whole replication process into 10 jobs, which can run concurrently.
- **NOTFOUND(CONTINUE|SKIP #)** When encounter a RECORD NOT FOUND condition in Batch mode, this option allows you to choose whether to continue processing next command or skip # command lines from the current DB2I2 Command. If no option specified, the job ends with a condition code 16. Specify NOTFOUND(Skip 0) is the same as NOTFOUND(CONTINUE) which does not skip any commands.
- **ODSN='output.dsname'** Specify ODSN to write the result out to a specified file.
- **ODD='output.ddname'** Specify ODD to write the result out to a specified DDNAME in batch mode. This option work with most of the DB2I2 commands except for those commands generate JCL, such as REORG, COPY etc.
- **ODSN='output.dsname(*)'** When use this option with DB2 utility JCL generation, you can specify a PDS with (*) as member and allows DB2I2 to use JOB name as member name for the generated JCL output. Since there could be multiple job cards generated from one DB2I2 utility command, multiple output members can be generated with one command. This command option can be used for all DB2I2 utility command which include REORG, COPY, RECOVER, REBUILD, QUIESCE, REPORT, LOAD, LISTDEF, RUNSTATS and UNLOAD.
- **SKIP=(#1,#2) or SKIP(#1)** In batch mode, specify SKIP option to skip commands. You can used this option to control process flow. #1 represents the number of command lines to skip from the current DB2I2 Command. #2 represents the number of time to skip. The current command must be successfully processed before the skip can occur. 0 is the default value for #1, which means no Command lines are skipped. Specify -1 repeat the current command. Specify -2 to process previous command. If #1 is greater than the last command line, job ends after current command completes(skip to EOJ). If you specify a negative #1 without #2, skip #1 will be processed forever (infinite loop?), so that if you skip backward without #2 specified, you must have a way to break the loop. For example, use SKIP(-1,3) to repeat the current command 3 times:
- **SQLID=sqlid** During DDL or DCL generation, SQLID option allows you to replace SQLID from existing SQLID recorded in the db2 catalog with the **sqlid** specified. (The SET CURRENT SQLID = current-user will be replaced with SET CURRENT SQLID = **sqlid**) The command option can be useful when used with command such as MIGR and COPYAUTH.
- **STEP#=###** In batch mode, when DB2I2 commands generates multiple job steps, use STEP#=### option to assign job step number for the JCL generated from current DB2I2 command. Use this option together with JOBCARD=N option to generate a multi-step job.
- **T=N** Specify T=N option to disable writing any title information, heading information, or footing information. This information contains description of the information generated from current command. However, when you need to reuse output from current command as input to next command, this information becomes invalid as reusable line object. Use T=N option to avoid this condition happens.

- **WKSP=# or WKSP=(type,pri,sec,dir)** When a work file or default output file is generated from DB2I2, the default space allocation used for those files is Track,(1,1,2). You use this option to override this default setting. Specify # changes the file allocation from default Track,(1,1,2) to Track,(#,1,2). You can also specify the long format of the WKSP=(type,pri,sec,dir) where type – C=cylinder T=track and pri, sec and dir if specified should all be numeric numbers.

Global Variable and Host Variable Usage

Global Variable

- Global variables are used for command substitution.
- Default System Global Variable (part of the installation output file) is installed with the following:
000001 GV &GVDSN='your.installation.GLOBAL.VARIABLE'
000002 GV &MYGV=DB2I2.GLOBAL.VARIABLE
- The format of global variable is &GGGGGGGG, where GGGGGGGG is alpha-numeric up to 8 position long
- It is defined with SETG command and last during your session until a RESETG command is issued.
- Use VIEWG command to view the current global variable setting.
- Used global variable when
 - you process DB2I2 command in online mode and the command options is too long
 - setup a set of global variable for your own standard
- Global variable can contain host variables
- To allow a set of global variables always available when you login, customize you DB2I2 environment by specifying a set of global variables in the dataset '**tsoid.DB2I2.GLOBAL.VARIABLE**'.

Host Variable

- Host variables are used with
 - BATCH ICMD= DB2I2 command to substitute the host variables in a set of predefined DB2I2 commands
 - RUN command to substitute host variables in a predefined set of SQL.
- The format of host variable is &HHHHHHHH, where HHHHHHHH is alpha-numeric up to 8 position long

When use Global Variables together with Host Variables, the host variable name must not be the same as global variable name.

Example

The following example demonstrates how global variable and host variable work.

When you issue a system defined UDF GENURLD with the following command:

```
BATCH ICMD=*GENURLD &JCLWS='your.work.jcl'
```

&JCLWS is a **host variable** which is used to substitute the host variable &JCLWS defined in a system UDF *GENURLD

GENURLD uses SETG command to set **global variables** &OP1 through &OP14 with the information from DDNAME //GLOBDD. These global variables are used to **define a set of host variables** and then passing to another pre-defined command GENURLD1.

The predefined system UDF – GENURLD contains the following in detail:

```
SETG      IDD=GLOBDD
BATCH     ICMD=*GENURLD1 +
          ODSN=&JCLWS +
          &OP1 &OP2 &OP3 &OP4 &OP5 &OP6 &OP7 &OP8 &OP9 +
          &OP10 &OP11 &OP12 &OP13 &OP14
TSO       SUBMIT &JCLWS
//GLOBDD  DD DATA,DLM=GG
GV &OP1=&TBDSWS=<GENERATED.TB>
GV &OP2=&UNLDJWS=<OUTPUT.JCL>
GV &OP3=&JOBCTL=<JOB.PARM>
GV &OP4=&LDTBWS=<LOAD.TB>
GV &OP5=&WHEREWS=<WHERE.DS>
GV &OP6=&PUTILL=<LOAD.PARMUTIL>
GV &OP7=&PUTILNC=<REPAIR.NCOPY.PARMUTIL>
GV &OP8=&PUTILRS=<RUNSTATS.PARMUTIL>
GV &OP9=&FROMLOC=<fromlocation>
GV &OP10=&TOLOC=<tolocation>
GV &OP11=&FROMCLAS=<from-job-class>
GV &OP12=&TOCLAS=<to-job-class>
GV &OP13=&DSPREWS=<dataset-prefix>
GV &OP14=&WKSPWS=<##>
GG
```

In this example, host variable &JCLWS will be substituted with 'your.work.jcl'. All the global variables &OP1 through &OP14 which defines a set of host variables from //GLOBDD and then pass to another system UDF – GENURLD1.

DB2I2 Command Description Summary

The following is a list of all DB2I2 commands and their description in alphabetical order:

<i>DB2I2 Command</i>	<i>DB2I2 Command Description</i>	<i>Allowed Line Object</i>
<p>[HOSTVAR can be used for PG or DM line object only to display host variable information] [NOSTATS used with PG or DM to disable display of statistics information – the output can be used with EXPALIN directly] [TRUNC=Y truncate the result if the result line length is > lrecl]</p>	Drill one level down of the selected line listed.	DB, TS, TP, TB, AL, SY, VW, IX, IP, PG, PL, SG, DM, DS, RI DT, FU, SP, TR,SH, IS,ISP,MT,OI,SQ, TSC,TPC,TBC,IXC,IPC,ISC,ISPC, TC,RO, PM, MS
AL	Generate Alias line object information from the selected Table, View, Plan or Package.	TB, VW, PL, PG, MT
ALTER	Generate ALTER DDL for the selected Database, Tablespace, Table, Index or StoGroup.	DB, TS, TB, IX, SG FU, SP, SQ, CO, MS, PM
AUTH	Display Authorization information for the selected line objects	AL, BP, CL, DB, PL, PG, SG, TS, TB, US,VW DT,FU,SH,SP,TR,SQ, MT, RO,TBC
<p>BATCH db2i2 command [Db2i2 command options] or ICMD=command.input [CLI=MC UTL clidsn(mem)] [JCL=N][IDSN=*DUMMY] [IDSN*=line.object.filename] [ODSN*=[JOBDD=N]</p>	<p>Generate BATCH JCL to run DB2I2 command in batch.</p> <p>Majority of the DB2I2 commands can be executed in batch mode, so that you can free your terminal for the long running command, or schedule it to run during the preferred time. Typical long running DB2I2 commands are MIGR, STATS, LISTC. Use ICMD option to process Db2I2 command from external file with host variable substitution.</p> <p>Use CLI=MC option allows you to execute multiple commands against line object one at a time.</p> <p>Use CLI=UTIL option to generate DB2 utility JCL one line object at a time with option to dynamically submit it.</p> <p>Use UTIL=clidsn(mem) option to specify your own REXX or CLIST routine.</p> <p>Specify JCL=N if the ICMD input is a prepared JCL file and you do not want DB2I2 generates any unnecessary DD names.</p>	
<p>BIND [O=owner] [CL=collection] [Q=qualifier] [MEM=*] [GRANT=Y] [EXPLAIN=Y]</p>	Generate DB2 BIND commands for selected DB2 package or plan.	PG, PL*
<p>BIND COPY [O=owner] [CL=collection] [Q=qualifier] [MEM=*] [GRANT=Y] [EXPLAIN=Y]</p>	Generate DB2 BIND COPY commands against selected package.	PG*

* Wildcard % is allowed for these line objects

<i>DB2I2 Command</i>	<i>DB2I2 Command Description</i>	<i>Allowed Line Object</i>
CANCEL [DDF] THREAD [DUMP] [NOBACKOUT]	Issue CANCEL THREAD or CANCEL DDF THREAD for a display thread line with Numeric Token as the last field on the line.	A display thread line with Numeric Token as the last field
CHECK [[PARMUTIL=]'parmutil. dsn'] [DFLTSP=(1,1 pri,sec)] [FNM=ID] [WKUNIT=TAPE_0#]	Generate DB2 CHECK DATA/INDEX/LOB utility JCL for selected table space or index	TS,IX,TP,IP,TSC, TPC, IXC, IPC, ISC, ISPC
COAUTH [GRANTOR GRANTEE]	Display Column Authorization information for the selected line objects	AL,TB,US,VW, MT, RO
CONNECT(location)	Connect to remote location. Use CONNECT(?) to list all location information	
CONNECT(RESET)	Reset connection to default local location	
COPY [[PARMUTIL= 'parmutil.dsn']	Generate DB2 IMAGE COPYMERGCOPY utility JCL for selected table space or index space	TS, TP, IX,IP, IS, ISP, TSC, TPC, IXC, IPC, ISC, ISPC
COPYAUTH [GRANTOR GRANTEE] [TO=to-userid] [CRE=new_creator] [DB=new_dbname] [CL=new_collid] [SG=new_stogroup] [SH=new_schema]	Generate GRANT DCL for specified DB2 user. This command is useful to copy all the authorizations from one DB2 user to another.	US, RO, AL,BP,CL,DB,DT,FU,MT,PG,P L,SG,SP,SQ,TB,TS,VW
CPY2CPY TEMPLATE= template.dsname [LISTDEF=listdef.dsname{(pat t*)}] [OPTIONS=options.dsname] [[PARMUTIL=]'parmutil.dsn']	Generate DB2 COPYTOCOPY utility JCL for selected table space or index space.	TS, TP, IX,IP, IS, ISP,TSC, TPC, IXC, IPC, ISC, ISPC
CREATE	Generate CREATE DDL for the selected line object.	AL, DB, DT, FU, IX, SG, SP, SY, TB, TR, TS, VW, MT, SQ, RO, TC, MS, PM
CURSORD	Generate DECLARE CURSOR embedded SQL statement for selected line object.	TB,AL,SY,VW, MT,TBC
DB	Display Database information from selected line object	TS, TP, IX, IP, IS, ISP, TB, MT, TSC, TPC, IXC, IPC, ISC, ISPC
DBAUTH [GRANTOR GRANTEE]	Display Database authorization	US,DB, RO
DBDSIZE [threshold]	Display Database DBD size Information	DB
DB2CMD [LIST]	Issue DB2 Command using the db2 command lines specified. LIST option display all available DB2 commands	db2 command lines
DCLGEN [[PARMUTIL=]'parmutil.dsn']	Generate DB2 DCLGEN command against selected Table, Alias or View.	TB,AL,VW, SY, MT, TBC

DB2I2 Command	DB2I2 Command Description	Allowed Line Object
DDL [%=### [CYL TRK _] [ALLOC=(type,pri,sec)] [SQLTERM(?)] [OBID][TR=N Y] [MASASGN] [CRE=new_creator] [DB=new_dbname] [CL=new_collid] [SG=new_stogroup] [VCAT=new_vcatname] [O=new_owner] [Q=new_qualifier] [SH=new_schema]	Generate DDL for the selected database, Tablespace, table, index, view or StoGroup. Use MIGR command instead of DDL if you want to generate all the DDL for the selected object as well as the entire dependent object. %=### Options allows you to adjust the space allocation as % of current allocation. Specify SQLTERM in Db2 V6 to assign SQL terminator. Use TR=Y to display trigger information fro a TB line	AL,DB,TS,TB,IX,SY, VW,SG, RI, DT,SP,FU,TR, MT, SQ, TBC, TC, MS, PM
DELETE [MAP='dclgen dsn']	Generate DELETE SQL	TB, AL, SY, VW, MT, TBC
DISPLAY [DB2 display command option except DB and SPACE option] [TSIX option to return TP or IP line objects] [EDIT=Y]	Display DB2 objects Issue any valid DB2 -DISPLAY command option w/o line object or any -DSIPLAY options except DB and SP with DB,TS,TP,IX,IP. Use EDIT=Y option to edit the DISPLAY result.	DB, TS, IX, TP, IP, IS, ISP, BP or no line object
DSADJ [%=### [CYL TRK _] [MOVE=YES NO] [ALLOC=(type,pri,sec)] [MAXSZ=(type,pri,sec)] [OVRD=reorg_override_table] [MACRO(your.ed.macro)] [ONESTEP=Y]	Generates JCL to adjust the space allocation. For DB2 manage Dataset, Alter storage is generated. MOVE=Y option generate DSN1CPY for the underline VSAM linear Dataset. Use %=### option to adjust space allocation as % of current allocation. Use ALLOC=(space_type,pri,sec) to override space allocation	TP,IP, ISP, TPC, IPC, ISPC
DSCOPY [ICOPY=Y N LB RP RB] [NOSTOPS] [NOSTOP][SORTKEYS] [HLQ=Copy-high-level-qualifier] [MACRO(your.ed.macro)] [CONT=Y] [ERRDSN=error.dsn] [DFLTSP=(1,1 pri,sec)] [CLONE=Y]	Generate DSN1COPY Utility JCL to copy Tablespace or indexspace with OBID translation.	SC,XC
DSNJU004 [BSDS='bsds.dsn']	Display DSNJU004 prints log map interactively. Optional BSDS='bsds.dsn' allow you to specify the input BSDS Dataset.	
DSNTEP2 [SQLTERM(?)]	Interface to DSNTEP2 interactively	SQL blocks
DSNTIAD [SQLTERM(?)] [RC0]	Interface to DSNTIAD interactively	SQL blocks
DSNTIAUL [DEV=D T] [SQL] [DSPRE=DatasetPrefix]	Generate DSNTIAUL utility JCL against selected table, view, alias or SQL blocks.	TB,VW,AL,SY, MT, TBC or SQL blocks
DSN1COPY	Generate DSN1COPY utility JCL against selected Tablespace or index.	TS,IX,TP,IP, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC
DSN1LOGP [BSDS ACTV ARCH]	Generate DSN1LOGP utility JCL against selected Tablespace or index. Use BSDS, ACTV or ARCH to specify DSDS, active log or archive log as input to the DSN1LOGP.	TS,IX,TP,IP,IS, ISP TSC, IXC, TPC, IPC, ISC, ISPC

DB2I2 Command	DB2I2 Command Description	Allowed Line Object
DSN1PRNT	Generate DSN1PRNT utility JCL against selected Tablespace or index.	TS,IX,TP,IP, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC
DT	Generates DT- distinct type line object from selected line object.	TB, TBC
ED [MACRO(macro.dsname) IDD=macrodd)]	Use ED command to edit sequential datasets or members of a PDS with the sam syntax you use in your ISPF edit session.	
EDIT 'edit.dsname' & END_EDIT	Use EDIT and END_EDIT pair commands in Batch mode to provide simulated batch ISPF edit command. For example, C 'aaa' 'bbb' all, to change all 'aaa' to 'bbb' in the specified 'edit.dsname'	Obsolete and replaced by ED command
EXEC [#] [ERROR(CONTINUE SKIP #)] [RC0] [SQLTERM(?)]	Execute DB2 Commands, IDCAMS commands, Non-select SQL or any of the DB2I2 scripts generated from various DB2I2 commands. RC0 allows continue execution after any non-zero SQL return code. ERROR(CONTINUE) continues execute next command after error returned	DB2I2 script
EXPLAIN [QNO=# 0] [O=planowner] [DESC=Y N] [DET=O N Y] [DEGREE=degree] [SQLTERM(?)] [Q=qualifier] [SU=service-unit-thread hold] [EDIT=Y] [T=N]	Invoke DB2 EXPLAIN and display EXPLAIN report. EXPLAIN use block number 0 for explain PLAN_TABLE output EXPLAIN # use block number # for PLAN_TABLE output Use O=planowner option to SET SQLID for unqualified name from DBRM statement (This option only allowed if you can issue SET SQLID command)	SQL statements or statements from PG or DM drilldown
EXPLAINP [O=plan-owner] [PG=Program-Name] [QNO=[##1-#2]] [PL=planname] [DESC=Y N] [DET=O N Y S] [GN=generation] [T=N] [SU=service-unit-thread hold] [HOSTVAR] [EDIT=Y]	Display PLAN_TABLE report Use PG=prognam to select program, PL=planname to select plan. You can use wildcard % on both progrname and planname field. Use QNO=# to display only for query no=#. Use QNO=#1-#2 to display query no between #1 and #2. Use DET=Y to enable catalog statistics drill down as well as db2 statistics. DET=O is the default which display detail statistics only if the selected SQL is a potential problem SQL. Use DESC=N to disable printing of report detail description. Use DET=S to display SQL statement only without db2 statistics. Use GN=generation to specify generation of explain output. Specify 0 for current generation, -1 for previous generation... The default (do not specify GN option) is all generations. Use SU= option to display only for those queries with service unit greater than or equal to the thread hold specified. Use HOSTVAR option to display host variable information. Specify EDIT=Y to Edit the output instead of Browse.	PL, PG, DM [stmt-no]
EXPLORE [TYPE=CHECK REBIND] [O=tsoid plan-owner] [CL=tsoid collection-id] [VER=C A] For TYPE=CHECK [CRIT=R,I,O,L,MJ,HJ,SO] ALL] [SU=service-unit-threahold] [CONT=AND OR]	A pricing add-on for JRH-DB2I2 EXPLORE can be used to check if db2 package requires REBIND. It also can be use to check if db2 package meets your specified criteria. Authorization required: SYSADM, or BINDADD with table access authorization. Use TYPE=REBIND to check if selected PG lines require REBIND. Use TYPE=CHECK default option together with CRIT and SU option to check if selected PG lines match your criteria.	PG
FETCH MAP='dclgen dataset'	Generate Cobol Embedded FETCH INTO statement for selected db2 objects MAP='dclgen.copybook.dsn' is used to map db2 column name with Cobol host variable	TB,AL,SY,VW, MT,TBC

DB2I2 Command	DB2I2 Command Description	Allowed Line Object
FLASH [MACRO(your.ed.macro)] [CONT=Y] [CLONE=Y]	Generates dfsmsDSS JCL to Flash Copy DB2 Dataset. The Source and Target must have the same database Structure which includes the same Index Names and Same OBIDs (MIGR with OBID) The generated JCL contains the following steps: - STOP Target Db2 Objects - IDCAMS Delete Target VSAM Datasets - START Source Db2 Objects with ACCESS(RO) - dfsmsDSS Flash Copy Source and Rename to Target objects - START Source Db2 Objects with ACCESS(RW) - START Target Db2 Objects with ACCESS(UT) When process Multiple line objects, use CONT=Y option to continue process the rest of the FC line object if there are any difference between source and destination object.	FC
FLIST 'flist.dsname' [132]lrec1]	List the content of a dataset. 'flist.dsname' is used to specify the name of the dataset.	N/A
FREE [KEEP=#][CURRENT]	Generate DB2 FREE commands against selected package or plan. Use KEEP=# option to specify the # of versions to be kept. Use CURRENT option to free the current version of package.	PG,PL*
FTP [INPUTDSN=input-dsn] [RESET=Y]	A pricing add-on with RXDB2I2 Use FTP to invoke and process FTP command thru RXDB2I2. It can be Processed as RXDB2I2 command option: RXDB2I2 FTP ... Or Processed inside RXDB2I2 RXDB2I2 idd=ddl Lineobj dname=ddl DB2I2REX "FTP ..." End_lineobj In online mode, you can select lines in your work bench and then issue RXDB2I2 FTP command to invoke FTP to PUT the selected lines to a remote server. The selected lines are stored under //DD:TEMP. Use INPUTDSN option to assign the dsn for FTP control statements If not specified, Use //INPUT DD file as INPUT to FTP or if no //INPUT DD, use yourID.DB2I2.FTP.INPUT as INPUT dsn to FTP. You can edit the FTP control statement to fit your need. Use RESET=Y to reset your entered password to 'password' to protect your password after FPT is done.	N/A
FU	Generates FU- function line object from selected line object.	TB,VW,PL,PG, MT, SQ
GENVCAT [VOL=*] [%=### [CYL TRK _]] [ALLOC=(alloc_type,pri,se)] [VCAT=new-vcatname] [DB=new-dbname]	Generates VCAT information from the selected line objects. VOL=* option generate IDCAMS define with VOLUMES (*). %=### and ALLOC=(alloc_type,pri,sec) can be used to adjust space allocation. Specify DB for new DB override.	TS, TP, IX, IP,DS, IS, ISP, TSC, TPC, IXC, IPC, DS, ISC, ISPC
GRANT [SQLID=id] [TO=to-user] [grant options]	Generates GRANT DCL for the selected line objects	AL,BP,CL,DB,PL,PG,SG,TS,TB, US,VW, DT,FU,SP,SH,TR,MT, SQ, RO,TBC

* Wildcard % is allowed for these line objects

DB2I2 Command	DB2I2 Command Description	Allowed Line Object
HELP or ? [db2i2 command]*UDF*db2i2 TSO command &globalvar]	Display Help screen and detail help information. HELP [db2i2 command] displays help information for specific db2i2 command.	Any line object
HELPL0 [valid db2i2 line object]	Display Help for line object with associated DB2I2 command. HELPL0 [db2i2 line object] displays DB2I2 command information for specific db2i2 line object.	
HMIGRATE	Invokes HSM HMIGRATE to migrate DS or AR line objects.	DS,AR
HRECALL	Invokes HSM HRECALL to recall DS or AR line objects.	DS,AR
IMPACT	Display dependent line object information from the selected line object	DB,TS,TB,IX,AL,SY, VW DT, FU, SP, TR, MT, SQ, TBC
INFO	Display SSID and connection information.	
INSERT [MAP='dclgen dsn'] [RUN='db2i2.run.output' COMMIT=# SQLTERM(?) CHAR=HEX]	Generate SQL INSERT statements for the selected table, alias, synonyms or view. Use INSERT MAP='dclgen.dsn' to generate embedded SQL. RUN option generates INSERT from RUN output. Use COMMIT=# to specify commit frequency. Use SQLTERM(?) to specify sql terminator. Use CHAR=HEX to generate CHAR or VARCHAR in HEX format	TB,AL,SY,VW, MT, TBC
IP	Shows IndexPart usage for selected line objects.	DB,TS,TB,MT, IX, IS, ISP, TP, TSC, TBC, IXC, ISC, ISPC, TPC
IPC	Shows IndexPart Clone usage for selected line objects.	DB,TS,TB,MT, IX, IS, ISP, TP, TSC, TBC, IXC, ISC, ISPC, TPC
IS	Shows Index Sapce usage for selected line objects	DB,TS,TB,MT, IX, IP,ISP, TP,PL,PG
ISC	Shows Index Sapce Clone usage for selected line objects	DB,TS,TB,MT, IX, IP,ISP, TP,PL,PG, TSC, TBC, IXC, IPC, ISPC, TPC
ISP	Shows IndexSpacePart usage for selected line objects	DB,TS,TB,MT, IX, IS, IP, TP
ISPC	Shows IndexSpacePart Clone usage for selected line objects	DB,TS,TB,MT, IX, IS, IP, TP, TSC, TBC, IXC, ISC, IPC, TPC
IX	Shows index usage for selected line objects.	DB,TS,TB,IS, ISP,PL,PG*
IXC	Shows index Clone usage for selected line objects.	DB,TS,TB,IS, ISP,PL,PG, TSC,TBC,ISC,ISPC
JOBCARD	Enter Jobcard Information. Please specify this if you want to use any of the JCL generated by DB2I2.	
LISTC [EXT(##)] [TSIX] [IN] [SIZE=####] [VOLSER=# volser]	Generate IDCAMS List Catalog information, which detail the space usage and extents information. EXT(##) option direct DB2I2 only list information when VSAM Dataset extents > ##. TSIX option allows DB2I2 to display TP or IP line objects instead default DS information. IN option insert output directly into you workbench edit session. Use SIZE option to generate <NEWJOB> token on the output line for every ### cylinders accumulated. Use VOLSER=volser to generate a list of DS for a specific volser only	TS,IX,TP,IP, DS,IS,ISP, TSC,IXC,TPC,IPC,ISC,ISPC
LISTDEF [IDSN='input.listdef.dsn'] [ODSN=odsn[*]] [LISTNM=mylist] [[PARMUTIL=]'parmutil. dsn']	This command is used to generate DB2 LISTDEF utility control statements for selected line objects.	DB, TS, TP, IX, IP, TB, IS, ISP, MT, TSC, TPC, IXC, IPC, TBC, ISC, ISPC

DB2I2 Command	DB2I2 Command Description	Allowed Line Object
LOAD [[PARMUTIL= 'parmutil.dsn'] [OVERRIDE] [FNM=ID] [WKUNIT=TAPE_0#] [LISTDEF= listdef.dsn[(patt*)]] [TEMPLATE= template.dsname] [OPTIONS=options.dsn]	Generate DB2 LOAD utility JCL for selected table with calculated space for work files. If multiple TB or MT line objects are selected, you can use # on each TB line to indicate the estimated # of rows to be loaded. Make sure you select 0 on the estimated row field on the selection screen to allow estimated row to be selected from the TB or MT line. Use SYSIN=sysin.dsn and SYSREC=sysrec.dsn at end of the TB line object to assign the SYSIN and SYSREC input. Use RESUME=YES at end of a TB line object to indicate the LOAD RESUME override for that particular TB line. Specify OVERRIDE command option to overrides the SYSIN control card with the information from the 'load.parmutil.dsname' file.	TB, TBC or MT tbnam [#] [SYSIN=sysin.dsn] [SYSREC=sysrec.dsn] [RESUME=YES]
MIGR AL=Y] [SY=Y] [VW=Y] [BIND=Y] [GRANT=Y] [RI=B]C P N][%=### [CYL TRK _] [ALLOC=(type,pri,cse)] [SQLTERM(?)] [DT=Y] [LO=Y] [FU=Y] [TR=Y] [SQ=Y] [MT=Y N S] [MAXASGN] [OBID] [CRE=new_creator] [DB=new_dbname] [CL=new_collid] [SG=new_stogroup] [VCAT=new_vcatname] [O=new_owner] [Q=new_qualifier] [SH=new_schema]	Generate migration DB2I2 script which includes all the DDL, IDCAM Defines, BIND plan, BIND package and DCL GRANT for selected objects. You can modify the generated DB2I2 scripts to fit your need. The scripts can then be executed by using the DB2I2 EXEC command. Specify %=### at the end of line object to override space allocation. Specify #PART=### and PTX=index-name to convert a non-partition TS to a Partition TS. Specify SQLTERM option in Db2 V6 to assign SQL terminator. Use MAXASGN option to generate DDL with Identity column START WITH maxassignal + increment. Use OBID option to generate OBID option for CREATE TABLE DDL. Use <u>POSTMIGR</u> ED macro to comment out duplicate VIEW, ALIAS, SYNONYMS, BIND and GRANT.	AL,DB,TS,TB,IX, SY,VW, SG, RI,DT,SP,FU,TR,MT, SQ,TBC, TC, MS, PM
MODIFY [[PARMUTIL= 'parmutil.dsn'] [LISTDEF= listdef.dsname[(patt*)]] [OPTIONS=opt.dsname]	Generate DB2 MODIFY utility JCL for selected Tablespace. Modify.parm.dsname option allows you to generate MODIFY utility JCL with the option directly input from modify.parmutil.dsname.	TS,TP, IX, IS, TSC, TPC
MS MT	Generate Column Mask line objects Shows Material query Table usage for selected line objects.	TB DB,TS,TP,TB,IX, IP,MT, AL, VW, SY, IS, ISP,PL,PG, DT, FU, SP, TR*
OI OPTIONS PACKIT [CL=collid] [O=owner] [Q=qualifier] [MEM=*]	Shows Object Id information for selected line objects. Generate DB2 utility OPTIONS control statement Generate DB2 BIND PACKAGE commands for selected DBRM or Plan. CL option allows you specify Collection ID of the package. O= options allows you specify the owner of the package. Q= option allows you to specify the qualifier of the package. MEM=* generate BIND PLAN with PKLIST=collectionID.*	DB,TS,TB,IX, MT, IS* DM,PL*

* Wildcard % is allowed for these line objects

DB2I2 Command	DB2I2 Command Description	Allowed Line Object
PARMUTIL [REORG COPY RUNSTATS MODIFY REPAIR RECOVER LOAD REBUILD CHECK REPORT DCLGEN CPY2CPY LISTDEF UNLOAD] [odsn=]'parmtuil.dsnname'	Generate utility parameter control statement file for DB2 REORG, COPY, RUNSTATS, MODIFY, REPAIR, RECOVER, LOAD, REBUILD, CHECK, REPORT, DCLGEN, CPY2CPY, LISTDEF, UNLOAD. The 'parmtuil.dsnname' can then be used as the input option parameter file, which can be used to generate appropriate DB2 utility JCL from DB2I2 batch interface.	N/A
PG [TR=Y N] [TYPE=A SIUD] [SARG=search-arg]	Show package line objects for specified Tablespace, table, index or plan.	TS,AL,TB,SY,VW, IX,PL,DB* FU, SP, TR, MT, SQ, PG,TSC, TBC, IXC, CO
PGAUTH [GRANTEE GRANTOR]	Display Package Authorization information for the selected line objects	PG,US, RO
PL [TR=Y N] [TYPE=A SIUD] [SARG=search-arg]	Show Plan line objects for specified Tablespace, table, index or package.	AL,TS,TB,SY,IX, PG, VW,DB* FU, SP, TR, MT, PL, TSC, TBC, IXC,CO
PLAUTH [GRANTEE GRANTOR]	Display Plan Authorization information for the selected line objects	PL,US, RO
PM	Display Row Permission information	TB
QBUILD [F1=field name 1] [F2=field name 2] [F3=field name 3] [F4=field name 4]	Generate DB2 QBUILD to generate SQL WHERE predicates from selected line objects.	Any line object except AC, AR, CI, CP, CU, GV, RL
QUIESCE [WRITE=NO] [TABLESPACESET] [LISTDEF= listdef.dsnname[(patt*)]] [OPTIONS= options.dsnname]	Generate DB2 QUIESCE utility JCL foe selected Tablespace.	TS,TP
RBA [yyyy-mm-dd] [JU004=dsnju004.output] [DETAIL=N Y]	Use DB2I2 RBA command to list all the RBA point from SYSIBM.SYSCOPY table with ICDATE greater or equal to the specified date. Specify DETAIL=Y to list detail RBA line information.	TS, TP ,IX, IP, IS, ISP, TSC, TPC ,IXC, IPC, ISC, ISPC
REBIND [DET=N Y] [EXPLAIN=Y]	Generate DB2 REBIND commands against selected package or plan. Use EXPLAIN=Y to generate EXPLAIN(YES) rebind option.	PG,PL*
REBUILD [[PARMUTIL= 'parmtuil.dsn'] [DFLTSP=(1,1 pri,sec)] [FNM=ID] [WKUNIT=TAPE,0 #] [LISTDEF= listdef.dsnname[(patt*)]] [TEMPLATE= template.dsnname] [OPTIONS=options.dsn]	Generate DB2 REBUILD Index utility JCL for selected Tablespace or index.	TS,IX,TP,IP, IS, ISP, TSC,IXC,TPC,IPC, ISC, ISPC

DB2I2 Command	DB2I2 Command Description	Allowed Line Object
RECOVER [[PARMUTIL=] 'parmutil.dsn'] [DFLTSP=(1,1 pri,sec)] [FNM=ID] [WKUNIT=TAPE_0#] [LISTDEF= listdef.dsname[(patt*)]] [TEMPLATE= template.dsname] [OPTIONS=options.dsn]	Generate DB2 RECOVER utility JCL for selected Tablespace or index.	TS,IX,TP,IP, IS, ISP, TSC,IXC,TPC,IPC, ISC, ISPC
REORG [[PARMUTIL=] 'parmutil.dsn'] [DFLTSP=(1,1 pri,sec)] [FNM=ID] [WKUNIT=TAPE_0#] [MAPTABLE=map_table] [LISTDEF= listdef.dsname[(patt*)]] [TEMPLATE= template.dsname] [OPTIONS=options.dsn]	Generate DB2 REORG utility JCL for selected Tablespace or index. If 'reorg.parm.dsname' is specified, the reorg option is generated based on the 'reorg.parm.dsname'. Use option RCHK to generate REORGCHK step before REORG step. Specify OVRD option to assign reorg override table for REORGCHK. Specify MAPTABLE option to override MAPTABLE option from REORG panel. In v6 for UNLOAD or DISCARD option, you can specify SYSREC= and SYSPUNCH= at the end of the line object to assign SYSREC and SYSPUNCH data set name. Specify MAPTABLE=rmapping_table at the end of the line object to override the MAPPING table for the online reorg. Mapping_table specified can include <JOBNM> as part of the specification which will replace it with the generated job name. Specify MAPTABLE=AA.<JOBNM> generates actual mapping_table with AA.J0001 if job name is J0001.	TS,IX, TP,IP, IS, ISP, TSC,IXC,TPC,IPC, ISC, ISPC [SYSREC=sysrec.dsn] [SYSPUNCH=sypsunch.dsn] [MAPTABLE=reorg_mapping_table]
REORGCHK [RC(0,1 CC1,CC2)] [OVRD=DB2I2.REORG OVERRIDE] your.reorg_override_table]	Use REORGCHK to set condition in batch mode to CC1 if a Tablespace, tablepart, index or indexpart does not require running REORG. Set to CC2 if it does require running REORG utility. Default CC1 is 0 and CC2 is 1.	TS, TP, IX, IP, IS, ISP, TSC,IXC,TPC,IPC, ISC, ISPC
REPAIR [[PARMUTIL=] 'repair.parmutil.dsname']	Generate DB2 REPAIR utility JCL for selected database, Tablespace, or index.	DB,TS,IX,TP,IP, IS, ISP, TSC,IXC,TPC,IPC, ISC, ISPC
REPORT	Generate DB2 REPORT utility JCL for selected Tablespace. REPORT with RECOVERY TO screen option allows you to narrow down the scope of a DB2 RECOVERY.	TS,TP ,IX, IP, IS, ISP
RESETG	Reset all global variables.	
REVOKE [FROM=from-user] [FROMROLE=from-role] [SQLID=sqliid for SET SQLID and By Granter] [INCLUDE=Y N to include dependent privileges]	Generates DB2 REVOKE DCL for the selected line objects.	AL,BP,CL,DB,PG,PL, SG, TS, TB, US, VW, DT, FU, SP, SH, MT, SQ, RO, TBC
REXX [IDD=rexx.source.ddname] [IDSN=rexx.source.dsname]	Use REXX DB2I2 command in batch mode to invoke REXX Exec or TSO CLIST with a stream of DB2I2 commands. IDD option indicates the source of REXX exec or CLIST is under DDNAME 'rexx.source.ddname'. IDSN option indicates the source of REXX exec or CLIST is from 'rexx.source.dsname'	

DB2I2 Command	DB2I2 Command Description	Allowed Line Object
RI [GRANTEE GRANTOR]	Use RI DB2I2 commands to generate Referential Integrity line from the select TB line object.	TB, DB* ⁴
RSAUTH [GRANTEE GRANTOR]	Display Resource Authorization for the selected objects	BP,CL,SG,TS,US*,DT, RO
RTAUTH [GRANTEE GRANTOR]	Display Routine Authorization for the selected objects	FU, SP,US, RO*
RUN LIMIT(300 #) [IN] [T=Y N] [host variable option] [DLM=?] [NOTFOUND(CONTINU E SKIP #)] [EDIT=N Y] [TRUNC=N Y] [DURATION=N Y] [BR=column-name]	Produces result for specified SQL SELECT statement. IN option returns result within your workbench edit session. Use the LIMIT(#) option to change the run result fetch limit 300. T=Y default option display the result column heading. T=N suppress heading display. The host variable option allows you to define host variable in Upper case &N=H-vvvv-H in the query and substitute them during run. NOTFOUND(SKIP #) option can be specified for batch to allow a not found bypass commands option. Specify DLM= to assign output field delimiter. You can use -INC to include part of your query from external source inside your SQL block. Specify DURATION=Y to display Start and End timestamp and the Duration information. Use BR option to create break point.	SQL blocks
RUNSTATS [[PARMUTIL=] 'runstats.parmutil.dsn']	Generate DB2 RUNSTATS utility JCL for selected Tablespace or indexes. If 'runstats.parmutil.dsnname' is specified. Runstats option is read in from the specified 'runstats.parmutil.dsnname'.	TS,IX,TP,IP
RXDB2I2 &hostvar=hostvar	DB2I2 Rexx Extender A pricing add-on for JRH-DB2I2. It allows you <ul style="list-style-type: none"> To process your REXX exec directly from your ISPF Edit session within DB2I2. To process DB2I2 commands directly from you REXX exec thru DB2I2REX interface Stream line process in both ONLINE and BATCH process. Use &hostvar=H-hostvar-H to pass host variable to rxdb2i2 script. Define your host variable in Upper case as &hostvar=H-hostvar-H to avoid conflict with existing Global variables or command options. <p>There are 3 extended Rexx Functions are added when using Rxdb2i2: RXNOTFND() Return Not Found Status from last command The vlaue can be 'Y' for Not Found and " or 'N' for others RXERROR() Return Error Status from last command The vlaue can be 'Y' for Not Found and " or 'N' for others RXERROR(N) Set Error status to 'N' RXRC() Return The RC or zisprfc from last Command RXRC(?) Set Last RC to ? where ? must be Numeric</p>	Any valid REXX statement plus Db2i2 line objects between LINEOBJ [DDNAME=DB2I2RXL DDNAM E=ddname] & END_LINEOBJ Db2i2 commands prefix with DB2I2REX
SDSF ISPF-SDSF-primary-menu- option [DDNAME=dd1,dd2..]	Interface to SDSF with job information specified. Spool selected jobs from output queue if running in batch mode or specify ODSN option.	JI
SELECT [PRE=corr-name] [JOIN=N Y LEFT RIGHT] FULL [TYPE=R M MATCH]]	Generate SQL SELECT statements for the selected table, alias, synonyms or view Use PRE option to assign correlation name. Use JOIN option to assign join type and use TYPE option to choose the type of join.	TB, AL,SY,VW, MT, TBC
SELPATHU	Generate SQL INSERT or UPDATE for Catalog tables which involves optimizer access path selection	CI, CU (from SELPATHV)
SELPATHV [OPTION=REPORT CUCI]	Generate CU and CI lines objects which display the information impact optimizer access path selection	TB,MT*
SETG	Use SETG to set the global variables.	GV

⁴ * Wildcard % is allowed for these line objects

DB2I2 Command	DB2I2 Command Description	Allowed Line Object
SETRBA [12-bytes-Hex] [+n] [-n] [T= B]	Uses SETRBA command to set the INCORE RBA which can then be used in the recover to RBA or recover to LOGPOINT process. To set INCORE T=I RBA or RestoreBefore RBA T=B just position the cursor on any of the valid RBA field and issue SETRBA or issue SETRBA command followed by a 12-bytes-Hex. Use +n or -n to add or subtract a integer number from the selected RBA.	Any edit line with valid RBA field
SHAUTH	Generates schema authorization information.	SH,US, RO
SNAPSHOT pgno[,pgno] [dsn1prnt-options]	Generate dsn1prnt output directly with the pgno (page number in hex) and dsn1prnt-otptions specified.	TS, IX, TP, IP, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC
SP	Display SP-stored procedure information	TB,VW,PL,PG,MT,SQ, TBC*
SPACE	Display detail space estimation information based on the WHATIF screen input parameters	TB,IX
SPACEADJ [CYL TRK PAGE]	Generate DSADJ line objects based on the row count for table partition and row count and key card for index partition. The output format is TP dbname.tsname partno ALLOC=(alloc_type,pri,sec) for table partition and IP ixcreator.ixname partno ALLOC=(alloc_type,pri,sec) for index partition. Use CYL or TRK to adjust to cylinder or track boundary.	AT,AI
SQ	Display SQ-sequence information	FU*
SQAUTH	Generates sequence authorization information.	SQ,US, RO*
SSID(ssid)	Set DB2 subsystem ID. Use SSID(?) to list all available SSID.	
START [db2 start command option]	Issue DB2 START command against selected database, Tablespace or indexes.	DB, TS, IX,TP,IP, IS, ISP, TSC, IXC,TPC,IPC, ISC, ISPC
STATS [TSIX] [OPTION=ALL REORGTSIX REORGTPIP]	Produce DB2 catalog statistic summary for selected database, Tablespace, table or indexes and their dependent objects. The statistics recommendation assists you to identify potential problem. Use TSIX option to return TS, TP, IX, and IP line objects for any potential problem db2 objects. Use OPTION=REORGTSIX or OPTION=REORGTPIP to return line object in TSIX or TPIP needed to be REORGed.	DB,TS, TB,IX,TP,IP,PL,PG, IS, ISP, CO, IC
STATUS	Use STATUS to display pageset status explanation. In Online mode, Point the cursor to the pageset-status-abbr from display output. Or enter pageset-status-abbr as part of the command option for both online and batch mode to display the explanation of pageset-status-abbr. The pageset-status-abbr can be the following seperate with , ARBDP AREO* ACHKP AREST AUXW CHKP COPY DEFER GRECP ICOPY INDBT LPL LSTOP OPENF PSRCP PSRBD RBDP RBDP* RECP REFP RELDP REORP REST RESTP RO RW STOP STOPE STOPP UT UTRO UTRW UTUT WEPR	Display output with pageset-status
STOP	Issue DB2 STOP command against selected database, Tablespace or indexes.	DB,TS, IX,TP,IP, IS, ISP, TSC, IXC,TPC,IPC, ISC, ISPC
STOSPCE	Use STOSPACE to generate Db2 STOSPACE Utility JCL against selected SG line objects.	SG
SUPERC newds oldds [ODSN='output.dsname'] superc-parm	Use SUPERC DB2I2 command to invoke IBM SUPERC utility to compare the content in the newds and oldds. Use optional superc-parm such as CMPCOLM 16:118 for comparing the output for EXPLAIN/EXPLAINP command.	

<i>DB2I2 Command</i>	<i>DB2I2 Command Description</i>	<i>Allowed Line Object</i>
SUPERCS search-dsn search-string [ODSN='output.dsnname']	Use SUPERCS to invoke IBM SUPERC SEARCH FOR to search the specified string from selected dataset.	N/A
SUPPORT	Use Support command to communicate with DB2I2 Support team.	N/A
SY	Use SY to generate SY line (Synonyms Dependency information from select line object.	TB, VW, PL, PG, MT, TBC
SYSIBM(prefix)	Set DB2I2 catalog table creator ID. Default is SYSIBM	
TAG	Use Tag command to tag a SYSCOPY line after you select a RECOVER or DSN1COPY db2i2 command, to identify RBA point or image copy Dataset name. Or tag a AC,AR,CP,RL RBA line after a DSN1LOGP command	Syscopy line, AC,AR,CP,RL
TB	Show table line object for specified line object	DB,TS,TP, IX,IP, IS, ISP,PL,PG, AL, SY, VW, DT,FU,TR, SP, MT, SQ, IXC, IPC, TSC, TPC, ISC, ISPC *
TBAUTH	Shows table authorizations for specified line objects.	AL, TB, VW, MT, TBC*
TBC	Show table Clone line object for specified line object	DB,TS,TP, IX,IP, IS, ISP,PL,PG, AL, SY, VW, DT,FU,TR, SP, MT, SQ, IXC, IPC, TSC, TPC, ISC, ISPC *
TB	Show table line object for specified line object	DB,TS,TP, IX,IP, IS, ISP,PL,PG, AL, SY, VW, DT,FU,TR, SP, MT, SQ, IXC, IPC, TSC, TPC, ISC, ISPC *
TEMPLATE template-option	Use TEMPLATE to generate TEMPLATE control statements to be used with other Db2 utilities. Template-option are: CHECKDATA, CHECKIX, CHECKLOB, COPY, CPY2CPY, LOAD, MERGECPY, REBUILD, REORGIX, REORGTS, UNLOAD.	N/A
TERM	Issues a Terminate Utility command against a select display utility line with UTILID = utilid	A display utility line with UTILID = utilid
TOKENSCN 'load.library[member]'	Use TOKENSCN to check the consistent token between a PG or DM line object and the specified 'load.library'. If member is specified then scan will be done interactively. Otherwise, a batch superc job is generated.	DM, PG
TP	Show TablePart usage for specified line object	DB, TB, TS, IX, IS, IP, ISP, MT, TBC, TSC, IXC, IPC, ISC, ISPC *
TPC	Show TablePart Clone usage for specified line object	DB, TB, TS, IX, IS, IP, ISP, MT, TBC, TSC, IXC, IPC, ISC, ISPC *
TR	Show TR-trigger information	DB,TB,PG*

DB2I2 Command	DB2I2 Command Description	Allowed Line Object
TRACK ODSN=output.dsnname ODD=outputDD [DDNAME=stepnm.ddnm, stepnm1.ddnm1..] [CURRENT]	A pricing add-on with RXDB2I2 Use TRACK in Batch Mode to Extract SDSF output. Use ODSN or ODD option to store the Extracted output. Use CURRENT option to Extract the Current Running Job information Line Object is not needed when you use CURRENT option. Use DDNAME option to specify stepname and ddname combination to be extracted. All selected Job information will be extracted when No DDNAME option specified. Specify DDNAME=JES2.JESMSG LG for Jes2 Job Log Information. Each stepname.ddname are separated by a comma	
TRAUTH	Generates TR – trigger authorization from selected line objects.	TR, US, RO
TS	Show Tablespace usage for specified line object	DB,TP,TB,IX, IP, IS, ISP, PL,PG, MT, TPC, TBC, IXC, IPC, ISC, ISPC *
TSC	Show Tablespace Clone usage for specified line object	DB,TP,TB,IX, IP, IS, ISP, PL,PG, MT, TPC, TBC, IXC, IPC, ISC, ISPC *
TSIX	Shows associated DB2 Tablepart or Indexpart line object from a selected DS line object.	DS*
TSO tso-command	Use TSO DB2I2 command in batch mode to invoke TSO command within a stream of DB2I2 commands.	
TSSET	Use TSSET to generate TableSpaceSET information in TS line object format from select DB or TS line object	DB, TS
UCASE(ON OFF)	Use UCASE(ON) to switch to Upper Case only. Use UCASE(OFF) to switch to mixed upper/lower case for line object	
UNLOAD [[PARMUTIL= 'parmutil.dsn'] [LISTDEF= listdef.dsnname[(patt*)] [HLQ=ImageCopy-high- level-qualifier] [TEMPLATE= template.dsnname] [OPTIONS= options.dsnname]	Use UNLOAD command in DB2 v7 or above to generate UNLOAD DB2 utility JCL to unload from TS, TP, TB or image copy of TS and TP. Specify FROM IMAGECOPY option allows unload from image copy. Use HLQ to specify High-level-qualifier for the selected Image Copy Dataset. You can not use this option with LISEDEF. Specify ICGEN=gg and ICDATE=YYMMDD at end of the selected line object to specify the generation of image copy gg as the input to the UNLOAD, gg if specified, must be <= 0. Current generation of image copy is selected if no ICGEN specified. Specify ICDATE=YYMMDD to select a specific date of image copy as input to the UNLOAD. Specify field specification and WHEN specification if desired. Specify SYSREC and SYSPUNCH options to specify the SYSPUNCH and SYSREC dataset information. You can use this information to override the template specified in TEMPLATE command or default unload dataset name. TEMPLATE command if required if you use LISTDEF to select your UNLOAD DB2 objects.	TS, TP, TB, TSC, TPC, TBC
UPDATE [MAP='dclgen dsn'] [RUN='db2i2.run.output' COMMIT=# SQLTERM(?) CHAR=HEX]	Generate SQL UPDATE statements for specified line object. Use MAP option to generate embedded SQL. Use RUN option to generate UPDATE from RUN output. Use COMMIT=# to specify commit frequency. Use SQLTERM(?) to specify sql terminator. Use CHAR=HEX to generate CHAR or VARCHAR in HEX format.	TB, AL, SY, VW, MT, TBC
USAUTH [GRANTEE GRANTOR]	Display User Authorization for the selected line objects	US, RO
VIEWG [_] global-var-wo-& global-var-wo-&%]	Display existing Global Variables. Default without option, display all global variables. Specify global-var-wo-& to display specific global variable. Specify global-var-wo-&% to display all global variables prefixed with global-var-wo-%.	

<i>DB2I2 Command</i>	<i>DB2I2 Command Description</i>	<i>Allowed Line Object</i>
VIRTUAL [IX=ixcreator.ixname]	Generates SQL INSERT for your DSN_VIRTUAL_INDEX table to simulate WHATIF condition for creating a new index or dropping an existing index. The information can then be evaluated with EXPLAIN to see the impact of the proposed changes. Use IX=ixcreator.ixname to populate the SLQ INSERT with the information clone from an existing index.	N/A
VTOC [ODSN=output.dsname [SHORT]]	Use VTOC to invoke IBM IEHLIST LISTVTOC to list the VTOC information. Use SHORT option to list Empty Space information only. SHORT option can only be used with ODSN option.	DS
VW	Use VW to generate VW line (View Dependency information from select line object.	TB, VW, AL, SY, PL, PG, FU, SP, MT, TBC
ZPARM [dsn-zparm-name]	Display ZPARM information. Use dsnzparm-name option to display the zparm other than the one the DB2 sub-system is currently using.	

DB2I2 Command Description Detail

Drill Down command

Command Syntax:	[HOSTVAR for PG or DM line object to display Host variable information] [NOSTATS for PG or DM line object to disable display of statistic information, output can be used as input directly to EXPLAIN command without modification]
Line objects allowed:	DB, TS, TP, TB, CO, AL, SY, VW, IX, IP, PG, PL, SG, DM, RI (DT, TR, SP, FU, SH for V6 or above) (IS, ISP, MT, OI, SQ for V8 or above) (TSC,TPC,TBC,IXC,IPC,ISC,ISPC,TC,RO for V9 or above) (MS,PM for V10 or above) PG and DM appended with STMT# to display only a specific statement information from the selected PG or DM
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Next level line object
Reusable:	Yes

Command Description

- Use drill down command (select line objects, leave command line blank, and press HOT key) to display the next level of detail of the selected line object.
- If DB2 wild card % is used in the line object field, then DB2I2 display all line objects qualified on the same object level. For example, you use q.% to display all QMF tables.
- Use ? instead of S on the line object to display HELP information of the select line object.
- When there is no S specified, Drill down command allows you to edit or browse a non-blank cursor position dataset.

Example

The following screen displays results from various drilldown commands.

Drill down detail heading description for TS-TableSpace line:

BP-Bufferpool **Part**-Partition No **LK**-Lock Rule **Er**-Erase Rule **Im**-Implicit
Cl-Close Rule **Seg**-Segments **PSID**-Pageset ID

```
Command ==>> croll ==> CSR
***** Top of Data *****
000001 DB DSQDBCTL
==MSG>    Tablespace Name.....BP...Part.Lk.Er.Im.Cl.Seg.PSID.
000002 TS DSQDBCTL.DSQTSTCT1    BP0 0    P N N N 0 2
000003 TS DSQDBCTL.DSQTSTCT2    BP0 0    P N N N 0 4
000004 TS DSQDBCTL.DSQTSTCT3    BP0 0    P N N N 0 6
000005 TS DSQDBCTL.DSQTSGOV    BP0 0    P N N N 0 27
000006 TS DSQDBCTL.DSQTSLOG    BP0 0    P N N N 0 22
```

Drill down detail heading description for TP-Table Part & TB-Table line:

Part-Partition No **Pqty**-Primary Quantity **Sqty**-Secondary Quantity
T-StorType **Storname**-StorageGroup Name **Vcatname**-VCAT name
Type-Table Type **OBID**-Internal Object ID

```
Command ==>> Scroll ==> CSR
000002 TS DSQDBCTL.DSQTSTCT1    BP0 0    P N N N 0 2
==MSG>    Tablespace Name.....Part.Pqty.....Sqty..T.Storname.Vcatname
000003 TP DSQDBCTL.DSQTSTCT1    0 50    5    I DSQSGCT1 TDB2
==MSG>    TBcreator.TBname.....Type.OBID.DBname..TSname..
000004 TB Q.OBJECT_DIRECTORY    T 7    DSQDBCTL DSQTSTCT1
```

Drill down detail heading description for CO-Column line:

Attribute-Column Attribute **Length**-Column Length **Scale**-Column Scale **N**-Null
D-Default

```
Command ==>> Scroll ==> CSR
000004 TB Q.OBJECT_DIRECTORY    T 7    DSQDBCTL DSQTSTCT1
==MSG>    Column Name.....Attribute.Length...Scale...N.D.....
000005 CO OWNER    CHAR 8    0    N N
000006 CO NAME    VARCHAR 18    0    N N
000007 CO TYPE    CHAR 8    0    N N
000008 CO SUBTYPE    CHAR 8    0    Y Y
000009 CO OBJECTLEVEL    INTEGER 4    0    N N
000010 CO RESTRICTED    CHAR 1    0    N N
000011 CO MODEL    CHAR 8    0    Y Y
000012 CO CREATED    TIMESTMP 10    0    Y Y
```

Drill down detail heading description for IX-Index & IP-Index Part line:

Cnt-Index Column Count **L**-Clustering **U**-Unique Rule
E-Eraserule **C**-Closerule **BP**-Bufferpool
ISOBID-Index Object ID
Part-Partition No **Pqty**-Primary Quantity **Sqty**-Secondary Quantity
T-StorType **Storname**-StorageGroup Name **Vcatname**-VCAT name

```
==MSG>    Index Name.....Cnt.L.U.E.C.BP..ISOBID.TBcrator.TBname
000015 IX Q.OBJECT_DIRECTORYX    2 Y U N N BP0 9 Q    OBJECT
000016 IC OWNER    A
000017 IC NAME    A
==MSG>    IXcreator.Ixname    Part.Pqty.....Sqty..T.Storname.Vcatna
000018 IP Q.OBJECT_DIRECTORYX    0 25    5    I DSQSGCT1 TDB2
```

Drill down detail heading description for RI-Referential Integrity line:

Relname-Relation Name **D**-Delete Rule (C-Cascade, N-Set Null, R-Restrict)

```
000020 TB DSN8410.DEPT T 11 DSN8D41A DSN8S41D
==MSG> Parent_creator.Parent_name.Child_creator.Child_name Relname D
000021 RI DSN8410.DEPT.DSN8410.DEPT RDD C
000022 RI DSN8410.EMP.DSN8410.DEPT RDE N
000023 RI DSN8410.DEPT.DSN8410.EMP RED N
000024 RI DSN8410.DEPT.DSN8410.PROJ DEPTNO R
```

Drill down detail heading description for AL-Alias line:

T-Type **Tbcreator.TBname**-Base table creator and table name

```
000013 al DSN8%.%
000014 -- AL DSN8%.%
==MSG> Alias.Name.....T.Tbcreator.TBname..
000015 AL DSN8310.MIBEMP A DSN8310.EMP
```

Drill down detail heading description for SY-Synonyms line:

Type-Type(V-View, T-Table) **Tbcreator.TBname**-Base table creator and table name

```
000013 sy JRHJ%.%
000015 -- SY JRHJ%.%
==MSG> Synonyms Name.....Tbcreator.TBname.....Type
000016 SY JRHJ.SS1 JRHJ.V2 V
000017 SY JRHJ.SS2 JRHJ.TN0A012 T
000018 SY JRHJ.SS3 JRHJ.TN0A012 T
000019 SY JRHJ.S1 JRHJ.V1 V
000020 SY JRHJ.S2 JRHJ.V2 V
000021 SY JRHJ.TN3B018 CM97.TN3B018 T
000022 SY JRHJ.TN3B022 CM97.TN3B022 T
```

Drill down detail heading description for SG-StoGroup line:

Creator-Creator **Vcatname**-VSAM catalog name **Space**-space in K
Spcdate-Last SPACE date **CreateBy**-Created By

```
000032 sg g%
000033 -- SG G%
==MSG> STGROUP..Creator..Vcatname.Password.Space.....Spcdate.CreatedBy
000036 SG GPTMP001 JRHJ TDB2 489890 99193 JRHJ
000037 SG GPTST001 JRHJ TDB2 456622 98175 JRHJ
```

Drill down detail heading description for PL-Plan line:

Name-Plan Name **Creator**-Plan Creator **Date**-Bind date **Time**-Bind time
Vld-Valid **Opv**-Operative **Vlt**-Validate **Exl**-Explain
BoundBy-Bound by

```
000032 pl JRH%
000033 -- PL JRH%
==MSG> Name.... Creator. Date.. Time.... Vld Opv Vlt Exl BoundBy
000034 PL JRH0110 JRHJ 971124 10173150 N Y B N JRHJ
000035 PL JRH0120 JRHJ 980224 11041246 Y Y B N JRHJ
000036 PL JRH0130 JRHJ 971124 10190456 N Y B N JRHJ
000037 PL JRH0140 JRHJ 990203 12344896 Y Y B N JRHJ
000038 PL JRH0180 JRHJ 980911 08351400 Y Y B N JRHJ
000039 PL JRH0190 JRHJ 971205 11245241 N Y B Y JRHJ
```

Drill down detail heading description for PG-Package line:

Location-Location Name (Blank it is a local package) **Collid**-Collection ID
Name-Package Name **Version**-Version (Blank if no version)

```
000032 pg .JRHJ.%
000033 -- PG .JRHJ.%
==MSG>      Location.Collid.Name.Version.....
000034 PG .JRHJ.N2BR010.
000035 PG .JRHJ.N3BR018.
000036 PG .JRHJ.PC1002.
```

Drill down detail heading description for DM-DBRM line:

Planname-Plan Name **DBRM**-DBRM name **PDS Dataset Name**-PDS name contains DBRM

```
000032 dm UJH1110.%
000033 -- DM UJH1110.%
==MSG>      PlanName.DBRM.... PDS Dataset Name.....
000034 DM UJH1110.N3BU018 SYST.DBRMLIB
000035 DM UJH1110.JH1120 SYST.DBRMLIB
000036 DM UJH1110.JH1122 SYST.DBRMLIB
000037 DM UJH1110.JH1130 SYST.DBRMLIB
```

Use ? to select the line object to display detail line object HELP information:

```
Command ==>
***** Top of Data *****
?00001 TB SYSADM.DSN%

File Edit Edit_Settings Menu Utilities Compilers Test Help Db2i2
SSID(DBAG) Connect() Ucase(ON) Sysibm(SYSIBM) ----- DB2I2 10.0

Comma E
**** e @HELPTB ----- DB2I2 Line Object HELP SCREEN - TB ----- e
00000 e Command: _____ (Enter a DB2I2 command Listed below) e
?0000 e e
00000 e Line Object: TB Table e
00000 e Format TB tbcreator.tbname e
00000 e Example TB JDUSER.TESTTBL e
00000 e DB2I2 Commands Allowed: e
00000 e _ AL _ ALTER _ AUTH _ COAUTH _ COPYAUTH _ CREATE e
00000 e _ CURSOR _ DB _ DCLGEN _ DDL _ DELETE _ DRILLDWN e
00000 e _ DSNTIAUL _ DT _ FETCH _ FU _ GRANT _ IMPACT e
00001 e _ INSERT _ IP _ IPC _ IS _ ISC _ ISP e
00001 e _ ISPC _ IX _ IXC _ LISTDEF _ LOAD _ MIGR e
00001 e _ MT _ OI _ PG _ PL _ QBUILD _ REVOKE e
00001 e _ RI _ SELECT _ SELPATHV _ SP _ SPACE _ STATS e
00001 e _ SY _ TBAUTH _ TP _ TPC _ TR _ TS e
00001 e _ TSC _ UNLOAD _ UPDATE _ VW e
00001 e PF3=Exit ENTER=Process Command e
00001 D
000018 TS DDB2ADM.DSNOPTTS BP2 0 R N N N 4 2
000019 TS DDB2ADM.HASHTS BP2 1 R N N N 4 26
```

DB2I2 Reference Manual

The following example demonstrates how to edit a dataset. With cursor position any of the highlighted character below and issuer default drill down.

```
Command ==>,                               Scroll ==> CSR
000016 ts  odsn='DB2ADM.pds.cntl(ts)'
```

The following screen is displayed which allows you to edit or browse the selected non-blank field (a possible dataset name).

```
** ----- **
** No lines explicitly selected          **
** Choose line 00000016 displayed below as default? **
** ----- **
ts  odsn='DB2ADM.pds.cntl(ts)'
```

Please Enter one of the following option below:

```
Enter  Y to Accept as it as a Line Object
        1 to Browse 'DB2ADM.pds.cntl(ts)'
        2 to Edit   'DB2ADM.pds.cntl(ts)'
        N to Reject
```

AL

Command Syntax:	AL
Line objects allowed:	TB, VW, PL, PG, MT, TBC
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object – AL
Reusable:	Yes

Command Description

- Use DB2I2 AL command to display AL-Alias line from the selected TB, VW, PG or PL line.

Example

The following example demonstrates how to display Alias information about DB2 package JD00.N0AR006.

```

Command ==> AL                               Scroll ==> CSR
000006 pg .JD00.%
==MSG> >>>> Begin of FREE DDL
==MSG> Collid.Name..... Valid. Operative
s00007 PG .JD00.N0AR006                      Y      Y
000008 -- FREE Package Name: JD00.N0AR006
    
```

The result from the previous AL command shows that there are two alias used in the selected db2 package.

```

Command ==>                               Scroll ==> CSR
000006 pg .JD00.%
000007 PG .JD00.N0AR006                      Y      Y
000008 AL JD00.A1
000009 AL JD00.A2
    
```

ALTER

Command Syntax:	ALTER
Line objects allowed:	DB, TS, TB, IX, and SG (FU, SP for V6 or above) (SQ, CO for V8 or above) (MS, PM for v10 or above) A Display Utility line with UTILID = utility-id format
Process Mode:	Online only
Support	
Multiple line object:	No
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	DDL skeleton or DB2 ALTER UTILITY command skelton
Reusable:	Yes

Command Description

- Use ALTER command to assist you preparing DB2 ALTER DDL statements.
- The generated ALTER statements can then be executed with DB2I2 EXEC command.

Example

The following screens demonstrate how to use ALTER to generate ALTER DDL for a TB line object Q.COMMAND_SYNONYMS.

First, you select the table Q.COMMAND_SYNONYMS.

```
Command ==> ALTER Scroll ==> CSR
S00005 TB Q.COMMAND_SYNONYMS T 28 DSQDBCTL DSQTSSYN
```

DB2I2 displays the following screen assist you to generate ALTER TABLE DDL statements.

```
Command ==> Scroll ==> CSR
***** Top of Data *****
==MSG> 1. Please make any ALTER changes (any lower case field) and
==MSG> 2. Select the line objects and issue DB2I2 EXEC command to Execute these changes
000001 ALTER TABLE Q.COMMAND_SYNONYMS
000002 ADD colname coltype
000003 NOT NULL WITH DEFAULT
000004 constant/USER/CURRENT SQLID/NULL
000005 REFERENCES tablename
000006 ON DELETE RESTRICT/CASCADE/SET NULL
000007 constraint CHECK(check condition)
000008 FIELDPROC program-name (parms,parms)
000009 VALIDPROC pgmname/NULL
000010 AUDIT NONE/CHANGES/ALL
000011 ADD PRIMARY KEY (colname,colname)
000012 ADD FOREIGN KEY constraint (colname,colname)
000013 REFERENCES tablename
000014 ON DELETE RESTRICT/CASCADE/SET NULL
000015 ADD CONSTRAINT constraint CHECK(check condition)
000016 DATA CAPTURE NONE/CHANGES
000017 DROP PRIMARY KEY
```

You can make changes to the displayed lines and delete any line you do not needed. Once you done your editing select all the lines and use DB2I2 command EXEC to execute the ALTER DDL.

The following is an example of how to generate ALTER UTILITY db2 command skelton from a display utility command with EDIT=Y option.

```
Command ==> display util(*) edit=y                               Scroll ==> CSR
***** ***** Top of Data *****
000001
```

```
Command ==> ALTER                                             Scroll ==> CSR
***** ***** Top of Data *****
000001      DSNU105I -DB1G DSNUGDIS - USERID = SYSADM 973
000002      MEMBER = DB1G
S00003      UTILID = REORGCP
000004      PROCESSING UTILITY STATEMENT 1
000005      UTILITY = REORG
000006      PHASE = LOG      COUNT = 0
000007      STATUS = ACTIVE
000008      DSNU347I -DB1G DSNUGDIS - 974
000009      DEADLINE = NONE
000010      DSNU384I -DB1G DSNUGDIS - 975
000011      MAXRO = DEFER
000012      LONGLOG = CONTINUE
000013      DELAY = 1200 SECONDS
000014      DSNU383I -DB1G DSNUGDIS - CURRENT ITERATION NUMBER = 4 976
000015      WRITE ACCESS ALLOWED IN THIS ITERATION = YES
000016      ITERATION BEFORE PREVIOUS ITERATION:
000017      ELAPSED TIME = 00:00:00
```

```
Command ==>                                               Scroll ==> CSR
***** ***** Top of Data *****
==MSG> 1. Please make any ALTER changes (any lower case field)
==MSG> 2. Select edited lines to be processed(with S or SS line command), and
==MSG> 3. Issue DB2I2 command DB2CMD to Execute these changes
000001 -ALTER UTILITY (REORGCP) REORG      -
000002 -- Deadline option
000003      DEADLINE(NONE|timestamp)      -
000004 -- Max Read Only in seconds
000005      MAXRO(DEFER|integer)          -
000006 -- Long Log option
000007      LONGLOG(CONTINUE|TERM|DRAIN)   -
000008 -- Delay option in seconds for LONGLOG action
000009      DELAY(integer)
***** ***** Bottom of Data *****
```

You can make changes to the displayed lines and delete any line you do not needed. Once you done your editing select all the lines and use DB2I2 command DB2CMD to execute the modified ALTER UTILITY command.

AUTH

Command Syntax:	AUTH [GRANTEE GRANTOR]
Line objects allowed:	AL, BP, CL, DB, PG, PL, SG, TS, and TB, US, VW (DT,FU,SH,SP,TR for V6 or above) (SQ, MT for V8 or above) (RO, TBC for V9 or above)
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 AUTH command to display authorization information for the selected line objects.
- You can choose either GRANTEE or GRANTOR information to display by specify GRANTOR or GRANTEE as command option on the command line. If you do not specify this command option, GRANTEE is used as default command option.

Example

The following screens demonstrate how to display table/column authorization information for a QMF table Q.OBJECT_DIRECTORY.

```
Command ==> auth                               Scroll ==> CSR
S00004  TB Q.OBJECT_DIRECTORY          T    7  DSQDBCTL DSQTSCT1
```

The results from the previous AUTH command displayed below.

```
Command ==>                                     Scroll ==> CSR
000004  TB Q.OBJECT_DIRECTORY          T    7  DSQDBCTL DSQTSCT1
000005                                     U                               R
000006                                     P                               E
000007                                     D                               F
000008                                     A   D   I   S   U   E
000009                                     T   A   I   N   E   P   R
000010                                     E   L   L   N   S   L   D   E
000011                                     C   T   E   D   E   E   A   N
000012                                     O   E   T   E   R   C   T   C
000013  GRANTEE  GRANTOR  TCREATOR.TNAME    L   R   E   X   T   T   E   E
000014  -----
000015  US DBXLQMF  DBEXCEL  Q.OBJECT_DIRECTORY          Y
000016  US DBXLQMF  DBEXCEL  Q.OBJECT_DIRECTORY          Y
000017  US DBXLQMF  DBEXCEL  Q.OBJECT_DIRECTORY          Y
000018
000019  GRANTEE  GRANTOR  CREATOR.TNAME      COLNAME
000020  -----
000021  US DSQ9SV   DB2ADM   Q.OBJECT_DIRECTORY          OBJECTLEVEL
```

BATCH

Command Syntax:	BATCH <u>db2i2-command</u> [db2i2-command-option] or BATCH ICMD=db2i2-command.input.dataset or BATCH ICMD= prepared-JCL [JCL=N] [Host-variable substitution] [CLI=MC UTIL clidsn(mem)][JOBDD=N]
Line objects allowed:	allowed line object used with db2i2-command
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	N/A
Output Type:	Depends on the selected db2i2-command
Reusable:	Depends on the selected db2i2-command

Command Description

- Use DB2I2 BATCH command to generate JCL to run most of the DB2I2 commands in batch mode. This command is extremely useful if you request a long running DB2I2 command such MIGR, LISTC, or STATS and you do not want to tie up your work station.
- Process DB2I2 command in batch mode allows you to stream line the process and execute multiple DB2I2 commands with one job execution.
- By using the IDSN and ODSN options, you can treat the output of one DB2I2 command as the input of the next DB2I2 command.
- Some of the advanced functions, such as UDF, are only available in batch mode through ICMD option. To use this facility, you prepare and save a set of DB2I2 commands in a file, and then use them in batch mode with ICMD option. Each time you invoke these UDF, You can assign different host variables (Host variable is defined as &VARNAME=H-VALUE-H all upper case to avoid conflict with existing global variables and/or command options) to it.
- In online mode, when you select line object for BATCH command, the length of the selected line objects must not greater than 80 characters long. If you have line objects which are > 80 bytes long, save these line objects in a file and use IDSN= option to invoke them. Use IDSN= and ODSN= to assign the input line object and output file for the BATCH command.
- Use IDSN*= and ODSN*= to assign the input line object and output file for the generated db2i2-command JCL.
- Because the generated JCL from DB2I2 BATCH command is 80 bytes long, if the line object selected is longer than 80 bytes, you should use IDSN=*DUMMY and IDSN*=input.line.object to assign the input line object and avoid the line object truncation.
- If ICMD input is a prepared JCL file, you can use JCL=N to disable the generation of unnecessary job cards and DD names which include SSID, DB2I2CMD and LINEOBJ. You can use this option to interface with third vendor party tool.
- DB2I2 supports Callable interface in BATCH mode. There are 2 sample CLI modules shipped with the product: MCCLI and UTILCLI.
- By default, DB2I2 BATCH executes one command against multiple line objects.

```

//DB2I2CMD DD *
DB2i2 command 1
DB2i2 command 2
//LINEOBJ DD *
line object 1
line object 2
line object 3
cmd1 ----> lobj1, lobj2, lobj3

```

cmd2 ----> lobj1, lobj2, lobj3

- Specify CLI=MC option to direct DB2I2 BATCH to execute multiple DB2I2 commands against line object one at a time.

```
Cmd1 , cmd2 ----> lobj1
Cmd1 , cmd2 ----> lobj2
Cmd1 , cmd2 ----> lobj3
Cmd1 , cmd2 ----> lobj4
```

- Using CLI=UTILCLI to generate Db2 utility one line object at a time with options to automatically submit the generated JCL. (See appendix for detail description of UTILCLI)
- You can use CLI=clidsn(mem) to use your own REXX or CLIST as pre-process and post-process. The clidsn should be defined as fixed length file **255 bytes long**. (Please refer to sample CLI routine for detail)
- Specify JOBDD=N to suppress the generation of the following //JOB CARD. This DD is not required if you do not want to generated JOBS from this BATCH command.

```
//JOB CARD DD DATA,DLM=JJ
```

```
.
JJ
```

- To invoke DB2I2 command in BATCH mode, you enter BATCH together with one of the DB2I2 command and press HOT key.

Example

The following example demonstrates how to run DB2I2 MIGR command to migrate Q.COMMAND_SYNONYMS QMF system table in batch mode.

```
Command ==> BATCH MIGR                               Scroll ==> CSR
S00005 TB Q.COMMAND_SYNONYMS          T 28  DSQDBCTL DSQTSSYN
```

The following pop up screen allows you to select migration options. These options include View, Alias, Synonyms, Bind and Grant. Select only those objects you need, so that you do not have to waste valuable CPU cycles.

```
Command ==> BATCH MIGR                               Scroll ==> CSR
000101 TB Q.COMMAND_SYNONYMS          T 28  DSQDBCTL DSQTSSYN
000102
000103 TB s .-----#MIGR ----- DB2I2 MIGR PROCESS OPTIONS ----- D
000104 |
000105 | #MIGR ----- DB2I2 MIGR PROCESS OPTIONS -----
000106 | Option Selection (Y/N)
000107 | ; VIEW Y
000108 VW ALIAS N
000109 VW SYNONYMS N
000110 VW BIND PLAN/PACKAGE Y
000111 VW GRANT N
000112 VW REFERENTIAL INTEGRITY B B-Both P-Parent C-Child N=None
000113 VW PF3=Exit ENTER=Process Your Selection
000114 VW
000115 PL UHP0113 ADMEUC.MDL_REF_H016
```

DB2I2 produces a migration JCL similar to the one display below.

```
Command ==>
000020 //SYSTSPRT DD SYSOUT=*
000021 //SYSTSIN DD *
000022 ISPSTART CMD(DB2I2)
000023 /*
000024 //SSID DD *
000025 DSN\ 5 SYSIBM
000026 //DB2I2CMD DD *
000027 MIGR VW=Y BIND=Y
000028 //LINEOBJ DD *
000029 TB Q.COMMAND_SYNONYMS T 28 DSQDBCTL DSQTSSYN
```

The DDNAME used in the batch DB2I2 consists of the following:

- **SSID** contains the information about DB2 subsystem-ID, location name, DB2 version, Catalog table prefix, as well as the restart command line information in the following format:
SSID(location-name DB2-version catalog-table-prefix [command restart line no])
The optional restart line no allows you to start the same batch job from different position Within the same set of DB2I2 Commands in DB2I2CMD.
- **DB2I2CMD** contains all DB2I2 commands for a batch run
(Because BATCH command is implemented as ISPF/TSO background job, to allow DB2I2 work properly, you either need to specify USER=a-true-TSOID on your job card, or specify TSO PROFILE PREFIX(Dataset-prefix) as the first DB2I2 command to allow authorization to create, updates and delete the work files needed for DB2I2 process)
Under DB2I2CMD, besides all regular DB2I2 command, you can use ICMD option to include UDF. UDF-user defined function, is a set of commands, prepared and saved in a file, which can contain host variables, and to be used with ICMD to perform a specific function. You can assign different value to host variables each time you invoke UDF. This facility allows you to save and reuse a set of commands to be used for specific function.

There are some limitations to use the ICMD option:
 - You can not embed an ICMD option within another ICMD option.
 - At the time when you submit the batch job, the ICMD option needs to be resolved. The data set needs to be presented. You can not dynamically build the data set and use it through ICMD at the time when you run the batch job.
- **LINEOBJ** is the default input DDNAME for the line objects.
(Use IDD command option to override the default line object DDNAME)

The previous example does the following function:

- Execute MIGR command against a table Q.COMMAND_SYNONYMS.
- The generated script contains the BIND and VIEW option.
- The current DB2 SSID is DSN with no connection location name specified.
- The DB2 catalog table prefix is SYSIBM.
- Line objects are entered under the default DDNAME LINEOBJ.

The following example demonstrates the advanced feature of batch interface. It allows you to execute multiple DB2I2 commands in sequence in one batch execution.

The intend of the following example is to generate a set of DB2 COPY utility JCL which include all the tablespace from the database DBTEST01 that does not have any full image copy ever taken.

The following example does the following functions:

- Executes a REXX Exec to delete T1, T2, and T3. The source of the REXX Exec is from the DDNAME REXXD1. Set CC to 0 when return back DB2I2 to avoid ABEND.

- Executes DB2I2 command Drill down to display all tablespace information with database DBTEST01. The input line object is from default DDNAME LINEOBJ, with ODSN=T1 option, DB2I2 writes drill down output to T1.
- Executes DB2I2 command STATS to display statistics information. IDSN=T1 and ODSN=T2 command options directs DB2I2 to use T1 as input and T2 as output for the STATS command. Request T=N to disable heading and footing print so that we can reuse the output. Request TSIX to output the result in TP, IP line object format.
- Execute FLIST to print T1 and T2.
- Invoke EDIT to edit T2.
- Excludes all line objects in T2 except the one contains 'NO FULL'.
- Save the edited file T2 by using the END_EDIT command.
- Invoke COPY DB2I2 command to generate the FULL IMAGE COPY Utility JCL. The input line object is from T2. The output of the run is in T3. The COPY parameter is retrieved from file UTIL.CNTL(COPY), which is prepared in advance with a PARMUTIL command.
- Use FLIST to print the T3.

```
//DB2I2CMD DD *
REXX IDD=REXXDD1
ODSN=T1
STATS IDSN=T1 ODSN=T2 T=N TSIX
FLIST T1
FLIST T2
EDIT T2
X ALL
F 'NO FULL' ALL
DEL X ALL
END_EDIT
COPY IDSN=T2 UTIL.CNTL(COPY) ODSN=T3
FLIST T3
//LINEOBJ DD *
DB DBTEST01
//REXXDD1 DD DATA,DLM=AA
/* REXX */
ADDRESS TSO "DELETE T1"
ADDRESS TSO "DELETE T2"
ADDRESS TSO "DELETE T3"
RETURN 0
AA
```

The following JCL invokes a user-defined function TUNETB, which provides all tuning information for a TB-table line object:

```
//DB2I2CMD DD *
  BATCH ICMD=*TUNETB          +
  ODSN=TEMP.OUT              +
  JOB#=10                    +
  &O1=O1                     +
  &SUFF=01                    +
  &TSOID=JRHJ                 +
//LINEOBJ DD *
TB JRHJ.TCUS01
```

TUNETB is a system UDF, which shipped with DB2I2 and contains the following predefined DB2I2 commands:

```
-- -----
-- TUNETB2: Tuning Assist Using TB Line Objects as Input      --
--   sample for ICMD host variable substitute                 --
--   File Usage:                                             --
--     &O1 - Output file .....                               --
--     &SUFF-File suffix for cuncurrent process.....        --
--     &TSOID - your TSOID .....                             --
--     WS - work file for TS and IX command .....           --
--     W1 - work file for PG command .....                   --
```

```

--      W2 - work file for DELETE FROM PLAN_TABLE....      --
--      W3 - work file for EXPLAINP.....                  --
--      W4 - work file for BIND COPY.....                  --
--      W5 - work file for BATCH .....                    --
--      W6 - work file for FREE PACKAGE.....                --
-----
REXX      IDD=DELWS
REXX      IDD=DELO
REXX      IDD=ALLOCO
SELPATHV  ODSN=&O1
TS        ODSN=WS&suff
STATS     IDSN=WS&suff ODSN=&O1 APPEND
IX        ODSN=WS&suff NOTFOUND(SKIP 7)
IDSN=WS&suff ODSN=&O1 APPEND
PG        ODSN=W1&suff NOTFOUND(SKIP 15)
EDIT     W1&suff
DISTINCT 1,30
END_EDIT
IDSN=W1&suff ODSN=&O1 APPEND
REXX     IDD=SQLDEL
EXEC     IDSN=W2&suff
BIND COPY O=USER CL=USER IDSN=W1&suff ODSN=W4&suff
EDIT     W4&suff
C 'EXPLAIN(NO)' 'EXPLAIN(YES)' ALL
C '(ADD)' '(REPLACE)' ALL
END_EDIT
EXEC     IDSN=W4&suff
EXEC     IDSN=W6&suff
BATCH   ICMD=W3&suff ODSN=W5&suff
TSO SUBMIT W5&suff
REXX     IDD=DELWS
//DELWS DD DATA,DLM=AA
/* ---REXX-----
DELWS: Delete work files
----- */
ADDRESS TSO
'DELETE WS&suff'
'DELETE W1&suff'
'DELETE W2&suff'
'DELETE W3&suff'
'DELETE W4&suff'
'DELETE W5&suff'
'DELETE W6&suff'
RETURN 0
AA
//DELO DD DATA,DLM=AB
/* ---REXX-----
DELO: Delete Output file
----- */
ADDRESS TSO
'DELETE &O1'
RETURN 0
AB
//ALLOCO DD DATA,DLM=BB
/* ---REXX-----
ALLOCO: Allocate Output file
----- */
ADDRESS TSO
"ALLOC FI(&O1) DS(&O1) NEW SPACE(1,1) TRACKS RECFM(F B) LRECL(133)",
"BLKSIZE(0) REUSE UNIT(SYSDA)"
"FREE FI(&O1)"
RETURN 0
BB
//SQLDEL DD DATA,DLM=CC
/* ---REXX-----
SQLDEL: Generates DELETE FROM PLAN_TABLE... in W2
Generates EXPLAINP commands ..... in W3
Generates FREE commands ..... in W6
Input W1 contains PG line objects
----- */
ADDRESS TSO
"ALLOC FI(W1) DS(W1&suff) SHR REUSE"
"ALLOC FI(W2) DS(W2&suff) NEW SPACE(1,1) TRACKS RECFM(F B) LRECL(80)",
"BLKSIZE(0) REUSE UNIT(SYSDA)"
"ALLOC FI(W3) DS(W3&suff) NEW SPACE(1,1) TRACKS RECFM(F B) LRECL(80)",
"BLKSIZE(0) REUSE UNIT(SYSDA)"
"ALLOC FI(W6) DS(W6&suff) NEW SPACE(1,1) TRACKS RECFM(F B) LRECL(80)",
"BLKSIZE(0) REUSE UNIT(SYSDA)"

```

DB2I2 Reference Manual

```
OUTREC.1="DELETE FROM PLAN_TABLE"
OUTREC.2=" WHERE PROGRAM IN ("
"EXECIO * DISKR W1(STEM INREC. FINIS"
J = 2
JJ= 0
DLM=' '
DO I = 1 TO INREC.0
  PARSE VALUE INREC.I WITH J1 '.' J2 '.' WS '.'
  J = J + 1
  OUTREC.J =DLM "'STRIP(WS)'"
  JJ= JJ+ 1
  OUTREC2.JJ ="EXPLAINP ODSN=&O1 APPEND PG="STRIP(WS)
  /* ----- */
  /* Modify the following &TSOID to your TSOID */
  /* ----- */
  OUTREC6.JJ ="DB2CMD FREE PACKAGE(&TSOID."STRIP(WS) ")"
  DLM=', '
END
J = J + 1
OUTREC.J=" );"
"EXECIO" J "DISKW W2(STEM OUTREC. FINIS"
"EXECIO" JJ "DISKW W3(STEM OUTREC2. FINIS"
"EXECIO" JJ "DISKW W6(STEM OUTREC6. FINIS"
"FREE FI(W1)"
"FREE FI(W2)"
"FREE FI(W3)"
"FREE FI(W6)"
RETURN 0
CC
```

BIND

Command Syntax:	BIND [CL=collection] [O=Owner] [Q=Qualifier] [MEM=*] [GRANT=Y] [EXPLAIN=Y]
Line objects allowed:	PG, PL
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	DB2 command
Reusable:	Yes

Command Description

- Use DB2I2 BIND command to generate DB2 BIND command statements from selected DB2 plan or package.
- The GRANT DCL associated with the selected packages and plans are also generated together with the DB2 BIND command.
- Specify CL, O, Q, and MEM options to generate DB2 BIND command with
 - a new collection ID
 - a new Owner ID
 - a new Qualifier
 - MEM=* package list option (for BIND PACKAGE only).
- Use CL=USER option to assign your TSOID as the collection ID.
- Use O=USER option to assign your TSOID as the owner field.
- Use GRANT=Y option to generate GRANT statements after the BIND command
- Use EXPLAIN=Y option to generate EXPLAIN(YES) BIND option.

Example

The following example demonstrates how to generate DB2 BIND command for DB2 package Q.DSQ8UPRF, with Q2 as new collection ID and Q2 as new owner name. Use GRANT=Y option to generate GRANT statements after the generated BIND command.

```
Command ==> BIND O=Q2 CL=Q2 GRANT=Y
000037 TB Q.PROFILES          T 18   DSQDBCTL DSQTSPRO
S00038 PG .Q.DSQ8UPRF
000039 PG .Q.DSQ9ICVS
000040 PG .Q.DSQ8ICVS
```

```
-- BIND Package: Q.DSQ8UPRF
-- Package Authorization
***
```


DB2I2 Reference Manual

The following screen displays the results from the previous BIND command. Notices that the GRANT DCL statements are also generated.

```
Command ===>                               Scroll ===> CSR
000037 TB Q.PROFILES                          T 18   DSQDBCTL DSQTSPRO
==MSG> >>>> Begin of BIND/REBIND DDL
000038   PG   .Q.DSQ8UPRF
000039 -- BIND Package: Q.DSQ8UPRF
000040 DB2CMD BIND PACKAGE(Q2) -
000041 DB2CMD OWNER(Q2) QUALIFIER(Q) -
000042 DB2CMD MEMBER(DSQ8UPRF) LIBRARY('SYSP.TEST.DSQDBRM') -
000043 DB2CMD ACTION(ADD) CURRENTDATA(NO) DEGREE(1) -
000044 DB2CMD ENABLE(*) -
000045 DB2CMD EXPLAIN(NO) FLAG(I) ISOLATION(CS) -
000046 DB2CMD SQLERROR(NOPACKAGE) VALIDATE(BIND)
000047
000048 -- Package Authorization
000049 SET CURRENT SQLID = 'ADM1';
000050 GRANT
000051     EXECUTE
000052     ON PACKAGE Q2.DSQ8UPRF TO PUBLIC
000053 ;
==MSG> >>>> End   of BIND/REBIND DDL
```

BIND COPY

Command Syntax:	BIND COPY [CL=collection] [O=Owner] [Q=Qualifier] [GRANT=Y] [OPTIONS(<u>COMPOSITE</u> COMMAND) v7 or above] [EXPLAIN=Y]
Line objects allowed:	PG
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	Yes
Output Type:	DB2 command
Reusable:	Yes

Command Description

- Use DB2I2 BIND COPY command to generate DB2 BIND COPY commands from an existing DB2I2 package line object.
- Use CL= collection option to override the collection ID option. Specify CL=USER to override the Collection ID with your TSOID.
- Use O option to override the OWNER option. Specify O=USER to override the Owner with your TSOID.
- Use Q option to override the QUALIFIER option.
- Use GRANT=Y option to generate GRANT statements after the BIND command
- Use OPTIONS(COMPOSITE) or OPTIONS(COMMAND) to select this BIND COPY option. Use COMPOSITE option to select the existing bind options. Use COMMAND option to set to use the current bind default options.
- Use EXPLAIN=Y option to generate EXPLAIN(YES) BIND option.

Example

The following example demonstrates how to generate DB2 BIND PACKAGE command for DB2 package Q.DSQ8UPRF and copy it to JD00.

```
Command ==> BINDCOPY O=JD00 CL=JD00                               Scroll ==> CSR
000037 TB Q.PROFILES T 18 DSQDBCTL DSQTSPRO
S00038 PG .Q.DSQ8UPRF
```

DB212 Reference Manual

The result from the previous BIND COPY command displayed below.

```
Command ==>
000037 TB Q.PROFILES          T 18   DSQDBCTL DSQTSPRO
000038 PG  .Q.DSQ8UPRF
000039 -- BIND Package: JD00.DSQ8UPRF
000040 DB2CMD BIND PACKAGE(JD00) -
000041 DB2CMD OWNER(JD00) QUALIFIER(Q) -
000042 DB2CMD COPY(Q.DSQ8UPRF) -
000043 DB2CMD ACTION(ADD) CURRENTDATA(NO) DEGREE(1) -
000044 DB2CMD ENABLE(*) -
000045 DB2CMD EXPLAIN(NO) FLAG(I) ISOLATION(CS) -
000046 DB2CMD SQLERROR(NOPACKAGE) VALIDATE(BIND

Scroll ==> CSR
```

CANCEL [DDF] THRAED

Command Syntax:	CANCEL THREAD CANCEL DDF THREAD [DUMP] [NOBACKOUT]
Line objects allowed:	A display tread line with <u>Numeric Token</u> as the last field on the line
Process Mode:	Online
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Result of the Cancel [DDF] Thread command
Reusable:	Yes

Command Description

- Use DB2I2 CANCEL [DDF] THREAD command to issue DB2 CANCEL[DDF] THREAD from a Display Thread output line with Numeric Token as the last field on the display line.
- You should use DISPLAY with EIDT=Y option to get access to the display line in edit mode.
- Use DUMP or NOBACKOUT options if desired for CANCEL DDF THREAD.

Example

```

Command ==> display thread(*) edit=y                               Scroll ==> CSR
***** ***** Top of Data *****
000001

Command ==> CANCEL THREAD                                         Scroll ==> CSR
***** ***** Top of Data *****
==MSG> -CAUTION- Profile changed to CAPS OFF (from CAPS ON) because data
==MSG>          contains lower case characters.
000001 DSNV401I  -DSN DISPLAY THREAD REPORT FOLLOWS -
000002 DSNV402I  -DSN ACTIVE THREADS -
000003 NAME      ST A  REQ ID      AUTHID   PLAN      ASID  TOKEN
S00004 DISCONN  DA *   505 NONE          NONE     DISTSERV 004A 74314
000005 V471-USHON000.U002DSN.BFB496965B15=74314
000006 SERVER   RA *   207 MSACCESS.EXE CP01     DISTSERV 004A 74312
000007 V437-WORKSTATION=TORAHLHHTLN81, USERID=cp01,
000008          APPLICATION NAME=MSACCESS.EXE
000009 V445-CF821C7B.03E7.061114182330=74312 ACCESSING DATA FOR
000010          207.130.28.123
000011 SERVER   RA *   186 w3wp.exe      CA@83     DISTSERV 004A 74307
000012 V437-WORKSTATION=AHFCPRIQS02, USERID=CA@83,
000013          APPLICATION NAME=w3wp.exe
000014 V445-CF821C7B.03E3.061114182259=74307 ACCESSING DATA FOR
000015          207.130.28.123
000016 SERVER   RA *   5195 IRPW23_5_5.e CDRPTEST DISTSERV 004A 73902

Command ==>                                                       Scroll ==> CSR
***** ***** Top of Data *****
DSNV426I  -DSN DSNVCT THREAD '74314' HAS BEEN CANCELED
***** ***** Bottom of Data *****

```

CHECK

Command Syntax:	CHECK [[PARMUTIL=]'parmutil.dsn'] [DFLTSP=(1,1 pri,sec)] [FNM=ID] [WKUNIT=TAPE_0 #] [LISTDEF=listdef.dsname[(patt*)]] [OPTIONS=options.dsname]
Line objects allowed:	TS, IX, TP, and IP TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use DB2I2 CHECK command to generate CHECK DATA, CHECK INDEX or CHECK LOB DB2 utility JCL.
- Any necessary work files associated with the CHECK utility are also generated with the appropriated space allocation specified.
- For DB2 v7 or above, you can specify LISTDEF option with PDS member pattern match. For example, By specifying LISTDEF=listdef(DB1*) will generate CHECK DB2 utility JCL for all members name prefix with DB1. Because DB2I2 process each LISTDEF one at a time, If you select to use LISTDEF pattern option, it will prompt selection screen one for each LISTDEF member. To avoid this inconvenience, you can use PARMUTIL command to generate PARMUTIL control and then use it as process option.
- Specify DFLTSP option to assign the space allocation information for DB2 CHECK work files.
- By default, DB2I2 constructs work file name using up to 8 character of the object creator/name information. Use FNM=ID to direct DB2I2 use DBID/PSID,OBID,ISOBID when construct work file name.
For example, TS db1.ts1 with dbid= 200 and psid=208
When no FNM specify the following name will be generated:
//SYSUT1 DD DSN=dspre.db1.ts1...
With FNM=ID, the following name will be generated:
//SYSUT1 DD DSN.dspre.DB000200.PS000208...
- By default, DB2I2 uses DASD for work files. Specify WKUNIT=TAPE,# to request work file to be generated as UNIT=TAPE if estimated primary allocation for work file is greater than #. # if not specified, is default to 0.

Example

The following example demonstrates how to generate CHECK DATA utility JCL for Tablespace DSQDBCTL.DSQTSPRO.

```
Command ==>> CHECK
000037 TB Q.PROFILES          T 18  DSQDBCTL DSQTSPRO          Scroll ==>> CSR
S00038  TS DSQDBCTL.DSQTSPRO
```

A pop up screen displayed below allows you to select either CHECK DATA or CHECK INDEX option. The following screen demonstrates CHECK DATA option is selected to generate JCL to run Check utility.

```
#CHECKA ----- DB2I2 CHECK DATA/INDEX/LOB PROCESS OPTIONS -----
Check Option: 1 (1-Check Data 2-Check INDEX 3-Check LOB)
PART (0-4096) _____ SHRLEVEL (REFERENCE/CHANGE) REFERENCE
DRAIN_WAIT (0-1800) _____ RETRY (0-255) _____ RETRY_DELAY _____ (1-1800)

1. Check DATA
SCOPE PENDING_____ (PENDING/ALL/AUXONLY/REFONLY/XMLSCHEMAONLY)
AUXERROR REPORT_____ (REPORT/INVALIDATE)
LOBERROR REPORT_____ (REPORT/INVALIDATE)
XMLERROR REPORT_____ (REPORT/INVALIDATE)
FOR EXCEPTION
DELETE - (Y/N/ ) LOG - (Y/N)
EXCEPTIONS 0_____ (0-9999) 0-Unlimited
INCLUDE XML TS - (Y/N)
2. Check INDEX
INDEX _____ (ALL)/
3. Check LOB
EXCEPTIONS _____ (0-9999) 0-Unlimited

SORTDEVT SYSALLDA SORTNUM 3__ (2-255)
CLONE _ (Y/N For Option 1,2)

PF3=Exit ENTER=Process Your Selection
```

The following screen displays the result JCL generated from the previous CHECK DATA command.

```
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//UTPRINT DD SYSOUT=*
//SORTOUT DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSERR DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSIN DD *
CHECK DATA
TABLESPACE DSQDBCTL.DSQTSPRO
SCOPE PENDING
AUXERROR REPORT
LOBERROR REPORT
XMLERROR REPORT
SORTDEVT SYSALLDA
SORTNUM 3
SHRLEVEL REFERENCE
```

COAUTH

Command Syntax:	COAUTH [GRANTEE GRANTOR]
Line objects allowed:	AL, TB, US, VW, MT, RO
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 COAUTH command to display column authorization information for selected line objects.
- The GRANTEE or GRANTOR option allows you to display either the GRANTEE or GRANTOR information of selected line object. The default option is GRANTEE.

Example

The following screen demonstrates how to display column authorizations for a QMF table Q.OBJECT_DIRECTORY.

```
Command ==>> coauth                               Scroll ==>> CSR
S00004  TB Q.OBJECT_DIRECTORY           T    7    DSQDBCTL DSQTSCT1
```

The result from the previous COAUTH command indicates that the column name OBJECTLEVLE has been granted to DSQ9SV by DB2ADM.

```
Command ==>>                               Scroll ==>> CSR
000004  TB Q.OBJECT_DIRECTORY           T    7    DSQDBCTL DSQTSCT1
000019      GRANTEE  GRANTOR  CREATOR.TNAME           COLNAME
000020  -----
000021  US DSQ9SV   DB2ADM   Q.OBJECT_DIRECTORY           OBJECTLEVEL
```

CONNECT(location) & CONNECT(RESET)

Command Syntax:	CONNECT(location-name) or CONNECT(RESET) or CONNECT(?)
Line objects allowed:	None
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	N/A

Command Description

- Use CONNECT command to connect to a remote location.
- The location name needed to be defined in the DB2 SYSIBM.SYSLOCATIONS (for DB2 V4) or SYSIBM.LOCATIONS table (for DB2 V5).
- The location name also needed to be specified in the ‘db2i2.clist.library(SSID)’ system ID setup file during the installation.
- Use CONNECT(RESET) to reset the connection to the default local location.
- If you do not know the name of the location, you can use CONNECT(?) to list all available location names.

Example

The following example Use CONNECT(?) to list all available locations.

```
Command ==>> connect (?)                               Scroll ==>> CSR
```

Result from the previous CONNECT(?) command displayed below.

```
** ----- Connection Information -----**
** Location: ACSCDB2A SSID: DB2A
** Location: ACSCDB2P SSID: DB2P
** Location: ACSCDB2D SSID: DB2D
** Location: ACSCDB2T SSID: DB2T
** Location: ACSCDB2M SSID: DB2M
**
```


The following example issue CONNECT(ACSCDB2P) to connect to a remote location ACSCDB2P.

```

Command ==> connect(acscdb2p) Scroll ==> CSR
***** Top of Data *****
==MSG> /* -----*/
==MSG> /* DB2I2 DB2 Catalog Interface Tool Box */
==MSG> /* By JRH GoldenState Software, Inc. */
==MSG> /* (C) Copyrighted 1997-2006 */
==MSG> /* Licensed to */
==MSG> /* DB2I2 Environment - SSID(DB2D) CONNECT( ) SYSIBM(SYSIBM) */
==MSG> /* -----*/

```

Result from the previous command displayed below.

```

EDIT JRHJ00.T41 (DB2D) (ACSCDB2P) (SYSIB
Command ==> Scroll ==> CSR
***** Top of Data *****
==MSG> /* -----*/
==MSG> /* DB2I2 DB2 Catalog Interface Tool Box */
==MSG> /* By JRH GoldenState Software, Inc. */
==MSG> /* (C) Copyrighted 1997-2011 */
==MSG> /* Licensed to */
==MSG> /* DB2I2 Environment - SSID(DB2D) CONNECT(ACSCDB2P ) SYSIBM(SYSIBM) */
==MSG> /* -----*/

```

The following example issue CONNECT(RESET) to reset connection.

```

EDIT JRHJ00.T41 (DB2D) (ACSCDB2P) (SYSIB
Command ==> CONNECT(reset) Scroll ==> CSR
***** Top of Data *****
==MSG> /* -----*/
==MSG> /* DB2I2 DB2 Catalog Interface Tool Box */
==MSG> /* By JRH GoldenState Software, Inc. */
==MSG> /* (C) Copyrighted 1997-2011 */
==MSG> /* Licensed to */
==MSG> /* DB2I2 Environment - SSID(DB2D) CONNECT(ACSCDB2P ) SYSIBM(SYSIBM) */
==MSG> /* -----*/

```

Result from the previous command displayed below.

```

Command ==> Scroll ==> CSR
***** Top of Data *****
==MSG> /* -----*/
==MSG> /* DB2I2 DB2 Catalog Interface Tool Box */
==MSG> /* By JRH GoldenState Software, Inc. */
==MSG> /* (C) Copyrighted 1997-2011 */
==MSG> /* Licensed to */
==MSG> /* DB2I2 Environment - SSID(DB2D) CONNECT( ) SYSIBM(SYSIBM) */
==MSG> /* -----*/

```

COPY

Command Syntax:	COPY [[PARMUTIL=] ['copy.parmutil.dsname'] [DSPRE=datasetprefix[TSOID] [LISTDEF=listdef.dsname[(patt*)]] [TEMPLATE=template.dsname] [OPTIONS=options.dsname]
Line objects allowed:	TS, TP IX, IP, IS, ISP for db2 v6 or above TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use DB2I2 COPY command to generate utility JCL to run DB2 Full Image COPY, Incremental COPY or Merge COPY against selected Tablespace or Tablespace Partition line objects. And Index and Index Part for db2 v6 or above.
- Use 'copy.parmutil.dsname' option to direct DB2I2 to read the COPY options from the specified file. The 'copy.parmutil.dsname' file is prepared with PARMUTIL command before you issue the COPY command.
- Use DSPRE to set the Dataset prefix for the generated utility work files. The default Dataset prefix is your TSOID.
- For DB2 v7 or above, you can specify LISTDEF option with PDS member pattern match. For example, By specifying LISTDEF=listdef(DB1*) will generate COPY DB2 utility JCL for all members name prefix with DB1. Because DB2I2 process each LISTDEF one at a time, If you select to use LISTDEF pattern option, it will prompt selection screen one for each LISTDEF member. To avoid this inconvenience, you can use PARMUTIL command to generate PARMUTIL control and then use it as process option.

Example

The following example demonstrates how to generate Full image copy utility JCL against multiple DB2 Tablespace DBTST001.STS00001 and DBTST001.STS00002.

```
Command ==> copy                               Scroll ==> CSR
000027 TB JD00.T2
ss0028 TS DBTST001.STS00001
ss0029 TS DBTST001.STS00002
```

A pop up copy option screen is displayed to allows you to select the following options:

- SHRLEVEL option
- QUIESCE option: before copy, after copy or both
- One copy per step option
- Local/Remote copy option
- Tape/Disk output option: if DASD is specified, then space required for output is calculated
- Staging tape option: appropriate LABEL, UNIT and VOL information is generated.
- Dataset Prefix option for the output copy file

```
#COPYA ----- DB2I2 IMAGE COPY PROCESS OPTIONS -----
Option: _
1. FULL, INCREMENTAL COPY or CHANGLIMIT options
DFSMS CONCURRENT N (Y/N) FILTERDDN N (Y/N) FULL Y (Y/N/C-CHANGLIMIT)
PERCENT VALUE 1 1__ PERCENT VALUE 2 __ REPORTONLY N (Y/N)
SHRLEVEL REFERENCE (REFERENCE/CHANGE) CHECKPAGE N (Y/N)
SYSTEMPAGES Y (Y/N) SCOPE (ALL/PENDING) ALL__
FLASHCOPY N (N/Y/C-Consistent)
FCCOPYDDN TYSCOPY_
2. MERGECOPY NEWCOPY Y (Y/N)
CLONE _ (Y/N)
DSNUM ALL_ (ALL/0-4096)
QUIESCE BO (B-Before A-After BO-Both N-None)
One Copy per STEP N (Y/N)
PARALLEL _ (0-99) TAPEUNITS __
Copy Type (Y/N) (T)ape/(D)ASD Tape Stag
Local 1 Y T Y
Local 2 N T Y
Remote 1 N T Y
Remote 2 N T Y
Dataset Prefix - Local _____
Dataset Prefix - Remote _____
GDG Generation Image Copy Y (Y/N)
PF3=Exit ENTER=Process Your Selection
```

The example shown below demonstrates how to select full image copy with the following options:

- SHRLEVEL REFERENCE
- Generate both QUIESCE job steps before and after image copy step
- Single local copy to TAPE
- Staging image output Dataset onto the same tape
- Use 'TP' Dataset Prefix for all image copy output files.

```
#COPYA ----- DB2I2 IMAGE COPY PROCESS OPTIONS -----
Option: 1
1. FULL, INCREMENTAL COPY or CHANGELIMIT options
DFSMS CONCURRENT N (Y/N) FILTERDDN N (Y/N) FULL Y (Y/N/C-CHANGELIMIT)
PERCENT VALUE 1 1__ PERCENT VALUE 2 __ REPORTONLY N (Y/N)
SHRLEVEL REFERENCE (REFERENCE/CHANGE) CHECKPAGE N (Y/N)
SYSTEMPAGES Y (Y/N) SCOPE (ALL/PENDING) ALL__
FLASHCOPY N (N/Y/C-Consistent)
FCCOPYDDN TYSCOPY_
2. MERGECOPY NEWCOPY Y (Y/N)
CLONE _ (Y/N)
DSNUM ALL_ (ALL/0-4096)
QUIESCE BO (B-Before A-After BO-Both N-None)
One Copy per STEP N (Y/N)
PARALLEL _ (0-99) TAPEUNITS __
Copy Type (Y/N) (T)ape/(D)ASD Tape Stag
Local 1 Y T Y
Local 2 N T Y
Remote 1 N T Y
Remote 2 N T Y
Dataset Prefix - Local TP
Dataset Prefix - Remote
GDG Generation Image Copy Y (Y/N)
PF3=Exit ENTER=Process Your Selection
```

The following screen displays the result JCL from the previous COPY command.

```

000013 //STEP001 EXEC DSNUPROC,REGION=4M,COND=(4,LT),
000014 //          SYSTEM='DSN',UID='JD00. QUIESCE',UTPROC=''
000016 //SYSPRINT DD SYSOUT=*
000017 //SYSIN   DD *
000018   QUIESCE
000019           TABLESPACE   DBTST001.STS00001
000020           TABLESPACE   DBTST001.STS00002
000022 //STEP002 EXEC DSNUPROC,REGION=4M,COND=(4,LT),
000023 //          SYSTEM='DSN',UID='JD00.COPY',UTPROC=''
000025 //SYSPRINT DD SYSOUT=*
000026 //UTPRINT  DD SYSOUT=*
000027 //LCPY1001 DD DSN=TP.DBTST001.STS00001.L1000(+1),
000028 //          DISP=(NEW,CATLG,CATLG),
000029 //          DCB=(BUFNO=30,RECFM=FB,LRECL=4096,BLKSIZE=0),
000030 //          UNIT=TAPE,
000031 //          LABEL=(1,SL,RETPD=30),
000032 //          VOL=(,RETAIN,,20)
000033 //LCPY1002 DD DSN=TP.DBTST001.STS00002.L1000(+1),
000034 //          DISP=(NEW,CATLG,CATLG),
000035 //          DCB=(BUFNO=30,RECFM=FB,LRECL=4096,BLKSIZE=0),
000036 //          UNIT=AFF=LCPY1001,
000037 //          LABEL=(2,SL,RETPD=30),
000038 //          VOL=(,,20,REF=*.LCPY1001)
000039 //SYSIN DD *
000040   COPY
000041           TABLESPACE   DBTST001.STS00001
000042           DSNUM          ALL
000043           FULL           YES
000044           COPYDDN        (LCPY1001)
000045           SHRLEVEL       REFERENCE
000046   COPY
000047           TABLESPACE   DBTST001.STS00002
000048           DSNUM          ALL
000049           FULL           YES
000050           COPYDDN        (LCPY1002)
000051           SHRLEVEL       REFERENCE
000053 //STEP003 EXEC DSNUPROC,REGION=4M,COND=(4,LT),
000054 //          SYSTEM='DSN',UID='JD00. QUIESCE',UTPROC=''
000056 //SYSPRINT DD SYSOUT=*
000057 //SYSIN   DD *
000058   QUIESCE
000059           TABLESPACE   DBTST001.STS00001
000060           TABLESPACE   DBTST001.STS00002

```

COPYAUTH

Command Syntax:	COPYAUTH [GRANTEE GRANTOR] [TO=grant-to] [SQLID=sqid] [CRE=new_creator] [DB=new_dbname] [CL=new_collid] [SG=new_stogroup] [SH=new_schema]
Line objects allowed:	US, RO, AL, BP, CL, DB, DT, FU, MT, PG, PL, SG, SH, SP, SQ, TB, TBC, TS, VW
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	DB2 GRANT
Reusable:	Yes

Command Description

- Use COPYAUTH command to generate GRANT DCL from selected line object.
- For US and RO line object:
 - User. This command is useful to copy authorization from one user to another user.
 - Use **GRANTEE** option to copy the authorization of **what the user has been granted**. For example, if a new comer join the application group, you want the new user to have the same authorization the other team member has.
 - Use the **GRANTOR** to copy the authorization which **the selected user is the grantor**. This option is useful for any user leave the company and you want to make sure that whatever he has granted for the other people to use have been reassigned by a valid user.
- The TO option is used to indicate the user you want to grant the copied authorization to.
- Use SQLID=sqid option to set the current SQLID precedes each of the generated GRANT statements.
- Use CRE, DB, CL, SG, SH to override Creator, dbname, collid, stogroup and schema information.

Example

The following example generates GRANT DCL from user TST1 to TST2.

```

Command ==> COPYAUTH TO=TST2
S00038 US TST1
                                                                    Scroll ==> CSR

-- Database Authorization
-- Table Authorization
-- Plan Authorization
-- Package Authorization
***

```

Partial result from the previous COPYAUTH command is displayed below.

```
Command ==>
000038  US  TST1
000039 -- Database Authorization
000040 GRANT
000041     DBADM
000042     ,DBCTRL
000043     ,DBMAINT
000044     ,CREATETAB
000045     ,CREATEETS
000046     ,DISPLAYDB
000047     ,DROP
000048     ,IMAGCOPY
000049     ,LOAD
000050     ,RECOVERDB
000051     ,REORG
000052     ,REPAIR
000053     ,STARTDB
000054     ,STATS
000055     ,STOPDB
000056     ON DATABASE DBTST01
000057     TO TST2
```

Scroll ==> CSR

CPY2CPY DB2 V7 or above only

Command Syntax:	CPY2CPY [[PARMUTIL=]'parmutil.dsn] TEMPLATE=cpy2cpy.template.dsn [LISTDEF=listdef.dsn[(patt*)]] [OPTIONS=options.dsn]
Line objects allowed:	TS, TP, IX, IP, IS, ISP TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use CPY2CPY command to generate COPYTOCOPY utility JCL from selected TS, TP, IX or IP line objects. The output JCL can be used to make image copies from the existing image copy datasets.
- If FROMCOPY option is selected, use ICGEN=# at the end of the line object to specify the generation of image copy to be used as input to CPY2CPY command. # if specified, must be <= 0. Specify ICGEN=0 means use the current generation of image copy will be selected.
- If desired, you can also specify ICDATE= YYMMDD at the end of the line object to specify the image copy of a specific date to be used as input to CPY2CPY.
- TEMPLATE command option is required to be used for CPY2CPY command.
- For DB2 v7 or above, you can specify LISTDEF option with PDS member pattern match. For example, By specifying LISTDEF=listdef(DB1*) will generate COPYTOCOPY DB2 utility JCL for all members name prefix with DB1. Because DB2I2 process each LISTDEF one at a time, If you select to use LISTDEF pattern option, it will prompt selection screen one for each LISTDEF member. To avoid this inconvenience, you can use PARMUTIL command to generate PARMUTIL control and then use it as process option.

Example

The following example demonstrates how to generate COPYTOCOPY utility JCL to make Local Backup imagecopies of tablespaces and indexspaces of database DBSYSADM from Local Primary Full image copy dataset, by using CPY2CPY DB2I2 command with LISTDEF, TEMPLATE and OPTIONS command options.

Where OPTIONS=options(o1) dataset contains OPTIONS PREVIEW and
TEMPLATE=tmp(t2) and LISTDEF=listdef(l1) contain the information as expanded in the generated COPYTOCOPY utility JCL.

```
Command ==> cpy2cpy &gl options=options(o1)                               Scroll ==> CSR
000023 db Dbsysadm
000024 gv &gl=listdef=listdef(l1) template=tmp(t2)
```


DB2I2 Reference Manual

#CPY2CPY ----- DB2I2 CPY2CPY Utility PROCESS OPTIONS -----

```

From Copy Information          2 (1,2,3,4)
  1. FROMLASTCOPY
  2. FROMLASTFULLCOPY
  3. FROMLASTINCRCOPY
  4. FROMCOPY
    LOCAL/REMOTE COPY (LP/RP) LP (LP-Local Primary RP-Remote Primary)
    FULL/INCREMENT (F/I) F

Dataset Specification
  LOCAL PRIMARY COPY (Y/N) N    LOCAL BACKUP COPY (Y/N) Y
  REMOTE PRIMARY COPY (Y/N) N    REMOTE BACKUP COPY (Y/N) N

CLONE (Y/N) _

PF3=Exit  ENTER=Process Your Selection
  
```

n

```

EDIT          SYSADM.SPFTEMP1.CNTL          Columns 00001 00072
Command ==>          Scroll ==> CSR
***** ***** Top of Data *****
000001 //SYSADM1 JOB (999,POK),'JERRY D',MSGLEVEL=(1,1),
000002 // CLASS=A,MSGCLASS=H,NOTIFY=&SYSUID
000003 //* -----**** DB2I2 DB2 CPY2CPY JCL GENERATION ****-----+
000004 //* DB2I2 DB2 WORK BENCH UTILITIES INTERFACE
000005 //* BY: JRH GOLDENSTATE SOFTWARE, INC.
000006 //* COPYRIGHTED 1997-2006 Rev. Date: 04/26/2006
000007 //* DATE: 02/05/03 TIME:05:24 CREATOR:SYSADM
000008 //*-----+
000009 //JOBLIB DD DISP=SHR,DSN=DSN710.SDSNLOAD
000010 // DD DISP=SHR,DSN=DSN710.SDSNEXIT
000011 // DD DISP=SHR,DSN=DSN710.RUNLIB.LOAD
000012 //DB2I2P JCLLIB ORDER=SYSADM.DB2I2.EVAL.ISPFLIB
000013 //* ----- **
000014 //STEP001 EXEC DSNUPROC,REGION=4M,COND=(4,LT),
000015 // SYSTEM='DSN1',UID='SYSADM1.STEP001',UTPROC=''
000016 //* ----- **
000017 //SYSPRINT DD SYSOUT=*
000018 //SYSLISTD DD DISP=SHR,DSN=SYSADM.LISTDEF(L1)
000019 //SYSTEMPL DD DISP=SHR,DSN=SYSADM.TMP(T2)
000020 //UTPRINT DD SYSOUT=*
000021 //* **-- LISTDEF detail from 'SYSADM.LISTDEF(L1)'
000022 //* LISTDEF L1
000023 //* INCLUDE
000024 //* TABLESPACES
000025 //* DATABASE DBSYSADM
000026 //* ALL
000027 //* INCLUDE
000028 //* INDEXSPACES
000029 //* COPY YES
000030 //* DATABASE DBSYSADM
000031 //* ALL
000032 //* **-- TEMPLATE detail from 'SYSADM.TMP(T2)'
000033 //* TEMPLATE TYSREC UNIT SYSALLDA DSN 'SYSADM.&JO..&SN..SYSREC'
000034 //* DISP (NEW,DELETE,CATLG)
000035 //* TEMPLATE TYSDISC UNIT SYSALLDA DSN 'SYSADM.&JO..&SN..SYSDISC'
000036 //* DISP (NEW,CATLG,CATLG)
000037 //* TEMPLATE TYSPUNCH UNIT SYSALLDA DSN 'SYSADM.&JO..&SN..SYSPUNCH'
000038 //* DISP (NEW,CATLG,CATLG)
000039 //* TEMPLATE TYSCOPY DSN 'SYSADM.&DB..&SN..D&JU..L1&TI.'
000040 //* UNIT SYSALLDA
000041 //* DISP (NEW,CATLG,CATLG)
000042 //* TEMPLATE TYSCOPY2 DSN 'SYSADM.&DB..&SN..D&JU..L2&TI.'
  
```

DB2I2 Reference Manual

```
000043 //*                UNIT SYSALLDA
000044 //*                DISP (NEW,CATLG,CATLG)
000045 //*  TEMPLATE TYSRCPY1  DSN 'SYSADM.&DB..&SN..D&JU..R1&TI.'
000046 //*                UNIT SYSALLDA
000047 //*                DISP (NEW,CATLG,CATLG)
000048 //*  TEMPLATE TYSRCPY2  DSN 'SYSADM.&DB..&SN..D&JU..R2&TI.'
000049 //*                UNIT SYSALLDA
000050 //*                DISP (NEW,CATLG,CATLG)
000051 //*  TEMPLATE TYSUT1     UNIT SYSALLDA DSN 'SYSADM.&JO..&SN..SYSUT1'
000052 //*                DISP (NEW,DELETE,CATLG)
000053 //*  TEMPLATE TORTOUT    UNIT SYSALLDA DSN 'SYSADM.&JO..&SN..SORTOUT'
000054 //*                DISP (NEW,DELETE,CATLG)
000055 //SYSIN DD *
000056  OPTIONS PREVIEW
000057          LISTDEFDD  SYSLISTD
000058          TEMPLATEDD SYSTEMPL
000059  COPYTOCOPY LIST      L1
000060          FROMLASTFULLCOPY
000061          COPYDDN( ,TYSOCOPY2)
```

CREATE

Command Syntax:	CREATE
Line objects allowed:	AL, DB, DT, FU, IX, SG, SP, SY, TB, TR, TS, VW MT, SQ for V8 or above RO, TC for V9 or above MS, PM for V10 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	DDL statement
Reusable:	Yes

Command Description

- Use DB2I2 CREATE command to generate CREATE DDL control statements for selected line objects.

Example

The following example demonstrates how to generate CREATE DDL for column mask.

```

Command ==> create
s01636 MS DB2ADM.SSN_MASK DB2ADM DB2ADM.EMPLOYEE

** ----- **
** 1. Please make Any ALTER changes (lower case field) **
** 2. Select lines to be processed(with S or SS) **
** 3. Issue DSNTIAD or EXEC DB2I2 Command **
** ----- **
CREATE MASK DB2ADM.SSN_MASK
ON table-name
AS correlation-name
FOR COLUMN column-name
RETURN case-expression
DISABLE|ENABLE
ENABLE

```

CURSORD

Command Syntax:	CURSORD
Line objects allowed:	TB, AL, SY, VW, MT TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Cobol Code
Reusable:	No

Command Description

- Use CURSORD command to generate DECLARE CURSOR embedded SQL statement for selected line object.

Example

```
Command ==> CURSORD                               Scroll ==> CSR
***** Top of Data *****
S00001 TB Q.PROFILES
```

The following screen shows the results from the previous CURSORD command.

```
Command ==>                                         Scroll ==> CSR
000001 TB Q.PROFILES
000002          EXEC SQL
000003          DECLARE CURSOR FOR
000004          SELECT  A.CREATOR, A.CASE, A.DECOPT, A.CONFIRM
000005                  , A.WIDTH, A.LENGTH, A.LANGUAGE, A.SPACE
000006                  , A.TRACE, A.PRINTER, A.TRANSLATION
000007                  , A.PFKEYS, A.SYNONYMS, A.RESOURCE_GROUP
000008                  , A.MODEL, A.ENVIRONMENT
000009          FROM Q.PROFILES A
000010          WHERE
000011          END-EXEC.
```

DB

Command Syntax:	DB
Line objects allowed:	TS, TP, IX, IP, IS, ISP, TB, MT TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	DB line object
Reusable:	Yes

Command Description

- Use DB command to display **database** information for the selected line objects.

Example

```
Command ===> db
s01638 TB DB2ADM.EMPLOYEE DB2ADM.SSN_MASK
```

```
Command ===>
001638 TB DB2ADM.EMPLOYEE DB2ADM.SSN_MASK
001639 DB DSN00027 DB2ADM.EMPLOYEE
```

DBAUTH

Command Syntax:	DBAUTH [GRANTEE GRANTOR]
Line objects allowed:	DB, US RO for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use DBAUTH command to display **database authorization** information for the selected line objects.
- The GRANTEE or GRANTOR option allows you to display the GRANTEE or GRANTOR of the selected line objects. The default is GRANTEE.

Example

The following example displays the users were granted with the database authorization for database DSQDBCTL.

```
Command ==>> DBAUTH                               Scroll ==>> CSR
S00012 DB DSQDBCTL
```

```
Command ==>>                                       Scroll ==>> CSR
000012 DB DSQDBCTL
000013 -- Database Authorization
000014
000015          C                               R
000016          R C           I           E
000017          E R           D D   M   C   S
000018          A E   D B I   A   O R T
000019          T A D B M S   G   R V E A S
000020          E T B C A P D C L E E P R T S
000021          T E A T I L R O O R A T A T
000022          DATABASE A T D R N A O P A R D I D T O
000023          GRANTEE  GRANTOR  NAME  B S M L T Y P Y D G B R B S P
000024 US  USR1      DB2ADM  DSQDBCTL G G G G G G G G G G G G G G
000025 US  USR2      DB2ADM  DSQDBCTL Y
```

DBDSIZE

Command Syntax:	DBDSIZE [threshold 300000] (default threshold is 300000)
Line objects allowed:	DB
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Report
Reusable:	No

Command Description

- Use DBDSIZE DB2I2 command to display **DBD size information** for the selected database line objects.
- The threshold is use to signal the display of warning message. If the size of DBD exceed the thread, a warning message
DBD size is > thread-hold bytes for dbame

The default threshold if not specified is 300000 bytes.

Example

The following example demonstrates the usage of DBDSIZE command with threshold 20000.

```
Command ==> dbdsiz 20000                               Scroll ==> CSR
000028 DB DCA%
ss0029 DB DJDOO      GTDB2000 BP0      581
ss0030 DB DJDOO1    GPDB2001 BP0      458
000031
```

The result screen displayed below shows that database DJDOO1 has DBD size exceeds 20000 specified threshold.

```
Command ==>                               Scroll ==> CSR
000028 DB DCA%
000029 DB DJDOO      GTDB2000 BP0      581
==MSG> + ===== +
==MSG> | DBD Size Information |
==MSG> + ===== +
==MSG> DBname... DBD Size..
==MSG> DJDOO      12104
==MSG> DJDOO1    32294
=NOTE= * DBD size is > 20000 bytes for DJDOO1
000030 DB DJDOO1    GPDB2001 BP0      458
```

DB2CMD

Command Syntax:	DB2CMD [LIST]
Line objects allowed:	db2 command line
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 DB2CMD to execute DB2 commands directly from your DB2I2 workbench within your ISPF editing session.
- Use the LIST option to list all the available DB2 commands options available. You can use this option to assist you to build a DB2 command.
- If you write your own DB2 command lines, the Db2 command lines entered should follow the same DB2 command syntax as you execute them through DB2 DSN command.

Example

The following example demonstrates how to use DB2CMD to issue DB2 DISPLAY THREAD.

```
Command ==> DB2CMD                               Scroll ==> CSR
S00039 -DISPLAY THREAD(*)
```

Partial result from previous –DISPLAY THREAD DB2CMD are displayed below.

```
Command ==>                                       Scroll ==> CSR
-DISPLAY THREAD(*)_____
DSNV401I -DSN DISPLAY THREAD REPORT FOLLOWS -
DSNV402I -DSN ACTIVE THREADS -
NAME      ST A   REQ ID          AUTHID   PLAN      ASID  TOKEN
IMSB      N     3              STC      0030     0
TSO       T   *   3 JD00          JD00     00CD    7516
TSO       N     1              CD20     0098     0
TSO       T     16 Cxxx      CXXX     PCD6005 0077  7503
```


DB2I2 Reference Manual

The following screen shows how to use DB2CMD command with LIST option specified.

```
Command ==> DB2CMD LIST                               Scroll ==> CSR
000039 -DISPLAY THREAD(*)
```

Partial result is displayed in the following screen. The screen displayed contains all available DB2 commands. You can find the Db2 command you are looking for, select the command, and issue DB2CMD again to process that command.

```
Command ==>                                           Scroll ==> CSR
***** ***** Top of Data *****
==MSG> 1. Please find the desired DB2 command first.
==MSG> 2. Make any necessary changes to those command lines.
==MSG> 3. Selected the DB2 command lines the same way you
==MSG> select the other line objects.
==MSG> 4. Issue DB2I2 command DB2CMD to Execute the selected lines.
000001 -- -----
000002 --                DB2 commands List for DB2 Version 4&5                --
000003 -- -----
000004 -- ALTER BUFFERPOOL: Alters attributes for the buffer pools          --
000005 -- -----
000006 -ALTER BUFFERPOOL (bpname)
000007         VPSIZE(integer) HPSIZE(integer) VPSEQT(integer)
000008         VPPSEQT(integer) HPSEQT(integer) DWQT(integer)
000009         VDWQT(integer) CASTOUT(YES/NO)
000010
000011 -- -----
000012 -- ALTER GROUPBUFFERPOOL: Alters attributes for the                    --
000013 --                group buffer pools                                    --
000014 -- -----
```

DCLGEN

Command Syntax:	DCLGEN [[PARMUTIL=]'parmutil.dsn']
Line objects allowed:	TB, AL, VW, SY, MT TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Cobol Code
Reusable:	N/A

Command Description

- Use DB2I2 DCLGEN command to generate DB2 DCLGEN command.
- The generated DB2 DCLGEN command can be executed with DB2I2 EXEC command.

Example

The following example demonstrates how to generate DCLGEN command for a DB2 table Q.PROFILES.

```
Command ==> DCLGEN                               Scroll ==> CSR
s00037  TB Q.PROFILES                               T 18   DSQDBCTL DSQTSPO
```

Please follow the on screen instruction, as displayed below, to make your DCLGEN option selection. The following example selects PDS.CNTL as the name of the library containing the DCLGEN output. Use PROFILE as the COBOL structure name.

```
#DCLGEN -----DB2I2 DCLGEN PROCESS OPTIONS-----
Library:      PDS.CNTL
Action:      A          (A-Add R-Replace)
Language:    COBOL      (COBOL/PLI/C/COB2)
Names:       (Name Prefix)
Structure:   PROFILE   (Structure Name)
APOST/QUOTE: _        (A-Apost Q-Quote)
Label:      N          (N/Y use Column Label)
DBCSSYMBOL  _        (G/N graphic data)
DBCSEDELIM  Y          (Y/N)
COLSUFFIX   N          (N/Y column name as suffix to Name)
INDVAR      N          (N/Y Indicator variable array)

PF3=EXIT  ENTER=PROCESS YOUR SELECTION
```

DB2I2 Reference Manual

Partial results from the previous DCLGEN command as shown below. You can use EXEC command to execute these DB2CMD lines.

```
Command ==>
000037  TB Q.PROFILES          T 18  DSQDBCTL DSQTSPRO          Scroll ==> CSR
000038  DB2CMD  DCLGEN  TABLE(PROFILES)          -
000039  DB2CMD          OWNER(Q)                  -
000040  DB2CMD          LIBRARY('JD00.PDS.CNTL(PROFILES)') -
000041  DB2CMD          LANGUAGE(COBOL)            -
000042  DB2CMD          STRUCTURE(PROFILE)         -
000043  DB2CMD          DBCSDELIM (YES)            -
000044  DB2CMD          INDVAR (NO)
```

DDL

Command Syntax:	DDL [%=### [CYL TRK __] [ALLOC=(alloc_type,pri,sec)] [SQLTERM(?) for V6 or above] [TR=N Y] [MAXASGN] [OBID] [CRE=new_creator] [DB=new_dbname] [CL=new_collid] [SG=new_stogroup] [VCAT=new_vcatname] [O=new_owner] [Q=new_qualifier] [SH=new_schema] [VOL=*]
Line objects allowed:	AL, DB, TS, TB, IX, SY, VW, SG, RI (DT, FU, SP, TR for V6 or above) (SQ, MT for V8 or above) (TBC, TC for V9 or above) (MS, PM for V10 or above)
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	DDL
Reusable:	Yes

Command Description

- Use DB2I2 DDL command to generate DDL statements for selected line objects.
- For DB, TS, and IX line object, you can use %=### command option to adjust the size as a % of the current allocation. Use optional sub-parameter CYL or TRK to round up to cylinder or track boundary. . You can also use ALLOC=(alloc_type,primary,secondary) command option to override the space allocation. Alloc_type specified must be CYL, TRK, or PAGE. The primary and secondary specified must be numeric. ..
- Both %=### and ALLOC=(alloc_type,primary,secondary) options can also be used on the line object as line object option to override the option from command line. Specify these options at the end of the line object to override command options.
- Use MIGR command instead of DDL if you want to generate DDL statements not just for the selected line objects but also generates **all the dependent objects**, which includes GRANT and BIND information as well.
- In db2 v6 or above, specify SQLTERM(?) to choose a SQL terminator to be used for the generated DDL control statements. ? can be any special character which is not frequently being used.
- Add command option TR=Y if trigger information is desirable.
- Use MAXASGN option to generate table DDL with START WITH the maxassignedval + increment.
- Use OBID option to generate OBID option for CREATE TABLE DDL.
- Use VOL=* option to generate VCAT with VOLUME(*)
- Use CRE=new_creator to change creator information.
- Use DB=new_dbname to change database name information
- Use CL=new_collid to change collection ID information
- Use SG=new_stogroup to change stogroup information
- UseVCAT=new_vcatnameto change VCAT information
- Use O=new_owner to change owner information
- Use Q=new_qualifier to change qualifier information
- Use SH=new_schema to change schema information

Example

The following example generates DDL statements for table Q.PROFILE.

```
Command ==> ddl sqlterm(~) Scroll ==> CSR
s00001 TB Q.PROFILES
```

The following screen shows partial results from the previous DDL command.

```
Command ==> Scroll ==> CSR
000001 TB Q.PROFILES
000002 -- Table Generation: Q.PROFILES
000003 SET CURRENT SQLID = 'Q      '~
000004 CREATE TABLE Q.PROFILES
000005 (
000006     CREATOR                CHAR(8)
000007     NOT NULL
000008     ,CASE                   CHAR(18)
000009     ,DECOPT                 CHAR(18)
000010     ,CONFIRM                CHAR(18)
000011     ,WIDTH                  CHAR(18)
000012     ,LENGTH                 CHAR(18)
000013     ,LANGUAGE               CHAR(18)
000014     ,SPACE                   CHAR(50)
000015     ,TRACE                  CHAR(18)
000016     ,PRINTER                CHAR(8)
000017     ,TRANSLATION            CHAR(18)
000018     NOT NULL
000019     ,PFKEYS                 VARCHAR(31)
```

The following example demonstrates how to use DDL command with global command option %= and line object override to generate DDL statements with space adjustment and line object option override:

```
Command:      DDL %=10 TRK
Line object:  TS db.ts1
              TS db.ts2 %=100
              TS db.ts3 ALLOC=(TRK,1,0)
```

DB2I2 generates DDL for 10% track boundary space adjustment for db.ts1, and
 100% adjustment for db.ts2, and
 1 track primary and no secondary space adjustment for db.ts3.

DELETE

Command Syntax:	DELETE [MAP='dclgen.dsn']
Line objects allowed:	AL, TB, SY, VW, MT TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Delete SQL Skeleton
Reusable:	Yes

Command Description

- Use DB2I2 DELETE command to generate DELETE SQL statements for the selected line objects.
- Use MAP option to generate static SQL for the embedded SQL statement. The 'dclgen.dsn' dataset contains the name of the DCLGEN command output.

Example

The following example generates DELETE SQL for table Q.PROFILES.

```
Command ==> DELETE                               Scroll ==> CSR
000037 TB Q.PROFILES                               T 18   DSQDBCTL DSQTSPRO
000038 -- Table Generation:  Q.PROFILES
000039 SET CURRENT SQLID = 'Q'                       ;
```

The result from the previous DELETE command as shown below.

```
Command ==>                               Scroll ==> CSR
000037 TB Q.PROFILES                               T 18   DSQDBCTL DSQTSPRO
000038   DELETE FROM Q.PROFILES WHERE
000039     CREATOR           = --CHAR           8   0
000040     AND CASE           = --CHAR           18  0
000041     AND DECOPT        = --CHAR           18  0
000042     AND CONFIRM       = --CHAR           18  0
000043     AND WIDTH         = --CHAR           18  0
000044     AND LENGTH        = --CHAR           18  0
000045     AND LANGUAGE      = --CHAR           18  0
000046     AND SPACE         = --CHAR           50  0
000047     AND TRACE         = --CHAR           18  0
000048     AND PRINTER       = --CHAR            8  0
000049     AND TRANSLATION   = --CHAR           18  0
000050     AND PFKEYS        = --VARCHAR          31  0
000051     AND SYNONYMS      = --VARCHAR          31  0
000052     AND RESOURCE_GROUP = --CHAR           16  0
000053     AND MODEL         = --CHAR            8  0
000054     AND ENVIRONMENT   = --CHAR            8  0
000055 -- Table Generation:  Q.PROFILES
000056 SET CURRENT SQLID = 'Q'                       ;
```

DISPLAY

Command Syntax:	DISPLAY [db2 command display options] [TSIX] [EDIT=Y]
Line objects allowed:	DB, TS, TP, IX, IP, IS, ISP, BP or no line object
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 DISPLAY command to issue DB2 DISPLAY command against the selected Database, BUFFERPOOL, Tablespace or Indexspace.
- The [db2 command display options] can be any valid DB2 display command option except DB(dbname) and SPACE(spacename) options. Both DB and SPACE option are derived directly from the line objects you selected. For example, DISPLAY RESTRICT LIMIT(*) to display restrict status with no limit display.
- If no line object specified, DISPLAY can be issued with any valid DB2 –DISPLAY command option. For Example, DISPLAY DB(mydb) RESTRICT is the same as DISPLAY RESTRICT with a DB mydb line object selected.
- Specify optional TSIX command option to request DB2I2 to return results in TP or IP line object format. The returned line objects can then be used for other actions. For example, a copy pending TP can be used to with COPY command to generate full image copy utility jobs or can be used for START command with option FORCE to reset copy pending restrict status.
- Specify EDIT=Y to return result in EDIT mode.

Example

The following example issue DB2 DISPLAY command against index Q.COMMAND_SYNONYMSX.

```

Command ==> DISPLAY
==MSG> Index Name.....Cnt.L.U.E.C.BP..ISOBID.TBcrator.TBname.
S00010 IX Q.COMMAND_SYNONYMSX      2  N U N N BP0  30  Q      COMMAND
000011      IC VERB                      A
000012      IC OBJECT                     A
==MSG> Part.Pqty.....Qty..T.Storname.Vcatname.FPG.%FR...
000013 IP 0      25      5      I DSQSGSYN TDB2      0      10
    
```

The result from previous DB2I2 DISPLAY command displayed below.

```

Command ==>
***** Top of Data *****
DSNT360I -DSN *****
DSNT361I -DSN * DISPLAY DATABASE SUMMARY
* GLOBAL
DSNT360I -DSN *****
DSNT362I -DSN DATABASE = DSQDBCTL STATUS = RW
DBD LENGTH = 8066
DSNT397I -DSN
NAME TYPE PART STATUS PHYERRLO PHYERRHI CATALOG PIECE
-----
COMMANDR IX RW
***** DISPLAY OF DATABASE DSQDBCTL ENDED *****
DSN9022I -DSN DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
***** Bottom of Data *****

```

The following example issue DB2 DISPLAY command without line object and the result is TSIX format and the result is returned in EDIT mode.

```

Command ==> display db(dbsysadm) tsix edit=y
Scroll ==> CSR

```

```

Command ==>
***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG> your edit profile using the command RECOVERY ON.
000001 TP DBSYSADM.DSN8S71D 0 RW
000002 TP DBSYSADM.SLARGE 001 RW
000003 TP DBSYSADM.SLARGE 002 RW
000004 TP DBSYSADM.SXNT 0 RW
000005 TP DBSYSADM.TSSYSADM 0 RW
000006 TP DBSYSADM.TSSYSAD1 0 RW
000007 TP DBSYSADM.TSSYSAD2 0 RW
000008 TP DBSYSADM.TSSYSAD3 0 RW
000009 TP DBSYSADM.TSTRACK 0 RW
000010 IP SYSADM.PLAN_X 0 RW
000011 IP SYSADM.XCOPY1 0 RW
000012 IP SYSADM.XCOPY2 0 RW
000013 IP SYSADM.XDEPT1 0 RW
000014 IP SYSADM.XDEPT2 0 RW
000015 IP SYSADM.XDEPT3 0 RW
000016 IP SYSADM.XLARGE1 001 RW
000017 IP SYSADM.XLARGE1 002 RW
000018 IP SYSADM.XLARGE2 0 RW

```


DSADJ

Command Syntax:	DSADJ [MOVE=Y N] [%=### [CYL TRK _] [ALLOC=(alloc_type,primary,secondary)] [MAXSZ=(type,primary,secondary)] [OVRD=reorg_override_table] [MACRO(your.ed.macro)] [ONESTEP=Y]
Line objects allowed:	TP, IP, ISP TPC, IPC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	JCL
Reusable:	Yes

Command Description

- Use DSADJ command to generate JCL to adjust the space allocation for the underlined VSAM linear dataset.
- If the table part or index part is a DB2 manage dataset, DSADJ also generates DB2 ALTER SQL statements.
- Command option MOVE=Y generates additional JCL steps to adjust the space with IDCAMS DEFINE, RENAME and DSN1COPY. The MOVE=Y option is the default option.
- Specify %=### to adjust the space allocation as a % of current allocation. Use optional sub-parameter CYL or TRK to round up to cylinder or track boundary. Use %=100 to adjust dataset to remove extents.
- You can also use ALLOC=(alloc_type,primary,secondary) option to override the space allocation. Specified alloc_type must be CYL, TRK, or PAGE. Default alloc_type from ICF catalog will be used if omits the alloc_type. The primary must be HIARBA, HIURBA or a numeric number. Default primary allocation information from ICF catalog will be used if omits this field. Secondary options if specified must be numeric. The following are a few examples: .
 ALLOC=(CYL,100,5)
 ALLOC=(,HIARBA,)
 ALLOC=(CYL,HIURBA,)
- Both %=### and ALLOC=(alloc_type,primary,secondary) options can be used on the line object as line object option to override the option from command line. Specify these options at the end of the line object to override command options.
- In online mode, if there is no %=### or ALLOC options present, DB2I2 display a detail screen which allows you specify detail space allocation information to be used as the space allocation parameters.
- Use %=100 to set the space allocation to the current calculated space.
- Specify MAXSZ option to set the maximum allocation for the adjusting line objects. DB2I2 set the primary and secondary allocation based on the MAXSZ option if the calculated primary allocation is greater than the primary specified in the MAXSZ. For example, specify MAXSZ=(CYL,500,500) to set the primary and secondary allocation both to 500 cylinders if the calculated primary allocation is greater than 500 cylinders. .
- Specify OVRD option to specify REORG override table and an INSERT SQL will be generated together with ALTER DDL. The table can then be used with REOGR command with RCHK and OVRD option.
- Use ONESTEP=Y to generate all DB2 ALTER statements in one job step.

Example

The following example demonstrates how to adjust the space allocation of an index space Dataset.

```
Command ==> DSADJ                               Scroll ==> CSR
***** Top of Data *****
S00001 IP  JD00.XN3B005
```

Since there are no %=### or ALLOC options specified, you will receive the following screen with all the detail information from both DB2 catalog and VSAM catalog. You can adjust the primary and secondary space allocation by entering the desired primary and secondary space allocation on the following screen.

```
#DSADJ ----- DB2I2 DB2 DATASET ADJUSTMENT PROCESS OPTIONS -----
DB2 Object:      JD00.XN3B005
Dataset Name:    TDB2.DSNDBD.DJDOO.XN3B005.I0001.A001
Storgroup Name:  GPDB2001
VcatNmae:       TDB2
Hi-A-RBA:       737280                (720 K / 1 CYLs)
Hi-U-RBA:       737280                (720 K / 1 CYLs)
Space Type:     CYLINDER

Primary Allocation:  1_____ (720 K)
Secondary Allocation: 1_____ (720 K)

PF3=Exit  ENTER=Process Your Selection
```

The results JCL, with default command option MOVE=Y, from previous example contains the following information:

```
Step1:
-STO DB(DJDOO) SPACE(XN3B005)
step2:
//ALTERDEF EXEC PGM=IDCAMS
//SYSIN DD *
ALTER -
  TDB2.DSNDBC.DNOP01U.XN3B005.I0001.A001 -
  NEWNAME(TDB2.DSNDBC.DNOP01U.XN3B005.I0001.A001.O)
ALTER -
  TDB2.DSNDBD.DNOP01U.XN3B005.I0001.A001 -
  NEWNAME(TDB2.DSNDBD.DNOP01U.XN3B005.I0001.A001.O)
DEFINE -
  CLUSTER -
  (NAME(TDB2.DSNDBC.DNOP01U.XN3B005.I0001.A001) -
  LINEAR -
  RECORDS(180 180) -
  VOLUMES (DS3006) -
  REUSE -
  SHAREOPTIONS(3 3) -
  ) -
  DATA -
  (NAME(TDB2.DSNDBD.DNOP01U.XN3B005.I0001.A001) -
step3:
//DSN1COPY EXEC PGM=DSN1COPY,PARM='RESET',COND=((0,NE,ALTERDEF))
//SYSPRINT DD SYSOUT=*
//SYSUT2 DD DSN=TDB2.DSNDBC.DNOP01U.XN3B005.I0001.A001,
// DISP=SHR,DCB=BUFNO=30
//SYSUT1 DD DSN=TDB2.DSNDBC.DNOP01U.XN3B005.I0001.A001.O,
// DISP=SHR,DCB=BUFNO=30
step4:
//DELETE EXEC PGM=IDCAMS,COND=((0,NE,DSN1COPY))
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE TDB2.DSNDBC.DNOP01U.XN3B005.I0001.A001.O PURGE
step5:
//* ----- *//
//* ADJUST PRIMARY/SECONDARY FOR STORAGE GROUP ALLOCATION *//
//* ----- *//
```

```
//ALTERDB EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB2T)
RUN PROGRAM(DSNTIAD) PLAN(DSNTIAD)
//SYSIN DD *
ALTER INDEX CM96.XN3B005
PRIQTY 720 SECQTY 720
step6:
//STARTDB EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DSN)
-DIS DB(DNOP01U) SPACENAM(XN3B005)
-STA DB(DNOP01U) SPACENAM(XN3B005) ACCESS(RW)
-DIS DB(DNOP01U) SPACENAM(XN3B005)
/*
```

The following example use ovrd option to specify REORG override table.

```
Command ==> dsadj ovrd=syadm.reorg_ovrd %=100 move=n          Scroll ==> CSR
000221 -- TS DBSYSADM.TSSYSAD2
s00222 TP DBSYSADM.TSSYSAD2          0

/* ----- */
/* ADJUST PRIMARY/SECONDARY FOR STORAGE GROUP ALLOCATION */
/* ----- */
//ALTRD1 EXEC PGM=IKJEFT1B,DYNAMNBR=20,COND=(4,LT)
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DSN1)
RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71)
//SYSIN DD *
ALTER TABLESPACE DBSYSADM.TSSYSAD2
PRIQTY 2448 SECQTY 48 ;
INSERT INTO
SYADM.REORG_OVRD
(OBJECT_NAME, PART_NO)
VALUES (
'DBSYSADM.TSSYSAD2'
,0) ;
/*
```

The example below demonstrates how to use DSADJ together with global command option and line object option override to adjust table partition space allocation:

Command:	DSADJ %=200 CYL	
Line objects:	TP DB.TS1 0	DB.TS1 space allocation adjusted to 200% of the current allocation and round up to cylinder boundary.
	TP DB.TS2 0 %=100	DB.TS2 space allocation adjusted to 100% instead of 200% of the current allocation with line object option override. It also round up to cylinder boundary.
	TP DB.TS3 ALLOC=(CYL,10,1)	DB.TS3 space allocation adjusted to 10 cylinders of primary allocation and 1 cylinder of secondary allocation with line object option override.

DSCOPY

Command Syntax: DSCOPY

	[ICOPY=Y N LB RP RB] [[HLQ=ImageCopy-high-level-qualifier] [SORTKEYS] [NOSTOP] [NOSTOPS] [MACRO(your.ed.macro)] [CONT=Y [ERRDSN=error.dsn]] [DFLTSP=(1,1 pri,sec)] [CLONE=Y for V9 or above]
Line objects allowed:	SC, XC
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	JCL
Reusable:	Yes

Command Description

- Use DB2I2 DSCOPY command to generate DSN1COPY utility JCL to copy base VSAM linear data set from source tablespace or indexspace to target tablespace or indexspace.
- DB2I2 always generate STOP and START steps for target TS or IX before and after DSN1COPY step, and by default DB2I2 generates STOP and START steps for the target TS or IX. If your specify ICOPY=N command option on the command line (VSAM to VSAM), use NOSTOPS to skip STOP and START steps for source TS or IX.
- DB2I2 validates the information between source and target to see if the copy from source to target is valid. The validation check includes:
 - Number of tables per tablespace
 - Number of columns per tables
 - Segment size
 If there are any mismatches, DB2I2 stops the process and display error message.
- OBID translation information is also extracted from both source and target DB2 catalogs to be used in the DSN1COPY job step to translate OBID information.
- If you do not specify ICOPY command option, DB2I2 uses the most current full image copy from source DB2 object as input to the DSN1COPY job step. Instead use full image copy, you can use command option ICOPY=N to copy directly from source VSAM dataset to target VSAM dataset.
- ICOPY=Y – Local Primary, ICOPY=LB – Local Backup, ICOPY=RP – Remote Primary, ICOPY=RB – Remote Backup, ICOPY=N – No image copy is used (Vsam to Vsam)
- Use HLQ to specify High-level-qualifier for the selected Image Copy Dataset.
- Once you decide to use ICOPY=Y,LB,RP,RB command option, you can
 - Use line object option ICGEN=# to indicate the generation of full image copy to be used as input to the DSCOPY. The default # for ICGEN is 0, which selects the most current full image copy as input. The #, if specified, should be less than or equal to 0. For example, use ICGEN=-1 to select -1 generation of the full image copy as input.
 - Use line object option ICDATE=YYMMDD to select a full image copy with specific date as input to the DSCOPY.
- The DB2I2 DSCOPY command supports source to target from **two different locations**. This feature allows you to **copy information directly from one DB2 sub-system to another DB2 sub-system**.
- When DSCOPY generates JCL to copy from source to target, If the line objects is SC, tablespace copy, the **recover index/rebuild index** step can also be generated so that all the indexes after the copy will be in sync with all the indexes of source tablespace.
- By default, DB2I2 generate recover/rebuild index steps after a table part is copied over. Specify line object option RCVRIX=N or RBLDIX=N to disable the recover/rebuild index generation for that specific line object.
- When Rebuild index step is generated, use SORTKEYS to generate SORTKEYS rebuild index option.
- Specify NOSTOP to generate no STOP and START steps for both source and target objects.
- Specify NOSTOPS to generate no STOP and START steps only for source objects.

- When multiple SC or XC are selected, use CONT=Y option to continue process the rest of the line objects if there are any difference between the source and destination object. Use ERRDSN option together with CONT=Y to record any error SC or XC lines.
- Specify DFLTSP option to assign the space allocation information for DSCOPY work files.
- Specify CLONE=Y for V9 to include CLONE tablespace and indexspace.

The SC line object contains the following format:

[SSID[LOC].]DBNAME.TSNAME [SSID[LOC].]DBNAME.TSNAME [PART#]

The XC line object contains the following format:

[SSID[LOC].]IXCREATOR.IXNAME [SSID[LOC].]IXCREATOR.IXNAME [PART#]

Example

The following example demonstrates how to copy a partition Tablespace 1st Dataset from source to target. Both tablespaces are from local default location so that you do not need to specify ssid or location name information. With ICOPY=N option specified, the copy process is done using VSAM to VSAM method without using any image copy file.

```
Command ==> DSCOPY ICOPY=N                               Scroll ==> CSR
S00280 SC DJDOO.DSN8S41E DJDOO.DSN8S41A 1
000281 SC DJDOO.DSN8S41E DJDOO.DSN8S41A 2
000282 SC DJDOO.DSN8S41E DJDOO.DSN8S41A 3
000283 SC DJDOO.DSN8S41E DJDOO.DSN8S41A 4
```

Partial result JCL from the previous command displayed below:

```
Step 1:  -STO DB(DJDOO) SPACE(DSN8S41E) PART(1)
Step 2:  -STO DB(DJDOO) SPACE(DSN8S41A) PART(1)
Step 3:  //STEP003 EXEC PGM=DSN1COPY,
//        PARM='OBIDXLAT,RESET,NUMPARTS(4) '
// * -----
//SYSUT1 DD DSN=TDB2.DSNDBC.DJDOO.DSN8S41E.I0001.A001,
//        DISP=SHR,
//        AMP='BUFND=24 '
//SYSUT2 DD DSN=TDB2.DSNDBC.DJDOO.DSN8S41A.I0001.A001,
//        DISP=SHR,
//        AMP='BUFND=24 '
//SYSPRINT DD SYSOUT=*
//SYSXLAT DD *
581,581
24,36
25,37
step4:  -STA DB(DJDOO) SPACE(DSN8S41E) PART(1)
step5:  -STA DB(DJDOO) SPACE(DSN8S41A) PART(1)
step6:  RECOVER
        INDEX (ALL)
        TABLESPACE DJDOO.DSN8S41A
```

The example below uses DSCOPY to generate DSN1COPY use current full image copy as input for a 4 partitions table space. Recover/rebuild index job step is generated only after the 4th partition finishes the DSN1COPY step.

- The local SSID is DB2P. Since there are no locations specified, the default local location is used for both source and target.
- Since no ICOPY specified, it defaults to use the most current full image copy as input.
- RCVRIX=N or RBLDIX=N can be coded for the first three partitions to disable the default recover index JCL step generation.

```
SC DB2P.DJRHHJ.SJRHHJ3 DB2P.DJRHHJ.SJRHHJ3 1 rcvrix=n
SC DB2P.DJRHHJ.SJRHHJ3 DB2P.DJRHHJ.SJRHHJ3 2 rcvrix=n
SC DB2P.DJRHHJ.SJRHHJ3 DB2P.DJRHHJ.SJRHHJ3 3 rcvrix=n
SC DB2P.DJRHHJ.SJRHHJ3 DB2P.DJRHHJ.SJRHHJ3 4
```

If there are no SSID or connection information specified, the default SSID and location name is used. The following lines without SSID specification produce the same results JCL as the above.

```
SC DJRHHJ.SJRHHJ3 DJRHHJ.SJRHHJ3 1 rcvrix=n
SC DJRHHJ.SJRHHJ3.DJRHHJ.SJRHHJ3 2 rcvrix=n
SC DJRHHJ.SJRHHJ3.DJRHHJ.SJRHHJ3 3 rcvrix=n
SC DJRHHJ.SJRHHJ3.DJRHHJ.SJRHHJ3 4
```

The example below uses the most current full image copy from DB2P as input to the DSCOPY. The target tablespace is at location LOCT.

```
SC DJRHHJ.SJRHHJ1 DB2P\LOCT.DJRHHJ.SJRHHJ1
```

The example below uses -2 generation of full image copy from DB2P as input.

```
SC DB2P.DJRHHJ.SJRHHJ1 DB2P.DJRHHJ.SJRHHJ1 ICGEN=-2
```

The example below uses the most current full image copy as of 2004-01-04 from DB2P as the input.

```
SC DB2P.DJRHHJ.SJRHHJ3 DB2P.DJRHHJ.SJRHHJ3 ICDATE=040104 ICGEN=0
```

DSNJU004

Command Syntax:	DSNJU004 [BSDS='bsds.dsn']
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	AC, AR, RL,CP
Reusable:	Yes

Command Description

- Use DB2I2 DSNJU004 command to run DSNJU004 print log map utility directly from DB2I2 workbench. The default BSDS dataset information is retrieved directly from the SSID system setup file.
- Use BSDS option to specify the input BSDS information. This option is required if the DB2 is not up and running.

Example

The example below display DSNJU004 information with default BSDS information.

```
Command ==> dsnju004                               Scroll ==> CSR
```

Partial result from previous command displayed below.

```
Command ==>                               Scroll ==> CSR
***** ***** Top of Data *****
==MSG> /* -----*/
==MSG> /*          DB2I2  DB2 Catalog Interface Tool Box          */
==MSG> /*          By JRH GoldenState Software, Inc.             */
==MSG> /*          (C) Copyrighted 1997-2011                     */
==MSG> /*          Licensed to EVALUATION COPY                    */
==MSG> /* DB2I2 Environment - SSID(DB2D) CONNECT( ) SYSIBM(SYSIBM) */
==MSG> /* -----*/
=NOTE= T=AC-active log AR-archive log CP-check point RL-Archieve Log command
=NOTE= Action you can take from this screen:
=NOTE= 1. Use DB2I2 command TAG to select RBA ranges and Log DSname
=NOTE= 2. Use DB2I2 command HRECALL to recall Archieve log datasets
=NOTE=    If Archive Log datasets are on DASD and have been migrated
=NOTE= 3. Position cursor on a RBA field and issue SETRBA command to
=NOTE=    set the incore RBA, whcih can be used in the Recovery process
===== T  Start RBA      Ending RBA      Date & Time      Dataset Name
000001 CP 01484E138000 01484E1412A7 2000.271 00:34:42
000002 AC 01484AC5D000 0148574C0FFF 2000.270 23:02:57 DB2D.LOGCOPY1.DS03
000003 CP 01484AC5D763 01484AC66390 2000.270 23:02:57
000004 CP 01484A901794 01484A90B000 2000.270 23:01:23
000005 CP 014846BED2F1 014846BF8EBF 2000.270 21:51:34
000006 CP 014844225398 01484422EA08 2000.270 21:10:49
000007 CP 014841B872C2 014841B952A6 2000.270 19:50:15
000008 AC 01483E3F9000 01484AC5CFFF 2000.270 19:45:33 DB2D.LOGCOPY1.DS01
```

```
000009 AR 01483E3F9000 01484AC5CFFF 2000.270 19:45:33 DB2D.ARCHLOG1.D00270.T160
000010 CP 01483E41D3FB 01483E428C00 2000.270 19:45:33
000011 CP 01483D574D92 01483D581152 2000.270 19:44:39
000012 CP 01483A2D0A8B 01483A2E01B8 2000.270 17:14:46
000013 CP 014837A6F03D 014837A9F1C6 2000.270 15:52:18
000014 AC 014835759000 01483E3F8FFF 2000.270 10:15:21 DB2D.LOGCOPY1.DS02
000015 AR 014835759000 01483E3F8FFF 2000.270 10:15:21 DB2D.ARCHLOG1.D00270.T124
000016 CP 01483575A674 014835761F4F 2000.270 10:15:21
000017 RL 014835758642 2000.270 10:15:21
000018 CP 0148354FC090 014835503452 2000.270 08:32:17
000019 CP 0148354F41C2 0148354FB575 2000.270 08:03:25
000020 CP 014833A5E0A7 014833A6955C 2000.269 23:07:33
000021 CP 0148311F7DE5 01483120F3F3 2000.269 22:47:15
000022 CP 01482CF587AF 01482CF618DE 2000.269 15:32:11
000023 AR 01482ABB4000 014835758FFF 2000.269 09:25:53 DB2D.ARCHLOG1.D00270.T031
```

.
.
.
.

All the information returned are in RBA descendant sequence. You can

- Use DB2I2 command **TAG** to select RBA ranges and Log Dsname
- Use DB2I2 command **HRECALL** to recall Archieve log datasets. If Archive Log datasets are on DASD and have been migrated.
- Position cursor on a RBA field and issue **SETRBA** command to set the incore RBA, which can be used in the Recovery process

DSNTEP2

Command Syntax:	DSNTEP2 [SQLTERM(?) for db2 v6 or above]
Line objects allowed:	valid SQL block followed by an ';' or a special SQLTERM character
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No - SQL statements only
Wild Card % allowed:	N/A
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 DSNTEP2 command to run DSNTEP2 IBM sample program directly from your DB2I2 workbench.
- The line objects specified should follow the same syntax as when you run DSNTEP2 in batch mode.
- Specify SQLTERM(?) in db2 v6 or above to indicate the special character to be use as the SQL terminator for the SQL specified.

Example

The following example run DSNTEP2 with the selected SQL statement – SELECT * FROM Q.PROFILES;

```
Command ==> dsntep2 sqlterm(~)                               Scroll ==> CSR
***** ***** Top of Data *****
000001 DB2CMD -DISPLAY THREAD(*)
s000002 SELECT COUNT(*) FROM Q.PROFILES~
```

The result from the previous DSNTEP2 command displayed below.

```
PAGE      1
***INPUT STATEMENT:
SELECT COUNT(*) FROM Q.PROFILES~
```

```
-----+
|                                           |
+-----+
1_|                                           | 48 |
+-----+
```

DSNTIAD

Command Syntax:	DSNTIAD [RC0] [SQLTERM(?) for db2 v6 or above]
Line objects allowed:	valid SQL block followed by an ';' or special SQLTERM character
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No – Non-SELECT SQL statements only
Wild Card % allowed:	N/A
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 DSNTIAD command to run IBM DSNTIAD sample program directly from DB2I2 workbench.
- The line objects should follow the same syntax as when you run DSNTIAD in batch mode.
- Specify SQLTERM(?) in db2 v6 or above to indicate the special character to be use as the SQL terminator for the SQL specified.

Example

The following example demonstrates how to execute DSNTIAD command against select SQL block to create a table, insert into the created table, delete from the table and drop the table.

```

Command ==> DSNTIAD                               Scroll ==> CSR
SS0017 CREATE TABLE T2 LIKE T1;
000018 COMMIT;
000019 INSERT INTO T2 SELECT * FROM T1;
000020 DELETE FROM T2;
000021 COMMIT;
000022 DROP TABLE T2;
SS0023 COMMIT;

```

The result from the previous DSNTIAD command as shown below.

```

Command ==>                                         Scroll ==> PAGE
***** Top of Data *****
DSNTIAD - SAMPLE DYNAMIC SQL PROGRAM 2.0

      CREATE TABLE T2 LIKE T1
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION

      COMMIT
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION

      INSERT INTO T2 SELECT * FROM T1
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION
DSNT416I SQLERRD = 0 0 13 1144486641 0 0 SQL DIAGNOSTIC INFORMATION
DSNT416I SQLERRD = X'00000000' X'00000000' X'0000000D' X'44377AF1' X'0000
INFORMATION

      DELETE FROM T2
SQL WARNING DURING EXECUTE IMMEDIATE
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION
DSNT418I SQLSTATE = 01504 SQLSTATE RETURN CODE
DSNT416I SQLERRD = 0 0 13 1129173655 0 0 SQL DIAGNOSTIC INFORMATION

```

DSNTIAUL

Command Syntax:	DSNTIAUL [DEV=D T] [SQL] [DSPRE=datasetprefix TSOID]
Line objects allowed:	TB, VW, AL, SY, MT or valid SQL block followed by an ‘;’ or TB tbcreator.tbname SYSPUNCH=syspunch.output SYSREC=sysrec.output TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	N/A
Output Type:	JCL
Reusable:	Yes

Command Description

- Use DB2I2 DSNTIAUL command to generate DSNTIAUL JCL against select line objects.
- The SQL command option is used if the selected line objects is a SQL block.
- The default device for output SYSREC is DASD (DEV=D).
- Use DEV=T command option to direct output SYSREC to tape.
- If TB line object selected and DEV=D option is specified as default, the space allocation for SYSREC is calculated based on the RECLENGTH and CARD(F) information from SYSIBM catalog table.
- Use DSPRE to set the Dataset prefix for the generated utility work files. The default is your TSOID.
- Specify SYSPUNCH=syspunch.output and SYSREC=sysrec.output to assign the name of the SYSPUNCH and SYSREC for the DSNTIAUL process. Otherwise the default SYSREC and SYSPUNCH names will be
 - Dstasetprefix.tbcreator.xxxxxxxx.SYSREC for SYSREC and
 - Dstasetprefix.tbcreator.xxxxxxxx.SYSPUNCH for SYSPUNCH
 Where xxxxxxxx is the first 8 position of table name with all “_” substituted with “#”

Example

The following example demonstrates how to generate DSNTIAUL JCL to unload a table Q.APPLICANT.

```
Command ==> dsntiaul                               Scroll ==> CSR
S00029  TB Q.APPLICANT                               T 39   DSQ1STBB DSQ1STBT
```

Result for the previous DSNTIAUL command displayed below.

```

Command ==>
000012 /* ----- Scroll ==> CSR **
000013 //STEP001 EXEC PGM=IEFBR14
000014 /* ----- **
000015 //DD1 DD DSN=JD00.Q.APPLICAN.SYSREC,
000016 // DISP=(MOD,DELETE,DELETE),
000017 // UNIT=SYSALLDA,
000018 // SPACE=(TRK,0)
000019 //DD2 DD DSN=JD00.Q.APPLICAN.SYSPUNCH,
000020 // DISP=(MOD,DELETE,DELETE),
000021 // UNIT=SYSALLDA,
000022 // SPACE=(TRK,0)
000023 /* ----- **
000024 //STEP002 EXEC PGM=IKJEFT01,DYNAMNBR=100
000025 /* ----- **
000026 //SYSTSPRT DD SYSOUT=*
000027 //SYSTSIN DD *
000028 DSN SYSTEM(DSN)
000029 RUN PROGRAM(DSNTIAUL PLAN(DSNTIAUL) PARM('SQL'))
000030 END
000031 //SYSPRINT DD SYSOUT=*
000032 //SYSUDUMP DD SYSOUT=*
000033 //SYSREC00 DD DSN=JD00.Q.APPLICAN.SYSREC,
000034 // SPACE=(TRK,(30,15),RLSE),
000035 // DISP=(,CATLG,DELETE)
000036 //SYSPUNCH DD DSN=JD00.Q.APPLICAN.SYSPUNCH,
000037 // SPACE=(TRK,(1,1),RLSE),
000038 // DISP=(,CATLG,DELETE)
000039 //SYSIN DD *
000040 SELECT * FROM Q.APPLICANT;
000041 /*
***** ***** Bottom of Data *****

```

DSN1COPY

Command Syntax:	DSN1COPY
Line objects allowed:	TX, TP, IX, IP, IS, ISP TXC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	JCL
Reusable:	No

Command Description

- Use DSN1COPY to generate DSN1COPY JCL against select line objects.

Example

The following example generates DSN1COPY JCL for Tablespace DBTST001.STS00002.

```
Command ==> dsn1copy                               Scroll ==> CSR
000027 TB JD00.T2
000028 TS DBTST001.STS00002
000029 CREATE TABLE T2 LIKE T1 IN DBTST001.STS00002;
000030 COMMIT;
000031 INSERT INTO T2 SELECT * FROM T1;
```

After you selected the line objects, the following pop up screen is displayed to assist you to make a DSN1COPY option selection. There are 8 preset options allow you to fully utilize the DSN1COPY utility.

1. Create a backup copy of a DB2 VSAM linear data set
2. Restore a backup copy of a DB2 VSAM linear data set
3. Move a DB2 VSAM linear data set to another DB2 VSAM linear data set
4. Perform validity checking on a DB2 VSAM linear data set
5. Perform validity checking on and print a DB2 VSAM linear data set
6. Restore a tablespace from a non-partition image copy
7. Restore a tablespace from a partition image copy
8. Perform RBA reset on a DB2 VSAM linear data set

```

Command ==> db2i2 dsnlcopy
000027 -----
S00028 | #DSN1CP0 -----DB2I2 DSN1COPY PROCESS OPTIONS-----
000029 | SELECT OPTION ==>
000030 |
000031 | 1. Create a backup copy of a db2 Dataset
000032 | 2. Restore a backup copy of a db2 Dataset
000033 | 3. Move a db2 Dataset to another db2 Dataset
000034 | 4. Perform validity checking on a db2 Dataset
000035 | 5. Perform validity checking on and print a db2 Dataset
000036 | 6. Restore a Tablespace from a non-partition image copy
000037 | 7. Restore a Tablespace from a partition image copy
000038 | 8. Perform RBA reset on a db2 Dataset
000039 |
000040 | PF3=EXIT ENTER=PROCESS YOUR SELECTION
000041 | -----
000042 | PCTFREE 10
000043 | COMPRESS YES
    
```

The following example demonstrates how to use option 1 to backup a DB2 Dataset.

```

Command ==> db2i2 dsnlcopy
000027 -----
S00028 | #DSN1CP0 -----DB2I2 DSN1COPY PROCESS OPTIONS-----
000029 | SELECT OPTION ==> 1
000031 | 1. Create a backup copy of a db2 Dataset
000032 | 2. Restore a backup copy of a db2 Dataset
000033 | 3. Move a db2 Dataset to another db2 Dataset
000034 | 4. Perform validity checking on a db2 Dataset
000035 | 5. Perform validity checking on and print a db2 Dataset
000036 | 6. Restore a Tablespace from a non-partition image copy
000037 | 7. Restore a Tablespace from a partition image copy
000038 | 8. Perform RBA reset on a db2 Dataset
000040 | PF3=EXIT ENTER=PROCESS YOUR SELECTION
000041 | -----
    
```

The result from the previous DSN1COPY option 1 selection is listed below. DB2I2 command automatically generates step to run DB2 STOP command before the DSN1COPY backup step and DB2 START command after the backup step. The OBID information is also provided for reference in the output JCL.

```

000012 /** ----- **
000013 /** STOP TABLESPACE: DBTST001.STS00002
000014 /**STEP001 EXEC PGM=IKJEFT01,DYNAMNBR=20
000015 /** ----- **
000016 /**SYSIN DD DUMMY
000017 /**SYSPRINT DD SYSOUT=*
000018 /**SYSPRINT DD SYSOUT=*
000019 /**SYSTSIN DD *
000020 DSN SYSTEM(DSN)
000021 -DIS DB(DBTST001) SPACE(STS00002) LOCKS RESTRICT LIMIT(*)
000022 -STO DB(DBTST001) SPACE(STS00002)
000023 -DIS DB(DBTST001) SPACE(STS00002)
000024 END
000034 /**STEP002 EXEC PGM=DSN1COPY
000035 /** ----- **
000036 /**SYSUT1 DD DSN=TDB2.DSNDBC.DBTST001.STS00002.I0001.A001,
000037 /** DISP=SHR,
000038 /** AMP='BUFND=24'
000039 /**SYSUT2 DD DSN=JD00.DSNDBC.DBTST001.STS00002.I0001.A001,
000040 /** DISP=(,CATLG,DELETE),
000041 /** DCB=(RECFM=FB,LRECL=4096,BLKSIZE=0,BUFNO=30),
000042 /** UNIT=SYSDA,SPACE=(CYL,(011,1),RLSE)
000043 /**SYSPRINT DD SYSOUT=*
000044 /**SYSXLAT DD *
000045 527, DBID
000046 325, PSID
000047 326, OBID-T2
000048 /** ----- **
000049 /** START TABLESPACE: DBTST001.STS00002
000050 /**STEP003 EXEC PGM=IKJEFT01,DYNAMNBR=20
    
```

DB2I2 Reference Manual

```
000051 /* ----- **
000052 //SYSIN DD DUMMY
000053 //SYSPRINT DD SYSOUT=*
000054 //SYSTSPRT DD SYSOUT=*
000055 //SYSTSIN DD *
000056 DSN SYSTEM(DSN)
000057 -DIS DB(DBTST001) SPACE(STS00002) LOCKS RESTRICT LIMIT(*)
000058 -STA DB(DBTST001) SPACE(STS00002)
000059 -DIS DB(DBTST001) SPACE(STS00002)
000060 END
```

The following example demonstrates how to use DSN1COPY option 6 to restore a Tablespace from a DB2 full image copy.

```
Command ==> db2i2 dsn1copy                               Scroll == CSR
27 -----
28 #DSN1CPO -----DB2I2 DSN1COPY PROCESS OPTIONS-----
29 SELECT OPTION ==> 6
30
31 1. Create a backup copy of a db2 Dataset
32 2. Restore a backup copy of a db2 Dataset
33 3. Move a db2 Dataset to another db2 Dataset
34 4. Perform validity checking on a db2 Dataset
35 5. Perform validity checking on and print a db2 Dataset
36 6. Restore a Tablespace from a non-partition image copy
37 7. Restore a Tablespace from a partition image copy
38 8. Perform RBA reset on a db2 Dataset
39
40 PF3=EXIT ENTER=PROCESS YOUR SELECTION
41 -----
```

DB2I2 searches SYSIBM.SYSCOPY table and generates the following screen with all available image copy data sets for restore function. The SYSCOPY selection screen displayed below is available for all DB2I2 commands that required a RBA recovery or an image copy recovery/restore.

```
Command ==>                                             Scroll ==> CSR
***** ***** Top of Data *****
=NOTE= T Type
==MSG> T=F:Full Copy          I:Increment Copy          P:Recover tocopy/torba
==MSG> R:load replace log(yes) S:Load Replace log(no) W:Reorg log(no)
==MSG> X:reorg log(yes)      Y:Load Resume log(no) Z:Load Resume log(yes)
==MSG> T:Term utility        Q:Quiesce
==MSG> S=STYPE
==MSG> :DB2 Image Copy      C:DFSMS Concurrent Copy
==MSG> R:Load Replace(Yes)   S:Load Replace(No)
==MSG> W:Reorg Log(NO)      S:Reorg Log(Yes)
==MSG> DSNUM 0=TableSpace Level Otherwise=Partition Level
=NOTE= Dsnum
==MSG> 0=TableSpace Level    Otherwise=Partition Level
=NOTE= -Start RBA---TS -date- -time- DSNUM DSNAME-----
000001 00E6B5ED4C73 F 971222 144021 0 TP.DBTST001.STS00002.P001.LCPY1002
***** ***** Bottom of Data *****
```

After you identify the restoring data set, select that line object and enter TAG command and press HOT key to generate DSN1COPY JCL.

```

Command ==> tag                               Scroll ==> CSR
=NOTE= T Type
==MSG> T=F:Full Copy           I:Increment Copy           P:Recover tocopy/torba
==MSG> R:load replace log(yes) S:Load Replace log(no) W:Reorg log(no)
==MSG> X:reorg log(yes)       Y:Load Resume log(no) Z:Load Resume log(yes)
==MSG> T:Term utility         Q:Quiesce
==MSG> S=STYPE
==MSG>   :DB2 Image Copy       C:DFSMS Concurrent Copy
==MSG> R:Load Replace(Yes)     S:Load Replace(No)
==MSG> W:Reorg Log(NO)        S:Reorg Log(Yes)
==MSG> DSNUM 0=TableSpace Level Otherwise=Partition Level
=NOTE= Dsnum
==MSG> 0=TableSpace Level     Otherwise=Partition Level
=NOTE= -Start RBA---TS -date- -time- DSNUM DSNAME-----
s00001 00E6B5ED4C73 F 971222 144021 0 TP.DBTST001.STS00002.P001.LCPY1002
    
```

Result from the previous DSN1COPY option 6 command is displayed below.

```

Command ==>                               Scroll ==> CSR
000014 //STEP001 EXEC PGM=IKJEFT01,DYNAMNBR=20
000015 /** ----- **
000016 //SYSIN DD DUMMY
000017 //SYSPRINT DD SYSOUT=*
000018 //SYSTSPRT DD SYSOUT=*
000019 //SYSTSIN DD *
000020 DSN SYSTEM(DSN)
000021 -DIS DB(DBTST001) SPACE(STS00002) LOCKS RESTRICT LIMIT(*)
000022 -STO DB(DBTST001) SPACE(STS00002)
000023 -DIS DB(DBTST001) SPACE(STS00002)
000024 END
000025 /**
000026 /**-----*
000027 /** NOTES: *
000028 /** 0. STOP TABLESPACE TO GET ACTUAL DATA *
000029 /** 1. FULLCOPY-FULL IMAGE COPY AS INPUT *
000030 /** 2. OBIDXLAT-OBID TRANSLATION *
000031 /** 2. RESET -RESET RBA FOR RECOVERY *
000032 /**-----**
000034 //STEP002 EXEC PGM=DSN1COPY,
000035 // PARM='FULLCOPY,OBIDXLAT,RESET'
000036 /** ----- **
000037 //SYSUT1 DD DSN=TP.DBTST001.STS00002.P001.LCPY1002,
000038 // DISP=SHR,
000039 // DCB=(BUFNO=30)
000040 //SYSUT2 DD DSN=TDB2.DSNDBC.DBTST001.STS00002.I0001.A001,
000041 // DISP=SHR,
000042 // AMP='BUFND=24'
000043 //SYSPRINT DD SYSOUT=*
000044 //SYSXLAT DD *
000045 527, DBID
000046 325, PSID
000047 326, OBID-T2
000048 /** ----- **
000049 /** START TABLESPACE: DBTST001.STS00002 **
000050 //STEP003 EXEC PGM=IKJEFT01,DYNAMNBR=20
000051 /** ----- **
000052 //SYSIN DD DUMMY
000053 //SYSPRINT DD SYSOUT=*
000054 //SYSTSPRT DD SYSOUT=*
000055 //SYSTSIN DD *
000056 DSN SYSTEM(DSN)
000057 -DIS DB(DBTST001) SPACE(STS00002) LOCKS RESTRICT LIMIT(*)
000058 -STA DB(DBTST001) SPACE(STS00002)
000059 -DIS DB(DBTST001) SPACE(STS00002)
    
```


DSN1LOGP

Command Syntax:	DSN1LOGP [BSDS='bsds.dsn'] [BSDS ACTV ARCH]
Line objects allowed:	TS, TP, IX, IP, IS, ISP TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online only
Support	
Multiple line object:	No
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Report
Reusable:	No

Command Description

- Use DSN1LOGP to generate DSN1LOGP JCL against select line objects.
- Use BSDS, ACTV or ARCH option to specify the input to the generated DSN1LOGP JCL is from BSDS information, active log Dataset or archive log Dataset.
- Use BSDS='bsds.dsn' information to specify the BSDS Dataset information when DB2 is not up and running.

Example

The example below use DSN1LOGP command to display and list DSNJU004-print log map information, TAG a active log line, and generate a batch DSN1LOGP job to run against a index line object.

```
Command ==>> dsn1logp                               Scroll ==>> CSR
S00012 IX PRIW1.IXACH1                               PRIW1.ACSC_CUSTOMER_HIST
```

The result from previous command is the same as the result from DSNJU004 DB2I2 command. It lists all active log RBA, archive log RBA, check point RBA, and archive log command RBA.

The screen below is a partial result from previous command.

```
Command ==>>                               Scroll ==>> CSR
***** ***** Top of Data *****
==MSG> /* -----*/
==MSG> /*          DB2I2  DB2 Catalog Interface Tool Box          */
==MSG> /*          By JRH GoldenState Software, Inc.              */
==MSG> /*          (C) Copyrighted 1997-2011                      */
==MSG> /*          Licensed to EVALUATION COPY                    */
==MSG> /* DB2I2 Environment - SSID(DB2D) CONNECT( ) SYSIBM(SYSIBM) */
==MSG> /* -----*/
=NOTE= T=AC-active log AR-archive log CP-check point RL-Archieve Log command
=NOTE= Action you can take from this screen:
=NOTE= 1. Use DB2I2 command TAG to select RBA ranges and Log DSname
=NOTE= 2. Use DB2I2 command HRECALL to recall Archieve log datasets
=NOTE=    If Archive Log datasets are on DASD and have been migrated
=NOTE= 3. Position cursor on a RBA field and issue SETRBA command to
=NOTE=    set the incore RBA, whcih can be used in the Recovery process
===== T Start RBA      Ending RBA      Date & Time      Dataset Name
000001 CP 01484E138000 01484E1412A7 2000.271 00:34:42
000002 AC 01484AC5D000 0148574C0FFF 2000.270 23:02:57 DB2D.LOGCOPY1.DS03
000003 CP 01484AC5D763 01484AC66390 2000.270 23:02:57
000004 CP 01484A901794 01484A90B000 2000.270 23:01:23
```

The screen below uses DB2I2 TAG command to tag a line to be used in the generated DSN1LOGP JCL output.

```

Command ==> tag                               Scroll ==> CSR
***** Top of Data *****
==MSG> /* -----*/
==MSG> /*          DB2I2 DB2 Catalog Interface Tool Box          */
==MSG> /*          By JRH GoldenState Software, Inc.             */
==MSG> /*          (C) Copyrighted 1997-2011                     */
==MSG> /*          Licensed to EVALUATION COPY                   */
==MSG> /* DB2I2 Environment - SSID(DB2D) CONNECT( ) SYSIBM(SYSIBM) */
==MSG> /* -----*/
=NOTE= T=AC-active log AR-archive log CP-check point RL-Archieve Log command
=NOTE= Action you can take from this screen:
=NOTE= 1. Use DB2I2 command TAG to select RBA ranges and Log DSname
=NOTE= 2. Use DB2I2 command HRECALL to recall Archieve log datasets
=NOTE=    If Archive Log datasets are on DASD and have been migrated
=NOTE= 3. Position cursor on a RBA field and issue SETRBA command to
=NOTE=    set the incore RBA, whcih can be used in the Recovery process
===== T Start RBA      Ending RBA      Date & Time      Dataset Name
000001 CP 01484E138000 01484E1412A7 2000.271 00:34:42
s00002 AC 01484AC5D000 0148574C0FFF 2000.270 23:02:57 DB2D.LOGCOPY1.DS03
000003 CP 01484AC5D763 01484AC66390 2000.270 23:02:57
000004 CP 01484A901794 01484A90B000 2000.270 23:01:23

```

The following is partial JCL from previous command.

```

000018 /** -----
000019 /** RBASTART(hexadecimal-constant)
000020 /**
000021 /** RBAEND(hexadecimal-constant)
000022 /**
000023 /** LRSNSTART(hex-constant)
000024 /**
000025 /** LRSNEND(hex-constant)
000026 /**
000027 /** DATAONLY(NO/YES)
000028 /**
000029 /** SYSCOPY(NO/YES)
000030 /**
000031 /** DBID(HEXADECIMAL-CONSTANT)
000032 /** OBID(HEXADECIMAL-CONSTANT)
000033 /**
000034 /** PAGE(HEXADECIMAL-CONSTANT)
000035 /** hexadecimal-constant can consist of a maximum of 8 digits.
000036 /** you can specify a maximum of 100 page keywords in any given
000037 /** dsnllogp job. you must also specify the dbid and obid keywords
000038 /** that correspond to those pages.
000039 /**
000040 /** RID(HEXADECIMAL-CONSTANT)
000041 /** specifies a record identifier, which is a 10-digit hexadecimal
000042 /** number, with the first 8 digits representing the page number and
000043 /** the last 2 digits representing the page id map entry number.
000044 /**
000045 /** URID(HEXADECIMAL-CONSTANT)
000046 /** specifies a hexadecimal unit of recovery identifier (urid).
000047 /**
000048 /** LUWID (LUWID)
000049 /** specifies up to 10 luwids to include information about in the
000050 /** summary report.
000051 /** luwid consists of three parts: an lu network name, an luw instance
000052 /** number, and a commit sequence number.
000053 /**
000054 /** TYPE(HEXADECIMAL-CONSTANT)
000055 /**      2          page set control record
000056 /**      4          syscopy utility record
000057 /**     10          system event record
000058 /**     20          ur control record
000059 /**    100          checkpoint record
000060 /**    200          ur-undo record
000061 /**    400          ur-redo record
000062 /**    800          archive quiesce record
000063 /**   1000 to 8000 assigned by the resource manager
000064 /**
000065 /** SUBTYPE(HEXADECIMAL-CONSTANT)

```

DB2I2 Reference Manual

```
000066 /**      1          update data page
000067 /**      2          format page or update space map
000068 /**      3          update space map bits
000069 /**      4          update to index space map
000070 /**      5          update to index page
000071 /**      6          dba table update log record
000072 /**      7          checkpoint dba table log record
000073 /**      9          dbd virtual memory copy
000074 /**      A          exclusive lock on page set partition or dbd
000075 /**      B          format file page set
000076 /**      C          format index page set
000077 /**      F          update by repair (first half if 32 kb)
000078 /**     10          update by repair (second half if 32 kb)
000079 /**     11          allocating or deallocating a segment entry
000080 /**     12          undo/redo log record for modified page or redo log
000081 /**          record for formatted page
000082 /**     14          savepoint
000083 /**     15          other db2 component log records written for rmid 14
000084 /**     17          checkpoint record of modified page set
000085 /**     19          type 2 index update
000086 /**     1A          type 2 index under/redo or redo log record
000087 /**     1B          type 2 index change notification log record
000088 /**     1C          type 2 index space map update
000089 /**     1D          dbet log record with exception data
000090 /**     1E          dbet log record with lpl/grecp data
000091 /**     65          data propagation diagnostic log
000092 /**     81          type 2 index dummy compensation log record
000093 /**
000094 /**     VALUE(HEXADECIMAL-CONSTANT)
000095 /**     specifies a value that must appear in a log record to be extracted
000096 /**     the subtype option must be specified before the value option.
000097 /**
000098 /**     OFFSET(HEXADECIMAL-CONSTANT)
000099 /**     specifies an offset from the log record header at which the
000100 /**
000101 /**
000102 /**     SUMMARY(YES/NO/ONLY)
000103 /**
000104 /**     FILTER
000105 /**     -----
000106 /**     //BSDS      DD DSN='DB2D.BSDS01',DISP=SHR
000107 /**     //SYSIN     DD *
000108 /**     RBASTART(01484AC5D000) RBAEND(0148574C0FFF)
000109 /**     DBID (0291) OBID(0005)
000110 /**
```

DSN1PRNT

Command Syntax:	DSN1PRNT
Line objects allowed:	TS, TP, IX, IP, IS, ISP TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online only
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	JCL
Reusable:	No

Command Description

- Use DB2I2 DSN1PRNT to generate DSN1PRNT JCL against select line objects.

Example

The following example generates DSN1PRNT JCL for Tablespace DBTST001.STS00002.

```

Command ===> DSN1PRNT
S00009 TS DBTST001.STS00002
000010 TP 0 1798 178 I SGTST1 TDB2 0 10
000011 TB JD00.ELEMENT T 307 DBTST001 STS00001
000012 TB JD00.ENTITY T 298 DBTST001 STS00001
000013 TB JD00.PLAN_TABLE T 311 DBTST001 STS00001
000014 TB JD00.STD_ABBR T 302 DBTST001 STS00001
  
```

Result from the previous DSN1PRNT command listed below.

```

Command ===>
000012 /*-----*
000013 /* NOTES: *
000014 /* 0. STOP TABLESPACE TO GET ACTUAL DATA *
000015 /* 1. CHANGE ?? FOR STARTING AND ENDING PAGE NUMBER IN HEX *
000016 /* 2. PRINT(0) FOR HEADER PAGE *
000017 /* PRINT(1) SPACE MAP PAGE *
000018 /* PRINT(2) FIRST DATAPAGE *
000019 /* 3. VALUE(123) OR VALUE('F1F2F3') WILL PRINT PAGES WITH 123*
000020 /*-----*
000021 /*-----**
000022 //STEP001 EXEC PGM=DSN1PRNT,
000023 // PARM='PRINT(??,??),FORMAT,NUMPARTS(0)'
000024 /* ----- CHANGE THE PARM WITH ADDITIONAL OPTIONS AS LISTED BELOW -----
000025 /* FULLCOPY ; INRCOPY
000026 /* VALUE(XXXXXXXXXXXXXXXXXXXX)
000027 /* VALUE('F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1')
000028 /*-----**
000029 //SYSUT1 DD DSN=TDB2.DSNDBC.DBTST001.STS00002.I0001.A001,
000030 // DISP=SHR
000031 //SYSPRINT DD SYSOUT=*
  
```

DT

Command Syntax:	DT (V6 or above only)
Line objects allowed:	TB TBC for V9 or above
Process Mode:	Batch and online
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	DT line object
Reusable:	Yes

Command Description

- Use DB2I2 DT command to generate DT-distinct type line object against select TB lines.
- DT command is only available for Db2 V6 or above.

Example

The following example generates DT – distinct type lines for Table DB2ADM.T1.

```
Command ==> DT
*****
S00001 TB DB2ADM.T1
```

ED

Command Syntax:	ED 'edit.dataset.name' [MACRO(edit.macro.dsname IDD=ddname *sysmacro)] [START=starting-member-name] [END=ending-member-name] [PASS=N Y M]
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	No

Command Description

- Use DB2I2 ED command to edit a sequential dataset, a group of sequential datasets, or a group of members of a PDS dataset. This command replaces the DB2I2 EDIT command.
Specify exact name to edit a dataset.
Specify an * in the dataset name to edit a group of datasets. The * must be stand alone between two dots or the last node.
Specify an * in the member name of a PDS to edit a group of members of a PDS dataset. The * must be the last character in the member name field.
For example,
ED 'my.dataset'
ED Dbdbid.*.SYSPUNCH
ED my.data.*
ED my.cntl(tst*)
When editing of pds members, use START=starting-member-name END=ending-member-name to narrow down the range of editing.
- Use MACRO option to process predefined ISPF edit commands against the 'edit.dataset.name'.
For example, enter and save the following ISPF EDIT command and save in a file called 'MM'
C 'aaaa' 'bbbb' all
C 'cccc' 'dddd' all
You can then invoke these predefined ISPF edit commands against the edit.dataset.name with
ED 'edit.dataset.name' MACRO(MM)
Use MACRO(*sysmacro) to access system defined ED macro. MYMENU is one of the system defined ED MACRO which can be invoked with MACRO(*MYMENU)
- Use PASS=Y option to pass the content of IDS, IDD or S selected lines down to your ED macro specified. The total number of line object is stored in a rexx variable LNOBJ.0 and the content of the line objects are stored in a rexx array LNOBJ..
- Use PASS=M option to pass the content S selected lines or LINEONJ DD for batch as ED macro specified.
 - Most of the ISPF edit commands can be used in the save MACRO dataset together with ED command:
C 'aaa' 'bbb' ALL to changes all 'aaa' to 'bbb' in batch mode
X ALL to exclude all lines
F .PG ALL to find all .PG line objects
DELETE X ALL to delete excluded lines

Besides the standard ISPF edit commands, DB2I2 provides the following additional ISPF edit macros:

LOAD cache-name

LOAD cache-name **IDD=input DDNAME | IDSN=input DSNAME|IDSN={rexx-var}**

For performance reason, use LOAD command to load a CACHE with cache-name. The information in cache can then be used with INSERT command.

INSERTB, INSERTA, INSERT

INSERTB

INSERTA **line#*|{rexx-var} 'character string' | IDD=input DDNAME | IDSN=input DSNAME|IDSN={rexx-var}|**
CACHE=cache-name

INSERT

Where line# is the line number where you want the character string or file to be inserted. Specify * means the current positioned line. Specify IDD for the inserting file DDNAME and IDSN for the input file DSNAME. The line or file to be inserted can substitute with REXX variables specified. {REXX_VAR}. For Example,

```
INSERTA * Database name: {ED_DBNAME}
```

Use CACHE=cache-name option to insert information from a LOADED cache. For Example,

```
LOAD CACHE1 IDSN=my.file  
INSERT CACHE=CACHE1
```

Insert a line after the current position line: Database name: ????????

Where ???????? is the current value stored in REXX variable ED_DBNAME

```
INSERTA * IDSN=MY.INSERTING.DSNAME
```

And MY.INSERTING.DSNAME may contains the following:

```
Database Name: {ED_DBNAME}  
TableSpace Name: {ED_TSNAME}
```

For backward compatibility, INSERT work just the same way as INSERTB.

For example,

```
INSERTA 3 This is line number 4
```

IDSN can also be a {rexx-var}, which allows you to dynamically insert different file depending on

The content of a rexx-var.

For example,

```
REXX mydsn = "MY.DSN1"  
INSERT * IDSN={mydsn}
```

During file insertion with rexx-var substitution, all {{ are replaced with single { and all }} are replaced With single {.

DELETE

DELETE line#*[{rexx-var} [# of lines to be deleted]_*[{rexx-var}]

Where line# is the starting line number to be deleted. Specify * means the current positioned line. The # of line to be deleted if not specified is default to 1, specify * for delete all lines from current positioned line. Both fields can be {rexx-var} substitutes.

For example,

```
DELETE *      deletes the current positioned line
DELETE 100 5  deletes 5 lines from line 100
```

CLINE

CLINE [[NEXT|PREV|WRITE]

Use CLINE to get the content of current edit line into REXX variable CLINE.

Use CLINE NEXT to get the content of next line into REXX variable CLINE. If end of file condition Reached, a REXX variable EOF is set to 'Y', which will end the LOOP process

Use CLINE PREV to get the content of previous line into REXX variable CLINE. If top of file condition Reached, a REXX variable EOF is set to 'Y', which will end the LOOP process

Use CLINE WRITE to set the content of current line to REXX variable CLINE.

For example,

```
F 'aaaa' First
CLINE                                     get the content of current line
REXX PARSE VAR CLINE ED_V1 ED_V2 ED_rest
REXX CLINE = ED_V1 'bbbb' ED_V3 ED_rest
CLINE WRITE                               Update current line with CLINE

-- Process until EOF
CLINE
LOOP EOF                                  Process LOOP until EOF

.
  Process CLINE information

.
  CLINE NEXT                               set EOF = 'Y' if end of file reached
  also set EOLOOP = 'Y' to end loop process
LOOP_END
```

CPOS [#]

CPOS

Use CPOS to set the current line number REXX variable CPOS. If # is use, CPOS as well as current line Position is set to #.

REXX

REXX single line REXX statement

Use REXX inside ED macro to execute one line rexx statement . You can use this to

- o Parse a line which obtained thru CLINE
- o Manipulate a line so that you can update the current with CLINE information
- o Set REXX variables so that they can be substituted with during INSERT processing
- o Extract the information from CLINE and output them to a file
- o Set EOLOOP special REXX variable to control when to end a LOOP process.
- o Since your REXX routine is co-exist with the DB2I2 REXX routines, to avoid process conflict, It is highly recommended to use rexx variable name with ED_ prefix.

For example,

```
REXX Parse VAR CLINE ED_V1 ED_V2 ED_REST
```

- o A special REXX variable **WSDSN** contains the current editing dataset name.

- If your REXX statement is too long to fit on single line, Use + at the end of the line and continue your REXX statement on the next line.
- Use SIGNAL FINISH in REXX statement to end of the current edit process
- Use LAST_RC REXX variable to check the return code of last command
- Use SKIP # tin REXX statement to SKIP commands
- Use NEXTLINE REXX variable to change the next line to be executed. For example, REXX NEXTLINE=3 set the next line to be executed to line no 3
- Use REXX ERROROFF='Y' to turn off ED warning/error message. This is useful if you do not want to see some of the warning message display during executing of ED macro. MYMENU use this to turn off message displayed.

DB2I2

DB2I2 db2i2 command

Use DB2I2 inside ED macro to execute DB2I2 command . You can use this to

- Invoke DB2I2 command after you process the editing dataset
- Use {varname} to substitute the DB2I2 command with the REXX variable information
- Manipulate a line so that you can update the current with CLINE information
- If your DB2I2 command is too long to fit on single line, Use + at the end of the line and continue your REXX statement on the next line.

LOOP & LOOP_END

LOOP [#|EOF|{rxvar}]

.
. .
.

LOOP_END

Use LOOP and LOOP_END inside ED macro to execute a set of ED statement repeatedly until either # of time reached, {rxvar} times reached, EOF condition reached or EOLOOP condition raised. Use REXX command to set rxvar and then use LOOP {rxvar} together with LOOP_END to loop thru rxvar times.

- If you did not specify #, EOF or {rxvar}, To prevent an infinite looping condition, DB2I2 requires you to enter a FIND before LOOP is required and a **FIND before LOOP_END**. By doing so, a NOTFOUND condition can be triggered to End a LOOP process **EOLOOP = 'Y'**.
- When use LOOP EOF, the last statement before LOOP_END must be **CLINE NEXT** so that the end of file condition get check for the next line. **EOF = 'Y'** if end of file condition reached. EOF condition set both EOF= 'Y' and EOLOOP = 'Y'
- You can also control your own loop process by setting a special REXX variable **EOLOOP**. Use a REXX statement REXX IF THEN EOLOOP = 'Y' to control when you want to exit from the current LOOP process.
- A special REXX variable **LP** contains the current looping count.
- Rest EOLOOP and EOF before another Find command or CLINE NEXT when multiple LOOP or multiple FIND are used. **REXX EOLOOP = 'N'; EOF = 'N'**

GETG

GETG rexx-variable=&GV

Use GETG inside ED macro to set a rexx variable to a DB2I2 global variable. The set rexx-variable can then be used for subsequent INSERT DB2I2 ED commands to substitute the {rexx-variable}in the inserting skeleton file.

SETG

SETG 'string-before' &GV 'string-after'

Use SETG inside ED macro to set DB2I2 global variable based on the pattern of the editing file. The set

DB2I2 global variable can then be used for subsequent DB2I2 commands.

Where &GV is the assigned global variable name - any alpha name prefixed with &, and 'string-before' and 'string-after' are a pair of strings which identify the global variable.

SETG2

SETG2 &GV =rexx-variable

Use SETG2 inside ED macro to set DB2I2 global variable to the content of a rexx-variable. The set DB2I2 global variable can then be used for subsequent DB2I2 commands. This way of assign global variable is easier than the SETG method.

TOP or BOTTOM

Use TOP or BOTTOM to position cursor to the first or last line of the edit file.

In addition, the following new ISPF EDIT macro can also be used with ED command. These new ISPF edit command are implemented as true EDIT macro. When you use them in online mode, you should use **ENTER** key to process them:

POSTMIGR

MIGR generates duplicate VIEW if a VIEW involves multiple tables or views. A new ISPF edit macro POSTMIGR is added to assist you to comment out duplicate CREATE VIEW, CREATE ALIAS, CREATE SYNONYMS, BIND and GRANT from MIGR output. The POSTMIGR edit macro can be used in both batch and online mode. To invoke POSTMIGR macro, you enter the following:

POSTMIGR [[:DLM]

The DLM is the delimiter which used with your MIGR command to generate the command scripts. Default if not specified is a semicolon.

DISTINCT

A new ISPF edit macro DISTINCT is added to assist you to delete duplicate lines. The DSITINCT edit macro can be used in both batch and online mode. To invoke DISTINCT macro, you enter the following:

DISTINCT starting-column,ending-column

NEWJOB

A new ISPF edit macro NEWJOB is added to assist you to add <NEWJOB> token to line object when data between starting-column and ending-column change. The NEWJOB edit macro can be used in both batch and online mode. To invoke NEWJOB macro, you enter the following:

NEWJOB starting-column,ending-column

FGET

A new ISPF edit macro FGET is added to assist you insert an external file before or after an existing line within your EDIT session. The FGET edit macro can be used in both online and batch mode.

To invoke FGET macro in online mode, you enter the following:

```
FGET 'file.information.tobe.inserted' [&hostvar=hostvar-value]
```

And either use a 'B' - before or 'A' - after line command to indicate where you want to insert the specified external file.

To invoke FGET macro in batch mode, you enter the following:

```
FGET 'file.information.tobe.inserted' TYPE=[B|A] POS=[*|line-no] [&hostvar=hostvar-value]
```

Both TYPE and POS are required options in batch mode. The specified file will be inserted before or after a specified line-no or current position (POS=*).

Use *name on the file name to get the system delivered UDF, MACRO,DDL from DB2I2.ISPFLIB.

For example,

```
FGET *DDLCOPYS
```

FPUT

A new ISPF edit macro FPUT is added to assist you to replacing or creating an external file. To invoke FPUT macro, you enter the following:

```
FPUT 'file.contain.extracted.information'
```

And either use 'CC' or 'MM' line command to either remove or copy the selected lines after replacing/creating the external file.

```
FPUT 'file.contain extracted.information' [POS=[*|line-no|*+##|*-# NOREC=# APPEND]
```

Both NOREC and POS are required options in batch mode. The output file can either be replaced or appended with extracted information with APPEND option.

MODEBTCH

A new ISPF edit macro MODEBTCH is added to simulate DB2I2 batch mode process when running RXDB2I2 in online mode. If you do not want to stop at end of each db2i2 command when bundle Db2I2 commands running Rxdb2i2 in online mode, MODEBTCH allows you to simulate batch process in online mode.

To invoke MODEBTCH macro in online mode, you enter the following:

```
MODEBTCH
```

MODERSTR

A new ISPF edit macro MODERSTR is added to return from simulate DB2I2 batch mode process after you are done with MODEBTCH. You must use MODERSTR after you issued MODEBTCH, otherwise your online session might run into some weird errors such as Lineobj missing errors.

To invoke MODERSTR macro in online mode, you enter the following:

```
MODERSTR
```

RESIZEIT

A new ISPF edit macro RESIZEIT is added to assist you to resize and add <NEWJOB> token on the line object depend on the accumulated size between starting-column and ending-column and the size info. If the accumulated size between starting-column and ending-column exceed the size specified, then a <NEWJOB> token is added onto the end of the line object. **Size** can be either a numeric number of **#JOB=##** to let DB2I2 to calculate the size for you based on the number of jobs requested (No of <NEWJOB> - 1).

Use LISTNM and LIST# options to append LISTNM=listnm## at end of each line. LIST# is not specified is default to 0

Use TP2TS option to resize it based on TS level instead of TP level.

```
RESIZEIT starting-column,ending-column,size|#JOB=##  
[LISTNM=listnm## [LIST#=#]] [TP2TS]
```

For Example,

```
RESIZEIT 49,54,#JOB=5 LISTNM=mydb01## LIST#=10
```

Will equally divide the selected list of line into five groups and each group of list will have LISTNM=mydb0111,LISTNM=mydb0112... append to the first line of each group.

The example below shows how to pick a RBA points from the result of DB2I2 DSNJU004 command, and assign to a global variable &RBA. The subsequent DB2I2 SETRBA command can then be used to assign it as the incore RBA, which then can be used with the subsequent RECOVER process.

```
//DB2I2CMD DD DATA,DLM=AA
DSNJU004   ODSN=T99
ED T99     MACRO(IDD=T99M)
SETRBA     &RBA
.
.
//T99M DD *
FIND 'RL ' 1 first
SETG 'RL ' &RBA ' '
/*
```

Which

- find the first line with 'RL ' on the first column
- set the global variable &RBA with the information between 'RL ' and ' '

Next example below uses FGET edit macro to insert a file before the first job step in all the members in a PDS file.

```
//DB2I2CMD DD DATA,DLM=AA
ED MY.JCL(MEM*)   MACRO(IDD=INSERTDD)
.
.
//INSERTDD DD *
FIND ' EXEC ' first
FGET jcl.file.to.be.inserted  TYPE=B POS=*
/*
```

Which

- find the first line with ' EXEC ' in MY.JCL will member name start with MEM
- insert jcl.file.to.be.inserted before the found ' EXEC ' line

Another example below demonstrates how to use LOOP to process a set of ED commands for a group of dataset until no more ' TABLE ' found.

```
//DB2I2CMD DD *
-- Issue REXX to prepare output dataset
REXX IDD=DD1
-- Scan all dataset with the following criteria and use instream edit macro M1
-- to extract TABLE information and generate a TB line object and output to the
ED 'SYSADM.DB00268.*.SYSPUNCH' MACRO(IDD=M1)
-- List the extracted file
FLIST EDF1
-- Delete the out file
TSO DELETE EDF1
//DD1      DD DATA,DLM=AA
/* REXX */
ADDRESS TSO "ALLOC FI(EDF1) DS(EDF1) REUSE NEW TRACKS SPACE(1,1)",
"UNIT(SYSALLDA) LRECL(80) BLKSIZE(8000)"
ADDRESS TSO "FREE  FI(EDF1)"
ADDRESS TSO "ALLOC FI(EDF1) DS(EDF1) REUSE MOD"
RETURN
AA
//M1      DD DATA,DLM=@@
F ' TABLE ' FIRST
LOOP
    CLINE
    REXX PARSE VALUE CLINE WITH ED_J1 ' TABLE ' ED_CREATOR ' ' ' ' +
    REXX ED_TBNAME ' ' ED_J2
    REXX ED_REC = 'TB ' ||STRIP(ED_CREATOR) || '.' ||ED_TBNAME
    REXX QUEUE ED_REC
    REXX "EXECIO 1 DISKW EDF1"
    F ' TABLE '
LOOP_END
```

Another example below demonstrates using ED to generating RECOVER to RBA process

```
//DB2I2CMD DD *
-- Input file I1 contains all TS lines needed to RECOVER to last QUIESCE point
-- Issue RBA to get all RBA points
RBA IDSN=I1 ODSN=O1
-- Find the first Quiesce line and set global variable &RBA to that Quiesce RBA
-- with M1 edit macro
ED O1 MACRO(IDD=M1)
-- set RBA with SETRBA command
SETRBA &RBA
REPORT parmutil(rptrba) IDSN=I1 ODSN=O2
ED O2 MACRO(IDD=M2)
//M1 DD DATA,DLM=@@
F ' Q ` 17 FIRST
SETG &RBA ` `
@@
//M2 DD DATA,DLM=MM
C 'DB2I2.REPORT.RECOVERY' 'MY.REPORT.RECOVERY'
F 'RBA='
CLINE
REXX PARSE VALUE CLINE WITH EDV1 'RBA=' ED_RBA ` ` EDV2
BOTTOM
INSERTA * IDD=RCVRDD
MM
//RCVRDD DD DATA,DLM=RR
..
..
//DB2I2CMD DD DATA,DLM=DD
SETRBA {ED_RBA}
RECOVER IDSN=MY.REPORT.RECOVERY parmutil(rcvrba) ODSN=O3
TSO SUB O3
DD
RR
```

Another example below demonstrates using ED and DB2I2 command inside ED macro to process each PDS member. The PDS member name is the creator of some DB2 table creator. After the execution of the following command, each PDS member contains the table creator and table name which belong to the PDS member.

```
ED ALL.TABLE(*) MACRO(IDD=DD1)
//DD1 DD DATA,DLM=AA
NUM OFF
REXX PARSE VALUE WSDSN WITH J1 '(' D2_CRE ')' J2
DB2I2 RUN +
DB2I2 T=N LIMIT(999) +
DB2I2 IDD=SQLDD +
DB2I2 ODSN='SYSADM.O2' +
DB2I2 &CRE={D2_CRE}
INSERTA 999 IDSN=O2
SAVE
DB2I2 FLIST {WSDSN} 79
AA
//SQLDD DD DATA,DLM=BB
SELECT CREATOR, NAME FROM SYSIBM.SYSTABLES
WHERE TYPE = 'T'
AND CREATOR = '&CRE';
BB
```

MYMENU

A new ISPF ED macro MYMENU is added to allow you to setup your own customization menu which allows you to

- o define your own selection menu from your predefined db2i2 script
- o substitute any rexx control variables with the information from your selection menu

The following screen shows you how to setup your selection menu and prepare skeleton to use MYMENU ED macro.

The screen below use SETG command to setup a global variable &v2 to MACRO(*MYMENU)

```
Command ==> SETG                               Scroll ==>CSR
***** Top of Data *****
s00001 GV &V2=MACRO(*MYMENU)
```

```
EDIT      JRHJ.DB2I2.WKBENCH                     Global Var &V2 is Replac
Command ==>                                       Scroll ==>CSR
***** Top of Data *****
000001 GV &V2=MACRO(*MYMENU)
```

The screen below use ED with &v2 to access MYMENU and store the output in my.output.jcl(myjcl)

```
Command ==> ed my.output.jcl(myjcl) &v2         Scroll ==>CSR
***** Top of Data *****
000001 GV &V2=MACRO(*MYMENU)
```

DB2I2 Reference Manual

After you enter the command, the following screen is displayed. This screen allows you to enter a dataset which contains your menu information. The information dataset should contain

- o The information of your panel library
- o And the description of this panel

```
Command ==> DB2I2 ed my.output.jcl(myjcl) &v2          Scroll ==> CSR
***** ***** Top of Data *****
00000 -----
00000 #MYMENU-----DB2I2 My Menu Setup Screen-----
00000 |
00000 | Please Enter The Name of Your Menu Information Dataset
00000 |
00000 | This Dataset Must Contain the Following Format
00000 | Dataset Information          Description
00000 | -----
00000 | your.ispfplib(xxxxxxxx)      My Menu Description
00001 | 'your.ispfplib(xxxxxxxx)'    My Menu Description
00001 |
00001 | DSN: 'JRHJ.MYPLIB(INFO)' _____
00001 |
00001 | PF3=Exit  Enter=Process Your Selection
00001 |-----
```

We have entered 5 panel and the description of each panel below.

```
EDIT      JRHJ.MYPLIB(INFO) - 01.07          Columns 00001 00072
Command ==>          Scroll ==>CSR
***** ***** Top of Data *****
=NOTE=* ----- *
=NOTE=*          DB2I2 Customize Menu Selection Screen For - JRHJ *
=NOTE=*          *
==MSG>*          Use DB2I2 TAG command to Select a Menu to Process *
=NOTE=*          *
=NOTE=* Information Entered Here Must Have The Following Format: *
=NOTE=* Dataset Information          Description *
=NOTE=* ----- *
==MSG> MYPLIB(RCVTOCP2)          Recover To Last Fullcopy2
==MSG> 'JRHJ.MYPLIB(RCVTOCPY)'    Recover To Last Fullcopy
=NOTE=* ----- *
000001 Dataset Information          Description
000002 -----
000003 'JRHJ.MYPLIB(RCVTOCPY)'    Recover To Last Fullcopy
000004 MYPLIB(RCVTOCP2)          Recover To Last Fullcopy
000005 'JRHJ.MYPLIB(RCVTOCP3)'    Recover To Last Fullcopy
000006 MYPLIB(REPAIR)            Repair option selection screen
000007 MYPLIB(UNLOAD)            Unload from Production Image Copy
***** ***** Bottom of Data *****
```


DB2I2 Reference Manual

Use TAG command to select the panel to be processed.

```

EDIT          JRHJ.MYPLIB(INFO) - 01.07                      Columns 00001 00072
Command ==> TAG                                           Scroll ==>CSR
***** Top of Data *****
=NOTE=* -----*
=NOTE=*          DB2I2 Customize Menu Selection Screen For - JRHJ          *
=NOTE=*          *
==MSG>*          Use DB2I2 TAG command to Select a Menu to Process      *
=NOTE=*          *
=NOTE=* Information Entered Here Must Have The Following Format:        *
=NOTE=* Dataset Information          Description                          *
=NOTE=* -----*
==MSG> MYPLIB(RCVTOCP2)          Recover To Last Fullcopy2
==MSG> 'JRHJ.MYPLIB(RCVTOCPY)'    Recover To Last Fullcopy
=NOTE=* -----*
000001 Dataset Information          Description
000002 -----*
s00003 'JRHJ.MYPLIB(RCVTOCPY)'    Recover To Last Fullcopy
000004 MYPLIB(RCVTOCP2)          Recover To Last Fullcopy
000005 'JRHJ.MYPLIB(RCVTOCP3)'    Recover To Last Fullcopy
000006 MYPLIB(REPAIR)          Repair option selection screen
000007 MYPLIB(UNLOAD)          Unload from Production Image Copy
***** Bottom of Data *****

```

EDIT	JRHJ.MYPLIB(INFO) - 01.07	Record Tagged
Command ==>	TAG	Scroll ==>CSR
***** Top of Data *****		
=NOTE=* -----*		*
=NOTE=* DB2I2 Customize Menu Selection Screen For - JRHJ *		*
=NOTE=* *		*
==MSG>* Use DB2I2 TAG command to Select a Menu to Process *		*
=NOTE=* *		*
=NOTE=* Information Entered Here Must Have The Following Format: *		*
=NOTE=* Dataset Information Description *		*
=NOTE=* -----*		*
==MSG> MYPLIB(RCVTOCP2) Recover To Last Fullcopy2		
==MSG> 'JRHJ.MYPLIB(RCVTOCPY)' Recover To Last Fullcopy		
=NOTE=* -----*		*
000001 Dataset Information Description		
000002 -----*		
000003 'JRHJ.MYPLIB(RCVTOCPY)'	Recover To Last Fullcopy	
000004 MYPLIB(RCVTOCP2)	Recover To Last Fullcopy	
000005 'JRHJ.MYPLIB(RCVTOCP3)'	Recover To Last Fullcopy	
000006 MYPLIB(REPAIR)	Repair option selection screen	
000007 MYPLIB(UNLOAD)	Unload from Production Image Copy	
***** Bottom of Data *****		

DB2I2 Reference Manual

RCVTCOPY is defined as follow:

```
)ATTR DEFAULT (%+_ )
  [ TYPE(INPUT) INTENS(HIGH) COLOR(RED) PAD('_')
  | TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
  @ TYPE(TEXT) INTENS(HIGH) COLOR(GREEN) HILITE(REVERSE)
)BODY WINDOW(65,10) EXPAND(!!)
|RCVTOCPY+!-!Recover To Last Full Copy Options-!-!
%
% SSID [Z + Sub-System ID (PROD,DSNT)
% TableSpace [Z +TableSpace Name
% Work DS [Z + Template Substitute for Work DS
% Job Class [Z+ (F/R)
%
% JCL skelton [Z +
%
@PF3=Exit Enter=Process Your Selection
)INIT
.ZVARS = '(SSID,TS,DS,CLS,JCLDS)'
.CURSOR = SSID
&SSID = 'DSNT'
&CLS = 'F'
&DS = '~DB..~TS..D~DT..T~TI.' /* Use ~ for & */
&JCLDS = ''JRHJ.V7.SKEL(RCVTOCPY)''
&ZPRIM = NO /* &ZPRIM=YES */
&ZPLACE = TOP
)PROC
VER(&SSID,NB)
VER(&TS,NB)
VER(&DS,NB)
VER(&CLS,NB)
VER(&JCLDS,NB)
VPUT (SSID,TS,CLS) SHARED
VPUT (DS,JCLDS) SHARED
)END
```

```
Command ==> DB2I2 ed my.output.jcl(myjcl) &v2 Scroll ==> CSR
***** ***** Top of Data *****
000001 GV &V2=MACRO(*MYMENU)
000002 GV &V1=MACRO(EDMACRO(MYMENU))
000003 -----
000004 | RCVTOCPY -----Recover To Last Full Copy Options-----
000005 |
000006 | SSID DSNT Sub-System ID (DSNT,PROD)
000007 | TableSpace DJRHJ.SJRHJ_____ TableSpace Name
000008 | Work DS ~DB..~TS..D~DT. Template Substitute for Work DS
000009 | Job Class F (F/R)
000010 |
000011 | JCL skelton 'JRHJ.V7.SKEL(RCVTOCPY)'_____
000012 |
000013 | PF3=Exit Enter=Process Your Selection
000014 | -----
```

DB2I2 Reference Manual

Skeleton is defined as follow:

```
EDIT          JRHJ.V7.SKEL(RCVTOCPY) - 01.01          Columns 00001 00072
Command ==>>          Scroll ==>> CSR
*****          ***** Top of Data *****
000001 //{USERID}RCVR JOB JRHJ001.J.DOONG,CLASS={CLS},
000002 //          MSGCLASS=R,MSGLEVEL=(1,1),NOTIFY={USERID}
000003 //STEP001 EXEC DSNUPROC,REGION=4M,COND=(8,LT),
000004 //          SYSTEM='{SSID}',UID='{USERID}.RCVR',UTPROC=''
000005 //SYSPRINT DD SYSOUT=*
000006 //SYSUDUMP DD SYSOUT=*
000007 //UTPRINT DD SYSOUT=*
000008 //* OPTIONS  PREVIEW
000009 //DFSPARM DD *
000010 DYNALLOC=(SYSALLDA,5)
000011 //SYSIN DD *
000012 TEMPLATE TYSUT1 DSN {USERID}.{DS}.SYSUT1
000013 UNIT SYSALLDA DISP(NEW,DELETE,CATLG)
000014 LISTDEF L1 INCLUDE TABLESPACE {TS}
000015 RECOVER LIST L1
000016 TOLASTFULLCOPY
000017 REBUILD INDEX LIST L1
000018 STATISTICS KEYCARD
*****          ***** Bottom of Data *****
```

The result jcl is displayed below.

```
EDIT          JRHJ.MY.JCL(MYJCL) - 01.00          Columns 00001 00072
Command ==>>          Scroll ==>> CSR
*****          ***** Top of Data *****
000001 //JRHJRCVR JOB JRHJ001.J.DOONG,CLASS=F
000002 //          MSGCLASS=R,MSGLEVEL=(1,1),NOTIFY=JRHJ
000003 //STEP001 EXEC DSNUPROC,REGION=4M,COND=(8,LT)
000004 //          SYSTEM='DSNT',UID='JRHJ.RCVR',UTPROC=''
000005 //SYSPRINT DD SYSOUT=*
000006 //SYSUDUMP DD SYSOUT=*
000007 //UTPRINT DD SYSOUT=*
000008 //* OPTIONS  PREVIEW
000009 //DFSPARM DD *
000010 DYNALLOC=(SYSALLDA,5)
000011 //SYSIN DD *
==CHG> TEMPLATE TYSUT1 DSN=JRHJ.&DB..&TS..D&DT..SYSUT1
000013 UNIT SYSALLDA DISP(NEW,DELETE,CATLG)
000014 LISTDEF L1 INCLUDE TABLESPACE DJRHJ.SJRHJ
000015 RECOVER LIST L1
000016 TOLASTFULLCOPY
000017 REBUILD INDEX LIST L1
000018 STATISTICS KEYCARD
*****          ***** Bottom of Data *****
```

You can use FGET ed macro to copy sample Panel and sample Skeleton
FGET *MYMENUPL to get a copy of the sample panel as displayed above, and
FGET *MYMENUSK to get a copy of the sample skeleton as displayed above.

EDIT obsoleted and replaced by ED command

Command Syntax:	EDIT & END_EDIT (obsolete and replaced by ED command)
Line objects allowed:	N/A
Process Mode:	Batch only
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	No

Command Description

- In batch mode, you can use DB2I2 EDIT and END_EDIT pair commands to process ISPF edit commands.
- You should use new ED command, which works both online and batch mode, instead of the EDIT command to edit a dataset.
- To invoke the ISPF EDIT in batch mode, you enter EDIT 'edit.dsname' to initiate the ISPF edit mode.
EDIT 'edit.dsname'
- Then you can enter the following ISPF edit commands:
 - C** 'aaa' 'bbb' ALL to changes all 'aaa' to 'bbb' in batch mode
 - INSERT** 1 //first line of job card to insert a line on the first line of the edit file
 - X** ALL to exclude all lines
 - F** .PG ALL to find all .PG line objects
 - DEL** X ALL to delete all excluded lines
- Use the **END_EDIT** command to end the ISPF edit session in batch mode.
END_EDIT
- A new ISPF edit macro DISTINCT is added to assist you to delete duplicate lines. The DSITINCT edit macro can be used in both batch and online mode. To invoke DISTINCT macro, you enter the following:
DISTINCT starting-column,ending-column
- A new ISPF edit macro RESIZEIT is added to assist you to resize and add <NEWJOB> token on the line object depend on the accumulated size between starting-column and ending-column and the size info. If the accumulated size between starting-column and ending-column exceed the size specified, then a <NEWJOB> token is added onto the end of the line object:
RESIZEIT starting-column,ending-column,sizeI

The following screen shows how you can use RESIZEIT edit macro to resize and add <NEWJOB> token to line object. The information displayed is generated with LISTC command. The size information, in cylinders, is located at column 47 and 54. The intend is to generate <NEWJOB> token for every 40 cylinders.

```
Command ==> resizeit 47,54,40                               Scroll ==> CSR
***** ***** Top of Data *****
000001   Name                               Part Extents   CYLS
ss0002 TP DJRHJ.SN3B017                     0     1         000088
000003 TP DJRHJ.SN3B018                     0     1         000029
000004 TP DJRHJ.SN3B020                     0     1         000022
000005 TP DJRHJ.SN3B021                     0     1         000035
000006 TP DJRHJ.SN3B022                     0     1         000004
ss0007 TP DJRHJ.SN3B023                     0     1         000002
```

The result of the RESIZEIT id displayed below.

000001	Name	Part	Extents	CYLS	
000002	TP DJRHJ.SN3B023	0	1	000002	0000000002.0
000003	TP DJRHJ.SN3B022	0	1	000004	0000000006.0
000004	TP DJRHJ.SN3B020	0	1	000022	0000000028.0
000005	TP DJRHJ.SN3B018	0	1	000029	0000000029.0 <NEWJOB>
000006	TP DJRHJ.SN3B021	0	1	000035	0000000035.0 <NEWJOB>
000007	TP DJRHJ.SN3B017	0	1	000088	0000000088.0 <NEWJOB>

You can then use the output as input to any of the DB2 utility generation command, which allow you to balance the workload based on the size of dataset.

EXEC

Command Syntax:	EXEC [restart line no] [ERROR(CONTINUE SKIP #)] [RC0] [SQLTERM(?)]
Line objects allowed:	DB2I2 scripts (include DDL, DML DB2 command, IDCAMS command)
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	N/A
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 EXEC command to execute **DB2I2 command scripts** which consists of the following:
 - DB2 command
 - IDCAMS command
 - DDL or DCL statements(NON-SELECT SQL statements)
- The formats of the DB2I2 scripts are:
 - DB2 command DB2CMD in position 1-6
 - IDCAMS command IDCAMS in position 1-6
 - Non-SELECT DDL or DCL commands any other line input does not contain DB2CMD or IDCAMS in position 1-6
- DB2I2 EXEC command invokes IBM DSNTIAD internal, you can specify RC0 command option to return RC=0 even with non-zero SQL return code.
- In batch mode, you can use ERROR(CONTINUE) or ERROR(SKIP #) option to continue execute the next command in sequence or skip # commands if there is an error occurs during the current command process.
- Use option restart line no to start the execution from the middle of the EXEC script instead from the beginning.
- Specify SQLTERM(?) to indicate that the SQL to be processed contain ? as SQL terminator instead of the default terminator ;. You can only use this command option if your DB2 environment is V6 or above.
- Use '+' on the last position of a DDL line to indicate the next DDL line is a continuation from the current DDL line.

For example, DB2I2 EXEC command treats the following DDL lines are all the same:

```

.....5.....1.....7..
comment on table DB2ADM.db2amrpt_table is      '123456789abcdef';
comment on table DB2ADM.db2amrpt_table is      '1234
56789abcdef';
comment on table DB2ADM.db2amrpt_table is      '123+
456789abcdef';

```

Example

The following example demonstrates how to use EXEC command to execute a combination of DB2 command, IDCAMS command and SQL DDL.

```

Command ==> exec
ss0001 DB2CMD -DISPLAY THREAD(*)
000002
000003 IDCAMS LISTC LVL(TDB2.DSNDBC.DBTST001)
000004
000005 CREATE TABLE T3 LIKE T2 IN DBTST001.STS00002;
000006 COMMIT;
000007 DROP TABLE JD00.T3;
000008 COMMIT;
000009
000010 DB2CMD -DISPLAY THREAD(*)
000011
ss0012 DB2CMD -DISPLAY DB(DBTST001) SPACE(*)
    
```

Scroll ==> CSR

The execution result from the previous EXEC command is displayed below.

```

Command ==>
***** Top of Data *****
** ===== **
**          DB2I2 Command/Batch Process          **
**          By JRH GoldenState Software Inc.      **
**          COPYRIGHTED 1997-2011                **
**                                          UserID: JD00 **
**                                          Date: 12/22/10 **
**                                          Time: 16:40:49 **
** ===== **
*          Process DB2CMD Between 0001 and 0001          *
-----
0001 DB2CMD -DISPLAY THREAD(*)
-----
DSNV401I -DSN DISPLAY THREAD REPORT FOLLOWS -
DSNV402I -DSN ACTIVE THREADS -
NAME  ST A  REQ ID          AUTHID  PLAN      ASID  TOKEN
TSO   N    89  CJ18          CJ18    PLAN    006E  0
TSO   N    89  CJ18          CJ18    PLAN    006E  0
      V444-USHON000.U002DSN.AFBFFEF09164=3001 ACCESSING DATA AT
      V446-AHMSSTR
.
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I -DSN DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION
*          Process IDCAMS Between 0002 and 0003          *
-----
0002
0003 IDCAMS LISTC LVL(TDB2.DSNDBC.DBTST001)
-----
IDCAMS SYSTEM SERVICES                                TIME: 16:40:53
LISTC LVL(TDB2.DSNDBC.DBTST001)
NONVSAM ----- TDB2.DSNDBC.DBTST001.IXCA4903.I0001.A001
IN-CAT --- ICF.CAT03.ICFCAT
-----
*          Process DDL Between 0004 and 0008          *
-----
0004
0005 CREATE TABLE T3 LIKE JD00.T2 IN DBTST001.STS00002;
0006 COMMIT;
0007 DROP T3;
0008 COMMIT;
-----
DSNTIAD - SAMPLE DYNAMIC SQL PROGRAM 2.0
CREATE TABLE T3 LIKE JD00.T2 IN DBTST001.STS00002
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION
COMMIT
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION
DROP T3
SQL ERROR DURING EXECUTE IMMEDIATE
COMMIT
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION
.
*          Process DB2CMD Between 0009 and 0010          *
-----
0009
0010 DB2CMD -DISPLAY THREAD(*)
-----
DSNV401I -DSN DISPLAY THREAD REPORT FOLLOWS -
DSNV402I -DSN ACTIVE THREADS -
NAME  ST A  REQ ID          AUTHID  PLAN      ASID  TOKEN
.
DSN9022I -DSN DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION
*          Process DB2CMD Between 0011 and 0012          *
-----
0011
0012 DB2CMD -DISPLAY DB(DBTST001) SPACE(*)
-----
DSNT360I -DSN *****
    
```

Scroll ==> CSR

EXPLAIN

Command Syntax:	EXPLAIN [QNO=Q#] [O=plan_table owner] [DESC=Y N] [DEGREE=degree] [DET=O N Y] [SQLTERM(?)] [SU=service-unit thresh hold] [Q=Qualifier] [EDIT=Y]
Line objects allowed:	SQL statement block or DBRM declare cursor statement block SQL statement drilldown directly from PG or DM
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	N/A
Output Type:	Report
Reusable:	No

Command Description

- Use EXPLAIN command to invoke DB2 Explain facility against a SQL block or a DBRM DECLARE CURSOR statement block drilled down from PL or PG commands.
- Use QNO=# option to assign the starting QUERYNO, if not specified default to 0.
- If you have system administration authorization, you can Specify O option to switch PLAN_TABLE used for the explain command. Otherwise, the PLAN_TABLE of your TSO id will be used for the explain output.
- Use Q= to override qualifier name for unqualified object name from DBRM or PACKAGE drill down. Otherwise, Db2I2 uses \$ Q=qualifier from drilldown command as the qualifier name.
- The default DET=O option to display catalog statistics ONLY for questionable SQL. These SQL statements include
 - Table Space scan (access type = R)
 - Merge scan join (method = 2)
 - Hybrid join (method = 4)
 - Sort with ORDER BY, GROUP BY, DISTINCT, UNION, IN and LIST PREFETCH (method = 3 and PREFETCH = L)
 - Index scan with no matching columns (access type = I and match column = 0)
- Use DET=Y or DET=N to enable or disable the display catalog statistics for selected SQL.
- Use DEGREE=Degree to assign the CURRENT DEGREE for the selected explained SQL. Degree if specified, must be a numeric number or ANY.
- By default, DB2I2 displays all detail description on the explain output. If you do not want to display these descriptions, you can use DESC=N to disable the generation of report detail description.
- Use SQLTERM(?) to assign SQL terminator if you want to explain multiple SQL and use SQL terminator other than the default terminator ;
- Use SU=service-unit thresh hold option to display only for those queries have service unit greater or equal to the service unit thresh hold specified.
- When EXPLAIN a DBRM or PACKAGE drill down, the QUERYNO fields are ignored and being treated as SQL terminator. The following lines are excluded:
 - on the first 2 position of the line
 - * on the first position of the line
 - Stmt# on the first 5 position of the line
 - Any non-SELECT, INSERT, UPDATE, DELETE sqls
- Use EDIT=Y to edit the result report instead of browse it.

Example

The following example demonstrates how to invoke explain facility against a SQL block with default query block number 0.

```

Command ==> EXPLAIN DET=Y                               Scroll ==> CSR
SS0095      SELECT  A.LOCATION, A.COLLDID, A.NAME, A.CONTOKEN
000096      , A.OWNER, A.CREATOR, A.TIMESTAMP, A.BINDTIME
000097      , A.QUALIFIER, A.PKSIZE, A.AVGSIZE, A.SYSENTRIES
000098      , A.VALID, A.OPERATIVE, A.VALIDATE, A.ISOLATION
000099      , A.RELEASE, A.EXPLAIN, A.QUOTE, A.COMMA, A.HOSTLANG
000100      , A.CHARSET, A.MIXED, A.DEC31, A.DEFERPREP, A.SQLError
000101      , A.REMOTE, A.PCTIMESTAMP, A.IBMREQD, A.VERSION
000102      , A.PDSNAME, A.DEGREE, A.GROUP_MEMBER, A.DYNAMICRULES
000103      , A.REOPTVAR, A.DEFERPREPARE, A.KEEPDYNAMIC
SS0104      FROM SYSIBM.SYSPACKAGE A
    
```

The result from the previous EXPLAIN command are displayed below. Notices that DB2I2 uses query block number 0 as the default query block number for the output of EXPLAIN command.

```

Command ==>                                             Scroll ==> CSR
***** Top of Data *****
DB2I2 - Dynamic Explain Result Report
***** SQL statements Evaluated *****
SELECT  A.LOCATION, A.COLLDID, A.NAME, A.CONTOKEN
, A.OWNER, A.CREATOR, A.TIMESTAMP, A.BINDTIME
, A.QUALIFIER, A.PKSIZE, A.AVGSIZE, A.SYSENTRIES
, A.VALID, A.OPERATIVE, A.VALIDATE, A.ISOLATION
, A.RELEASE, A.EXPLAIN, A.QUOTE, A.COMMA, A.HOSTLANG
, A.CHARSET, A.MIXED, A.DEC31, A.DEFERPREP, A.SQLError
, A.REMOTE, A.PCTIMESTAMP, A.IBMREQD, A.VERSION
, A.PDSNAME, A.DEGREE, A.GROUP_MEMBER, A.DYNAMICRULES
, A.REOPTVAR, A.DEFERPREPARE, A.KEEPDYNAMIC
FROM SYSIBM.SYSPACKAGE A
*****
DB2I2-EXPLAIN
          Q          B          M          I          A          A          J          J          S          S          P          M          W
          U          L          P          M          C          C          E          L          X          D          P          D          P          C          N          A          J          G          I          N
          E          O          L          E          T          C          H          X          O          O          O          O          O          O          O          S          F          F          O          E          G          E          G          P          P          L          O          R          N          O
          R          C          A          T          A          T          C          O          R          R          R          R          R          R          R          L          E          N          P          G          R          G          R          G          M          N          A          T          P
          Y          K          N          H          B          Y          O          N          T          T          T          T          T          T          T          O          T          E          S          R          P          R          P          R          O          C          O          N          Y          T
          N          N          N          O          T          A          B          L          E          N          P          L          A          C          C          C          C          C          C          C          V          E          E          I          E          I          P          D          O          G          P          I
QBLOCK
PROGNAME O      O      O      D      NAME      O      E      S      NAME      Y      U      J      O      G      U      J      O      G      K      H      L      Q      E      D      E      D      I      D      I      D      E      L      E      E      M      TYPE
-----
DB2I2A  0      ***** Table Space Scan
          + =====+
          | TableSpace Information |
          + =====+
          DBname.TSname..... Stats.Date NACTIVE
          DSNDB06.SYSPKAGE      09/17/2000 93369
          + =====+
          | TablePart Information |
          + =====+
          DBname.TSname      PT Stats Date CARD      FARINDREF      NEARINDREF      %Act %Drp
          DSNDB06.SYSPKAGE  0 09/17/2000 1638488      0      0      95 0
          + =====+
          | Table Information |
          + =====+
          Table Name..... Stats Date CARDF..... NPAGES.... RECLENGTH
          SYSIBM.SYSPACKAGE  09/17/2000 33392      1488      265
          + =====+
          | Column Information |
          + =====+
          COLNAME      COLCARDF      HIGH2KEY      LOW2KEY
          LOCATION      1
          COLLID      537      WPRT3      AADB2COL
          NAME      19638      YPB472      ADSQID00
          CONTOKEN      26828      | / & .-...
          OWNER      86      VE0012      CMNP01
          CREATOR      70      VE0012      CMNT01
          TIMESTAMP      76405      .....q .k... .
          BINDTIME      18600      ..... .k.... &
          QUALIFIER      168      WPRG2      ARCD1
          PKSIZE      448      .m      .
          AVGSIZE      3161      . 8      .
          SYSENTRIES      1      .
          VALID      3      N      N
          OPERATIVE      2      Y      N
          VALIDATE      2      R      B
          ISOLATION      5      T      R
          RELEASE      3      C      C
          EXPLAIN      2      Y      N
          QUOTE      1      N      N
          COMMA      1      N      N
          HOSTLANG      6      P      B
          CHARSET      2      K      A
          MIXED      2      Y      N
          DEC31      1      N      N
          DEFERPREP      2      C      B
    
```

DB2I2 Reference Manual

```

SQLERROR          2          N          C
REMOTE            3          N          N
PCTIMESTAMP      26828      . . . . .g .j. . .
IBMREQD          4          H          E
VERSION          256        2000-07- V230
PDSNAME          116        TEST.DBR CMNPN.B1
DEGREE           3          ANY        ANY
GROUP_MEMBER     1
DYNAMICRULES     3          B          B
REOPTVAR         1          N          N
DEFERPREPARE     2          N
KEEPDYNAMIC      1          N          N
+ =====+
| ColumnDist Information |
+ =====+
COLNAME          FREQUENCYF  TYPE  CARDF          NUMCOL  COLGROUPCOLNO  COLVALUE
COLLID           364          F     -1          1          PRIBBIND
COLLID           365          F     -1          1          PRIG5
COLLID           365          F     -1          1          PRID5
COLLID           365          F     -1          1          PRIE1
COLLID           367          F     -1          1          PRIG1
COLLID           392          F     -1          1          PRIABIND
COLLID           421          F     -1          1          PRISBIND
COLLID           464          F     -1          1          PRITBIND
COLLID           756          F     -1          1          HUNOB1
COLLID           906          F     -1          1          HUNSB1
LOCATION          10000       F     -1          1
+ =====+
| Index Information |
+ =====+
Index Name: SYSADMD.XSNKX01          Table Name: SYSIBM.SYSPACKAGE
+ =====+
| IndexColumn Information |
+ =====+
IC COLLID        ASC
IC NAME          ASC
Stats.Date CG CD CTO TYPE FIRSTKEYCARDF  FULLKEYCARDF  NLEAF  NLEVEL
09/17/2000 N Y 99 2 537 32576 318 3
Index Name: SYSIBM.DSNKX01          Table Name: SYSIBM.SYSPACKAGE
+ =====+
| IndexColumn Information |
+ =====+
IC LOCATION      ASC
IC COLLID        ASC
IC NAME          ASC
IC VERSION       ASC
Stats.Date CG CD CTO TYPE FIRSTKEYCARDF  FULLKEYCARDF  NLEAF  NLEVEL
09/17/2000 Y Y 99 2 1 33392 955 3
Index Name: SYSIBM.DSNKX02          Table Name: SYSIBM.SYSPACKAGE
+ =====+
| IndexColumn Information |
+ =====+
IC LOCATION      ASC
IC COLLID        ASC
IC NAME          ASC
IC CONTOKEN      ASC
Stats.Date CG CD CTO TYPE FIRSTKEYCARDF  FULLKEYCARDF  NLEAF  NLEVEL
09/17/2000 N Y 99 2 1 33392 478 3
+ =====+
| IndexPart Information |
+ =====+
Index Name      IP Stats Date CARDF          FAROFFPOS      NEAROFFPOS      LEAFDIST
XSNKX01         0 09/17/2000 33392          1              0                0
DSNKX01         0 09/17/2000 33392          1              0                1
DSNKX02         0 09/17/2000 33392          1              0                1
DB2I2A 0 1 1 0 SYSPACKAGE 1 R 0          N N N N N N N N N N IS S 0          SELECT

```

Column Name	Description
PROGNAME	The name of the program or package containing the statement being explained. Applies only to embedded EXPLAIN statements and to statements explained as the result of binding a plan or package. Blank if not applicable.
QUERYNO	A number intended to identify the statement being explained. For a row produced by an EXPLAIN statement, you can specify the number in the SET QUERYNO clause; otherwise, DB2 assigns a number based on the line number of the SQL statement in the source program. Values of QUERYNO greater than 32767 are reported as 0. Hence, in a very long program, the value is not guaranteed to be unique. If QUERYNO is not unique, the value of TIMESTAMP is unique.
QBLOCKNO	The position of the query in the statement being explained (1 for the outermost query, 2 for the next query, and so forth). For better performance,

The example below uses O (PLAN_TABLE owner option) to explain the DECLARE CURSOR block from a DB2 package drill down and save the explain output in specified JDOO.PLAN_TABLE. You can only use this option if

you have authority to execute DB2 SET CURRENT SQLID command.

If drill down of a DB2 package or DBRM DECLARE CURSOR statement does not qualify the table name, to use EXPLAIN command correctly, you either have to insert the creator name before the table name to inform DB2I2 which table to be used for EXPLAIN or use Q=qualifier option to assign unqualified object name . The example below, we insert JRHJ. In front of TN0A007 as table creator to be used in the DB2I2 EXPLAIN command.

```

Command ==> explain o=JD00                               Scroll ==> CSR
***** ***** Top of Data *****
000001 pg .JD00.%
==MSG> Collid.Name..... Valid. Operative
000002 PG .JD00.AIXCOB          N      Y
000003 Stmt# -----SQL Statements-----
000004 00079 DECLARE JD00 . TN0A007 TABLE ( SERV_ID CHAR ( 8 ) NOT NULL ,
000005         CALLED_SERV_ID CHAR ( 8 ) NOT NULL )
ss0006 00099 DECLARE C1 CURSOR FOR SELECT SERV_ID , CALLED_SERV_ID FROM
ss0007         JRHJ . TN0A007 WHERE SERV_ID = : H
000008 00111 WHENEVER NOT FOUND CONTINUE
000009 00113 WHENEVER SQLWARNING GO TO DBERROR
000010 00114 WHENEVER SQLERROR GO TO DBERROR
    
```

The result from the previous EXPLAIN command with O option is displayed below.

```

Command ==>
***** ***** Top of Data *****                               Scroll ==> CSR
DB2I2-EXPLAIN
          M          A          I
          E          C          N
          T          C          D
          H          T          X
          O          A          O          T          P          M
          O          S          C          L          E          P          R          J
          O          B          T          N          S          S          S          S          S          S          S          S          O          T          L          M          I          X          L          C          N
          O          N          Y          L          ACCESS          L          N          N          N          C          C          C          C          C          C          F          O          P          M          O          T
          O          P          S          CREATOR          ACCESSNAME          Y          U          J          O          G          U          J          O          G          K          H          E          SEQ          D          L          Y
-----
APPLNAME  PROGNAME  QN  QB  PL  D  CREATOR  TNAME  O  P  S  CREATOR  ACCESSNAME  Y  U  J  O  G  U  J  O  G  K  H  E  SEQ  D  L  Y
-----
          DB2I2A  0  1  1  0  JRHJ  TN0A007  1  I  1  JD00  XN0A0070  Y  N  N  N  N  N  N  N  N  N  N  IS  0
          .
          .
          .
    
```

EXPLAIN

Command Syntax:	EXPLAIN [O=plan table owner] [PG=program name] [PL=Planname] [CL=collection] [DESC=Y N] [DET=O N Y S] [GN=generation] [QNO=[##1-#2]][SU=service-unit thresh hold] [HOSTAVR] [EDIT=Y]
Line objects allowed:	PL, PG, DM PG and DM appended with STMT# to display only a specific statement information from the selected PG or DM
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	Report
Reusable:	No

Command Description

- Use EXPLAINP command to display the content from an existing PLAN_TABLE.
- By default, the PLAN_TABLE of your own TSO ID is used as input to the EXPLAINP command. You can use O (owner) option to switch to different PLAN_TABLE as input to the EXPLAINP command. You must have SELECT authorization of the PLAN_TABLE you try to switch to.
- Use PG (program name) option to select specific program from plan table to report.
- Use PL (plan name) option to select specific plan from plan table to report.
- Use CL (collection) option to select specific collection from plan table to report
- Use QNO=# (query no) option to select specific query no from plan table to report
- Use QNO=#1-#2 to select a range of query no between #1 and #2 to report
- You can use any combination of PL, PG and CL options.
- Wild card % can be used for PL, PG and CL options.
- When your PLAN_TABLE contains multiple generations of explain output for the same program, by default, EXPLAINP will report all of those generations. Specify GN (generation) option select only one of those generations to report. Specify GN=0 to report current generation explain output, GN=-1 to report previous generation, etc.
- The default DET=O option to display catalog statistics ONLY for questionable SQL. These SQL statements include
 - Table Space scan (access type = R)
 - Merge scan join (method = 2)
 - Hybrid join (method = 4)
 - Sort with ORDER BY, GROUP BY, DISTINCT, UNION, IN and LIST PREFETCH (method = 3 and PREFETCH = L)
 - Index scan with no matching columns (access type = I and match column = 0)
 - Multiple-indexes access
- Use DET=Y or DET=N to enable or disable the display catalog statistics for selected SQL.
- Use DET=S to display SQL statements only without catalog statistics.

DB2I2 Reference Manual

- By default, EXPLAIN command generates a detail report description, which helps you to understand the content of the report. It takes about 2 pages. If you already familiar with the content of the EXPLAIN output and do not want to print these detail description, you can use DESC=N option to disable it.
- Use SU=service-unit thread hold option to display only for those queries have service unit greater or equal to the service unit thread hold specified.
- Use HOSTAVR option to display host variable information.
- Use EDIT=Y to edit the result instead of browse it.

Example

The following example demonstrates how to use EXPLAIN command to display all the content from PLAN_TABLE with specific program name.

```
Command ==> explain pg=mc674src desc=n det=0                               Scroll ==> CSR
***** Top of Data *****
```

The result from the previous EXPLAIN command is displayed below.

```
DB2I2-EXPLAIN
          Q          M          I          A A J J S S P M          W
          B          A          N          P C M C C O O R R A R          J H
          L P M          C          D          R O I C C N N T T R G P O E
          O L E          C          E          E L X D P D P C N A J G I N
          C A T          T C H          X O O O O O O O O S          F F O E G E G P P L O R N O
          K N H          A T C          O R R R R R R R R R L          E N P G R G R G G M N A T P
          N N O TABLE          B Y O          N T T T T T T T T O          T E S R P R P R P R O C N Y T
          O O O D NAME          O E S          L N N N N C C C C C          C V E E I E I P P D O G P I QBLOCK
          PROGNAME          ACCESS          Y U J O G U J O G K          H L Q E D E D I D I D E L E E M TYPE
-----
MC674SRC 1206 1 1 0 PRODUCT 3 I 2 PROX0 Y N N N N N N N N IS 0 SELECT
MC674SRC 1206 1 2 1 CUST_LOB_ROLE_PROD 2 I 3 CLRX0 N N N N N N N N N IS 0 SELECT
MC674SRC 1206 1 3 1 CUSTOMER 1 I 1 CUSX0 N N N N N N N N N IS 0 SELECT
MC674SRC 1206 1 4 1 CUST_LOB 4 I 3 CLBX0 Y N N N N N N N N IS 0 SELECT
MC674SRC 1251 1 1 0 PRODUCT 3 I 2 PROX0 Y N N N N N N N N IS 0 SELECT
MC674SRC 1251 1 2 1 CUST_LOB_ROLE_PROD 2 I 3 CLRX0 N N N N N N N N N IS 0 SELECT
MC674SRC 1251 1 3 1 CUSTOMER 1 I 1 CUSX0 N N N N N N N N N IS 0 SELECT
MC674SRC 1251 1 4 1 CUST_LOB 4 I 3 CLBX0 Y N N N N N N N N IS 0 SELECT
MC674SRC 1371 ***** Table Space Scan
+ =====+
+ | TableSpace Information |
+ =====+
DBname.TSname..... Stats.Date NACTIVE
DBMRDG21.LODS 11/18/2000 18
+ =====+
+ | TablePart Information |
+ =====+
DBname.TSname PT Stats Date CARD FARINDREF NEARINDREF %Act %Drp
DBMRDG21.LODS 0 11/18/2000 134 0 0 7 0
+ =====+
+ | Table Information |
+ =====+
Table Name..... Stats Date CARDF..... NPAGES.... RECLENGTH
MRDG2.LODGE 11/18/2000 134 2 46
+ =====+
+ | Column Information |
+ =====+
COLNAME COLCARDF HIGH2KEY LOW2KEY
ID_NUMBER 128 cX .
FK_ACL_CLUB_C 5 601 018
LODGE_NAME 134 WHITTIER AAA DISN
EFFECTIVE_DATE 2 ....
EXPIRATION_DATE 1 rr.. rr..
+ =====+
+ | ColumnDist Information |
+ =====+
COLNAME FREQUENCYF TYPE CARDF NUMCOL COLGROUPCOLNO COLVALUE
FK_ACL_CLUB_C 74 F -1 1 1 999
FK_ACL_CLUB_C 298 F -1 1 1 018
FK_ACL_CLUB_C 522 F -1 1 1 601
FK_ACL_CLUB_C 1716 F -1 1 1 252
FK_ACL_CLUB_C 7388 F -1 1 1 004
+ =====+
+ | Index Information |
+ =====+
Index Name: MRDG2.LODX0 Table Name: MRDG2.LODGE
+ =====+
+ | IndexColumn Information |
+ =====+
IC FK_ACL_CLUB_C ASC
IC ID_NUMBER ASC
Stats.Date CG CD CTO TYPE FIRSTKEYCARDF FULLKEYCARDF NLEAF NLEVEL
11/18/2000 N Y 97 2 5 134 1 2
+ =====+
+ | IndexPart Information |
```

DB212 Reference Manual

```

+ =====+
Index Name      IP Stats Date CARDF      FAROFFPOS      NEAROFFPOS      LEAFDIST
MC674SRC 1371  1 1 0 LODGE      1 R 0          0              0                0
MC674SRC 1371  ***** Hybrid Join
+ =====+
| TableSpace Information |
+ =====+
DBname.TSname..... Stats.Date NACTIVE
DBMRDG21.LOZS      0 11/18/2000 360
+ =====+
| TablePart Information |
+ =====+
DBname.TSname      PT Stats Date CARD      FARINDREF      NEARINDREF      %Act %Drp
DBMRDG21.LOZS      0 11/18/2000 10020      0              0                15  0
+ =====+
| Table Information |
+ =====+
Table Name..... Stats Date CARDF..... NPAGES.... RECLENGTH
MRDG2.LODGE_ZIPCD  11/18/2000 10020      66             25
+ =====+
| Column Information |
+ =====+
COLNAME           COLCARDF      HIGH2KEY LOW2KEY
ID_NUMBER         93            .         .
EFFECTIVE_DATE    2             .m..     .r..
EXPIRATION_DATE   1             rr..     rr..
ZIP_CODE_5        10020        .:~      .r..
FK_ACL_CLUB_C     5            601      018
+ =====+
| ColumnDist Information |
+ =====+
COLNAME           FREQUENCYF    TYPE CARDF      NUMCOL COLGROUPCOLNO  COLVALUE
FK_ACL_CLUB_C     143           F -1            1        018
FK_ACL_CLUB_C     463           F -1            1        601
FK_ACL_CLUB_C     1593          F -1            1        999
FK_ACL_CLUB_C     2554          F -1            1        004
FK_ACL_CLUB_C     5244          F -1            1        252
ID_NUMBER         265           F -1            1        b}
ID_NUMBER         339           F -1            1        bB
ID_NUMBER         481           F -1            1        bI
ID_NUMBER         498           F -1            1        bC
ID_NUMBER         500           F -1            1        b
ID_NUMBER         631           F -1            1        b
ID_NUMBER         696           F -1            1        b
ID_NUMBER         885           F -1            1        bA
ID_NUMBER         1169          F -1            1        -
ID_NUMBER         1593          F -1            1        x.
ZIP_CODE_5        0             F -1            1        ...
ZIP_CODE_5        0             F -1            1        ...
ZIP_CODE_5        0             F -1            1        ...
ZIP_CODE_5        0             F -1            1        ...
ZIP_CODE_5        0             F -1            1        ...
ZIP_CODE_5        0             F -1            1        ...
ZIP_CODE_5        0             F -1            1        ...
ZIP_CODE_5        0             F -1            1        ...
ZIP_CODE_5        0             F -1            1        ...
ZIP_CODE_5        0             F -1            1        ...
ZIP_CODE_5        0             F -1            1        ...
+ =====+
| Index Information |
+ =====+
Index Name: MRDG2.LOZX0          Table Name: MRDG2.LODGE_ZIPCD
+ =====+
| IndexColumn Information |
+ =====+
IC FK_ACL_CLUB_C      ASC
IC ZIP_CODE_5        ASC
IC ID_NUMBER         ASC
IC EFFECTIVE_DATE    ASC
Stats.Date CG CD CTO TYPE FIRSTKEYCARDF FULLKEYCARDF NLEAF NLEVEL
11/18/2000 N Y 98 2 5 10020 53 2
Index Name: MRDG2.LOZX1          Table Name: MRDG2.LODGE_ZIPCD
+ =====+
| IndexColumn Information |
+ =====+
IC ID_NUMBER         ASC
Stats.Date CG CD CTO TYPE FIRSTKEYCARDF FULLKEYCARDF NLEAF NLEVEL
11/18/2000 N N 55 2 93 93 14 2
Index Name: MRDG2.LOZX2          Table Name: MRDG2.LODGE_ZIPCD
+ =====+
| IndexColumn Information |
+ =====+
IC ZIP_CODE_5        ASC
Stats.Date CG CD CTO TYPE FIRSTKEYCARDF FULLKEYCARDF NLEAF NLEVEL
11/18/2000 N Y 98 2 10020 10020 34 2
+ =====+
| IndexPart Information |
+ =====+
Index Name      IP Stats Date CARDF      FAROFFPOS      NEAROFFPOS      LEAFDIST
LOZX0           0 11/18/2000 10020      113            93                0
LOZX1           0 11/18/2000 10020      60             72                0
LOZX2           0 11/18/2000 10020      115            94                0
MC674SRC 1371  1 2 4 LODGE_ZIPCD  2 M 0          N N Y N N N Y N IS L 0
MC674SRC 1371  1 2 4 LODGE_ZIPCD  2 M 1 LOZX1    Y N N N N N N N IS 1
.
.
.

```

EXPLORE* A Priced Add-on

Command Syntax:	EXPLORE [TYPE=CHECK REBIND] [O=tsoid plan table owner] [CL=tsoid collection ID] [VER=C A] [CRIT=R,I0,L,MX,MJ,HJ,SO ALL] [SU=service-unit thresh hold] [COND=AND OR]
Line objects allowed:	PG
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	Yes
Output Type:	DB2I2 REBIND or explain type of output
Reusable:	Yes

Command Description

- To use this add-on feature, the user need to have SYSADM or BINDADD authorization as well as table authorization to based line db2 tables. You also need to define both PLAN_TABLE and DSN_STATEMNT tables which are also used by EXPLORE.
- Use EXPLORE command with TYPE=REBIND option to generate REBIND command if any of the selected PG lines which require a REBIND.
- By default, the PLAN_TABLE of your own TSO ID is used by EXPLORE command. You can use O (owner) option to switch to different PLAN_TABLE as work tables for EXPLORE command.
- By default, your TSO ID is used by EXPLORE command to generate temporary collection ID for temporary package. You can use CL option to switch to assign temporary collection ID for EXPLORE command.
- By default, only the current version of the selected PG are being checked. Specify VER=A to check all versions of the selected PG.
- For a TYPE=CHECK option,
 - Use CRIT option to specify criteria to be checked:
 - R – Table Space Scan
 - I0 – Index match column 0
 - L – List Prefetch
 - MX – Multiple Indexes Access
 - MJ – Merge Scan Join
 - HJ – Hybrid Join
 - SO – Sort
 - ALL – all the above
 - Use SU option to specify service unit threshold for the CHECK option. For example, SU=20 means any queries whose service unit is greater than 20
 - User COND option to specify how CRIT and SU should be qualified. COND=AND indicates that both SU and CRIT should qualify before a PG will be selected. COND=OR indicates either SU or CRIT qualify the PG will be selected. The default if not specify is COND=AND.

FETCH

Command Syntax:	FETCH MAP='DCLGEN data set'
Line objects allowed:	TB, AL, SY, VW, MT TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Cobol Code
Reusable:	No

Command Description

- Use DB2I2 FETCH command to generate FETCH INTO embedded SQL statements against selected line objects.
- The MAP command option specifies a file, which contains the output from DCLGEN command. When DB2I2 generates the EXEC SQL FETCH statement, all host variables defined in the file from DCLGEN are mapped onto DB2 column name.

Example

The following example demonstrates how to use FETCH command to map all columns of Q.PROFILES with the DCLGEN output saved in PDS.CNTL(PROFILE).

```
Command ==> fetch map=pds.cntl(profile)                               Scroll ==> CSR
***** ***** Top of Data *****
S00001 TB Q.PROFILES
```

The screen below displays the result from the previous FETCH command.

```
000001 TB Q.PROFILES
000002         EXEC SQL
000003             FETCH cursor-name INTO
000004                 :CREATOR
000005                 , :CASE:NULL2
000006                 , :DECOPT:NULL3
000007                 , :CONFIRM:NULL4
000008                 , :WIDTH:NULL5
000009                 , :LENGTH:NULL6
000010                 , :LANGUAGE:NULL7
000011                 , :SPACE:NULL8
000012                 , :TRACE:NULL9
000013                 , :PRINTER:NULL10
000014                 , :TRANSLATION
000015                 , :PFKEYS:NULL12
000016                 , :SYNONYMS:NULL13
000017                 , :RESOURCE-GROUP:NULL14
000018                 , :MODEL:NULL15
000019                 , :ENVIRONMENT:NULL16
000020         END-EXEC.
```


FLASH

Command Syntax:	FLASH [MACRO(your.ed.macro)] [CONT=Y] [CLONE=Y (V9 or above)]
Line objects allowed:	FC
Process Mode:	Online and Batch
Support	
Multiple line object:	Y
Multiple type of line object:	N
Wild Card % allowed:	N
Output Type:	JCL
Reusable:	Yes

Command Description

- Use FLASH to generate dfsmsDSS JCL to Flash Copy DB2 Dataset.
- The Source and Target Table Space must have the same database Structure which includes the **same Index Names** and **Same OBIDs** (MIGR with OBID).
- The generated JCL contains the following steps:
 - STOP Target Db2 Objects
 - IDCAMS Delete Target VSAM Datasets
 - START Source Db2 Objects with ACCESS(RO)
 - dfsmsDSS Flash Copy Source and Rename to Target objects
 - START Source Db2 Objects with ACCESS(RW)
 - START Target Db2 Objects with ACCESS(UT)
 - DB2 Repair LevelID and Repair LOCATE/VERIFY/REPLACE
 - START Target Db2 Objects with ACCESS(RW)
- The format for FC Flash Copy:
FC FLoc.db.ts TLoc.db.ts [CLONE=Y]
 Where FLoc and TLoc has the following format:
 SSID or SSID\LocationName Use current SSID if no Floc and Tloc specified
- use MACRO(your.ed.macro) option to edit output with your ED macros.
- By default, FLASH Stop processing line objects once it encounters an error use CONT=Y option to allow FLASH to continue generating Flash Copy jobs even if there are errors during the process of the selected FC lines.
 A processing summary report will be generated under DDNAME - ERROR.
- Use CLONE=Y option to select CLONE TS for FLASH process. (V9 or above)

Example

The example below demonstrates how to use DB2I2 FLASH to generate dfsmsDSS to Flash Copy a local Table Space from DSN8D91A.DSN8S91D to DSN8T91A.DSN8S91D.

```
Command ==> FLASH Scroll ==> CSR
***** Top of Data *****
S00001 FC DSN8D91A.DSN8S91D DSN8T91A.DSN8S91D
```

And the output JCL is displayed below:

```
//* $FLASH1----- **
//STEP001 EXEC PGM=IKJEFT1B,DYNAMNBR=20,COND=(4,LT)
// * ----- **
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB9G)
-DISPLAY DB(DSN8T91A) SPACE(DSN8S91D,XDEPT1,XDEPT2,XDEPT3 -
)
-STOP DB(DSN8T91A) SPACE(DSN8S91D,XDEPT1,XDEPT2,XDEPT3 -
)
-DISPLAY DB(DSN8T91A) SPACE(DSN8S91D,XDEPT1,XDEPT2,XDEPT3 -
)
//* $FLASH2----- **
/* IDCAMS Delete Target VSAM Datasets Before dfsmsDSS Flash Copy
/* ----- **
//STEP002 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE DB2ADM9.DSNDBC.DSN8T91A.DSN8S91D.I0001.A001 CLUSTER
DELETE DB2ADM9.DSNDBC.DSN8T91A.XDEPT1.I0001.A001 CLUSTER
DELETE DB2ADM9.DSNDBC.DSN8T91A.XDEPT2.I0001.A001 CLUSTER
DELETE DB2ADM9.DSNDBC.DSN8T91A.XDEPT3.I0001.A001 CLUSTER
//* $FLASH1----- **
//STEP003 EXEC PGM=IKJEFT1B,DYNAMNBR=20,COND=(4,LT)
// * ----- **
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB9G)
-DISPLAY DB(DSN8D91A) SPACE(DSN8S91D,XDEPT1,XDEPT2,XDEPT3 -
)
-START DB(DSN8D91A) SPACE(DSN8S91D,XDEPT1,XDEPT2,XDEPT3 -
) ACCESS(RO)
-DISPLAY DB(DSN8D91A) SPACE(DSN8S91D,XDEPT1,XDEPT2,XDEPT3 -
)
/* $FLASH3----- **
/* dfsmsDSS Flash Copy from Source to Target
/* ----- **
//STEP004 EXEC PGM=ADRDSSU ,PARM='TYPRUN=NORUN'
/*DD1 DD UNIT=SYSALLDA,VOL=SER=?????,DISP=OLD
/*COPY OUTDD(DD1)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COPY DATASET
( INCLUDE (
DSN910.DSNDBC.DSN8D91A.DSN8S91D.I0001.A001
,DSN910.DSNDBC.DSN8D91A.XDEPT1.I0001.A001
,DSN910.DSNDBC.DSN8D91A.XDEPT2.I0001.A001
,DSN910.DSNDBC.DSN8D91A.XDEPT3.I0001.A001
) )
RENAMEU(
(DSN910.DSNDBC.DSN8D91A.DSN8S91D.I0001.A001,
DB2ADM9.DSNDBC.DSN8T91A.DSN8S91D.I0001.A001),
(DSN910.DSNDBC.DSN8D91A.XDEPT1.I0001.A001,
DB2ADM9.DSNDBC.DSN8T91A.XDEPT1.I0001.A001),
(DSN910.DSNDBC.DSN8D91A.XDEPT2.I0001.A001,
DB2ADM9.DSNDBC.DSN8T91A.XDEPT2.I0001.A001),
(DSN910.DSNDBC.DSN8D91A.XDEPT3.I0001.A001,
DB2ADM9.DSNDBC.DSN8T91A.XDEPT3.I0001.A001),
)
CATALOG
DEBUG(FRMSG(DETAILED))
```

```

/* * $FLASH1----- **
//STEP005 EXEC PGM=IKJEFT1B,DYNAMNBR=20,COND=(4,LT)
/* * ----- **
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB9G)
-START DB(DSN8D91A) SPACE(DSN8S91D,XDEPT1,XDEPT2,XDEPT3 -
) ACCESS(RW)
-DISPLAY DB(DSN8D91A) SPACE(DSN8S91D,XDEPT1,XDEPT2,XDEPT3 -
)
/* * $FLASH1----- **
//STEP006 EXEC PGM=IKJEFT1B,DYNAMNBR=20,COND=(4,LT)
/* * ----- **
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB9G)
-START DB(DSN8T91A) SPACE(DSN8S91D,XDEPT1,XDEPT2,XDEPT3 -
) ACCESS(UT)
-DISPLAY DB(DSN8T91A) SPACE(DSN8S91D,XDEPT1,XDEPT2,XDEPT3 -
)
/* * $FLASH4----- **
/* * REPAIR LEVEL ID & LOCATE, VERIFY AND REPLACE **
//STEP007 EXEC DSNUPROC,COND=(4,LT),
/* * LIB='DSN910.SDSNLOAD',
// * SYSTEM='DB9G',UID='DB2ADM01.STEP007',UTPROC=' '
/* * ----- **
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
REPAIR LEVELID TABLESPACE DSN8T91A.DSN8S91D
REPAIR OBJECT LOG YES
LOCATE TABLESPACE DSN8T91A.DSN8S91D PAGE 0
VERIFY OFFSET X'0C'
DATA X'01090002'
REPLACE OFFSET X'0C'
DATA X'011D0002'
REPAIR LEVELID INDEXSPACE DSN8T91A.XDEPT1
REPAIR OBJECT LOG YES
LOCATE INDEXSPACE DSN8T91A.XDEPT1 PAGE 0
VERIFY OFFSET X'0C'
DATA X'0109000D'
REPLACE OFFSET X'0C'
DATA X'011D0004'
REPAIR LEVELID INDEXSPACE DSN8T91A.XDEPT2
REPAIR OBJECT LOG YES
LOCATE INDEXSPACE DSN8T91A.XDEPT2 PAGE 0
VERIFY OFFSET X'0C'
DATA X'0109000F'
REPLACE OFFSET X'0C'
DATA X'011D0006'
REPAIR LEVELID INDEXSPACE DSN8T91A.XDEPT3
REPAIR OBJECT LOG YES
LOCATE INDEXSPACE DSN8T91A.XDEPT3 PAGE 0
VERIFY OFFSET X'0C'
DATA X'01090011'
REPLACE OFFSET X'0C'
DATA X'011D0008'
/* * $FLASH1----- **
//STEP008 EXEC PGM=IKJEFT1B,DYNAMNBR=20,COND=(4,LT)
/* * ----- **
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB9G)
-START DB(DSN8T91A) SPACE(DSN8S91D,XDEPT1,XDEPT2,XDEPT3 -
) ACCESS(RW)
-DISPLAY DB(DSN8T91A) SPACE(DSN8S91D,XDEPT1,XDEPT2,XDEPT3 -
)

```

FLIST

Command Syntax:	FLIST 'print.file.name' [lrecl 132]
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 FLIST command to print the content of a sequential file or a member of a partition data set.
- You can use optional lrecl to specify the logical record length of the input file. If you do not specify the lrecl option, the default length of 132 characters long will be used.

Example

The example below demonstrates how to use DB2I2 FLIST to print an 80 characters long file T1.

```
Command ==> FLIST T1 80                               Scroll ==> CSR
***** Top of Data *****
000001 TB Q.PROFILES
```

FREE

Command Syntax:	FREE [KEEP=0#] [CURRENT]
Line objects allowed:	PL, PG
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	DB2 FREE Command
Reusable:	Yes

Command Description

- Use DB2I2 FREE command to generate DB2 FREE command statements against selected plans or packages.
- If you have multiple versions of the same DB2 package, you can use KEEP command option to decide how many versions of the same DB2 package you want to keep. By default # is set to 0, which generates FREE DB2 command all versions of selected packages. For example, you can specify KEEP=2 command option to keep the most current 2 generations of all the selected packages.
- Use CURRENT option to generate FREE command for current version of the selected PG line. This option is mutual exclusive with KEEP option. When you specify CURRENT option, you can not also specify KEEP option for the same FREE DB2I2 command.
- The generated DB2 FREE commands can then be executed with DB2I2 EXEC command.

Example

The example below demonstrates how to use FREE command to generate DB2 FREE command statements for package JD00.N0AR006.

```

Command ==> free                               Scroll ==> CSR
***** Top of Data *****
000001 pg .JD00.%
==MSG> Collid.Name..... Valid. Operative
s00002 PG .JD00.N0AR006          Y      Y
000003 PG .JD00.N3BR010        Y      Y

```

The screen below displays the result from the previous FREE command. The FREE PACKAGE DB2 command statements were generated together with the last bind timestamp. The version information, if available, is also displayed so that you can select which version you want to execute the DB2 FREE command.

```

Command ==>                               Scroll ==> CSR
***** Top of Data *****
000001 pg .JD00.%
==MSG> >>>> Begin of FREE DDL
==MSG> Collid.Name..... Valid. Operative
000002 PG .JD00.N0AR006          Y      Y
000003 -- FREE Package Name: JD00.N0AR006
000004 -- Timestamp: 1997-10-28-17.01.36.053576
000005 DB2CMD FREE PACKAGE(JD00.N0AR006)

```

FTP** a price add-on with RXdb2i2

Command Syntax:	FTP [INPUTDSN=input-dsname] [RESET=Y]
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	Report
Reusable:	No

Command Description

- Use FTP to invoke and process FTP command thru RXDB2I2. It can be Processed as RXDB2I2 command option: RXDB2I2 FTP ... Or Processed inside RXDB2I2
RXDB2I2 idd=dd1
Lineobj ddname=dd1
DB2I2REX "FTP ..."
End_lineobj
- In online mode, you can select lines in your work bench and then issue RXDB2I2 FTP command to invoke FTP to PUT the selected lines to a remote server.
- The selected lines are stored under //DD:TEMP.
- Use INPUTDSN option to assign the dsn for FTP control statements. If not specified, use //INPUT DD file as INPUT to FTP or if no //INPUT DD, use yourID.DB2I2.FTP.INPUT as INPUT dsn to FTP.
- You can edit the FTP control statement to fit your need.
- Use RESET=Y to reset your entered password to 'password' to protect your password after FPT is done.

Example

The example below demonstrates how to use FPT to get information from remote server, if not successful, issue MAIL and send message to a user. (The following script and the script to invoke the following are in db2i2.ispfliib. You can retrieve these scripts by FGET *FTPMAIL or FGET *FTPMAILS or ? *FTPMAIL or ? *FTPMAILS)

In db2i2.ispfliib(FTPMAILS):

```

-----*
-- Issue Rxdb2i2 Command to Invoke DB2I2 Batch to Build JCL          -*
-- to Execute FTP and check for Status of FTP Execution.            -*
-- &H-WAITH and &H-CNT-H are Used to control number of tries and    -*
-- wait time between each Try.                                       -*
-- After reach limit set by &H-CNT-H and &H-WAIT-H,                  -*
-- Send Email message if still not successful                          -*
-- Specify Host variables:                                           -*
--   &H-CNT-H      - Number of FTP Tries Before Send An Email        -*
--   &H-FTPDSN-H   - FTP Control Statement Input                     -*
--   &H-WAIT-H     - Wait period between each FTP Try                 -*
--   &H-CLASS-H   - SMTP output Class                                -*
--   &H-WRITER-H   - SMTP Output Writer                              -*
--   &H-PRINT-H   - Print e-mail                                     -*
--   &H-HOST-H    - Host Name                                        -*
--   &H-FROM-H    - From Address                                    -*
--   &H-TO-H     - To Addresses                                     -*

```

```

--      &H-CC-H      - CC Addresses                                -*
--                  Specify &H-CC-H=Cc= for No CC addresses      -*
-----*
DB2I2REX "Batch ICMD=*ftpmail",
          "jobdd=N lineobj=N",
          "&H-CNT-H=5",
          "&H-FTPDSN-H='jrh.demo(ftpinput)'",
          "&H-WAIT-H=00:00:05",
-- Mail Parmns
          "&H-CLASS-H=P",
          "&H-WRITER-H=SMTP",
          "&H-PRINT-H=Y",
          "&H-HOST-H=etjrh.dal-ebid.ihost.com",
          "&H-FROM-H=jdoong@jrh-inc.com",
          "&H-TO-H=(dsn='jrh.demo(toaddr)')",
          "&H-CC-H=Cc="

```

In db2i2.ispfliB(FTPMAIL):

```

RXDB2I2
//LINEOBJ DD *
-----*
-- This Routine demonstrates the Usage of RXDB2I2 FTP,WAIT,MAIL,TRACK commands
-- . Process FTP GET with FTP command
-- . Check Status of FTP GET
-- . If There is an Error
-- .   Wait for 5 Minutes with WAIT And Go Back to Try FTP GET Again
-- .   If More Than 5 Tries and Not Successful
-- .     Send a Message/Email with MAIL Command
-- .     Set Error and Return Code to 16
-- DDNAME Defined:
-- BODYMSG : MAIL body text
-- Note:
-- 'JRH.DEMO(FTPINPUT)' contains the FTP GET control Statements below
-- xxx.xx.xx.xx 21
-- userid password
-- ascii
-- get 'my.ftp.file'
-- quit
-----*
LINEOBJ DDNAME=BODYMSG

Hi Jerry

    This script demonstrates how to use RXDB2I2 with FTP and MAIL

Jerry Doong
12345

The Following is the FPT Execution Result

END_LINEOBJ
-----*
-- Allocate Output DD  below for FTP Error Traping & Reporting
-----*
Address Tso "Alloc Fi(Output) New Reuse",
            "Space(1,1) Tracks Unit(sysallda)",
            "Recfm(F B) Lrecl(133) Blksize(13300)"
loop_cnt = 1
LOOP:
  DB2I2REX "FTP INPUTDSN=&H-FTPDSN-H"
  Say "*** FTP Execution Results Follows: ***"
  "Execio * Diskr Output(Stem rl. Finis"
  Do I = 1 to rl.0
    Say Left(rl.i,120)
  End
  If RXerror() = 'Y' THEN
  Do
    If loop_cnt < &H-CNT-H Then
    Do
      loop_cnt = loop_cnt + 1
      x = WAIT('PERIOD &H-WAIT-H')
      Signal LOOP
    End
  Else
  Do
    Say "*** FTP Processing Fail After "loop_cnt" Trials"

```

DB2I2 Reference Manual

```
-- Set Upper Case Off to Send Subject in Upper/Lower Case
Db2i2rex "UCASE(OFF)"
Db2i2rex "Mail Class=&H-CLASS-H",
         "Writer=&H-WRITER-H",
         "Print=&H-PRINT-H",
         "Host=&H-HOST-H",
         "From=&H-FROM-H",
         "To=&H-TO-H",
         "&H-CC-H",
         "Subject=FTP Fail with RC="RXrc()"@" ,
-- Body of the MAIL from 2 DD, BODYMSG and OUTPUT from FTP
         "Body=(dd=bodymsg,output)"
Db2i2rex "UCASE(ON)"
-- Set Error and RC with RXError and RXrc function
x = RXError(Y)
x = RXrc(16)
End
End
-- End of Routine
```

Where 'JRH.DEMO(FTPINPUT)' contains the following FTP control statements:

```
EDIT          JRH.DEMO(FTPINPUT) - 01.00          Columns 00001 00072
Command ==>>          Scroll ==>> CSR
***** ***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
000001 111.22.33.44 21
000002 db2adm pppppppp
000003 ascii
000004 get t99
000005 quit
***** ***** Bottom of Data *****
```


Next example select the lines and issue RXDB2I2 FTP to send the select lines to a remote server:

```
Command ==> rxdb2i2 ftp                               Scroll ==> CSR
***** ***** Top of Data *****
s70001 lines
000002 to
000003 be
000004 FTP
000005 to
000006 Remote
000007 Server
, , , , ,
```

If this is the first time use rxdb2i2 ftp, rxdb2i2 will create the following dataset which allow you to modify to fit your environment.

```
EDIT          yourID.DB2I2.FTP.INPUT                  Columns 00001 00072
Command ==>                                           Scroll ==> CSR
***** ***** Top of Data *****
=NOTE= *-----*
=NOTE= * 1st Line: Machine Name (Host Name or IP Address) port *
=NOTE= * 2nd Line: UserID Password *
=NOTE= * Other FTP Subcommands: *
=NOTE= * ascii : ASCII transfer of text file *
=NOTE= * binary : BINARY transfer of binary file *
=NOTE= * get Filename LocalFileName: Get a file from Remote host *
=NOTE= * put Filename RemoteFileName: Put a file onto Remote host *
=NOTE= *                                     //DD:TEMP contains the selected lines *
=NOTE= * quit: exit FTP *
=NOTE= *-----*
000001 hostname
000002 yourID password
000003 put //DD:TEMP remote.file.name
000004 quit
***** ***** Bottom of Data *****
```

Modify the FTP control statement and press End key - F3 to process FTP. Issue Cancel to cancel FTP command.

```
220-FTPSERVE IBM FTP CS V1R9 at ETJRH.DAL-EBIS.IHOST.COM, 00:15:21 on 2010-04-13.
220 Connection will close if idle for more than 5 minutes.
331 Send password please.
230 DB2ADM is logged on. Working directory is "DB2ADM.".
200 Representation type is Ascii NonPrint
200 SITE command was accepted
200 Port request OK.
125 Storing data set DB2ADM.FROMJRH
250 Transfer completed successfully.
221 Quit command received. Goodbye.
```

FU

Command Syntax:	FU (for V6 or above only)
Line objects allowed:	TB, VW, PL, PG, MT, SQ
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	FU line object
Reusable:	Yes

Command Description

- Use DB2I2 FU command to generate FU – function line object for the selected line object.

Example

The example below demonstrates how to generate FU – function lines.

```
Command ==>> FU
S011100 PG .DSN8%.DSN8%.

Command ==>>
0011100 PG .DSN8%.DSN8%.
0011101 SP SYSPROC.DSNUTILS DSN8EP10.DSN8EPU
```

GENVCAT

Command Syntax:	GENVCAT [VOL=*][%=### [CYL TRK _] [ALLOC=(alloc_type,primary,secondary)] [VCAT=new-vcatname] [DB=new-dbname]
Line objects allowed:	TS, TP, IX, IP, DS, IS, ISP TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	IDCAMS DEFINE Command
Reusable:	Yes

Command Description

- Use DB2I2 GENVCAT command to generate IDCAMS DEFINE statements for the selected line object.
- By default, DB2I2 generates IDCAMS DEFINE scripts using the existing ICF catalog information.
- To override the VOLUME information, you specify VOL=* option, which assigns VOLUME(*) in the generated IDCAMS DEFINE scripts instead of using the information from ICF catalog.
- To adjust space allocation as a percentage of the current allocation, you use %=### command option. The primary and secondary allocations will be adjusted as a percentage ###% of the values from ISF catalog. You also have options to adjust the allocation to cylinder or track boundary by specifying CYL and TRK options.
- You can also use ALLOC=(alloc_type,primary,secondary) option to override the space allocation. Specified alloc_type must be CYL, TRK, or PAGE. Default alloc_type from ICF catalog will be used if omits the alloc_type. The primary must be HIARBA, HIURBA or a numeric number. Default primary allocation information from ICF catalog will be used if omits this field. Secondary options if specified must be numeric. The following are a few examples: .
 ALLOC=(CYL,100,5)
 ALLOC=(,HIARBA,)
 ALLOC=(CYL,HIURBA,)
- Both %= and ALLOC= options can be specified at the end of line object, which can be used to override what is defined from the command line.
- Use VCAT option to assign new VCAT name.
- Use DB option to assign new DB name.

Example

The example below demonstrates how to generate IDCAMS DEFINE for index SYSIBM.DSNTX02 with all the allocation information from ICF catalog.

```
Command ==> GENVCAT                               Scroll ==> CSR
S00171  IX  SYSIBM.DSNTX02                          SYSIBM.SYSTABLES
```

The screen below displays the result from the previous GENVCAT command.

```
Command ==>                               Scroll ==> CSR
000171  IX  SYSIBM.DSNDTX02                SYSIBM.SYSTABLES
000172  IDCAMS DEFINE                      -
000173  IDCAMS CLUSTER                     -
000174  IDCAMS (NAME(TDB2.DSNDBD.DSNDB06.DSNDTX02.I0001.A001)) -
000175  IDCAMS LINEAR                     -
000176  IDCAMS REUSE                       -
000177  IDCAMS SHAREOPTIONS(3 3)          -
000178  IDCAMS VOLUMES(VL3024)           -
000179  IDCAMS TRACKS(30,3)              -
000180  IDCAMS DATA                       -
000181  IDCAMS (NAME(TDB2.DSNDBD.DSNDB06.DSNDTX02.I0001.A001))
```

The example below demonstrates various ways to generate IDCAMS statements with command option %=100 override as well as line object option override:

Command:	GENVCAT %=100	100% command option space allocation override
Line object:	TP DBJD.TS1 0	100% space allocation from command option
	TP DBJD.TS2 0 %=200	200% space allocation from line object option
	TP DBJD.TS3 0 ALLOC=(CYL,1,0)	override with 1 cylinder primary and no secondary

GRANT

Command Syntax:	GRANT [SQLID=id] [TO=grantee ID] [grant options]
Line objects allowed:	AL, BP, CL, DB, PL, PG, SG, TS, TB, VW, US, CO (DT, FU, SP, SH, TR for V6 or above) (MT, SQ for V8 or above) (RO, TBC for V9 or above)
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	DCL GRANT statements
Reusable:	Yes

Command Description

- Use DB2I2 GRANT command to generate DB2 GRANT DCL statements against selected line object.
- Specify SQLID command option to override SET SQLID information for the generated DCL.
- Use TO command option to specify the grantee's user ID.
- Use grant option to limit the authorization to be grant. For example, use SELECT option as grant option to grant only the select authorization of a table to a user. If no grant options specified, all available options will be displayed.

Example

The example below demonstrates how to use DB2I2 GRANT command to generate DB2 GRANT DCL to grant usage of BUFFERPOOL BP0 to a user U1.

```
Command ==> Grant SQLID=jd to=u1                               Scroll ==> CSR
s00001 BP bp0
```

The screen below displays the result from the previous GRANT command.

```
Command ==>                                                    Scroll ==> CSR
000171 BP bp0
000172 SET CURRENT SQLID = 'JD';
000173 GRANT
000174     USE OF BUFFERPOOL
000175     BP0
000176     TO U1
000177 ;
```

DB2I2 Reference Manual

The example below demonstrates how to use DB2I2 GRANT command with grant options SELECT, INSERT. It also demonstrates usage of SQLID global command option.

```
Command ==> grant to=a SQLID=sysadm select,insert          Scroll ==> CSR
S00001  TB JDOO.TBL1
```

The screen below displays the result from the previous GRANT command.

```
Command ==>                                               Scroll ==> CSR
000001  TB JDOO.TBL1
000002  SET CURRENT SQLID = 'SYSADM';
000003  GRANT
000004  SELECT,INSERT
000005  ON TABLE
000006  JDOO.TBL1
000007  TO A
000008  ;
```

HELP or ?

Command Syntax:	HELP [db2i2 command *UDF *db2i2 TSO command &global Variable]
Line objects allowed:	N/A
Process Mode:	Online only
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	N/A

Command Description

- Use DB2I2 HELP command in online mode to display Help information for all DB2I2 commands. The result from DB2I2 HELP command contains
 - How to use these commands
 - Line objects associated with these commands
 - Detail description for these commands
- There are two ways to invoke help for a DB2I2 command:
 - Specify desired DB2I2 command as the command option together with DB2I2 HELP command.
 - Specify Db2I2 HELP command without command option. It lists all available DB2I2 commands, so that you can choose desired DB2I2 command for detail HELP.
- Replacing [db2i2 command] with *UDF or *db2i2 TSO command, HELP can display the detail usage information for specified system defined UDF or system delivered DB2I2 TSO command. For example, to see how to use GENAT UDF, you type HELP *GENAT.
- Use HELP &global variable to display global variable information. Help & display all currently defined global variables. ? &abc display all global variables prefixed with &abc.
- You can use ? or HELP interchangeable

Example

The example below demonstrates how to use DB2I2 HELP command without command option to list all DB2I2 commands.

Position cursor to new DB2I2 tool bar or enter ? or Help on command and press Hotkey to bring up DB2I2 Help Menu.

```
File Edit Edit_Settings Menu Utilities Compilers Test Help Db2i2
SSID(DB9G) Connect() Ucase(ON) Sysibm(SYSIBM) ----- DB2I2 10.0
EDIT          DB2ADM.DB2I2.LO                      Columns 00001 00072
Command ==> help or ?                               Scroll ==> CSR
***** Top of Data *****
```

```
File Edit Edit_Settings Menu Utilities Compilers Test Help Db2i2
SSID(DB9G) Connect() Ucase(ON) Sysibm(SYSIBM) ----- DB2I2 10.0
EDIT          DB2ADM.DB2I2.LO                      Columns 00001 00072
```

```
#HELPM0 ----- JRH DB2I2 For DB2 OS/390 and UDB zOS Help Menu -----
OPTION ==> _____
```

```
-----
|                JRH DB2I2 By JRH GoldenState Software, Inc.                |
|                (c) CopyRighted 1997-2008                                |
|                -----
```

- | | | | |
|-----|---------------------------|------|---------------------------|
| _ 1 | How JRH DB2I2 works | _ 2 | Line Object Definition |
| _ 3 | How to Select Line Object | _ 4 | Global Line Object Option |
| _ 5 | DB2I2 Execution Output | _ 6 | Global Command Option |
| _ 7 | DB2I2 Command - Summary | _ 8 | DB2I2 Command - List |
| _ 9 | *UDF, *UDQ, *Tso Command | _ 10 | Tutorial Mode |

* Select An Option by Either Enter The Option Name in The Option Field
Or Position The Cursor On The Place Holder and Press Enter.

Select an Option and Press ENTER to Preceed PF3=Exit

Option 1 shows how db2i2 works.

```
#HELPM1 ----- JRH DB2I2 For DB2 OS/390 and UDB zOS Help Screen -----
OPTION ==> _____
```

```
-----
|                JRH DB2I2 By JRH GoldenState Software, Inc.                |
|                (c) CopyRighted 1997-2011                                |
|                -----
```

How JRH DB2I2 works

JRH DB2I2 has been Implemented as a Superset of ISPF Edit command.
It works just like Any of The Other Builtin ISPF Commands.
To Direct DB2I2 to Perform A Task, You need To Do The Following
In Sequence:

- Enter Line Object(s) In ISPF Edit Line
- Enter DB2I2 Command On ISPF Command Line
- Select Line Object(s) with S or SS or IDSN Line Command option
- Press Preset HOTKEY To Execute The Command
- Use ISPF KEYS Command To Assign A FKEY To DB2I2 as HOTKEY
- F4 - The unused FKEY is a good choice for DB2I2 HOTKEY

PF3=Exit

Option 2 shows how to define line object and all available Line objects.

```
#HELPM2 ----- DB2I2 for OS/390 and zOS Line Object Help Screen ----- TUTORIAL
OPTION ==> _____
|                                     |
|      JRH DB2I2 By JRH GoldenState Software, Inc.      |
|      (c) CopyRighted 1997-2011                       |
|_____|_____|
More:      +

Line Object Definition
Line Objects(LO) Are Defined As Follows:
  LO-Abbr(eviation) LO-Name
Line Object can be Defined Any Where in ISPF Edit Line.
However, No Data Allowed Before LO-Abbr
LO-Name Can be Full Name or Partial Name (Name With Wild Card Symbol % )
For Example,
  TB Q.PROFILES
  TB Q.PROF%
  TB Q.%.

Line Objects Abbreviation
_ AC Active log line                    (From DSN1LOGP or DSNJU004)
-----
***Type A Line Object Or Position Cursor on A Place Holder ***
***And Press Enter to See Detail Description***
PF3=Exit PF7=PgUp PF8=PgDn
```

Option 3 shows how to select line object(s).

```
#HELPM3 ----- JRH DB2I2 For DB2 OS/390 and UDB zOS Help Screen -----
OPTION ==> _____
|                                     |
|      JRH DB2I2 By JRH GoldenState Software, Inc.      |
|      (c) CopyRighted 1997-2011                       |
|_____|_____|
How To Select Line Object(s)
More:      +

Use IDSN='input.dsn' Command Option
  To Select Line Objects From Input.dsn

Use S A New Addin ISPF Line Command
  To Select Single Line Object

Command ==>
S 0142 TS DDB2ADM.SDB2ADM  BP8K0 0 P N N N 16
000143 TS DDB2ADM.SQSCAN  BP8K0 0 P N N N 16

Use A Pair of SS A New Addin ISPF Block Line Commands
  To Select A Block of Line Objects

Command ==>
PF3=Exit PF7=PgUp PF8=PgDn
```

Option 4 shows all Global Line Object options.

```
#HELPM4 ----- JRH DB2I2 For DB2 OS/390 and UDB zOS Help Screen -----
OPTION ==>> _____

          |-----|
          | JRH DB2I2 By JRH GoldenState Software, Inc. |
          | (c) CopyRighted 1997-2011 |
          |-----|

Global Line Object Option
%=###
%=### is Used To OVERRIDE Space Allocation
Specify %=### Anywhere After Line Object Detail To OVERRIDE
Space Allocation Adjustment.
### Must Be A Positive Numeric Number.
The Primary Usage of This Option is with DB2I2 MIGR, DDL, DSADJ or
Any DB2 Utilities Related Commands, Such As REORG
To OVERRIDE The Command option %=###. Default is %=100 (No Adjustment)
<NEWJOB>
Use <NEWJOB> Anywhere After Line Object Detail To direct DB2I2
To Begin A New Job. The Primary Usage of this option is
with DB2 Utilities Related command such as REORG, COPY..
To Balance The Work Load Of Each Generated Utility Jobs.
```

PF3=Exit

Option 5 shows where the output of each db2i2 command go.

```
#HELPM5 ----- JRH DB2I2 For DB2 OS/390 and UDB zOS Help Screen -----
OPTION ==>> _____

          |-----|
          | JRH DB2I2 By JRH GoldenState Software, Inc. |
          | (c) CopyRighted 1997-2011 |
          |-----|

The DB2I2 execution output
If ODSN='output.dsname' Command Option Present, The Result Of The
Execution will Be Stored in 'output.dsname'. Otherwise,
For Online Interactive Mode:
    tsoid.DB2I2.LISTC.OUTPUT      for LISTC command
    tsoid.DB2I2.DSNTIAD.OUTPUT   for DSNTIAD command
    tsoid.DB2I2.DSNTPEP2.OUTPUT  for DSNTPEP2 command
    tsoid.DB2I2.EXEC.OUTPUT      for EXEC command
    tsoid.DB2I2.DB2CMD.OUTPUT    for DB2CMD, START, STOP and DISPLAY
    tsoid.DB2I2.EXPLAIN.OUTPUT  for EXPLAIN and EXPLAINP command
    tsoid.DB2I2.DSCOPY.OUTPUT    for DSCOPY command
    tsoid.DB2I2.RUN.OUTPUT       for RUN command
    tsoid.DB2I2.OUTPUT           for All The Other Commands
For Batch Mode, A Time Suffix is Added To The Back of The Above Dataset
Use APPEND option to Append the output to the existing dataset.
```

PF3=Exit

Option 6 shows all the Global Command Options.

```
#HELPM6 ----- JRH DB2I2 For DB2 OS/390 and UDB zOS Help Screen -----
OPTION  ===> _____

-----
|                JRH DB2I2 By JRH GoldenState Software, Inc.                |
|                (c) CopyRighted 1997-2011                                |
-----

Global Command Options

%#### CYL|TRK|                               More:      +
%#### Adjustment From The Current Allocation.
### if present, Must be A Positive Number.
The Default ### is 100. Use this option with MIGR
DDL, DSADJ and DB2 utility. Use sub-option CYL
or TRK to Round Up To Cylinder or Track Boundary

APPEND                                         Append the output To Existing Sequential Dataset.
PDS is NOT Allowed.

CLI=MC|UTIL|clidsn(mem)                      Specify CLI with BACTH command to allow DB2I2 to
process DB2I2 Callable Interface. Use MC option to
process Multiple DB2I2 commands against line object
ONE at a time. Use UTIL option to generate DB2
utility JCL for each line object one at a time and
PF3=Exit PF7=PgUp PF8=PgDn
```

Option 7 shows all Db2I2 commands in Summary format.

```
#HELPM7 ----- JRH DB2I2 For DB2 OS/390 and UDB zOS Help Screen -----
OPTION  ===> _____

-----
|                JRH DB2I2 By JRH GoldenState Software, Inc.                |
|                (c) CopyRighted 1997-2011                                |
-----

DB2I2 Commands - Summary Format

AL      ALTER  AUTH    BATCH   BIND    BINDCOPY  More:      +
CANCEL  CHECK  COAUTH  CONNECT COPY    COPYAUTH
CPY2CPY CREATE  CURSOR  DB      DBAUTH  DBDSIZE
DB2CMD  DCLGEN  DDL     DELETE  DISPLAY DRILLDWN
DSADJ   DSCOPY  DSNJU004 DSNTIAD DSNTEP2 DSNTIAUL
DSN1COPY DSN1LOGP DSN1PRNT DT      ED      EDIT
EXEC    EXPLAIN EXPLAINP EXPLORE  FETCH   FLIST
FREE    FU      GENVCAT GRANT   HELP*   HELPLO
HMIGRATE HRECALL IMPACT  INFO    INSERT  IP
IPC     IS      ISC     ISP     ISPC    IX
IXC    JOBCARD LISTC   LISTDEF LOAD    MIGR

-----
Type A DB2I2 Command or a &GlobalVariable and Press Enter to See Detail.
A &GlobalVariable can be wild card with % For Example, &A%
PF3=Exit PF7=PgUp PF8=PgDn
```

DB2I2 Reference Manual

Position cursor on DB2CMD and Press Enter:

```
#HELPM7 ----- JRH DB2I2 For DB2 OS/390 and UDB zOS Help Screen -----
OPTION ===> _____

-----
EeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeN
e @DB2CMD ----- DB2I2 HELP SCREEN - DB2CMD ----- e
e
e SYNTAX: DB2CMD LIST e
e Line Object(s) Allowed: A block of DB2 command lines with "-" e +
e on the first byte of first line e
e Access Mode : Batch and Online e
e Multiple Line Objects Allowed: No - 1 DB2 command only e
e Different Line Objects Allowed: No e
e Wild Card Line Objects Allowed: No e
e
e Use DB2CMD to execute a DB2 command. e
e Use LIST option in online mode to list all DB2 commands, e
e so that you can choose and modify them. e
e NO Line Objects are needed if LIST option is selected. e
e
----- e PF3=Exit e -----
Type DssssssssssssssssssssssssssssssssssssssssssssssssssssssssssM
A &GlobalVariable can be wild card with % For Example, &A%
PF3=Exit PF7=PgUp PF8=PgDn
```

Option 8 shows Db2I2 commands in List Format:

```
#HELPM8 ----- JRH DB2I2 For DB2 OS/390 and UDB zOS Help Screen -----
OPTION ===> _____

-----
| JRH DB2I2 By JRH GoldenState Software, Inc. |
| (c) CopyRighted 1997-2011 |
-----

DB2I2 Commands - List Format

More: +

_ DRILLDWN Drill Down; Edit or Browse Dataset
_ AL Generate ALias Information
_ ALTER Generate ALTER DDL or
Generate ALTER UTILITY DB2 command from output of DISPLAY UTIL
with the format of UTILID=utility-id
_ AUTH Display AUTHorizaion Information
_ BATCH Generate BATCH JCL to run DB2I2 command
_ BIND Generate DB2 BIND commands against selected DB2 Plan or Package
_ BIND COPY Generate DB2 BIND COPY commands against selected DB2 Package
-----
Type A DB2I2 Command or a &GlobalVariable and Press Enter to See Detail.
A &GlobalVariable can be wild card with % For Example, &A%
PF3=Exit PF7=PgUp PF8=PgDn
```

Option 9 shows how to display UDF, UDQ and TSO command.

```
#HELPM9 ----- JRH DB2I2 For DB2 OS/390 and UDB zOS Help Screen -----
OPTION ==> *gen%_____

-----
                JRH DB2I2 By JRH GoldenState Software, Inc.
                (c) CopyRighted 1997-2011
-----

User Defined Function, Queries and DB2I2 TSO command

DB2I2 Allows User to define Your own Command, Queries as well as TSO commands
All These Information are stored in JRH.DB2I29.ISPFLIB.N3
So that they can be shared and accessed by all DB2I2 users.

To See these Pre-defined UDF, UDQ and TSO commands,
Simply Type the name prefixed with *
For Example,
*GENAT   Display Predefined Query which Generates AT line objects
*G%     Display all members name prefixed with G

To Access *GENAT, Issue RUN ISDN=*GENAT

-----
*** Type A *member And Press Enter to See Detail Description ***
PF3=Exit
```

```
Menu Functions Utilities Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
BROWSE   JRH.DB2I29.ISPFLIB.N3                      Row 00001 of 00011
Command ==>                                          Scroll ==> CSR
Name      Prompt              Size   Created          Changed          ID
. GENAI   .                          13    2008/01/01      2008/06/17 00:00:00 JRHJ
. GENAT   .                          21    2008/01/01      2008/06/17 00:00:00 JRHJ
. GENSC   .                          35    2008/01/01      2008/06/17 00:00:00 JRHJ
. GENSCCMD .                        70    2008/01/01      2008/06/17 00:00:00 JRHJ
. GENSC2  .                        125   2008/01/01      2008/06/17 00:00:00 JRHJ
. GENUNLDQ .                       15    2008/01/01      2008/06/17 00:00:00 JRHJ
. GENUNLD1 .                      27    2008/01/01      2008/06/17 00:00:00 JRHJ
. GENURLD .                       65    2008/01/01      2008/06/17 00:00:00 JRHJ
. GENURLDE .                      9     2008/01/01      2008/06/17 00:00:00 JRHJ
. GENURLD1 .                     304   2008/01/01      2008/06/17 00:00:00 JRHJ
. GENXC   .                       32    2008/01/01      2008/06/17 00:00:00 JRHJ
**End**
```

Option 10 allows you enter Tutorial Mode of DB2I2.

```
File Edit Edit_Settings Menu Utilities Compilers Test Help Db2i2
SSID(DB9G) Connect() Ucase(ON) Sysibm(SYSIBM) ----- DB2I2 10.0
EDIT DB2ADM.DB2I2.LO                               Columns 00001 00072
Command ==>                                          Scroll ==> CSR
***** ***** Top of Data *****
=NOTE= /* -----*/
=NOTE= **                ALL Line Objects Are Listed Below                **
=NOTE= **                Select Desired Line Object(s) with S or SS                **
=NOTE= **                And Issue ? to Select Available DB2I2 Commands                **
=NOTE= **                Or Issue ? CMD to See All DB2I2 Commands                **
=NOTE= /* -----*/
000001 -- -----
000002 -- Active log line --
000003 -- -----
000004 -- Get this information from DSNJU004 command
000005 -- start-rba      end-rba      date      time      log dataset
000006 AC 014724D84000 0147315E7FFF 2000.258 19:32:10 DB2D.ACTLOG
000007
000008 -- -----
000009 -- Adjust Index Partition line --
000010 -- -----
000011 -- ixcreator.ixname partno row-count keycard
000012 AI USERJD.XN3B001 0      200000 5000
000013
```

DB2I2 Reference Manual

Select a line and enter ? or Help command to display a list of all possible Db2I2 command available for the selected line object(s):

```

File Edit Edit_Settings Menu Utilities Compilers Test Help Db2i2
SSID(DB9G) Connect() Ucase(ON) Sysibm(SYSIBM) ----- DB2I2 10.0
EDIT      DB2ADM.DB2I2.WKBENCH9                      Columns 00001 00072
Command ==> ?                                       Scroll ==> CSR
000116   TS      DDB2ADM.SBASE      DB2ADM.BASE_TABLE
000117   TS      DDB2ADM.SQSCAN     DB2ADM.QSCAN_TABLE
000118   TB      DB2ADM.BASE_TABLE          T      3      DDB2ADM SBASE

```

Enter Command name or position cursor on the place holder and press Enter to select the DB2I2 command for Processing:

```

File Edit Edit_Settings Menu Utilities Compilers Test Help Db2i2
SSID(DB9G) Connect() Ucase(ON) Sysibm(SYSIBM) ----- DB2I2 10.0
EDIT      DB2ADM.DB2I2.WKBENCH9                      Columns 00001 00072
EsxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxN
e @HELPTS ----- DB2I2 Line Object HELP SCREEN - TS ----- e
e Command: _____ (Enter a DB2I2 command Listed below) e
e e
e Line Object: TS TableSpace e
e Format      TS dbname.tsname e
e Example     TS DBJD.TSJD e
e DB2I2 Commands Allowed: e
e  _ ALTER    _ AUTH      _ CHECK      COPY      _ CPY2CPY  _ CREATE e
e  _ DB       _ DDL       _ DISPLAY  DRILLDWN  _ DSN1COPY  _ DSN1LOGP e
e  _ DSN1PRNT _ GENVCAT  _ GRANT    _ IMPACT   _ IP        _ IPC e
e  _ IS       _ ISC       _ ISP      _ ISPC     _ IX        _ IXC e
e  _ LISTC    _ MIGR      _ MODIFY   _ MT       _ PG        _ PL e
e  _ QBUILD   _ QUIESCE   _ RBA      _ RECOVER  _ REORG     _ REORGCHK e
e  _ REPAIR   _ REPORT    _ REVOKE   _ RSAUTH   _ RUNSTATS  _ SNAPSHOT e
e  _ START*   _ STATS     _ STOP     _ TB       _ TBC       _ TP e
e  _ TPC      _ TSSET     _ UNLOAD e
e PF3=Exit ENTER=Process Command e
DsxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxM
000134 TB      DB2ADM.DSN_FUNCTION_TABLE          DDB2ADM.SDB2ADM
000135 TB      DB2ADM.DSN_STATEMENT_CACHE_TABLE  DDB2ADM.SDB2ADM

** ----- **
** Processing Selected DB2I2 Commnad As Follows: **
** DRILLDWN **
** ----- **
***

```

```

File Edit Edit_Settings Menu Utilities Compilers Test Help Db2i2
SSID(DB9G) Connect() Ucase(ON) Sysibm(SYSIBM) ----- DB2I2 10.0
EDIT      DB2ADM.DB2I2.WKBENCH9                      SSID: DB9G
Command ==>                                       Scroll ==> CSR
000116   TS      DDB2ADM.SBASE      DB2ADM.BASE_TABLE
=====
TableSpace Name Part.Pqty.....Sqty.....T.StorName VcatName FRG
000117   TP      DDB2ADM.SBASE      1      -1      -1      I GDB2ADM DB2ADM9 0
=====
TBcreator.TBname Type.OBID DBname TSname
000118   TB      DB2ADM.BASE_TABLE          T      3      DDB2ADM SBASE

```

```

File Edit Edit_Settings Menu Utilities Compilers Test Help Db2i2
SSID(DB9G) Connect() Ucase(ON) Sysibm(SYSIBM) ----- DB2I2 10.0
EDIT DB2ADM.DB2I2.WKBENCH9 Columns 00001 00072
E#####N
e @HELPTS ----- DB2I2 Line Object HELP SCREEN - TS ----- e
e Command: _____ (Enter a DB2I2 command Listed below) e
e e e
e Line Object: TS TableSpace e
e Format TS dbname.tsname e
e Example TS DBJD.TSJD e
e DB2I2 Commands Allowed: e
e _ ALTER _ AUTH _ CHECK _ COPY _ CPY2CPY _ CREATE e
e _ DB _ DDL _ DISPLAY _ DRILLDWN _ DSN1COPY _ DSN1LOGP e
e _ DSN1PRNT █ GENVCAT _ GRANT _ IMPACT _ IP _ IPC e
e _ IS _ ISC _ ISP _ ISPC _ IX _ IXC e
e _ LISTC _ MIGR _ MODIFY _ MT _ PG _ PL e
e _ QBUILD _ QUIESCE _ RBA _ RECOVER _ REORG _ REORGCHK e
e _ REPAIR _ REPORT _ REVOKE _ RSAUTH _ RUNSTATS _ SNAPSHOT e
e _ START* _ STATS _ STOP _ TB _ TBC _ TP e
e _ TPC _ TSSET _ UNLOAD e
e PF3=Exit ENTER=Process Command e
D#####M
000134 TB DB2ADM.DSN_FUNCTION_TABLE DDB2ADM.SDB2ADM
000135 TB DB2ADM.DSN_STATEMENT_CACHE_TABLE DDB2ADM.SDB2ADM

```

```

File Edit Edit_Settings Menu Utilities Compilers Test Help Db2i2
SSID(DB9G) Connect() Ucase(ON) Sysibm(SYSIBM) ----- DB2I2 10.0
EDIT SYS08217.T174249.RA000.DB2ADM.R0100140 Columns 00001 00072
Command ==> GO Scroll ==> CSR
***** ***** Top of Data *****
=NOTE= /* -----*/
=NOTE= ** Modify The Listed DB2I2 Command Skelten **
=NOTE= ** And Issue DB2I2 GO to Process It **
=NOTE= /* -----*/
000001 -- Use GENVCAT to Generate IDCAMS DEFINE Information --
000002 -- Use %=### to Assign Allocation As % of the Current Allocation --
000003 -- roundup - CYL for Cylinder boundary, TRK for Track boundary --
000004 -- Use ALLOC to override Space Allocation, TRK information from ICF --
000005 -- alloc-type can be CYL, TRK, PAGE --
000006 -- pri can be HIARBA, HIURBA or a number --
000007 -- Use VOL=* option to assign * to VOLUME --
000008 -- -----
000009 GENVCAT %=100 cyl ALLOC=(cyl,hiarba,) VOL=*

** Global Var &GO is Added with GENVCAT %=100 cyl ALLOC=(cyl,hiarba,) VOL=*
** ----- **
** Processing Selected DB2I2 Commnad As Follows: **
** GENVCAT %=100 cyl ALLOC=(cyl,hiarba,) VOL=* **
** ----- **
***

```

```

File Edit Edit_Settings Menu Utilities Compilers Test Help Db2i2
SSID(DB9G) Connect() Ucase(ON) Sysibm(SYSIBM) ----- DB2I2 10.0
EDIT DB2ADM.DB2I2.WKBENCH9 SSID: DB9G
Command ==> Scroll ==> CSR
000116 TS DDB2ADM.SBASE DB2ADM.BASE_TABLE
000117 IDCAMS DEFINE -
000118 IDCAMS CLUSTER -
000119 IDCAMS (NAME(DB2ADM9.DSNDBC.DDB2ADM.SBASE.I0001.A001)) -
000120 IDCAMS LINEAR -
000121 IDCAMS REUSE -
000122 IDCAMS SHAREOPTIONS(3 3) -
000123 IDCAMS VOLUMES(*) -
000124 IDCAMS CYLINDERS(0000000001 0000000001)) -
000125 IDCAMS DATA -
000126 IDCAMS (NAME(DB2ADM9.DSNDBC.DDB2ADM.SBASE.I0001.A001))

```

To display detail help of a particular DB2I2 command, you simply enter the desired DB2I2 command on the option field and press **Enter key**. The example below demonstrates how to use HELP to display detail information on how the DB2I2 ALTER command works.

```
#HELP----- DB2 DB2I2 Development Tool Help Screen ----- TUTORIAL
OPTION  ==> ALTER

      |-----|
      |          DB2I2 By JRH Goldenstate Software Inc.          |
      |          copyrighted 1999-2011                          |
      |-----|

-----DB2I2 HELP SCREEN - ALTER-----
ALTE | @ALTER -----DB2I2 HELP SCREEN - ALTER----- | +
     | SYNTAX: ALTER                                     |
     |                                                    |
BATC | Line Object Allowed: DB, TS TB IX VW, SG          |
     |                                                    |
BIND | Use ALTER to generate ALTER DDL                    |
     |                                                    |
BIND | The Generated DDL can be process through EXEC or DSNTIAD command | ed
     | PF3=Exit                                          |
CHEC |-----|
CONNECT                                Connect to remote location.
                                       CONNECT(location name) or CONNECT(RESET)
```


HELPL0

Command Syntax:	HELPL0 [DB2I2 line object]
Line objects allowed:	N/A
Process Mode:	Online only
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	N/A

Command Description

- Use DB2I2 HELPL0 command in online mode to display valid DB2I2 commands for a line object.
- There are two ways to invoke HELPL0:
 - Specify desired DB2I2 line object as the command option of the HELPL0 command.
 - Specify DB2I2 HELPL0 command without command option to list all available DB2I2 line objects. Once all the line objects are displayed, you can then select a line object from the display screen to display all valid DB2I2 commands associated with the selected line object.

Example

The example below demonstrates how to use HELPL0 with line object option TB to list all associated DB2I2 commands for line object TB.

```
Command ==> HELPL0 TB                               Scroll ==> CSR
000001 TB Q.PROFILES
```

The screen below displays the result from previous HELPL0 command.

```
000012 @HELPTB ----- DB2I2 Line Object HELP SCREEN - TB -----
000013
000014 Line Object: TB table
000015
000016 DB2I2 Commands Allowed:
000017     AL ALTER AUTH COAUTH CURSOR drilldown DCLGEN DDL
000018     DELETE DSNTIAUL FETCH GRANT IMPACT INSERT IX LOAD
000019     MIGR PG PL REVOKE RI SELECT SELPATHV SPACE STATS
000020     SY TS UPDATE VW
000021
000022
000023 PF3=Exit
```

HMIGRATE

Command Syntax:	HMIGRATE
Line objects allowed:	DS, AR
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	N/A
Reusable:	N/A

Command Description

- Use DB2I2 HMIGRATE command to invoke HSM HMIGRATE to migrate a DS-Dataset line object or AR-archive log line object.

Example

```
Command ==> hmigrate                               Scroll ==> CSR  
S00541      DS  TDB2.DSNDBC.DBJD001.SJD001.I0001.A001
```

HRECALL

Command Syntax:	HRECALL
Line objects allowed:	DS, AR
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	N/A
Reusable:	N/A

Command Description

- Use DB2I2 HRECALL command to invoke HSM HRECALL to recall a DS or AR line object if the selected line object has been migrated.

Example

```
Command ==> HRECALL                               Scroll ==> CSR
S00541      DS TDB2.DSNDBC.DBJD001.SJD001.I0001.A001
```

IMPACT

Command Syntax:	IMPACT
Line objects allowed:	DB, TS, TB, IX, AL, SY, VW (DT, FU, SP, TR for V6 or above) (MT, SQ for V8 or above) (TBC for V9 or above)
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	line object
Reusable:	Yes

Command Description

- Use DB2I2 IMPACT command to display all dependent line object information from the selected line objects.

Example

The example below demonstrates how to use DB2I2 IMPACT command to list all dependent line object information for a selected table space DSQDBCTL.DSQTSPRO.

```
Command ==> IMPACT Scroll ==> CSR
***** Top of Data *****
S00002 TS DSQDBCTL.DSQTSPRO Q.PROFILES
```

The screen below display the results from the previous IMPACT command.

```
Command ==> Scroll ==> CSR
***** Top of Data *****
==MSG> -- DB2I2 Impact Analysis Start At: 16 Aug 1999 15:56:15
000002 TS DSQDBCTL.DSQTSPRO Q.PROFILES
000003 TS DSQDBCTL.DSQTSPRO
000004 TB Q.PROFILES
000005 IX Q.PROFILEX
000006 VW Q.VPROFILE
000007 PG .Q.DSQ8UPRF
000008 PG .Q.DSQ9ICVS
000009 PG .Q.DSQ8ICVS
000010 PG .Q.DSQ9UPRF
==MSG> -- DB2I2 Impact Analysis End At: 16 Aug 1999 15:56:42
```

INFO

Command Syntax:	INFO
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	N/A

Command Description

- Use DB2I2 INFO command to display SSID, current connected location, as well as the DB2 catalog table prefix information.
- The format of the result contains the following: (SSID) (location) (db2 catalog table prefix)

Example

The example below demonstrates how to use DB2I2 INFO command.

```
Command ==> Info                                     Scroll ==> CSR
000541      DS TDB2.DSNDBC.DBJD001.SJD001.I0001.A001
```

The screen below displays the result from the previous INFO command. It shows connect db2 SSID is DSN with no remote location connected and catalog table prefix is default SYSIBM.

```
EDIT          JD.DB2I2.wkbench                         (DSN) ( ) (SYSIBM)
Command ==>                                           Scroll ==> CSR
```

INSERT

Command Syntax:	INSERT [MAP='dclgen Dataset'] [RUN='db2i2.run.output' [COMMIT=#] [SQLTERM(¿?)] [CHAR=HEX]]
Line objects allowed:	TB, AL, SY, VW, MT TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	INSERT SQL Skeleton
Reusable:	Yes

Command Description

- Use DB2I2 INSERT command to generate SQL INSERT statements for the select line object. The generated SQL can be used either in dynamic SQL or embedded in static SQL.
- To generate static SQL, It requires you to use MAP option to map the column information with the host variable name information from a DCLGEN output file.
- Use RUN option to read in the output from RUN command and generate INSERT SQL.
- Use COMMIT=# to specify commit frequency.
- Use SQLTERM(¿?) to specify sql terminator.
- Use CHAR=HEX to generate CHAR or VARCHAR in HEX format if single quote is part of the data from RUN output.

Example

The example below demonstrates how use DB2I2 INSERT to generate INSERT dynamic SQL for JD00.T1 table.

```
Command ==> insert                               Scroll ==> CSR
***** Top of screen *****
S00001 TB JD00.t1
```

The screen below displays the result from the previous INSERT command. The results contain a list of all the column names together with the column attribute information. You can use this information to specify correct column attributes for you INSERT SQL statement.

```
Command ==>                               Scroll ==> CSR
000001 TB JD00.t1
000002  INSERT INTO JD00.T1 VALUES(
000003      --COL1                CHAR      7      0
000004      ,--COL2                CHAR      40     0
```

The following run a query and use the output of run to insert into a table:

```
Command ==> run odsn=t1
000019  TB  DB2ADM.DSN_STATEMNT_TABLE  DDB2ADM.DSNOPTTS
ss00020  SELECT  "QUERYNO", "APPLNAME", "PROGNAME", "COLLID"
000021      , "GROUP_MEMBER", "EXPLAIN_TIME", "STMT_TYPE"
000022      , "COST_CATEGORY", "PROCMS", "PROCSU", "REASON"
000023      , "STMT_ENCODE", "TOTAL_COST", "SECTNOI", "VERSION"
000024  FROM  "DB2ADM"."DSN_STATEMNT_TABLE"
ss00025  FETCH FIRST 1 ROWS ONLY

Command ==> insert run=t1
s000019  TB  DB2ADM.DSN_STATEMNT_TABLE  DDB2ADM.DSNOPTTS

Command ==>
000019  TB  DB2ADM.DSN_STATEMNT_TABLE  DDB2ADM.DSNOPTTS
000020  INSERT INTO "DB2ADM"."DSN_STATEMNT_TABLE" (
000021  QUERYNO , APPLNAME , PROGNAME , COLLID , GROUP_MEMBER
000022  , EXPLAIN_TIME , STMT_TYPE , COST_CATEGORY , PROCMS , PROCSU
000023  , REASON , STMT_ENCODE , TOTAL_COST , SECTNOI , VERSION
000024  ) VALUES (
000025  2 , '' , 'DB2I2A9' , 'DB2I2A' , ''
000026  , '2011-03-02-14.58.54.210000' , 'INSERT' , 'A' , 1 , 2 , '' , 'U'
000027  , +0.0000000000000000E+00 , 0 , 'DB2I2A-V91'
000028  ); COMMIT;
```

IP

Command Syntax:	IP
Line objects allowed:	DB, TS, TB, IS, ISP, IX, MT, TP TSC, TBC, ISC, ISPC, IXC, TPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	IP
Reusable:	Yes

Command Description

- Use DB2I2 IP command to display Index Partition line object information from selected line objects.

Example

The example below demonstrates how to use DB2I2 IP command to display IP-Index Partition information for index JD00.%.

```

Command ==> ip                               Scroll ==> CSR
000006 pg jdoo.%
s000007 ix jdoo.%                            Y      Y
000008 -- FREE Package Name: JD00.N0AR006
    
```

The screen below displays the result from the previous IP command. Since wild card is used with selected IX line, DB2I2 displays all Index Partition information with JD00 as index creator.

```

Command ==>                                     Scroll ==> CSR
000006 pg .JD00.%
000007 ix JD00.%                               Y      Y
000008 IP JD00.XN0A0060                       0
000009 IP JD00.XN0A0050                       0
000010 IP JD00.XN0A0061                       0
000011 IP JD00.XN0A0070                       0
    
```


IPC

Command Syntax:	IPC (for V9 or above)
Line objects allowed:	DB, TS, TB, IS, ISP, IX, MT, TP TSC, TBC, ISC, ISPC, IXC, TPC
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	IPC
Reusable:	Yes

Command Description

- Use DB2I2 IPC command to display Index Partition Clone line object information from selected line objects.

IS

Command Syntax:	IS
Line objects allowed:	DB, TS, TB, ISP, IX, IP, MT, TP, PL, PG TSC, TBC, ISPC, IXC, IPC, TPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	IS
Reusable:	Yes

Command Description

- Use DB2I2 IS command to display Index Space line object information from selected line objects.

Example

The example below demonstrates how to use DB2I2 IS command to display IS-Index Space information for index JD00.%.

```

Command ==> is                                     Scroll ==> CSR
000006 pg jdo0.%
s00007 ix jdo0.%                                     Y      Y
000008 -- FREE Package Name: JD00.N0AR006
    
```

The screen below displays the result from the previous IS command. Since wild card is used with selected IX line, DB2I2 displays all Index Space information with JDOO as index creator.

```

Command ==>                                     Scroll ==> CSR
000006 pg .JD00.%
000007 ix jD00.%                                     Y      Y
000008 IS DBJD001.XN0A0060
000009 IS DBJD001.XN0A0050
000010 IS DBJD001.XN0A0061
000011 IS DBJD001.XN0A0070
    
```

ISC

Command Syntax:	ISC (for V9 or above)
Line objects allowed:	DB, TS, TB, ISP, IX, IP, MT, TP, PL, PG TSC, TBC, ISPC, IXC, IPC, TPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	ISC
Reusable:	Yes

Command Description

- Use DB2I2 ISC command to display Index Space Clone line object information from selected line objects.

ISP

Command Syntax:	ISP
Line objects allowed:	DB, TS, TB, IS, IX, IP, MT, TP TSC, TBC, ISC, IXC, IPC, TPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	ISP
Reusable:	Yes

Command Description

- Use DB2I2 ISP command to display Index Space Partition line object information from selected line objects.

Example

The example below demonstrates how to use DB2I2 ISP command to display ISP-Index Space Partition information for index JD00.%.

```

Command ==> isp                               Scroll ==> CSR
000006 pg jdo0.%
s00007 ix jdo0.%                               Y       Y
000008 -- FREE Package Name: JD00.N0AR006
    
```

The screen below displays the result from the previous ISP command. Since wild card is used with selected IX line, DB2I2 displays all Index Space Partition information with JDOO as index creator.

```

Command ==>                                     Scroll ==> CSR
000006 pg .JD00.%
000007 ix jD00.%                               Y       Y
000008   ISP DBJD001.XN0A0060                 0
000009   ISP DBJD001.XN0A0050                 0
000010   ISP DBJD001.XN0A0061                 0
000011   ISP DBJD001.XN0A0070                 0
    
```

ISPC

Command Syntax:	ISPC (for V9 or above)
Line objects allowed:	DB, TS, TB, IS, IX, IP, MT, TP TSC, TBC, ISC, IXC, IPC, TPC
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	ISPC
Reusable:	Yes

Command Description

- Use DB2I2 ISPC command to display Index Space Partition Clone line object information from selected line objects.

IX

Command Syntax:	IX
Line objects allowed:	DB, TS, TB, IS, ISP, PL, PG TSC, TBC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object IX
Reusable:	Yes

Command Description

- Use Db2I2 IX command to display index line object information from selected line objects.

Example

The example below demonstrates how to use IX to display Index information for a DB2 package JD00.N0AR006.

```

Command ==> ix                               Scroll ==> CSR
000006 pg .JD00.%
==MSG> >>>> Begin of FREE DDL
==MSG> Collid.Name..... Valid. Operative
s00007 PG .JD00.N0AR006                      Y      Y
000008 -- FREE Package Name: JD00.N0AR006
    
```

The screen below displays the result from the previous IX command.

```

Command ==>                               Scroll ==> CSR
000006 pg .JD00.%
000007 PG .JD00.N0AR006                      Y      Y
000008 IX JD00.XN0A0060
000009 IX JD00.XN0A0050
000010 IX JD00.XN0A0061
000011 IX JD00.XN0A0070
    
```

IXC

Command Syntax:	IXC
Line objects allowed:	DB, TS, TB, IS, ISP, PL, PG TSC, TBC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object IXC
Reusable:	Yes

Command Description

- Use Db2I2 IXC command to display index line clone object information from selected line objects.

JOBCARD

Command Syntax:	JOBCARD
Line objects allowed:	N/A
Process Mode:	Online only
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	N/A

Command Description

- Use DB2I2 JOBCARD command to setup Job card information. You must setup job card information before you can use DB2I2 batch facility with DB2I2 BATCH command.
- If you want to generate multiple jobs and potential execute those jobs concurrently, you must use # as the job name suffix on the first job line. When DB2I2 needs to generate a job card, It starts # from 1 and increment # by 1. The result number is then used to replace the # fields defined on the first job card.
- For example, Entering JOB#### as job name on the first job card line allows you to generate multiple jobs with job name start with JOB0001, then JOB0002.... This is important because DB2I2 can generate multiple jobs for a single DB2I2 command, generating a unique job name allows you to run all the generated jobs concurrently.
- Since db2i2 batch execution with cc > 4 considered to be server error, specify COND=(4,LT) when you prompted to enter your job card information.

Example

The example below demonstrates how to use Db2I2 command JOBCARD to define job card information.

```
Command ==> jobcard                               Scroll ==> CSR
000006 pg JD00.%
```

The screen below uses JD00#### as job card name. It allows you to generate up to 9999 jobs and potential submitting those jobs and run them concurrently. (0001-9999)

```
#JOBCARD -----DB2I2 JOBCARDS SETUP-----

Job card to be used in the DB2I2 generated JCL

//JD00#### JOB (account info).user-name,CLASS=0,_____
//          MSGCLASS=R,MSGLEVEL=(1,1),NOTIFY=JD00,_____
//          COND=(0,NE)_____
// *_____

PF3=EXIT
```


LISTC

Command Syntax:	LISTC [EXT(0 ##)] [TSIX] [IN] [T=N] [SIZE=####] [VOLSER=# volser]
Line objects allowed:	TS, TP, IX, IP, DS, IS, ISP TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report or line object TP or IP
Reusable:	Yes; if use TSIX option

Command Description

- Use DB2I2 LISTC command to display summary IDCAMS LISTCAT information against the base VSAM linear data set of the selected line objects.
- By default, DB2I2 generates LISTCAT information for all the selected line objects. You can use EXT command option to selectively list only the underline DB2 VSAM dataset which have extents exceed the specified ##. For example, specify EXT(10) option to list all dataset with more than 10 extents.
- By default, DB2I2 generates LISTCAT result with DS line object format. You can TSIX command option to override DS line object format with in TP or IP line object format. This command option allows you to reuse the output from LISTC as input to other DB2I2 commands, such as DSADJ command to remove extents.
- By default, DB2I2 writes result to an output file. If you want the results returned within your workbench session in online mode, you can use IN command option.
- By default, DB2I2 generates heading and footing information to describe the result. If you want to reuse the result from LISTC with other command in batch mode, these heading and footing become invalid line objects. To avoid this problem, DB2I2 provides you with T=N command option to disable heading and footing printing.
- Please follow the rules below for wild card with,DS,line object:
 - Use,*,to represents one qualifier
 - Use,**,to represents one or more qualifiers
 - A double asterisk cannot precede or follow any characters it must be preceded or followed by either a period or a blank,
Example:
DS vcat.dsndbd.db*.* (correct)
DS vcat.dsndbd.db** (not correct)
- To further control the subsequence DB2I2 command, you can use SIZE=#### option to generate <NEWJOB> token on the output line for every ### of accumulated cylinders. Use this option allows you to break the output into pieces with <NEWJOB>. When the output is used with the other utility generation, each piece is generated with a new job card. so that they can be run concurrently. The size accumulation is **based on the sequence of the line objects specified, which is not in size sequence**. You can use **RESIZEIT** EDIT macro to resize the accumulation method in ascending order by size. (Please refer to ED command for detail)
- Use VOLSER=# to list all dataset with VOLSER is Numeric. Use VOLSER=volser to list all dataset with a specific volser.

Example

The example below demonstrates how to use LISTC to display summary information about VSAM Linear data sets for tablespace DBTST001.STS00001 and DBTST001.STS00002.

```

Command ==> listc                                     Scroll ==> CSR
000001 TB JD00.entl%
==MSG> Table.Name.....T.OBID.DBname..T.Sname..
000002 TB JD00.ENTITY                                T 298 DBTST001 STS00001
ss0003 TS DBTST001.STS00001
ss0004 TS DBTST001.STS00002
    
```

The screen below displays the result from the previous LISTC command.

```

Command ==>
CLUSTER-NAME                                VOLUME SPACE-TYPE    EXT HI-ALLOC-RBA  CYLS  # Ks
DS TDB2.DSNDBC.DBTST001.STS00001.I0001.A001  DS3022 CYLINDER      1  7372800         10   7200K
DS TDB2.DSNDBC.DBTST001.STS00002.I0001.A001  DS3005 CYLINDER      1  7372800         10   7200K
DS Total Cylinders Used: 10 Total Ks :7200K
    
```

The example below uses TSIX option to display the output in TP or IP format. SIZE=200 command option is used to generate a <NEWJOB> token for every 200 cylinders of accumulated space. IN command option is used to direct the LISTC output result return back within your workbench.

```

Command ==> listc tsix in size=200                 Scroll ==> CSR
000001 ts Dbdpdp01.%
000003 TS DBDPSP01.DOSS                            BP5  0      A  N  N  N  16  2
000004 TB DPSP.D_O_SERVICE_SITE                    10000000
ss0004 TS DBDPSP01.OFSS                            BP5  0      A  N  N  N  16  4
000041 TS DBDPSP01.OSSS                            BP5  0      A  N  N  N  32  6
000042 TS DBDPSP01.PDAS                            BP5  4      A  N  N  Y  0  8
000043 TS DBDPSP01.REGS                            BP6  0      A  N  N  N  8  10
000044 TS DBDPSP01.SEAS                            BP5  0      A  N  N  N  8  12
000045 TS DBDPSP01.SECS                            BP5  4      A  N  N  Y  0  14
000046 TS DBDPSP01.SESS                            BP6  0      A  N  N  N  16  16
ss0047 TS DBDPSP01.SRAS                            BP5  4      A  N  N  Y  0  18
000048 TS DBDPSP01.SRQS                            BP5  4      A  N  N  Y  0  20
    
```

The screen below displays the Result from the previous command

```

Command ==>
Name                                Part Extents  CYLS  Accumulated-Cyls
000048 Name
000049 TP DBDPSP01.OFSS                0  1      000001.1  0000001.1
000050 TP DBDPSP01.OSSS                0  1      000010.0  0000011.1
000051 TP DBDPSP01.PDAS                1  1      000100.0  0000111.1
000052 TP DBDPSP01.PDAS                2  1      000278.0  0000389.1
000053 TP DBDPSP01.PDAS                3  1      000417.0  0000417.0 <NEWJOB>
000054 TP DBDPSP01.PDAS                4  1      000278.0  0000278.0 <NEWJOB>
000055 TP DBDPSP01.REGS                0  1      000001.1  0000001.1 <NEWJOB>
000056 TP DBDPSP01.SEAS                0  1      000001.1  0000002.2
000057 TP DBDPSP01.SECS                1  1      000112.0  0000114.2
000058 TP DBDPSP01.SECS                2  1      000250.0  0000364.2
000059 TP DBDPSP01.SECS                3  1      000400.0  0000400.0 <NEWJOB>
000060 TP DBDPSP01.SECS                4  1      000250.0  0000250.0 <NEWJOB>
000061 TP DBDPSP01.SESS                0  1      000001.1  0000001.1 <NEWJOB>
000062 TP DBDPSP01.SRAS                1  1      000250.0  0000251.1
000063 TP DBDPSP01.SRAS                2  1      000400.0  0000400.0 <NEWJOB>
000064 TP DBDPSP01.SRAS                3  1      000600.0  0000600.0 <NEWJOB>
000065 TP DBDPSP01.SRAS                4  1      000400.0  0000400.0 <NEWJOB>
    
```

You can use RESIZEIT edit macro to resize and regenerate the <NEWJOB> token with the size being sorted in ascending sequence. Since RESIZEIT implemented as an edit macro, When you use it in online mode, you should use **ENTER key** instead of predefined hot key to execute it. The example below uses RESIZEIT macro to generate <NEWJOB> token with size information between column 47 and column 54 for every 200 cylinders.

```
Command ==>> resizeit 47,54,200
000048      Name                               Part Extents  CYLS      Scroll ==>> CSR
          Accumulated-Cyls
ss00049 TP DBDPSP01.OFSS                       0      1      000001.1 0000001.1
000050 TP DBDPSP01.OSSS                       0      1      000010.0 0000011.1
000051 TP DBDPSP01.PDAS                        1      1      000100.0 0000111.1
000052 TP DBDPSP01.PDAS                        2      1      000278.0 0000389.1
000053 TP DBDPSP01.PDAS                        3      1      000417.0 0000417.0 <NEWJOB>
000054 TP DBDPSP01.PDAS                        4      1      000278.0 0000278.0 <NEWJOB>
000055 TP DBDPSP01.REGS                       0      1      000001.1 0000001.1 <NEWJOB>
000056 TP DBDPSP01.SEAS                       0      1      000001.1 0000002.2
000057 TP DBDPSP01.SECS                       1      1      000112.0 0000114.2
000058 TP DBDPSP01.SECS                       2      1      000250.0 0000364.2
000059 TP DBDPSP01.SECS                       3      1      000400.0 0000400.0 <NEWJOB>
000060 TP DBDPSP01.SECS                       4      1      000250.0 0000250.0 <NEWJOB>
000061 TP DBDPSP01.SESS                       0      1      000001.1 0000001.1 <NEWJOB>
000062 TP DBDPSP01.SRAS                       1      1      000250.0 0000251.1
000063 TP DBDPSP01.SRAS                       2      1      000400.0 0000400.0 <NEWJOB>
000064 TP DBDPSP01.SRAS                       3      1      000600.0 0000600.0 <NEWJOB>
ss00065 TP DBDPSP01.SRAS                       4      1      000400.0 0000400.0 <NEWJOB>
```

The screen displays result from the previous command. You can see all the small size objects are grouped on top of the list so that they can all be processed within a single job.

```
Command ==>>
000048      Name                               Part Extents  CYLS      Scroll ==>> CSR
          Accumulated-Cyls
000049 TP DBDPSP01.OFSS                       0      1      000001.1 0 0000000001.1
000050 TP DBDPSP01.REGS                       0      1      000001.1 0 0000000002.2
000051 TP DBDPSP01.SEAS                       0      1      000001.1 0 0000000003.3
000052 TP DBDPSP01.SESS                       0      1      000001.1 0 0000000004.4
000053 TP DBDPSP01.OSSS                       0      1      000010.0 0 0000000014.4
000054 TP DBDPSP01.PDAS                        1      1      000100.0 0 0000000114.4
000055 TP DBDPSP01.SECS                       1      1      000112.0 0 0000000112.0 <NEWJOB>
000056 TP DBDPSP01.SECS                       2      1      000250.0 0 0000000250.0 <NEWJOB>
000057 TP DBDPSP01.SECS                       4      1      000250.0 0 0000000250.0 <NEWJOB>
000058 TP DBDPSP01.SRAS                       1      1      000250.0 0 0000000250.0 <NEWJOB>
000059 TP DBDPSP01.PDAS                        2      1      000278.0 0 0000000278.0 <NEWJOB>
000060 TP DBDPSP01.PDAS                        4      1      000278.0 0 0000000278.0 <NEWJOB>
000061 TP DBDPSP01.SECS                       3      1      000400.0 0 0000000400.0 <NEWJOB>
000062 TP DBDPSP01.SRAS                       2      1      000400.0 0 0000000400.0 <NEWJOB>
000063 TP DBDPSP01.SRAS                       4      1      000400.0 0 0000000400.0 <NEWJOB>
000064 TP DBDPSP01.PDAS                        3      1      000417.0 0 0000000417.0 <NEWJOB>
000065 TP DBDPSP01.SRAS                       3      1      000600.0 0 0000000600.0 <NEWJOB>
```

LISTDEF DB2 v7 or above

Command Syntax:	LISTDEF (for Db2 V7 or above only) [[PARMUTIL=]'parmutil.dsn'] [IDSN='input.listdef.skelton'] [ODSN=odsn[*]] [LISTNM=mylist] [ODSN=my.pds[*]]
Line objects allowed:	DB, TX, TP, IX, IP, TB, IS, ISP, MT TXC, TPC, IXC, IPC, TBC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	LISTDEF control statement
Reusable:	Yes

Command Description

- In DB2 V7 or above, use DB2I2 LISTDEF command to generate DB2 LISTDEF control statements, which can then be used with the other DB2 utilities to easily select a group of DB2 objects for a specific DB2 utility job.
- Specify ODSN=my.pds[*] together with LISTNM=mylistnm at end of each line object, DB2I2 use the mylistnm at end of the line to override the listnm specified from entry panel and mylistnm specified also are used as the member name of the output pds. LISTNM=mylistnm can be easily generated by using the ED RESIZEIT macro. (See ED command with RESIZEIT for detail)

Example

The example below demonstrates how to use LISTDEF command to generate LISTDEF Db2 utility control statements, which include all table space and index space with COPY YES from database DBSYSADM and DBSYSAD2. The generated LISTDEF control statement is stored in a pds LISTDEF with DBSYSADM and DBSYSAD2 as member name, which can then be used as LISTDEF option of a COPY command to generate DB2 COPY utility JCL to copy all table spaces and index spaces with COPY YES.

```
Command ==>> LISTDEF ODSN=LISTDEF(*)                               Scroll ==>> CSR
S00023 db Dbsysadm          LISTNM=DBSYSADM
000024 db Dbsysad2         LISTNM=DBSYSAD2
```

DB2I2 Reference Manual

```

Comman E
S00023 e #LISTDEF ----- DB2I2 LISTDEF PROCESS OPTIONS ----- e R
000024 e                                     L1_____ e
000025 e LIST NAME                                     e
000026 e                                     e
000027 e INCLUDE/EXCLUDE I (I-Include/X-Exculde) e
000028 e TYPE SPECIFICATION B (S-Tablespaces/X-Indexspaces) e
000029 e                                     (B-Both ) e
000030 e COPY (Y/N) Y (For Indexspaces Only) e
000031 e                                     e
000032 e REFERENCED LIST NAME _____ e
000033 e                                     e
000034 e PARTLEVEL _____ (0-999) e
000035 e RI N (Y/N) e
000036 e ALL/BASE/LOB ALL_ (ALL/BASE/LOB) e
000037 e                                     e
000038 e PF3=Exit ENTER=Process Your Selection e
000039 E

```

```

Command ==>                                     Scroll ==> CSR
***** ***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG> your edit profile using the command RECOVERY ON.
000001 LISTDEF dbsysadm
000002 INCLUDE
000003 TABLESPACES
000004 DATABASE DBSYSADM
000005 ALL
000006 INCLUDE
000007 INDEXSPACES
000008 COPY YES
000009 DATABASE DBSYSADM
000010 ALL
***** ***** Bottom of Data *****

```

```

Command ==>                                     Scroll ==> CSR
***** ***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG> your edit profile using the command RECOVERY ON.
000001 LISTDEF dbsysad2
000002 INCLUDE
000003 TABLESPACES
000004 DATABASE DBSYSAD2
000005 ALL
000006 INCLUDE
000007 INDEXSPACES
000008 COPY YES
000009 DATABASE DBSYSAD2
000010 ALL
***** ***** Bottom of Data *****

```

LOAD

Command Syntax:	LOAD [[PARMUTIL=] 'load.parmutil.dsname' [DSPRE=datasetprefix[TSOID] [OVERRIDE] [FNM=ID] [WKUNIT=TAPE_Q #]
Line objects allowed:	TB or MT tablename [row-count] [SYSREC=sysrec.input] [SYSIN=sysin.input] [RESUME=YES] TBC for V9 or above [LISTDEF=listdef.dsname[(patt*)]] [TEMPLATE=template.dsname] [OPTIONS=options.dsname]
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Utility JCL LOAD
Reusable:	Yes

Command Description

- Use DB2I2 LOAD command to generate DB2 LOAD utility JCL to load a DB2 table.
- Specify optional 'load.parmutil.dsname' command option to retrieve LOAD option parameters from a file prepared by PARMUTIL LOAD command. You must use this command option when you process DB2I2 LOAD in batch mode.
- Specify optional DSPRE command option to set utility work file prefix. The default, if not specified, is your TSOID.
- Use optional **row-count** line object option on TB line to override row count information from the selection screen. When you choose to override the row-count with line object option, you must specify **0** on the estimated rows field on the LOAD selection screen to allow this override.
- Specify **OVERRIDE** command option to override the DB2 load control parameters from the load control input file with the information from the selection screen input or PARMUTIL LOAD.
- Specify optional SYSIN=sysin.input and SYSREC=sysrec.input at the end of the TB line object to assign the name of the SYSIN and SYSREC for the LOAD process.
- Specify optional **RESUME=YES** at end of the TB line object to indicate a LOAD RESUME override only for that specific TB line. Use this option allows you to load multiple DB2 tables from the same segment tablespace with one invocation of DB2 LOAD. The first TB line will be loaded with RESUME NO REPLACE (options specified from the selection screen), and the rest of the TB line objects with RESUME YES (by specifying RESUME at the end of the TB line).
- For DB2 v7 or above, you can specify LISTDEF option with PDS member pattern match. For example, By specifying LISTDEF=listdef(DB1*) will generate LOAD DB2 utility JCL for all members name prefix with DB1. Because DB2I2 process each LISTDEF one at a time, If you select to use LISTDEF pattern option, it will prompt selection screen one for each LISTDEF member. To avoid this inconvenience, you can use PARMUTIL command to generate PARMUTIL control and then use it as process option
- Use **DUMMY** or **NULLFILE** on **SYSREC INPUT DATASET** field to create a LOAD to clear Table/tablespace.

- By default, DB2I2 constructs work file name using up to 8 character of the object creator/name information. Use FNM=ID to direct DB2I2 use DBID/PSID,OBID,ISOBID when construct work file name.
For example, TS db1.ts1 with dbid= 200 and psid=208
When no FNM specify the following name will be generated:
//SYSUT1 DD DSN=dspre.db1.ts1...
With FNM=ID, the following name will be generated:
//SYSUT1 DD DSN.dspre.DB000200.PS000208...
- By default, DB2I2 uses DASD for work files. Specify WKUNIT=TAPE,# to request work file to be generated as UNIT=TAPE if estimated primary allocation for work file is greater than #. # if not specified, is default to 0.

Example

The example below demonstrates how to use LOAD command to generate LOAD utility JCL for table Q.PROFILES.

```
Command ==> load                               Scroll ==> CSR
S00001 TB Q.PROFILES
```

The selection screen below is displayed, which allows you to enter LOAD option parameters for your DB2 LOAD job. The selection screen contains the following fields:

Load control input	<ul style="list-style-type: none"> ➤ Specify SYSIN=sysin.input to use the specified file as the load control ➤ Leave out this field if you want DB2I2 to generate load control card. ➤ If you enter DSNTIAUL on this field, then DB2I2 uses the naming convention of the SYSPUNCH output file name from DB2I2 DSNTIAUL command as load control input. ➤ If you are running DB2 Version 6 or Version 5 with PTF, you can enter REORG on this field, which directs DB2I2 to use the SYSPUNCH output from REORG UNLOAD EXTERNAL as load control input file. ➤ Use UNLOAD to use SYSPUNCH from UNLOAD.
Sysrec input Dataset	<ul style="list-style-type: none"> ➤ A required field. ➤ Specify SYSREC=sysrec.input to use the specified load input file ➤ If you enter DSNTIAUL here, DB2I2 uses the naming convention of the SYSREC output file from DSNTIAUL command will be used as the SYSREC input file name. ➤ If you are running DB2 Version 6 or Version 5 with PTF, you can enter REORG on this field, which directs DB2I2 to use the SYSREC output from REORG UNLOAD EXTERNAL as sysrec input file. ➤ Use UNLOAD to use SYSREC from UNLOAD.
Estimated Rows	<ul style="list-style-type: none"> ➤ A required field which DB2I2 uses it to calculate space for all DB2 LOAD work files. If you have multiple TB needed to be loaded and each has different estimated rows, you must leave this field with 0 and enter row count information at the end of TB line object to indicate estimated rows to be loaded for each TB line object.

The definition of all the other onscreen LOAD selection fields is the same as the DB2 LOAD. Please refer to DB2 command and utility reference for detail description.

The following selection screen is displayed.

```

#LOADA ----- DB2I2 LOAD Utility PROCESS OPTIONS -----
                                                    More:  +
LOAD CONTROL DATASET
INCURSOR DATASET
SYSREC INPUT DATASET
ESTIMATED ROWS (NUMRECS) 1000 0-Estimated Rows from TB line object
PREFORMAT (Y/N) N SHRLEVEL (NONE/CHANGE) NONE
RESUME (Y/N) N REPLACE (Y/N) Y
KEEPDICTIONARY (Y/N) Y COPYDICTIONARY
ROWFORMAT (BRF/RRF) PartNo when KEEDICTIONARY = N
  Copy Type (Y/N) (T)ape/(D)ASD (Y/N) (T)ape/(D)ASD
  Local 1, 2 N T N T
  Remote 1, 2 N T N T
  Copy Dataset Prefix GDG (Y/N) Y
LOG (Y/N) Y SORTKEYS (0/N/##) N
FORMAT (UNLOAD/SQLDS/SPANNED/ ) EBCDIC/ASCII/UNICODE(E/A/U)
DELIMITED COLDEL _ , CHARDEL _ " DECPT _ .
ENFORCE (CONSTRAINTS/NO) CONSTRAINTS CCSID
DISCARDS (0 - 2147483949) 0 IGNOREFIELDS(Y/N)
SORTDEVT SYSALLDA SORTNUM 3
PART (0-4096) NOSUB (Y/N) N
NOCOPYPEND (Y/N) N REUSE (Y/N) N
STATISTICS (Y/N) N FLOAT (S390/IEEE)
FLASHCOPY (N/Y/C-Consistent) N
FCCOPYDDN TYSCOPY_

PF3=Exit ENTER=Process Your Selection PF7=PgUp PF8=PgDn
  
```


DB212 Reference Manual

The example below demonstrates how to generate DB2 LOAD JCL to

- Generate default load control card. (Leave space on the LOAD CONTROL DATASET field)
- Use qprofile.sysrec as SYSREC INPUT DATASET.
- Use 1000 as estimated rows to be loaded.

The screen below displays the result from the previous LOAD command.

```
000017 //SYSREC DD DSN='JDOO.QPROFILE.SYSREC' ,
000018 // DISP=SHR
000019 //SYSDISC DD DSN=JDOO.Q.PROFILES.LOADSYSDISC ,
000020 // DISP=(NEW,DELETE,CATLG) ,
000021 // DCB=*.SYSREC ,
000022 // UNIT=SYSDA ,
000023 // SPACE=(CYL,(1,1))
000024 //SYSUT1 DD DSN=JDOO.Q.PROFILES.LOAD.SYSUT1 ,
000025 // DISP=(NEW,DELETE,CATLG) ,
000026 // UNIT=SYSDA ,
000027 // SPACE=(CYL,(0001,0001))
000028 //SORTOUT DD DSN=JDOO.Q.PROFILES.LOAD.SORTOUT ,
000029 // DISP=(NEW,DELETE,CATLG) ,
000030 // UNIT=SYSDA ,
000031 // SPACE=(CYL,(0001,0001))
000032 //SYSMAP DD DSN=JDOO.Q.PROFILES.LOAD.SYSMAP ,
000033 // DISP=(NEW,DELETE,CATLG) ,
000034 // UNIT=SYSDA ,
000035 // SPACE=(CYL,(0001,0001))
000036 //UTPRINT DD SYSOUT=*
000037 //SYSERR DD DSN=JDOO.Q.PROFILES.LOAD.SYSERR ,
000038 // DISP=(NEW,DELETE,CATLG) ,
000039 // UNIT=SYSDA ,
000040 // SPACE=(CYL,(1,1))
000041 //SORTWK01 DD DSN=JDOO.Q.PROFILES.LOAD.SORTWK01 ,
000042 // DISP=(NEW,DELETE,CATLG) ,
000043 // UNIT=SYSDA ,
000044 // SPACE=(CYL,(2,2))
000045 //SYSIN DD *
000046 LOAD DATA
000047 INDDN SYSREC
000048 WORKDDN (SYSUT1 , SORTOUT)
000049 ERRDDN SYSERR
000050 MAPDDN SYSMAP
000051 DISCARDN SYSDISC
000052 RESUME NO
000053 REPLACE
000054 KEEPDICTIONARY
000055 LOG YES
000056 ENFORCE CONSTRAINTS
000057 DISCARDS 0
000058 SORTDEVT SYSDA
000059 SORTNUM 3
000060 INTO TABLE Q.PROFILES (
000061 CREATOR POSITION(*) CHAR(8)
000062 , CASE POSITION(*) CHAR(18)
000063 , NULLIF (CASE = )
000064 , DECOPT POSITION(*) CHAR(18)
000065 , NULLIF (DECOPT = )
000066 , CONFIRM POSITION(*) CHAR(18)
000067 , NULLIF (CONFIRM = )
000068 , WIDTH POSITION(*) CHAR(18)
000069 , NULLIF (WIDTH = )
000070 , LENGTH POSITION(*) CHAR(18)
000071 , NULLIF (LENGTH = )
000072 , LANGUAGE POSITION(*) CHAR(18)
000073 , NULLIF (LANGUAGE = )
000074 , SPACE POSITION(*) CHAR(50)
000075 , NULLIF (SPACE = )
000076 , TRACE POSITION(*) CHAR(18)
000077 , NULLIF (TRACE = )
000078 , PRINTER POSITION(*) CHAR(8)
000079 , NULLIF (PRINTER = )
000080 , TRANSLATION POSITION(*) CHAR(18)
000081 , PFKEYS POSITION(*) VARCHAR
000082 , NULLIF (PFKEYS = )
000083 , SYNONYMS POSITION(*) VARCHAR
000084 , NULLIF (SYNONYMS = )
000085 , RESOURCE_GROUP POSITION(*) CHAR(16)
000086 , NULLIF (RESOURCE_GROUP = )
000087 , MODEL POSITION(*) CHAR(8)
000088 , NULLIF (MODEL = )
000089 , ENVIRONMENT POSITION(*) CHAR(8)
000090 , NULLIF (ENVIRONMENT = )
000091 )
```

DB2I2 Reference Manual

The example below demonstrates how to use DB2I2 LOAD command to generate LOAD utility JCL for table Q.PROFILES with row count 2000000 override from the TB line object. The SYSREC and SYSPUNCH are from the output of a REORG UNLOAD EXTERNAL run output (DB2 V6 or V5 with PTF only).

```
Command ==> load                               Scroll ==> CSR
000001 TB Q.PROFILES 2000000
```

On the load selection screen, specify REORG, UNLOAD or DSNTIAUL on both LOAD CONTROL DATASET and SYSREC INPUT DATASET fields, and specify 0 on the ESTIMATED ROWS field.

```
#LOADA ----- DB2I2 LOAD Utility PROCESS OPTIONS -----
                                                    More:  +
LOAD CONTROL DATASET          REORG
INCURSOR DATASET
SYSREC INPUT DATASET          REORG
ESTIMATED ROWS (NUMRECS)      0 0-Estimated Rows from TB line object
PREFORMAT (Y/N)               N          SHRLEVEL (NONE/CHANGE) NONE__
RESUME (Y/N)                  N          REPLACE (Y/N) Y
KEEPDICTIONARY (Y/N)         Y          COPYDICTIONARY
ROWFORMAT (BRF/RRF)          ___        PartNo when KEEDICTIONARY = N
  Copy Type (Y/N)             (T)ape/(D)ASD (Y/N) (T)ape/(D)ASD
  Local 1, 2 N                 T N       T
  Remote 1, 2 N                T N       T
  Copy Dataset Prefix
LOG (Y/N) Y                    SORTKEYS (0/N/##) N
FORMAT (UNLOAD/SQlds/SPANNED/ ) EBCDIC/ASCII/UNICODE(E/A/U) _
DELIMITED COLDEL _ , CHARDEL _ " DECP _ .
ENFORCE (CONSTRAINTS/NO)     CONSTRAINTS CCSID
DISCARDS (0 - 2147483949) 0 IGNOREFIELDS(Y/N)
SORTDEVT SYSALLDA             SORTNUM 3
PART (0-4096)                NOSUB (Y/N) N
NOCOPYPEND (Y/N) N           REUSE (Y/N) N
STATISTICS (Y/N) N           FLOAT (S390/IEEE)
FLASHCOPY (N/Y/C-Consistent) N
FCCOPYDDN TYSCOPY_

PF3=Exit ENTER=Process Your Selection PF7=PgUp PF8=PgD
```

On the load selection screen, specify DUMMY SYSREC INPUT DATASET field to generate //SYSREC DD DSN=NULLFILE,DISP=SHR to clear a Table/tablespace/

```
#LOADA ----- DB2I2 LOAD Utility PROCESS OPTIONS -----
                                                    More:  +
LOAD CONTROL DATASET
INCURSOR DATASET
SYSREC INPUT DATASET          DUMMY
ESTIMATED ROWS (NUMRECS)      0 0-Estimated Rows from TB line object
PREFORMAT (Y/N)               N          SHRLEVEL (NONE/CHANGE) NONE__
RESUME (Y/N)                  N          REPLACE (Y/N) Y
KEEPDICTIONARY (Y/N)         Y          COPYDICTIONARY
ROWFORMAT (BRF/RRF)          ___        PartNo when KEEDICTIONARY = N
  Copy Type (Y/N)             (T)ape/(D)ASD (Y/N) (T)ape/(D)ASD
  Local 1, 2 N                 T N       T
  Remote 1, 2 N                T N       T
  Copy Dataset Prefix
LOG (Y/N) Y                    SORTKEYS (0/N/##) N
FORMAT (UNLOAD/SQlds/SPANNED/ ) EBCDIC/ASCII/UNICODE(E/A/U) _
DELIMITED COLDEL _ , CHARDEL _ " DECP _ .
ENFORCE (CONSTRAINTS/NO)     CONSTRAINTS CCSID
DISCARDS (0 - 2147483949) 0 IGNOREFIELDS(Y/N)
SORTDEVT SYSALLDA             SORTNUM 3
PART (0-4096)                NOSUB (Y/N) N
NOCOPYPEND (Y/N) N           REUSE (Y/N) N
STATISTICS (Y/N) N           FLOAT (S390/IEEE)
FLASHCOPY (N/Y/C-Consistent) N
FCCOPYDDN TYSCOPY_

PF3=Exit ENTER=Process Your Selection PF7=PgUp PF8=PgD
```

MAIL** a price add-on with RXdb2i2

Command Syntax:	MAIL	CLASS=smtp-class WRITER=smtp-writer [PRINT=Y] HOST=your.host.com FROM=your-name@your.host.com TO=to1@host.com,to2@host2.com,. or TO=(DD=todd1,todd2,..) or TO=(DSN=todsn1,todsn2..) CC=cc1@host.com,cc2@host2.com,.. Or CC=(DD=ccdd1,ccdd2,..) or CC=(DSN=ccdsn1,ccdsn2,..) SUBJECT=mail-subect@ BODY=(DD=dd1,dd2,..) or BODY=(DSN=body.dsname) [ATTACH=dsn1,newname1;dsn2,newname2...]
Line objects allowed:	N/A	
Process Mode:	Online and Batch	
Support		
Multiple line object:	N/A	
Multiple type of line object:	N/A	
Wild Card % allowed:	N/A	
Output Type:	Report	
Reusable:	No	

Command Description

- Use MAIL to Send an Email/Message thru SMTP server
- All options Except CC and ATTACH are Required fields.
- Where CLASS is SMTP Output Class,
- WRITER is SMTP writer;
- Optional PRINT=Y to send the copy of message to SYSOUT
- HOST is your Host domain name for SMTP;
- FROM is you email address;
- TO has 3 different format:
 - to addresses separated by comma; or
 - (DD=todd1,todd2,..) from address under DDNAME
 - (DSN=todsn1,todsn2,..) from address files todsn1,todsn2..
- CC has the same formats as TO, from addresses, DD or DSN.
- SUBJECT is the subject of this email; A @ at end of the SUBJECT text is required to end the Subject.
- BODY is the body text from either multiple input DD with IDD=dd1,dd2.. or with IDSN – input dataset name;
- Use optional ATTACH option to attach text files with format of
source-dsname,target-dsname;s2-dsname,t2-dsname;... I

Example

The example below demonstrates how to use FPT to get information from remote server, if not successful, issue MAIL and send message to a user. (The following script and the script to invoke the following are in db2i2.ispfliib. You can retrieve these scripts by FGET *FTPMAIL or FGET *FTPMAILS or ? *FTPMAIL or ? *FTPMAILS) In db2i2.ispfliib(FTPMAILS):

```

-- -----*
-- Issue Rxdb2i2 Command to Invoke DB2I2 Batch to Build JCL          -*
-- to Execute FTP and check for Status of FTP Execution.           -*
-- &H-WAITH and &H-CNT-H are Used to control number of tries and  -*

```

```

-- wait time between each Try.                                -*
-- After reach limit set by &H-CNT-H and &H-WAIT-H,          -*
-- Send Email message if still not successful                 -*
-- Specify Host variables:                                    -*
--   &H-CNT-H        - Number of FTP Tries Before Send An Email -*
--   &H-FTPDSN-H     - FTP Control Statement Input             -*
--   &H-WAIT-H       - Wait period between each FTP Try       -*
--   &H-CLASS-H     - SMTP output Class                       -*
--   &H-WRITER-H    - SMTP Output Writer                     -*
--   &H-PRINT-H     - Print e-mail                           -*
--   &H-HOST-H      - Host Name                               -*
--   &H-FROM-H      - From Address                            -*
--   &H-TO-H        - To Addresses                           -*
--   &H-CC-H        - CC Addresses                           -*
--                   Specify &H-CC-H=Cc= for No CC addresses  -*
-----*
DB2I2REX "Batch ICMD=*ftpmail",
         "jobdd=N lineobj=N",
         "&H-CNT-H=5",
         "&H-FTPDSN-H='jrh.demo(ftpinput)'",
         "&H-WAIT-H=00:00:05",
-- Mail Parmas
         "&H-CLASS-H=P",
         "&H-WRITER-H=SMTP",
         "&H-PRINT-H=Y",
         "&H-HOST-H=etjrh.dal-ebid.ihost.com",
         "&H-FROM-H=jdoong@jrh-inc.com",
         "&H-TO-H=(dsn='jrh.demo(toaddr)')",
         "&H-CC-H=Cc="

```

In db2i2.ispfliB(FTPMAIL):

```

RXDB2I2
//LINEOBJ DD *
-----*
-- This Routine demonstrates the Usage of RXDB2I2 FTP, WAIT, MAIL, TRACK commands
-- . Process FTP GET with FTP command
-- . Check Status of FTP GET
-- . If There is an Error
-- .   Wait for 5 Minutes with WAIT And Go Back to Try FTP GET Again
-- .   If More Than 5 Tries and Not Successful
-- .     Send a Message/Email with MAIL Command
-- .     Set Error and Return Code to 16
-- DDNAME Defined:
-- BODYMSG : MAIL body text
-- Note:
-- 'JRH.DEMO(FTPINPUT)' contains the FTP GET control Statements below
-- xxx.xx.xx.xx 21
-- userid password
-- ascii
-- get 'my.ftp.file'
-- quit
-----*
LINEOBJ DDNAME=BODYMSG

Hi Jerry

    This script demonstrates how to use RXDB2I2 with FTP and MAIL

Jerry Doong
12345

The Following is the FPT Execution Result

END_LINEOBJ
-----*
-- Allocate Output DD  below for FTP Error Traping & Reporting
-----*
Address Tso "Alloc Fi(Output) New Reuse",
           "Space(1,1) Tracks Unit(sysallda)",
           "Recfm(F B) Lrecl(133) Blksize(13300)"

loop_cnt = 1
LOOP:
  DB2I2REX "FTP INPUTDSN=&H-FTPDSN-H"
  Say "*** FTP Execution Results Follows: ***"
  "Execio * Diskr Output(Stem r1. Finis"
  Do I = 1 to r1.0

```

```

    Say Left(r1.i,120)
End
If RXerror() = 'Y' THEN
Do
  If loop_cnt < &H-CNT-H Then
  Do
    loop_cnt = loop_cnt + 1
    x = WAIT('PERIOD &H-WAIT-H')
    Signal LOOP
  End
  Else
  Do
    Say "*** FTP Processing Fail After "loop_cnt" Trials"
-- Set Upper Case Off to Send Subject in Upper/Lower Case
    Db2i2rex "UCASE(OFF)"
    Db2i2rex "Mail Class=&H-CLASS-H",
              "Writer=&H-WRITER-H",
              "Print=&H-PRINT-H",
              "Host=&H-HOST-H",
              "From=&H-FROM-H",
              "To=&H-TO-H",
              "&H-CC-H",
              "Subject=FTP Fail with RC="RXrc()"@" ,
-- Body of the MAIL from 2 DD, BODYMSG and OUTPUT from FTP
              "Body=(dd=bodymsg,output)"
    Db2i2rex "UCASE(ON)"
-- Set Error and RC with RXError and RXrc function
    x = RXerror(Y)
    x = RXrc(16)
  End
End
-- End of Routine

```

Where 'JRH.DEMO(FTPINPUT)' contains the following FTP control statements:

```

EDIT          JRH.DEMO(FTPINPUT) - 01.00          Columns 00001 00072
Command ==>>          Scroll ==>> CSR
***** ***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
000001 111.22.33.44 21
000002 db2adm pppppppp
000003 ascii
000004 get t99
000005 quit
***** ***** Bottom of Data *****
An email will be sent to jdoong@jrh-inc.com as well as mobil phone.

```

MIGR

Command Syntax:	MIGR [AL=N Y] [SY=N Y] [VW=N Y] [BIND=N Y] [GRANT=N Y] [RI=B P C N] [%=###[CYL TRK _]] [ALLOC=(alloc_type,pri,sec)] (The following options are for V6 or above only) [SQLTERM(?)] [DT=Y] [FU=Y] [LO=Y] [TR=Y] [SQ=Y] [MT=Y N S][MAXASGN] [OBID] [VOL=*] [CRE=new_creator] [DB=new_dbname] [CL=new_collid] [SG=new_stogroup] [VCAT=new_vcatname] [O=new_owner] [Q=new_qualifier] [SH=new_schema]
Line objects allowed:	DB, TS, TB, IX, VW, and SG, RI TS with #PART=### and PTX=index-name option (BATCH mode only) (DT, FU, SP, TR for V6 or above only) (MT, SQ for V8 or above) (TBC, TC for V9 or above) (MS, PM for V10 or above)
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	db2i2 script (DDL, DCL, db2 command, IDCAMS command)
Reusable:	Yes

Command Description

- Use DB2I2 MIGR command to generate DB2I2 migration scripts.
- The generated migration scripts consist of all the CREATE DDL, GRANT DCL, IDCAMS DEFINE for VCAT – user managed dataset, as well as BIND package/plan commands for the selected line objects and ALL their dependent objects.
- Use AL command option to specify whether migration script should include DB2 alias information. The default, if not specify, is N.
- Use SY command option to specify whether migration script should include DB2 synonym information. The default, if not specify, is N.
- Use VW command option to specify whether migration script should include DB2 view information. The default, if not specify, is N.
- Use GRANT command option to specify whether migration script should include GRANT DCL. The default, if not specify, is N.
- Use RI command option to control Referential Integrity DDL generation. Specify RI=B to generate both Parent and Child relationship, Specify RI=P to generate Parent relationship only. Specify RI=C to generate Child relationship only. Specify RI=N if you do not want to generate any referential integrity information. If no RI option specified, RI=B option is the default.
- Global command option %=### can be used with MIGR command to adjust space allocation as ### percent of the existing space allocation. The default is %=100, which mean no space adjustment. Use option sub-parameter CYL or TRK to round the space allocation for all the generated script to either Cylinder or Track boundary .
- Use **ALLOC=(alloc_type,primary,secondary)** global command option to override space allocation. The specified alloc_type must be CYL for cylinder, TRK for track, or PAGE. .
- Both %=### and ALLOC=(alloc_type,primary,secondary) option can also be entered at the end on the line object to serve as line object option override which override the option from command line.
- To convert a non-partition tablespace to a partition tablespace, you can specify **#PART=###** line object option at the end of the line object to indicate that the selected tablespace to be convert to a ### partitions partition

tablespace. The ### specified represents the number of partitions to convert, if specified, must be a numeric between 1 and 255. You can also specify **PTX=partition-index-name** line object option together with #PART option to choose one of the existing index as partition index. By default, when you specify #PART option, DB2I2 use existing cluster index as converted partition index. So if you do not specify PTX option, you must have a clustering index defined for the select table space. The generated script contains equally divided space allocation from the current definition for all partitions. Limit key information, if possible, are also derived from the chosen partition index.

- Specify **SQLTERM(?)** to override SQL terminator for the generated SQL from DB2I2 MIGR command. You can only specify this command option if your Db2 environment is V6 or above.
- The scripts generated from MIGR command might contains duplicates if a multiple table view exists. Use **POSTMIGR ED macro** to comment out all the duplicate occurrence of CREATE VIEW, CREATE ALIAS, CREATE SYNONYMS, BIND PLAN/PACKAGE and GRANT but leave the last occurrence of these duplicate command script. Please refer to ED command for detail information on how to use the POSTMIGR ED macro.
- Use MT=S option in V8 or above to include Material Query Table together with the base table space information.
- Use MAXASGN option to generate table DDL with START WITH the maxassignedval + increment.
- Use OBID option to generate OBID option for CREATE TABLE DDL.
- Use VOL=* option to generate VCAT with VOLUME(*)
- Use CRE=new_creator to change creator information.
- Use DB=new_dbname to change database name information
- Use CL=new_collid to change collection ID information
- Use SG=new_stogroup to change stogroup information
- UseVCAT=new_vcatnameto change VCAT information
- Use O=new_owner to change owner information
- Use Q=new_qualifier to change qualifier information
- Use SH=new_schema to change schema information

These generated scripts can then be executed through EXEC command.

Example

The example below demonstrates how to use DB2I2 MIGR command to generate migration script for Tablespace DBTST001.STS00002. Since there is no % option specified, DB2I2 use the current allocation to generate migration script.

```

Command ==>> migr                               Scroll ==>> CSR
000001 TB JD00.entit%
==MSG>      Table.Name.....T.OBID.DBname...TSname..
000002 TB JD00.ENTITY                T 298  DBTST001 STS00001
000003 TS DBTST001.STS00001
s00004 TS DBTST001.STS00002
    
```

Since there are no migration option specified on the command line, the following migration option screen is displayed for you to pick your migration options.

```

E#####N
e #MIGR8 ----- DB2I2 MIGR PROCESS OPTIONS ----- e
e
e   Option Selection          (Y/N)                       e
e     VIEW                    N                           e
e     ALIAS                    N                           e
e     SYNONYMS                 N                           e
e     BIND PLAN/PACKAGE       N                           e
e     GRANT                    N                           e
e     DISTINCT TYPE           N                           e
e     LOB TS                   N                           e
e     USER DEFINED FUNCTION   N                           e
e     TRIGGER                  N                           e
e     SEQUENCE                 N                           e
e     MATERIALIZED QUERY TABLE N (Y-YES,N-NO,S-include tableSpace) e
e     REFERENCE INTEGRITY     N (B-BOTH,P-PARENT,C-CHILD,N=NONE)   e
e
e PF3=Exit  ENTER=Process Your Selection                  e
D#####M
    
```

Once you make your selection and press Enter key, DB2I2 starts generating migration scripts and displays the following information.

```

Start of Migration 23 Dec 2010 13:57:22
-- Tablespace Process: DBTST001.STS00002
-- Table Generation: JD00.T2
-- End of table MIGR
End of Migration 23 Dec 2010 13:57:31
***
    
```


The screen below display the result from the previous MIGR command.

```

000005 -- Tablespace Process: DBTST001.STS00002
000006 SET CURRENT SQLID = 'JD00  ';
000007 CREATE TABLESPACE STS00002
000008         IN DBTST001
000009         USING STOGROUP  SGTST1
000010         PRIQTY  7192
000011         SECQTY  712
000012         ERASE YES
000013         FREEPAGE 0
000014         PCTFREE 10
000015         COMPRESS YES
000016         BUFFERPOOL BP0
000017         LOCKSIZE TABLE
000018         CLOSE   NO
000019         SEGSIZE 64
000020 ;
000021 -- Table Generation: JD00.T2
000022 SET CURRENT SQLID = 'JD00  ';
000023 CREATE TABLE JD00.T2
000024 (
000025     COL1                CHAR(7)
000026                     NOT NULL
000027     ,COL2                CHAR(40)
000028                     NOT NULL
000029 )
000030         IN DBTST001.STS00002
000031     AUDIT NONE
000032     OBID 326
000033     DATA CAPTURE NONE
000034 ;
000035 -- End of table MIGR

```

The following example shows how to use #PART line object option to convert a non-partitioned TS to a partitioned TS. SQLID=SYSADM command option is used to assign SQLID for all the generated migration script. %=100 option is used to specify there are no space adjustment. CYL option specifies the cylinder boundary. On the second line object, we specify #PART=3 to convert the tablespace to a 3-partition partitioned tablespace. Since we do not specify PTX option, the current clustering index is used as partition index.

```

Command ==>  MIGR SQLID=SYSADM CYL %=100                               Scroll ==> CSR
***** Top of Data *****
SS0001 TS DTEST.STEST
SS0002 TS DTEST.STEST #PART=3

```

MIGRATION SCRIPT WITHOUT #PART CONVERTING TO PARTITION OPTION:

```

SET CURRENT SQLID = 'SYSADM  ';
CREATE TABLESPACE STEST
        IN DTEST
        USING STOGROUP  GTEST001
        PRIQTY  12960
        SECQTY  720
        .
        .
SEGSIZE  64
        .
        .
SET CURRENT SQLID = 'DB2ADM';
CREATE TYPE 2 UNIQUE
        INDEX JRHJ.XTEST01
ON      JRHJ.TTEST01
( CO_CD ASC
,RCVR_NO ASC
,RCVR_LOAD_ID ASC
,RCVR_LINE_NO ASC
)
CLUSTER
USING STOGROUP  GTEST001
PRIQTY  1440

```

```

SECQTY      720
.
.
PIECESIZE   2097152 K
.
.

```

MIGRATION SCRIPT WITH #PART=3 TO CONVERTE TO 3 PARTITION TS OPTION:

- Space for each partition are equally divided
- Segsize and piecesize are both removed from #part option
- NUMPARTS and LIMITKEY information are added to the result script

```

SET CURRENT SQLID = 'SYSADM  ';
CREATE TABLESPACE STEST
      IN DTEST
      NUMPARTS 3 (
PART 1
      USING STOGROUP GTEST001
      PRIQTY 4320
      SECQTY 720
      .
PART 2
      USING STOGROUP GTEST001
      PRIQTY 4320
      SECQTY 720
      .
PART 3
      USING STOGROUP GTEST001
      PRIQTY 4320
      SECQTY 720
      .
      )
.
.
SET CURRENT SQLID = 'SYSADM';
CREATE TYPE 2 UNIQUE
      INDEX JRHJ.XTEST01
ON      JRHJ.TTEST01
( CO_CD ASC
,RCVR_NO ASC
,RCVR_LOAD_ID ASC
,RCVR_LINE_NO ASC
)
CLUSTER
(
PART 1 VALUES( '11010',332059048,'ZZ',00004.)
USING STOGROUP GTEST001
      PRIQTY 1440
      SECQTY 720
      .
, PART 2 VALUES( '11010',673299176,'ZZ',00011.)
USING STOGROUP GTEST001
      PRIQTY 1440
      SECQTY 720
      .
, PART 3 VALUES(????????????????????)
USING STOGROUP GTEST001
      PRIQTY 1440
      SECQTY 720
      .
      )
BUFFERPOOL BP0
CLOSE NO
; COMMIT;
.
.

```

MODIFY

Command Syntax:	MODIFY [[PARMUTIL=] 'modify.parmutil.dsn'] [LISTDEF=listdef.dsname[(patt*)]] [OPTIONS=options.dsname]
Line objects allowed:	TS, TP, IX, IS TSC, TPC for V9 modify recovery
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use DB2I2 MODIFY command to generate DB2 MODIFY RECOVERY utility JCL for selected line object.
- Use 'modify.parmutil.dsn' command option to retrieve MODIFY command option parameters from a file prepared by PARMUTIL MODIFY command. You must specify 'modify.parmutil.dsn' if you process DB2I2 MODIFY command in batch mode.
- For DB2 v7 or above, you can specify LISTDEF option with PDS member pattern match. For example, By specifying LISTDEF=listdef(DB1*) will generate MODIFY DB2 utility JCL for all members name prefix with DB1. Because DB2I2 process each LISTDEF one at a time, If you select to use LISTDEF pattern option, it will prompt selection screen one for each LISTDEF member. To avoid this inconvenience, you can use PARMUTIL command to generate PARMUTIL control and then use it as process option

Example

The example below demonstrates how to use DB2I2 MODIFY command to generate DB2 MODIFY RECOVERY JCL for a tablespace DBTST001.STS00002.

```

Command ==> modify                               Scroll ==> CSR
000001 TB JD00.entl%
==MSG>      Table.Name.....T.OBID.DBname...TSname..
000002 TB JD00.ENTITY                T 298 DBTST001 STS00001
000003 TS DBTST001.STS00001
==MSG> >>>> Begin of Migration DDL
s00004 TS DBTST001.STS00002

```

Since there is no 'modify.parmutil.dsn' specified on command line, Modify recovery options screen is displayed for you to specify MODIFY utility options. The DELETE AGE or DATE option allows you to select whether you want to delete recover information older than AGE or before than DATE. The following example demonstrates how to delete all recovery information older than 7 days for the selected tablespace.

```

E#####N
e #MODIFY9 ----- DB2I2 MODIFY RECOVERY/STATISTICS OPTIONS ----- e
e
e Options:  R (R-Recovery S-Statistics) e
e
e DSNUM          (####/ALL) ALL_      Option R only e
e DELETE e
e AGE            (* /0-32767) 7      Order Than e
e DATE          (* /YYYYMMDD) _____ Before e
e CLONE          (Y/N) _ e
e
e For Modify Recovery e
e RETAIN e
e LAST/LOGLIMIT/GDGLIMIT _____ e
e For LAST      ##### _____ # of Syscopy To Retain e
e For GDGLIMIT e
e LAST          ##### _____ e
e LOGLIMIT      (Y/N) _ e
e Generate TSO Delete (Y/N) N      Option R only e
e
e For Modify Statistics e
e DELETE TYPE          1 (1-ALL 2-ACCESSPATH 3-SPACE) e
e PF3=Exit ENTER=Process Your Selection e
D#####M

```

The screen below displays partial JCL result from previous Db2I2 MODIFY command. There is a DELETE job step, which delete the image copy dataset that will be removed from SYSCOPY after the MODIFY RECOVERY job is run.

```

000017 //SYSIN DD *
000018 MODIFY RECOVERY
==CHG> TABLESPACE DBTST001.STS00002
000020 DSNUM ALL
000021 DELETE AGE(7)
000022 //STEP002 EXEC PGM=IKJEFT1B,REGION=0M
000023 //SYSTSPRT DD SYSOUT=*
000024 //SYSPRINT DD SYSOUT=*
000025 //SYSTSIN DD *
==CHG> DELETE 'TP.DBTST001.STS00002.P000.LCPY1003'

```

MS

Command Syntax:	MS
Line objects allowed:	TB
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	Yes
Output Type:	line object MS – Column Mask
Reusable:	Yes

Command Description

- Use DB2I2 MS command to generate Column Mask - MS line object for the selected TB line objects.

Example

The example below demonstrates how to use DB2I2 MS command to display column Mask information for a DB2 table .

```
Command ==> ms
s01683 TB DB2ADM.EMPLOYEE

Command ==>
001683 TB DB2ADM.EMPLOYEE
001684 MS DB2ADM.SSN_MASK DB2ADM.EMPLOYEE Enable
```

MT

Command Syntax:	MT
Line objects allowed:	DB, TS, TP, TB, MT, IX, IP, IS, ISP, VW, AL, SY, PL, PG , DT, FU, SP,TR
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object MT
Reusable:	Yes

Command Description

- Use DB2I2 MT command to generate Material Query Table - MT line object for the selected line objects.

Example

The example below demonstrates how to use DB2I2 MT command to display Material Query Table information for a DB2 package JD00.N0AR006.

```

Command ==> MT                               Scroll ==> CSR
000006 pg JD00.%
==MSG> >>>> Begin of FREE DDL
==MSG>      Collid.Name..... Valid. Operative
s00007 PG JD00.N0AR006                        Y      Y
000008 -- FREE Package Name: JD00.N0AR006
    
```

The screen below displays the result from previous DB2I2 MT command.

```

Command ==>                               Scroll ==> CSR
000006 pg JD00.%
000007 PG JD00.N0AR006                        Y      Y
000008 MT JD00.M1
000009 MT JD00.M2
    
```

OI

Command Syntax:	OI
Line objects allowed:	DB, TS, TB, IX, MT, IS
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object OI
Reusable:	Yes

Command Description

- Use DB2I2 OI command to generate Object ID - OI line object for the selected line objects. The OI format for DB is dbid, for TB and MT is dbid.obid, for TS is dbid.psid, for IX and IS is dbid.isobid.

Example

The example below demonstrates how to use DB2I2 MT command to display Material Query Table information for a DB2 package JD00.N0AR006.

```

Command ==> OI                               Scroll ==> CSR
S40270 db dbSYSADM
000271 TS DBSYSADM.DSN8S41A                   DBSYSADM
000272 TB SYSADM.EMPA                         DBSYSADM.DSN8S41A
000273 IX SYSADM.XEMPLA                      SYSADM.EMPA
    
```

The screen below displays the result from previous DB2I2 OI command.

```

Command ==>                               Scroll ==> CSR
000270 db dbSYSADM
000271 TS DBSYSADM.DSN8S41A                   DBSYSADM
000272 TB SYSADM.EMPA                         DBSYSADM.DSN8S41A
000273 IX SYSADM.XEMPLA                      SYSADM.EMPA
000274 OI 266.80                             SYSADM.XEMPLA
000275 OI 266.77                             SYSADM.EMPA
000276 OI 266                               DBSYSADM
000277 OI 266.67                             DBSYSADM.DSN8S41A
    
```

OPTIONS

Command Syntax:	OPTIONS [IDSN=options.indsn] [ODSN=options.outdsn]
Line objects allowed:	N/A
Process Mode:	Online only
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	No
Output Type:	OPTIONS Utility control statement
Reusable:	Yes

Command Description

- Use DB2I2 OPTIONS command to generate DB2 OPTIONS control statements.
- The output of the OPTIONS can then be used as command options with other DB2I2 utility commands, such as COPY, REORG.
- Use IDSN command option to clone an existing OPTIONS dataset.

Example

The example below display an example of an OPTIONS command.

```
Command ==> options odsn=options(o1)
```

```
Command ==>                               Scroll ==> CSR
***** ***** Top of Data *****
000001  OPTIONS PREVIEW
000002  OPTIONS PREVIEW EVENT(ITEMERR  HALT/SKIP  WARNING  RC4/RC0/RC8)
000003  OPTIONS OFF
000004  OPTIONS KEY key-value given by IBM
000005  -- ===== --
000006  -- Please delete and only keep one of the above OPTIONS line --
000007  -- ITEMERR HALT and WARNING RC4 is the default value --
000008  -- ===== --
```


PACKIT

Command Syntax:	PACKIT [CL=collection] [O=owner] [Q=qualifier] [MEM=*]]
Line objects allowed:	DM, PL
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	DB2 command BIND
Reusable:	Yes

Command Description

- Use DB2I2 PACKIT command to generate BIND PACKAGE and BIND PLAN commands for selected DBRM or PLAN. You use this DB2I2 command to convert an existing DBRM/PLAN to a PACKAGE/PLAN.
- You can use the CL (collection command option), O (owner command option), and Q (qualifier command option) to assign the collection-ID, owner and qualifier parameters for the generated BIND PACKAGE and BIND PLAN statements.
- When you use PACKIT to convert a DBRM/PLAN to a PACKAGE/PLAN, by default, DB2I2 generates PKLIST with all the DBRM specified.

PLAN1	PLAN1
Dbrm1, dbrm2, dbrm3	PKLIST(collid.dbrm1,collid.dbrm2,collid.dbrm3)
- Use MEM=* command option to specify PKLIST(collection-ID.*) when BIND PLAN statements are generated.

PLAN1	PLAN1
Dbrm1, dbrm2, dbrm3	PKLIST(collid.*)

Example

The example below converts a DBRM to a Package.

```
Command ==> PACKIT                               Scroll ==> CSR
S00691  DM TESTPLN.TEST01      SYST.TST.DBRMLIB
```

The screen below displays result from previous PACKIT command.

```
Command ==>                               Scroll ==> CSR
000691  DM TESTPLN.TEST01      SYST.TST.DBRMLIB
000692  -- DBRM to PACKAGE: TESTPLN.TEST01 --> TST1.TEST01
000693  DB2CMD BIND PACKAGE (TST1)          -
000694  DB2CMD OWNER(TST1) QUALIFIER(TST1)    -
000695  DB2CMD LIBRARY('SYST.TST.DBRMLIB')    -
==CHG> DB2CMD MEMBER(TEST01)                 -
000697  DB2CMD ACTION(ADD)                    -
000698  DB2CMD CURRENTDATA(NO) DEGREE(1)      -
000699  DB2CMD ENABLE (*)                     -
000700  DB2CMD EXPLAIN(NO) FLAG(I) ISOLATION(CS) -
000701  DB2CMD RELEASE(COMMIT)  SQLERROR(NOPACKAGE) VALIDATE(BIND)
```

Notice that the DBRM TESTPLN.TEST01 has been converted to a package TST1.TEST01 where TST1 is the owner of the TESTPLN plan. You can override this default by specify CL (collection ID option). All the other bind package options are generated, by copying existing bind options from the selected DBRM and Plan.

The example below converts a DBRM-Plan to a Package-Plan. The difference between PACKIT a DBRM vs. PACKIT a Plan is that you can convert all of your DBRMs to packages without impacting your existing plan execution. Once you have converted all DBRM to package, you can then convert DBRM/PLAN to PACKAGE/PLAN by binding the plan with all packages associated with it.

```
Command ==> packit                               Scroll ==> CSR
000664 PL TESTPLN TST0
```

The screen below displays the result from previous PACKIT command. Because there is no MEM=* option specified, the PKLIST option inside BIND PLAN command contains all the members of the original plan. You can override this with MEM=* option so that the new plan will be bound with PKLIST(TST1.*).

```
Command ==>                                         Scroll
==> CSR
000664 PL TESTPLN TST0
000665 -- DBRM to PACKAGE: TESTPLN.TEST01 --> TST1.TEST01
000666 DB2CMD BIND PACKAGE (TST1) -
000667 DB2CMD OWNER(TST1) QUALIFIER(TST1) -
000668 DB2CMD LIBRARY('SYST.TST.DBRMLIB') -
==CHG> DB2CMD MEMBER(TEST01) -
000670 DB2CMD ACTION(ADD) -
000671 DB2CMD CURRENTDATA(NO) DEGREE(1) -
000672 DB2CMD ENABLE (*) -
000673 DB2CMD EXPLAIN(NO) FLAG(I) ISOLATION(CS) -
000674 DB2CMD RELEASE(COMMIT) SQLERROR(NOPACKAGE) VALIDATE(BIND)
000675
000676 -- PACKIT Plan: TESTPLN
000677 DB2CMD BIND PLAN (TESTPLN) -
000678 DB2CMD OWNER(TST1) QUALIFIER(TST1) -
000679 DB2CMD PKLIST(
000680 DB2CMD TST1.TEST01
000681 DB2CMD ) -
000682 DB2CMD NODEFER(PREPARE) ACQUIRE(USE) -
000683 DB2CMD CURRENTDATA(NO) DEGREE(1) -
000684 DB2CMD ACTION(REPLACE) RETAIN -
000685 DB2CMD CACHESIZE(1024) DISCONNECT(EXPLICIT) -
000686 DB2CMD ENABLE (*) -
000687 DB2CMD EXPLAINNO FLAG(I) ISOLATION(CS) -
000688 DB2CMD RELEASE(COMMIT) SQLRULES(DB2) VALIDATE(BIND)
000689
000690 DM TESTPLN.TEST01 SYST.TST.DBRMLIB
000691 -- DBRM to PACKAGE: TESTPLN.TEST01 --> TST1.TEST01
000692 DB2CMD BIND PACKAGE (TST1) -
000693 DB2CMD OWNER(TST1) QUALIFIER(TST1) -
000694 DB2CMD LIBRARY('SYST.TST.DBRMLIB') -
==CHG> DB2CMD MEMBER(TEST01) -
000696 DB2CMD ACTION(ADD) -
000697 DB2CMD CURRENTDATA(NO) DEGREE(1) -
000698 DB2CMD ENABLE (*) -
000699 DB2CMD EXPLAINNO FLAG(I) ISOLATION(CS) -
000700 DB2CMD RELEASE(COMMIT) SQLERROR(NOPACKAGE) VALIDATE(BIND)
```

PARMUTIL

Command Syntax:	PARMUTIL [REORG COPY RECOVER RUNSTATS MODIFY REPAIR LOAD REBUILD CHECK REPORT D CLGEN] [LISTDEF UNLOAD CPY2CPY for DB2 V7 or above] [ODSN=]'parmutil.output.dsn'
Line objects allowed:	N/A
Process Mode:	Online only
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	DB2 utility parameter file
Reusable:	Yes

Command Description

- Use DB2I2 PARMUTIL command to prepare utility parameters.
- The output generated from DB2I2 PARMUTIL command can be used with one of the utility command, such as REORG, to generate DB2 utility JCL dynamically.
- The selection screen displayed for DB2I2 PARMUTIL command is the same as the utility selection screen displayed when a DB2I2 utility command was entered. The only difference is that the output from a PARMUTIL command is saved in the 'parmutil.output.dsn' specified.
- When you invoke a DB2I2 utility command, such as REORG, by specify a prepared 'parmutil.output.dsn' file, DB2I2 read the content to retrieve utility parameter information, instead display the selection screen for you to specify the utility options.
- To process DB2I2 utility command in **batch mode**, you **must use a prepared 'parmutil.output.dsn'** as command option.
- When you issue DB2I2 PARMUTIL command against an existing 'parmutil.output.dsn', the contents of the file are displayed. To change the utility parameter options, you simply type over whatever the changes, and press Enter to modify the contents of these prepared 'parmutil.output.dsn' file.

Example

The example below generates REORG PARMUTIL options and saves it in util.cntl(reorg). If util.cntl does not exist, DB2I2 creates it.

```
EDIT          JD.DB2I2.wkbench          Columns 00001 00072
Command ==> PARMUTIL REORG util.cntl(reorg)          Scroll ==> CSR
```

The same selection screen as you issue DB2I2 REORG command is displayed, which allows you to prepare and save REORG utility options in a specified UTIL.CNTL(REORG) file for future reuse.

```
#REORGA ----- DB2I2 REORG INDEX/TABLESPACE PROCESS OPTIONS -----
Option: _ (1-Reorg TABLESPACE 2-Reorg INDEX)
LOG Y (Y/N) NOSYSREC N (Y/N) SORTDATA Y (Y/N)
SORTKEYS Y (Y/N) KEEPDICTIONARY Y (Y/N) CLONE _ (Y/N)
Copy Type (Y/N) (T)ape/(D)ASD (Y/N) (T)ape/(D)ASD
Local 1/2 N _ N _
Remote 1/2 N _ N _
Local Copy DSN Prefix _____
Remote Copy DSN Prefix _____ Copy GDG Y (Y/N)
FLASHCOPY (N/Y/C-Consistent) N
FCCOPYDDN _____ FYSCOPY_ _____ AUTOESTSPACE _ (Y/N)
SHRLEVEL 1 (1-NONE 2-REFERENCE 3-CHANGE) FASTSWITCH Y (Y/N)
DEADLINE spec for REFERENCE and CHANGE (NONE/timestamp/label-duration)
DEADLINE _____
DRAIN spec DRAIN-WAIT _____ RETRY _____ RETRY-DELAY 300_
For CHANGE
MAPPINGTABLE _____ MAXRO 300_ (MAX RO Log Apply)
LONGLOG 1 (1-CONTINUE 2-TERM 3-DRAIN) DELAY 1200_ (In Seconds)
DRAIN 1 (1-WRITERS 2-ALL) TIMEOUT 1 (1-ABEND 2-TERM)
FORCE _____ (NONE/READERS/ALL) AUX _ (N/Y)
PART _____ UNLOAD 1 (1-CONTINUE 2-PAUSE 3-ONLY 4-EXTERNAL)
SORTDEVT SYSALLDA SORTNUM _____
PREFORMAT Y (Y/N) % SPACE ADJUSTMENT 0_____ (0-FROM CATGL STATS)
REUSE N (Y/N) STATISTICS N (Y/N) (for unload continue and pause only)

PF3=Exit ENTER=Process Your Selection PF7=PgUp PF8=PgDn
```

Once you make your choice and press Enter, the following message is displayed on screen to signal a file UTIL.CNTL(REORG) has been successfully saved with REORG utility option parameters. You can then use this file with a DB2I2 REORG command.

```
** PARMUTIL Command Requested Successfully. REORG UTIL.CNTL(REORG)
***
```

PG

Command Syntax:	PG [TR=Y N] [TYPE=A SIUD][SARG=search-arg]
Line objects allowed:	TS, AL, TB, SY, VW, IX, PL, DB (SP, FU, TR for V6 or above) (MT, PG for V8 or above) (TSC, TBC, IXC for V9 or above)
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object PG
Reusable:	Yes

Command Description

- Use DB2I2 PG command to display DB2 package line object information for selected line objects.
- When you have to drop, modify or recreate an existing DB2 object, such as TS, TB, VW or IX, you can use this DB2I2 PG command to display all DB2 packages, which get impacted.
- Use TR=N to generate NO Trigger packages.
- Once you have a list of impacted packages, you can then use DB2I2 BIND or REBIND command to generate necessary BIND or REBIND command scripts, so that after your modification of the existing DB2 object, you can BIND or REBIND those impacted packages.
- Specify TYPE= option to qualify Type of SQL. I for Insert, U for Update, D for Delete, S for Select and A for all SQL type and is the Default type option.
- Specify SARG=search-arg to specify search arguments. For Example, SARG=INSERT or SARG=INSERT%PRODUCT.

Example

The example below demonstrates how to use DB2I2 PG command to display DB2 package information for table JD00.TN0A007.

```

Command ==> pg                               Scroll ==> CSR
000044  TB  JD00.TN0A006                          T
000045  TB  JD00.TN0A007                          T
000046  TB  JD00.TN0A005                          T
000047  IX  JD00.XN0A0060

```

The screen below displays the result from the previous PG command.

```

Command ==>                               Scroll ==> CSR
000044  TB  JD00.TN0A006                          T
000045  TB  JD00.TN0A007                          T
000046  PG  .JD00.N0AR006.
000047  PG  .JD00.N0AR006.
000048  TB  JD00.TN0A005                          T
000049  IX  JD00.XN0A0060

```

PGAUTH

Command Syntax:	PGAUTH [GRANTEE GRANTOR]
Line objects allowed:	US, PG RO for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 PGAUTH command to display package authorization information for the selected line objects.
- When DB2I2 displays package authorization information, if there are no command option specified. DB2I2 displays GRANTEE information as default. (Whom the selected package have been granted to)
- Specify GRANTOR command option to display grantor information instead of grantee information.

Example

The example below demonstrates how to used PGAUTH to display the grantee information of a DB2 package DB2I2A.DB2I2A.

```
Command ==> pgauth                               Scroll ==> CSR
S02389 PG .DB2I2A.DB2I2A.
```

The screen below display the result from previous PGAUTH command.

```
Command ==>                               Scroll ==> CSR
002389 PG .DB2I2A.DB2I2A.
002390 -- Package Authorization
002391                                     E
002392                                     X
002393                                     E
002394                                     B C C
002395                                     I O U
002396                                     N P T
002397 GRANTEE GRANTOR LOCATION.COLLID.NAME   D Y E
002398 -----
002399 US PUBLIC DB2ADM .DB2I2A.DB2I2A      Y
```

PL

Command Syntax:	PL [TYPE=A SIUD][SARG=search-arg]
Line objects allowed:	TS, AL, TB, SY, VW, IX, PG, DB (SP, FU, TR for V6 or above) (MT, PL for V8 or above) (TSC, TBC, IXC for V9 or above)
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object PL
Reusable:	Yes

Command Description

- Use DB2I2 PL command to display DB2 plan line object information for the selected line objects.
- When you have to drop, modify or recreate an existing DB2 object, such as VW, TS, TB, or IX, you can use this DB2I2 PL command to display all DB2 plans get impacted.
- Once you have a list of impacted plans, you can use DB2I2 BIND or REBIND command to generate necessary DB2 BIND or REBIND command scripts, so that after your modification of the existing DB2 object, you can BIND or REBIND those impacted plans.
- Specify TYPE= option to qualify Type of SQL. I for Insert, U for Update, D for Delete, S for Select and A for all SQL type and is the Default type option.
- Specify SARG=search-arg to specify search arguments. For Example, SARG=INSERT or SARG=INSERT%PRODUCT.

Example

The example below demonstrates how to use DB2I2 PL to display DB2 plan information for package DB2I2A.

```
Command ==> PL                               Scroll ==> CSR
S00006 pg .db2i2a.db2i2a.
000007 SET CURRENT SQLID = 'JD00  ';
000008 CREATE TABLESPACE STS00002
```

The screen below displays the result from the previous PL command.

```
Command ==>                               Scroll ==> CSR
000006 pg .db2i2a.db2i2a.
000007 PL DB2I2A
000008 SET CURRENT SQLID = 'JD00  ';
```

PLAUTH

Command Syntax:	PLAUTH [GRANTEE GRANTOR]
Line objects allowed:	US, PL RO for V9 or above
Process mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 P`LAUTH` command to display DB2 plan authorization for the selected line objects.
- By default, DB2I2 P`LAUTH` command displays GRANTEE information for the selected PL line object. Specify GRANTOR command option if GRANTOR information display is desirable.

Example

The example below demonstrates how to use DB2I2 P`LAUTH` command to display the Plan authorization of the DB2I2 plan DB2I2A.

```
Command ==>> PLAUTH                               Scroll ==>> CSR
S02390  PL DB2I2A
```

The screen below displays the result from previous DB2I2 P`LAUTH` command.

```
Command ==>>                               Scroll ==>> CSR
002390  PL DB2I2A
002391  -- Plan Authorization
002392                                     E
002393                                     X
002394                                     E
002395                                     B C
002396                                     I U
002397                                     N T
002398  GRANTEE  GRANTOR  NAME  D E
002399  -----  -----  -----  - -
002400 US PUBLIC  JDOO    DB2I2A  Y
```


PM

Command Syntax:	PM
Line objects allowed:	TB
Process mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	Yes
Output Type:	PM line object – Row Permission
Reusable:	Yes

Command Description

- Use DB2I2 PM command to display DB2 row permission for a selected Table.

Example

The example below demonstrates how to use DB2I2 PM command to display the row permission for a selected db2 table – DB2ADM.EMPLOYEE.

```
Command ==> pm
01683 TB DB2ADM.EMPLOYEE

Command ==>
001683 TB DB2ADM.EMPLOYEE
001684 PM DB2ADM.SSN_ROW_ACCESS DB2ADM.EMPLOYEE Enable
```

QBUILD

Command Syntax:	QBUILD [F1=field name 1] [F2=field name 2] [F3=field name 3] [F4=field name 4]
Line objects allowed:	Any valid line objects except AC, AR, CI, CP, CU, GV, RL
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	Yes
Output Type:	SQL WHERE predicates
Reusable:	Yes

Command Description

- Use DB2I2 QBUILD command to generate SQL WHERE predicates from selected line objects. The generated WHERE predicates can then be used with -INC in a SQL block to be executed with RUN command.
- For most the DB2I2 line object, except the ones list above, use the following rules to specify field name values:
 - Specify F1=field name 1, for line object has only one value.
For example,
Specify QBUILD F1=DBNAME for a DB DSNDB% line object generates the following SQL WHERE predicate:
((DBNAME LIKE 'DSNDB%'))
The line objects within this category are: BP, CL, DB, PL, SG, SH (v6 or above), US, VL
 - Specify F1=field name 1 and/or F2=field name 2, for line object has two values.
For example,
Specify QBUILD F1=DBNAME F2=TSNAME for a TS DSNDB06.% line object generates following SQL WHERE predicate:
((DBNAME = 'DSNDB06' AND TSNAME LIKE '%'))
Most of the line objects belong to this category: AL, DM, DT (v6 or above), FU (v6 or above), IX, SP (v6 or above), SY, TB, TR (v6 or above), TS, VW
 - Specify F1=field name 1 and/or F2=field name 2 and/or F3=field name 3, for line object has three values.
For example,
Specify QBUILD F1=DBNAME F2=TSNAME F3=PARTITION for a TP DSNDB06.% 0 line object generates the following SQL WHERE predicates:
((DBNAME = 'DSNDB06' AND TSNAME LIKE '% ' AND PARTITION=0))
The line objects belong to this category are: AI, AT, IP, SC, TP, XC
For SC [ssid[\loc].]dbanmef.tsnamef [ssid[\loc].]dbnamet.tsnamef [part#]
F1=dbnamef F2=tsnamef F3=part#
For XC [ssid[\loc].]ixcreatorf.ixnamef [ssid[\loc].]ixcreator.ixname [part#]
F1=ixcreatorf F2=ixnamef F3=part#
 - Specify F1=field name 1 and/or F2=field name 2 and/or F3=field name 3 and/or F4=field name 4, for line object has four values.
For example,
Specify QBUILD F1=LOCATION F2=COLLID F3=NAME F4=VERSION for a PG .COLLID1.NAME1.VERSION1 line object generates the following SQL WHERE predicates:
((LOCATION = ' ' AND COLLID = 'COLLID1' AND NAME = 'NAME1' AND VERSION='VERSION1'))
The line objects belong to this category are: DS, PG, RI

DB2I2 Reference Manual

- DB2I2 QBUILD generates LIKE predicate for line object with % - wild card.
- The results from QBUILD command can be used inside SQL block with DB2I2 RUN command.

Example

The example below demonstrates how to use DB2I2 QBUILD command to generate SQL WHERE predicates for tablespaces DBTST001.STS00001 and DBTST001.STS00002.

```
Command ==> QBUILD F1=DBNAME F2=NAME ODSN=W1 Scroll ==> CSR
ss0003 TS DBTST001.STS00001
ss0004 TS DBTST001.STS00002
000005 -- Tablespace Process: DBTST001.STS00002
```

The screen below displays result from previous DB2I2 QBUILD command saved in dataset W1.

```
Command ==> Scroll ==> CSR
000001 - QBUILD begins
000002 ( (DBNAME = 'DBTST001' AND NAME = 'STS00001')
000003 OR
000004 (DBNAME = 'DBTST001' AND NAME = 'STS00002')
000005 )
000006 - QBUILD End
```

The screen below demonstrates how to use the result from previous DB2I2 QBUILD command and invoke DB2I2 RUN command to process a predefined query saved in dataset Q1 as follows:

```
Command ==> RUN IDSN=Q1 Scroll ==> CSR
```

Where dataset Q1 contains the following information where W1 – output from QBUILD, is included with –INC inside the query:

```
000001 SELECT * FROM SYSIBM.SYSTABLESPACES
000002 WHERE
000003 -INC W1
```

QUIESCE

Command Syntax:	QUIESCE [TABLESPACESET] [WRITE=NO] [LISTDEF=listdef.dsname[(patt(*))] [OPTIONS=options.dsname]
Line objects allowed:	TS, TP, TSC, TPC
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use DB2I2 QUIESCE command to generate DB2 QUIESCE utility JCL for selected Table Space or Table Partition.
- For DB2 V6 or above, you can specify TABLESPACESET command option to QUIESCE TABLESPACESET.
- For DB2 V6 or above, you can also specify WRITE=NO command option to request WRITE NO option.
- For DB2 v7 or above, you can specify LISTDEF option with PDS member pattern match. For example, By specifying LISTDEF=listdef(DB1*) will generate QUIESCE DB2 utility JCL for all members name prefix with DB1.

Example

The example below demonstrates how to use DB2I2 QUIESCE command to generate DB2 QUIESCE JCL for table spaces DBTST001.STS00001 and DBTST001.STS00002.

```
Command ==> quiesce                               Scroll ==> CSR
ss0003  TS  DBTST001.STS00001
ss0004  TS  DBTST001.STS00002
000005  -- Tablespace Process:DBTST001.STS00002
```

The screen below displays partial JCL result from previous DB2I2 QUIESCE command.

```
000012  /* -----**
000013  //STEP001  EXEC  DSNUPROC,REGION=4M,COND=( 4,LT) ,
000014  //          SYSTEM='DSN',UID='JD00.QUIESCE',UTPROC=''
000015  /* -----**
000016  //SYSPRINT DD  SYSOUT=*
000017  //SYSIN   DD  *
000018          QUIESCE
000019          TABLESPACE  DBTST001.STS00001
000020          TABLESPACE  DBTST001.STS00002
***** ***** Bottom of Data *****
```

RBA

Command Syntax:	RBA [yyyy-mm-dd] [JU004='dsnju004.output'] [DETAIL=Y]
Line objects allowed:	TS, TP IX, IP, IS, ISP for v6 or above TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	SYSCOPY RBA lines
Reusable:	Yes with SETRBA command

Command Description

- Use DB2I2 RBA command to display all SYSCOPY RBA lines for the selected line objects.
- By default, DB2I2 RBA command displays all SYSCOPY RBA lines for the selected line objects. Specifies optional **yyyy-mm-dd**, to limit to display only those RBA lines, which has event date **not less than** the specified date.
- Use JU004 option to merge the RBA output with the output from a previous DSNJU004 command.
- Use DETAIL=Y to list detail RBA line information
- You can use SETRBA command to pick an INCORE RBA for a later RECOVER activity.

Example

The example below demonstrates how to use DB2I2 RBA to list all RBA/LRSN points with date greater or equal to 2010-10-01 from SYSIBM.SYSCOPY.

```
Command ==> rba 2010-01-01 detail=y
ss0340 TS DDB2ADM.DSN8L10X BP32K 0 A N N Y 0 74
000341 TS DDB2ADM.DSN8S10X BP8K0 0 P N N N 16 36
ss0342 TS DDB2ADM.DSNOPTTS BP2 0 R N N N 4 2
```

The following screen displays the result from previous RBA command.

```

=NOTE= *-----*
=NOTE= * Position to a RBA field and Issue DB2I2 SETRBA To Set Incore RBA *
=NOTE= *-----*
=NOTE= I=Instance
=NOTE= B:Base C:Clone
=NOTE= T=ICTYPE
=NOTE= A:Alter B:Rebuild Index C:Create
=NOTE= D:Check Data LOG(NO) E:Recover To Current F:Full Copy
=NOTE= I:Incremental Copy M:Modify Recovery P:Recover To Copy/RBA
=NOTE= Q:Quiesce R:Load Replce LOG(YES) S:Load Replace LOG(NO)
=NOTE= T:Term utility V:Repair Version W:Reorg LOG(NO)
=NOTE= X:Reorg LOG(YES) Y:Load Resume LOG(NO) Z:Load Resume LOG(YES)
=NOTE= S=STYPE
=NOTE= :DB2 Image Copy A:Add Part/Reset Reorg C:Con Copy/Col Added to
=NOTE= E:Clone Table Exchange F:Copy Full Yes G:Index Regenerated
=NOTE= I:Copy Full No J:Concurrent Copy L:Alter Logged/Rec PIT
=NOTE= M:Rec to PIT consistent N:Alter Ix Not Padded O:Alter Not Logged
=NOTE= P:Alter Ix Padded R:Load Replace/Rotate S:Load Replace(No)
NOTE= T:First materializing V:Key Col Altered W:Reorg Log(NO)
NOTE= X:Reorg Log(Yes) Z:Key IX Col Alter(V8-)
NOTE= IC=ICBACKUP
NOTE= :Local Primary LB:Local Backup
NOTE= RP:Recovery site Primary RB:Recovery site Backup
NOTE= DS#=DSNUM
NOTE= 0:TableSpace Level Other:Partition Level
===== I -Start RBA-- TSIC ---Date--- --time-- DS# DSNAME----- JOBNAME
00001 B 00001DA978A3 FTFC 2011-04-11 13.48.15 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1848.P001 DB2ADM01
00002 -- Full Copy: FlashCopy copy is consistent.
00003 B 00001DA7BC23 FTFC 2011-04-11 13.42.39 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1842.P001 DB2ADM01
00004 B 00001D77F1D6 FTFC 2011-04-06 17.09.37 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11096.T2209.P001 DB2ADM01
00005 B 00001D60A6A0 FTFC 2011-04-04 18.46.32 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11094.T2346.P001 DB2ADM01
00006 B 00001D4821C1 F LB 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.D11091.T0107.P000 DB2ADM01
00007 -- Full Copy: DB2 image copy

```

You can then use SETRBA DB2I2 command to select a RBA point for later REPORT and RECOVER command.

```

Command ==> setrba
===== I -Start RBA-- TSIC ---Date--- --time-- DS# DSNAME----- JOBNAME
s00001 B 00001DA978A3 FTFC 2011-04-11 13.48.15 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1848.P001 DB2ADM01
000002 -- Full Copy: FlashCopy copy is consistent.
000003 B 00001DA7BC23 FTFC 2011-04-11 13.42.39 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1842.P001 DB2ADM01

EDIT DB2ADM.DB2I2.RBA.OUTPUT Incore RBA: 00001DA978A3
Command ==>
Scroll ==> CSR
===== I -Start RBA-- TSIC ---Date--- --time-- DS# DSNAME----- JOBNAME
000001 B 00001DA978A3 FTFC 2011-04-11 13.48.15 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1848.P001 DB2ADM01
000002 -- Full Copy: FlashCopy copy is consistent.

```

REBIND

Command Syntax:	REBIND [DET=N Y] [EXPLAIN=Y] [O=owner] [CL=collectionID] [Q=qualifier]
Line objects allowed:	PL, PG
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	DB2 Command REBIND
Reusable:	Yes

Command Description

- Use DB2I2 REBIND command to generate DB2 REBIND commands script for the selected DB2 PG-package or DB2 PL-plan.
- By default, DB2I2 REBIND command does not generate all DB2 REBIND options. If desired, you can use command option DET=Y to generate all DB2 REBIND command options, which then can be edited with ED command to change the existing bind options for your REBIND.
- Use EXPLAIN=Y option to force REBIND with EXPLAIN(YES) option.
- Use O=owner to assign new owner.
- Use Q=qualifier to assign new qualifier.
- Use CL=collectionID to assign new collection ID.

Example

The example below demonstrates how to use DB2I2 REBIND to generate DB2 REBIND command for package JD00.N0AR006 and JD00.N0AR007. The optional command option DET=Y is specified so that all the detail DB2 REBIND option is displayed.

```
Command ==> rebind det=Y                               Scroll ==> CSR
ss0048   PG   .JD00.N0AR006.
ss0049   PG   .JD00.N0AR007.
000050   TB   JD00.TN0A005                               T
000051   IX   JD00.XN0A0060

-- REBIND Package: JD00.N0AR006
-- REBIND Package: JD00.N0AR007
***
```

The result from the previous REBIND command is displayed below.

```
000048 PG JD00.N0AR006
000049 -- REBIND Package: JD00.N0AR006
000050 DB2CMD REBIND PACKAGE(JD00.N0AR006) -
000051 DB2CMD OWNER(JD00) QUALIFIER(JD00) -
000052 DB2CMD CURRENTDATA(NO) DEGREE(1) -
000053 DB2CMD ENABLE(*) -
000054 DB2CMD EXPLAIN(NO) FLAG(I) ISOLATION(CS) -
000055 DB2CMD SQLERROR(NOPACKAGE) VALIDATE(BIND)
000056
000057 -- REBIND Package: JD00.N0AR006
000058 DB2CMD REBIND PACKAGE(JD00.N0AR006) -
000059 DB2CMD OWNER(JD00) QUALIFIER(JD00) -
000060 DB2CMD CURRENTDATA(NO) DEGREE(1) -
000061 DB2CMD ENABLE(*) -
000062 DB2CMD EXPLAINYES) FLAG(I) ISOLATION(CS) -
000063 DB2CMD SQLERROR(NOPACKAGE) VALIDATE(BIND)
000064
==MSG> >>>> End of BIND/REBIND DDL
```


REBUILD

Command Syntax:	REBUILD [[PARMUTIL=] 'rebuild.parmutil.dsn'] [DSPRE=datasetprefix TSOID] [DFLTSP=(1,1 pri,sec)] [FNM=ID] [WKUNIT=TAPE_0 #] [LISTDEF=listdef.dsname[(patt*)]] [TEMPLATE=template.dsname] [OPTIONS=options.dsname]
Line objects allowed:	TS, TP, IX, IP, IS, ISP TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use DB2I2 REBUILD command to generate DB2 REBUILD INDEX utility JCL for selected line objects
- You can use a prepared 'rebuild.parmutil.dsn' utility parameter file generated from PARMUTIL REBUILD to generate DB2 REBUILD utility JCL. Instead display a screen for you to select the REBUILD options, DB2I2 retrieve REBUILD options directly from the specified 'rebuild.parmutil.dsn' file.
- By default, the generated utility work files are prefixed with your TSO ID, you can use DSPRE command option to override this default prefix.
- DB2I2 REBUILD command is only available for DB2 V5 and above.
- For DB2 v7 or above, you can specify LISTDEF option with PDS member pattern match. For example, By specifying LISTDEF=listdef(DB1*) will generate REBUILD DB2 utility JCL for all members name prefix with DB1. Because DB2I2 process each LISTDEF one at a time, If you select to use LISTDEF pattern option, it will prompt selection screen one for each LISTDEF member. To avoid this inconvenience, you can use PARMUTIL command to generate PARMUTIL control and then use it as process option.
- Specify DFLTSP option to assign space allocation for work files generated for Db2 REBUILD jobs that does not have RUNSTATS information. The default is not specified is DFLTSP=(1,1).
- By default, DB2I2 constructs work file name using up to 8 character of the object creator/name information. Use FNM=ID to direct DB2I2 use DBID/PSID,OBID,ISOBID when construct work file name.
For example, TS db1.ts1 with dbid= 200 and psid=208
When no FNM specify the following name will be generated:
//SYSUT1 DD DSN=dspre.db1.ts1...
With FNM=ID, the following name will be generated:
//SYSUT1 DD DSN.dspre.DB000200.PS000208...
- By default, DB2I2 uses DASD for work files. Specify WKUNIT=TAPE,# to request work file to be generated as UNIT=TAPE if estimated primary allocation for work file is greater than #. # if not specified, is default to 0.

Example

DB2I2 Reference Manual

The example below use DB2I2 REBUILD to generate REBUILD INDEX JCL for tablespace DDB2ADM.DSN8L10X.

```
Command ==> rebuild                               Scroll ==> CSR
S00003 TS DDB2ADM.DSN8L10X
```

Since there is no 'rebuild.parmutil.dsn' command option used, DB2I2 display the following selection screen for you to enter DB2 REBUILD options.

```
#RBLDA ----- DB2I2 REBUILD INDEX PROCESS OPTIONS -----
INDEX          (ALL)      (ALL) for Rebuild TS INDEX All option
PART           _____ otherwise leave with space
SORTDEVT       SYSALLDA
SORTNUM        _____ (2-255)
One Job per step N        (Y/N)
SORTKEYS       N        (Y/N)
REUSE          N        (Y/N)
STATISTICS     N        (Y/N)
SCOPE          ALL_____ (ALL/PENDING)
CLONE          -        (Y/N)
SHRLEVEL       1        (1-REFERENCE 2-CHANGE)
DRAIN Specification
  DRAIN-WAIT   _____ RETRY _____ RETRY-DELAY _____
FLASHCOPY     N        (N/Y/C-Consistent)
FCCOPYDDN     FYSCOPY_
For SHRLEVEL CHANGE
  MAXRO       _____ (MAX RO Log Apply)
  LONGLOG     1        (1-CONTINUE 2-TERM 3-DRAIN)
  DELAY       1200____ (In Seconds)

PF3=Exit  ENTER=Process Your Selection
```

The screen below displays partial result JCL from previous DB2I2 REBUILD command, after you enter desired REBUILD options and press Enter key.

```
Command ==>                               Scroll ==> CSR
000036 REBUILD
000037 INDEX (ALL)
000038 TABLESPACE DDB2ADM.DSN8L10X
000039 SORTDEVT SYSALLDA
000040 SCOPE ALL
000041 SHRLEVEL REFERENCE
000042 FLASHCOPY NO
```

RECOVER

Command Syntax:	RECOVER [[PARMUTIL=]'recover.parmutil.dsn'] [DSPRE=datasetprefix[TSOID]] [DFLTSP=(1,1 pri,sec)] [FNM=ID] [WKUNIT=TAPE_0 #] [LISTDEF=listdef.dsname[(patt*)]] [TEMPLATE=template.dsname] [OPTIONS=options.dsname]
Line objects allowed:	TS, TP, IX, IP, IS, ISP Use REBUILD instead RECOVER to rebuild index after DB2 V5 TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use DB2U2 RECOVER command to generate DB2 RECOVER TABLESPACE or RECOVER INDEX JCL for selected line objects.
- Specify 'recover.parmutil.dsn' command option to retrieve recover utility options directly from a prepared PARMUTIL RECOVER output file.
- Use DSPRE command option to set DB2 RECOVER work file prefix. If you do not specify this option, it defaults to your TSOID.
- Once you select your recover options and press Enter key, If the request are TORBA or TOLOGPOINT, DB2I2 displays all available RBA points so that you can decide which RBA point you want to recover back to. To select which RBA point you want to recover back to, you select a syscopy line and use DB2I2 TAG command to set a RBA point for your recovery.
- Beside TO RBA, TO COPY and TO CURRENT recover option, DB2I2 include INCORE RBA point recovery and last full image copy recovery as additional recovery options. To use INCORE RBA recovery, you must use SETRBA command to pick a RBA point from edit screen or an external file. These RBA can be generated from a query result, from DSNJU004 command. Once an INCORE RBA is set, you can use INCORE RBA option to recover the selected tablespace to a preset RBA point.
- For DB2 v7 or above, you can specify LISTDEF option with PDS member pattern match. For example, By specifying LISTDEF=listdef(DB1*) will generate RECOVER DB2 utility JCL for all members name prefix with DB1. Because DB2I2 process each LISTDEF one at a time, If you select to use LISTDEF pattern option, it will prompt selection screen one for each LISTDEF member. To avoid this inconvenience, you can use PARMUTIL command to generate PARMUTIL control and then use it as process option.
- For partition tablespace recovery, you can override the rebuild index option with RBLDIX line object option. Uses RBLDIX=N line object option to disable rebuild indexes for the selected TP line. RBLDIX=Y is the default option which use rebuild index panel option as the rebuild index option. Use RBLDIX=A to rebuild all indexes after the selected TP line has been recovered.
- Specify DFLTSP option to assign space allocation for work files generated for Db2 RECOVER jobs that does not have RUNSTATS information. The default is not specified is DFLTSP=(1,1).
- By default, DB2I2 constructs work file name using up to 8 character of the object creator/name information. Use

FNM=ID to direct DB2I2 use DBID/PSID,OBID,ISOBID when construct work file name.

For example, TS db1.ts1 with dbid= 200 and psid=208

When no FNM specify the following name will be generated:

```
//SYSUT1 DD DSN=dspre.db1.ts1...
```

With FNM=ID, the following name will be generated:

```
//SYSUT1 DD DSN.dspre.DB000200.PS000208...
```

- By default, DB2I2 uses DASD for work files. Specify WKUNIT=TAPE,# to request work file to be generated as UNIT=TAPE if estimated primary allocation for work file is greater than #. # if not specified, is default to 0.

Example

The example below uses DB2I2 RECOVER command to generate RECOVER TABLESPACE JCL for tablespace DDB2ADM.DSNOPTTS.

```
Command ==> recover
S00241 ts DDB2ADM.DSNOPTTS
```

DB2I2 display the screen below for you to select your recover options. If the point of time recover (TOLOGPOINT, TORBA or TOCOPY) is requested, you will be prompted with the SYSCOPY selection panel, which allows you to select a point of time for your recover with a TAG command.

Under the Performance Enhancement section of the selection, you can select recover indexes after a tablespace recovery option to automatically generated recovery indexes job steps. You can also select allocate the last full image copy as a DD card on the generated recovery JCL to request for a static allocation for the image copy required for your recovery job. Retain multiple dataset tape after recover option allows you to generate recovery JCL with RETAIN option on a multi-dataset tape image copy backup. You can use SYSCOPY log from number of day option to narrow down the SYSCOPY search ranges. The default for this option is 7 means that only SYSCOPY records within 7 days will be selected.

```
#RECOVA ----- DB2I2 RECOVER TABLESPACE/INDEX PROCESS OPTIONS -----
DSNUM          ALL_ (ALL/0-4096)          ---- Performance Enhancement ----
CURRENT        Y (Y/N)                    Rebuild IX After TS Rcvry Y (Y/N)
TORBA          N (Y/N/I-Incore RBA)       SORTDEVT SYSALLDA SORTNUM _ (0-9)
TOLOGPOINT     N (Y/N/I-Incore RBA)
LOGONLY        N (Y/N)                    Alloc DD for Last Copy DS Y (Y/N)
TOCOPY         N (Y/N/L-LastCopy/F-LastFullCopy)
COPYDSN PREFIX _____

PAGE/CONTINUE _____ RETAIN for Tape Dataset Y (Y/N)
ERROR RANGE    N (Y/N)
SITE OPTION    A (A-All 1-LOCALSITE 2-RECOVERYSITE)
REUSE          N (Y/N) PARALLEL _ (0-9)
LOGRANGES     _ (Y/N) CURRENTCOPYONLY _ (Y/N)
TAPEUNITS     _ (0-9)
FROMDUMP       _ (Y/N) DUMPCLASS _____
RESTOREBEFORE N (Y-Restore Before RBA/N)
SYSCOPY within 007 Days (1-999) (All SYSCOPY records for last ? days)
One Job per step N (Y/N)
One Recovery Job
for all objects N (Y/N)
VERIFYSET (Y/N) _ ENFORCE (Y/N) _ BACKOUT (Y/N) _
PF3=Exit ENTER=Process Your Selection
```

DB2I2 Reference Manual

The screen below demonstrates that a TORBA recovery with 7 days of SYSCOPY option is selected.

```
#RECOVA ----- DB2I2 RECOVER TABLESPACE/INDEX PROCESS OPTIONS -----
DSNUM          ALL_ (ALL/0-4096)          ---- Performance Enhancement ----
CURRENT        N   (Y/N)                  Rebuild IX After TS Rcvry Y (Y/N)
TORBA          Y   (Y/N/I-Incore RBA)     SORTDEVT SYSALLDA SORTNUM _ (0-9)
TOLOGPOINT     N   (Y/N/I-Incore RBA)
LOGONLY        N   (Y/N)                  Alloc DD for Last Copy DS Y (Y/N)
TOCOPY         N   (Y/N/L-LastCopy/F-LastFullCopy)
COPYDSN PREFIX _____

PAGE/CONTINUE  _____                RETAIN for Tape Dataset   Y (Y/N)
ERROR RANGE    N   (Y/N)
SITE OPTION    A   (A-All 1-LOCALSITE 2-RECOVERYSITE)
REUSE          N   (Y/N) PARALLEL _ (0-9)
LOGRANGES     _   (Y/N) CURRENTCOPYONLY _ (Y/N)
TAPEUNITS     _   (0-9)
FROMDUMP      _   (Y/N) DUMPCLASS _____
RESTOREBEFORE N   (Y-Restore Before RBA/N)
SYSCOPY within 007 Days (1-999) (All SYSCOPY records for last ? days)
One Job per step N (Y/N)
One Recovery Job
for all objects N (Y/N)
VERIFYSET (Y/N) _ ENFORCE (Y/N) _ BACKOUT (Y/N) _
PF3=Exit ENTER=Process Your Selection
```

With previous recover option selected, DB2I2 displays all RBA points within last 7 days for you to select.

```
Command ==>>
=NOTE= 0:TableSpace Level Other:Partition Level
===== -Start RBA-- TSIC ---Date--- --time-- DS# DSNAME-----
000001 00001DA978A3 FTFC 2011-04-11 13.48.15 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1848.P001
000002 00001DA7BC23 FTFC 2011-04-11 13.42.39 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1842.P001
000003 00001D77F1D6 FTFC 2011-04-06 17.09.37 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11096.T2209.P001
000004 00001D60A6A0 FTFC 2011-04-04 18.46.32 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11094.T2346.P001
000005 00001D4821C1 F LB 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.D11091.T0107.P000
000006 00001D4821C1 F 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.L10000.D11090.T1811
000007 00001C573513 FTFC 2011-03-14 19.16.30 0 BKUP.LPF.DDB2ADM.DSNOPTTS.D11074.T0016.P001
000008 00001C515E64 FTFC 2011-03-14 16.56.46 0 BKUP.LPF.DDB2ADM.DSNOPTTS.D11073.T2156.P001
000009 00001A77E312 CL 2011-02-15 19.21.05 0 DDB2ADM.DSNOPTTS
```

The screen below demonstrates how to use DB2I2 TAG command to select a RBA line, with RBA point 00E6B559E52D as the point of time DB2 recover.

```
Command ==>> TAG
=NOTE= 0:TableSpace Level Other:Partition Level
===== -Start RBA-- TSIC ---Date--- --time-- DS# DSNAME-----
000001 00001DA978A3 FTFC 2011-04-11 13.48.15 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1848.P001
000002 00001DA7BC23 FTFC 2011-04-11 13.42.39 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1842.P001
000003 00001D77F1D6 FTFC 2011-04-06 17.09.37 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11096.T2209.P001
000004 00001D60A6A0 FTFC 2011-04-04 18.46.32 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11094.T2346.P001
000005 00001D4821C1 F LB 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.D11091.T0107.P000
000006 00001D4821C1 F 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.L10000.D11090.T1811
000007 00001C573513 FTFC 2011-03-14 19.16.30 0 BKUP.LPF.DDB2ADM.DSNOPTTS.D11074.T0016.P001
000008 00001C515E64 FTFC 2011-03-14 16.56.46 0 BKUP.LPF.DDB2ADM.DSNOPTTS.D11073.T2156.P001
000009 00001A77E312 CL 2011-02-15 19.21.05 0 DDB2ADM.DSNOPTTS
```

DB2I2 displays the following screen to notify you that a RBA record line is selected.

```
EDIT DB2ADM.DB2I2.RCVR.OUTPUT
Command ==>>
=NOTE= 0:TableSpace Level Other:Partition Level
===== -Start RBA-- TSIC ---Date--- --time-- DS# DSNAME-----
000001 00001DA978A3 FTFC 2011-04-11 13.48.15 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1848.P001
000002 00001DA7BC23 FTFC 2011-04-11 13.42.39 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1842.P001
000003 00001D77F1D6 FTFC 2011-04-06 17.09.37 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11096.T2209.P001
000004 00001D60A6A0 FTFC 2011-04-04 18.46.32 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11094.T2346.P001
000005 00001D4821C1 F LB 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.D11091.T0107.P000
000006 00001D4821C1 F 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.L10000.D11090.T1811
000007 00001C573513 FTFC 2011-03-14 19.16.30 0 BKUP.LPF.DDB2ADM.DSNOPTTS.D11074.T0016.P001
000008 00001C515E64 FTFC 2011-03-14 16.56.46 0 BKUP.LPF.DDB2ADM.DSNOPTTS.D11073.T2156.P001
```

Once you select a RBA point for your recovery with DB2I2 TAG command, you can end the selection by pressing

PF3 KEY (End key) to generate the Recover JCL.

The screen below displays partial result JCL from the previous RECOVER and TAG command.

```
Command ==>
000036 //SYSIN      DD *
000037     RECOVER
000038     TABLESPACE DDB2ADM.DSNOPTTS DSNUM ALL
000039     TORBA X'00001D4821C1'
000040     LOCALSITE
000041     REBUILD INDEX(ALL) TABLESPACE DDB2ADM.DSNOPTTS
000042     WORKDDN(SYSUT1)
000043     SORTDEVT SYSALLDA
```

The following example demonstrates how to use RBLDIX to control indexes rebuilding after TP recovery. It recovers partition 1,3 and 7 and only after partition 7 recovery it will rebuild all indexes.

```
TP db1.ts1 1    RBLDIX=N
TP db1.ts1 3    RBLDIX=N
TP db1.ts1 7    RBLDIX=Y

RECOVER TABLESPACE db1.ts1 PART 1
..
RECOVER TABLESPACE db1.ts1 PART 3
..
RECOVER TABLESPACE db1.ts1 PART 7
REBUILD INDEX(ALL) TABLESPACE db1.ts1
```

REORG

Command Syntax:	REORG [[PARMUTIL=] 'reorg.parmutil.dsname'] [RCHK [OVRD=reorg_override_table]] [DSPRE=datasetprefix TSOID [DFLTSP=(1,1 pri,sec)] [FNM=ID] [WKUNIT=TAPE_Q #] [MAPTABLE= online_reorg_mapping_table_override] [LISTDEF=listdef.dsname[(patt*)]] [TEMPLATE=template.dsname] [OPTIONS=options.dsname]
Line objects allowed:	TS, TP, IX, IP, IS, ISP or TS or TP ..[SYSREC=sysrec.output] [SYSPUNCH=syspunch.output] [MAPTABLE=online_reorg_mapping_table_override] TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use DB2I2 REORG command to generate DB2 REORG TABLESPACE or REORG INDEX utility JCL for selected line objects.
- Specify 'reorg.parmutil.dsn' command option to retrieve reorg utility options directly from a prepared PARMUTIL REORG output file.
- Use RCHK option to generate REORGCHK job step, which is inserted before the generated REORG job step. The REORGCHK job step invoke DB2I2 REORGCHK command to check DB2 catalog statistics and set different Return Code based on whether the selected line objects need to be reorganized. Please refer to REORGCHK command for detail. Use OVRD option to specify REORG override table.
- Use DSPRE command option to set DB2 REORG work file prefix. If you do not specify this option, it defaults to your TSOID.
- For Db2 v6 or above, specify SYSPUNCH=syspunch.output and SYSREC=sysrec.output at the end of a TS or TP line object to assign the name of the SYSPUNCH and SYSREC for the REORG UNLOAD EXTERNAL and DISCARD process (v6 only). Otherwise the default SYSREC and SYSPUNCH names will be
 - Dstasetprefix.dbname.tsname.REORGppp.SYSREC for SYSREC and
 - Dstasetprefix.dbname.tsname.REORGppp.SYSPUNCH for SYSPUNCH
 Where ppp is the partition number
- Use MAPTABLE=online_reorg_mapping_table_override at the end of selected line object to override the online reorg mapping table. The online_reorg_mapping_table_override at end of the line override the online_reorg_mapping_table_override on the command line. The online_reorg_mapping_table can also use <JOBNM> as a token to substitute it with the generated job name.
- For DB2 v7 or above, you can specify LISTDEF option with PDS member pattern match. For example, By specifying LISTDEF=listdef(DB1*) will generate REORG DB2 utility JCL for all members name prefix with DB1. Because DB2I2 process each LISTDEF one at a time, If you select to use LISTDEF pattern option, it will prompt selection screen one for each LISTDEF member. To avoid this inconvenience, you can use PARMUTIL command to generate PARMUTIL control and then use it as process option.

- Specify DFLTSP option to assign space allocation for work files generated for Db2 REORG jobs that does not have RUNSTATS information. The default is not specified is DFLTSP=(1,1).
- By default, DB2I2 constructs work file name using up to 8 character of the object creator/name information. Use FNM=ID to direct DB2I2 use DBID/PSID,OBID,ISOBID when construct work file name.
For example, TS db1.ts1 with dbid= 200 and psid=208
When no FNM specify the following name will be generated:
//SYSUT1 DD DSN=dspre.db1.ts1...
With FNM=ID, the following name will be generated:
//SYSUT1 DD DSN.dspre.DB000200.PS000208...
- By default, DB2I2 uses DASD for work files. Specify WKUNIT=TAPE,# to request work file to be generated as UNIT=TAPE if estimated primary allocation for work file is greater than #. # if not specified, is default to 0.
- By default, DB2I2 constructs work file name using up to 8 character of the object creator/name information. Use FNM=ID to direct DB2I2 use DBID/PSID,OBID,ISOBID when construct work file name.
For example, TS db1.ts1 with dbid= 200 and psid=208
When no FNM specify the following name will be generated:
//SYSUT1 DD DSN=dspre.db1.ts1...
With FNM=ID, the following name will be generated:
//SYSUT1 DD DSN.dspre.DB000200.PS000208...
- By default, DB2I2 uses DASD for work files. Specify WKUNIT=TAPE,# to request work file to be generated as UNIT=TAPE if estimated primary allocation for work file is greater than #. # if not specified, is default to 0.

Example

The example below demonstrates how to use DB2I2 REORG to generate DB2 REORG JCL for tablespaces
DDB2ADM.DSNOPTTS.

```
Command ==> REORG
S00244 TS DDB2ADM.DSNOPTTS BP2 0 R N N N 4 2
```

The following selection screen is display for you entering the REORG options.

```
#REORGA ----- DB2I2 REORG INDEX/TABLESPACE PROCESS OPTIONS -----
Option: _ (1-Reorg TABLESPACE 2-Reorg INDEX)
LOG Y (Y/N) NOSYSREC N (Y/N) SORTDATA Y (Y/N)
SORTKEYS Y (Y/N) KEEPDICTIONARY Y (Y/N) CLONE _ (Y/N)
Copy Type (Y/N) (T)ape/(D)ASD (Y/N) (T)ape/(D)ASD
Local 1/2 N _ N _
Remote 1/2 N _ N _
Local Copy DSN Prefix _____
Remote Copy DSN Prefix _____ Copy GDG Y (Y/N)
FLASHCOPY (N/Y/C-Consistent) N
FCCOPYDDN _____ FYSOCOPY_ _____ AUTOESTSPACE _ (Y/N)
SHRLEVEL 1 (1-NONE 2-REFERENCE 3-CHANGE) FASTSWITCH Y (Y/N)
DEADLINE spec for REFERENCE and CHANGE (NONE/timestamp/label-duration)
DEADLINE _____
DRAIN spec DRAIN-WAIT _____ RETRY _____ RETRY-DELAY 300_
For CHANGE
MAPPINGTABLE _____ MAXRO 300_ (MAX RO Log Apply)
LONGLOG 1 (1-CONTINUE 2-TERM 3-DRAIN) DELAY 1200_ (In Seconds)
DRAIN 1 (1-WRITERS 2-ALL) TIMEOUT 1 (1-ABEND 2-TERM)
FORCE _____ (NONE/READERS/ALL) AUX _ (N/Y)
PART _____ UNLOAD 1 (1-CONTINUE 2-PAUSE 3-ONLY 4-EXTERNAL)
SORTDEVT SYSALLDA SORTNUM _____
PREFORMAT Y (Y/N) % SPACE ADJUSTMENT 0_ (0-FROM CATGL STATS)
REUSE N (Y/N) STATISTICS N (Y/N) (for unload continue and pause only)
LEAFDISTLIMIT _____ OFFPOSLIMIT _____ INDREFLIMIT _____ REPORTONLY N (Y/N)
DISCARD N (Y/N) NOPAD Y (Y/N) SCOPE _____ ALL/PENDING REBALANCE _ (Y/N)

PF3=Exit ENTER=Process Your Selection PF7=PgUp PF8=PgDn
```



```
#REORGA ----- DB2I2 REORG INDEX/TABLESPACE PROCESS OPTIONS -----
Option: 1 (1-Reorg TABLESPACE 2-Reorg INDEX)
LOG Y (Y/N) NOSYSREC N (Y/N) SORTDATA Y (Y/N)
SORTKEYS Y (Y/N) KEEPDICTIONARY Y (Y/N) CLONE _ (Y/N)
Copy Type (Y/N) (T)ape/(D)ASD (Y/N) (T)ape/(D)ASD
Local 1/2 N _ N _
Remote 1/2 N _ N _
Local Copy DSN Prefix _____
Remote Copy DSN Prefix _____ Copy GDG Y (Y/N)
FLASHCOPY (N/Y/C-Consistent) N
FCCOPYDDN _____ FYSOCOPY_ _____ AUTOESTSPACE _ (Y/N)
SHRLEVEL 1 (1-NONE 2-REFERENCE 3-CHANGE) FASTSWITCH Y (Y/N)
DEADLINE spec for REFERENCE and CHANGE (NONE/timestamp/label-duration)
DEADLINE _____
DRAIN spec DRAIN-WAIT _____ RETRY _____ RETRY-DELAY 300_
For CHANGE
MAPPINGTABLE _____ MAXRO 300_ (MAX RO Log Apply)
LONGLOG 1 (1-CONTINUE 2-TERM 3-DRAIN) DELAY 1200_ (In Seconds)
DRAIN 1 (1-WRITERS 2-ALL) TIMEOUT 1 (1-ABEND 2-TERM)
FORCE _____ (NONE/READERS/ALL) AUX _ (N/Y)
PART _____ UNLOAD 1 (1-CONTINUE 2-PAUSE 3-ONLY 4-EXTERNAL)
SORTDEVT SYSALLDA SORTNUM _____
PREFORMAT Y (Y/N) % SPACE ADJUSTMENT 0_ (0-FROM CATGL STATS)
REUSE N (Y/N) STATISTICS N (Y/N) (for unload continue and pause only)
LEAFDISTLIMIT _____ OFFPOSLIMIT _____ INDREFLIMIT _____ REPORTONLY N (Y/N)
DISCARD N (Y/N) NOPAD Y (Y/N) SCOPE _____ ALL/PENDING REBALANCE _ (Y/N)

PF3=Exit ENTER=Process Your Selection PF7=PgUp PF8=PgDn
```

Screen below displays partial JCL result from the previous REORG command.

```
000019 //STEP001 EXEC DSNUPROC,COND=(4,LT),
000020 //* LIB='DSNA10.SDSNLOAD',
000021 // SYSTEM='DBAG',UID='DB2ADM01.STEP001',UTPROC=''
000022 //* $REORGC6----- **
000023 //* ----- **
000024 //SYSPRINT DD SYSOUT=*
000025 //STPRIN01 DD SYSOUT=*
000026 //* ----- ADJUST FOLLOWING SYSREC SPACE ALLOCATION IF COMPRESSION = Y
000027 //SYSREC DD DSN=DB2ADM.DDB2ADM.DSNOPTTS.REORG.SYSREC,
000028 // DISP=(NEW,DELETE,CATLG),
000029 // UNIT=SYSALLDA,
000030 // SPACE=(CYL,(00002,00001),RLSE)
000031 //UTPRINT DD SYSOUT=*
000032 //SYSERR DD DSN=DB2ADM.DDB2ADM.DSNOPTTS.REORG.SYSERR,
000033 // DISP=(NEW,DELETE,CATLG),
000034 // UNIT=SYSALLDA,
000035 // SPACE=(CYL,(1,1))
000036 //* $DYN SORT----- **
000037 //DFSPPARM DD *
000038 DYNALLOD=(SYSALLDA,5)
000039 //SYSIN DD *
000040 REORG TABLESPACE DDB2ADM.DSNOPTTS
000041 PREFORMAT
000042 LOG YES
000043 SORTDATA YES
000044 FLASHCOPY NO
000045 SHRLEVEL NONE
000046 UNLOAD CONTINUE
000047 KEEPDICTIONARY
000048 SORTDEVT SYSALLDA
```

The screen below demonstrates how to specify a <JOBNM> token for SHRLEVEL CHANGE online reorg.

```

Option: 1 (1-Reorg TABLESPACE 2-Reorg INDEX)
LOG Y (Y/N) NOSYSREC N (Y/N) SORTDATA Y (Y/N)
SORTKEYS Y (Y/N) KEEPDICTIONARY Y (Y/N) CLONE _ (Y/N)
Copy Type (Y/N) (T)ape/(D)ASD (Y/N) (T)ape/(D)ASD
Local 1/2 N _ N _
Remote 1/2 N _ N _
Local Copy DSN Prefix _____
Remote Copy DSN Prefix _____ Copy GDG Y (Y/N)
FLASHCOPY (N/Y/C-Consistent) N
FCCOPYDDN FYSCOPY_ AUTOESTSPACE _ (Y/N)
SHRLEVEL 3 (1-NONE 2-REFERENCE 3-CHANGE) FASTSWITCH Y (Y/N)
DEADLINE spec for REFERENCE and CHANGE (NONE/timestamp/label-duration)
DEADLINE NONE
DRAIN spec DRAIN-WAIT _ RETRY _ RETRY-DELAY 300_
For CHANGE
MAPPINGTABLE MY.MAPPING_<JOBNM> _ MAXRO 300_ (MAX RO Log Apply)
LONGLOG 1 (1-CONTINUE 2-TERM 3-DRAIN) DELAY 1200_ (In Seconds)
DRAIN 1 (1-WRITERS 2-ALL) TIMEOUT 1 (1-ABEND 2-TERM)
FORCE _ (NONE/READERS/ALL) AUX _ (N/Y)
PART _ UNLOAD 1 (1-CONTINUE 2-PAUSE 3-ONLY 4-EXTERNAL)
SORTDEVT SYSALLDA SORTNUM _
PREFORMAT Y (Y/N) % SPACE ADJUSTMENT 0_ (0-FROM CATGL STATS)
REUSE N (Y/N) STATISTICS N (Y/N) (for unload continue and pause only)
LEAFDISTLIMIT _ OFFPOSLIMIT _ INDREFLIMIT _ REPORTONLY N (Y/N)
DISCARD N (Y/N) NOPAD Y (Y/N) SCOPE _ ALL/PENDING REBALANCE _ (Y/N)

PF3=Exit ENTER=Process Your Selection PF7=PgUp PF8=PgDn
    
```

The result from the above selection screen:

```

000039 //SYSIN DD *
000040 REORG TABLESPACE DDB2ADM.DSNOPTTS
000041 PREFORMAT
000042 SORTDATA YES
000043 FLASHCOPY NO
000044 SHRLEVEL CHANGE
000045 DEADLINE NONE
000046 RETRY_DELAY 300
000047 MAPPINGTABLE MY.MAPPING_DB2ADM01
000048 MAXRO 300
000049 DRAIN WRITERS
000050 TIMEOUT ABEND
000051 LONGLOG CONTINUE
000052 DELAY 1200
000053 UNLOAD CONTINUE
000054 KEEPDICTIONARY
000055 SORTDEVT SYSALLDA
    
```

REORGCHK

Command Syntax:	REORGCHK RC(0 CC1,1 CC2) [OVRD=DB2I2.REORG_OVERRIDE your.reorg_override_table]
Line objects allowed:	TS, TP, IX, IP, IS, ISP TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	No
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use DB2I2 REORGCHK command to check if the execution of a DB2 REORG job is required for the selected line object.
- In online mode, DB2I2 set condition code to CC1 after review the DB2 catalog statistics and decided that no REORG is required for the selected line objects. It Sets condition code to CC2 if they do require a REORG.
- The criteria for setting condition code to CC2 – REORG criteria are as follows:
 - No RUNSTATS for Tablespace
 - No RUNSTATS for Tablespace Part
 - NEARINDREF+FARINDREF > 10% of CARD
 - % DROP is > 10
 - No RUNSTATS for Table
 - No RUNSTATS for Index
 - Clustering Index Ratio < 90%
 - No RUNSTATS for Index Part
 - FAROFFPOS is > 10% of the CARD
 - LEAFDIST > 200 & FREEPAGE not= 0
- The default value for the return CC is 0 for CC1 and 1 for CC2.
- Specify OVRD option, which use either the default DB2I2.REORG_OVERRIDE table or your own REORG override table to force DB2 REORG against registered db2 object. Please Use HELP *DDLREORG to get the information of the REORG override table definition and create the default or your own override table before use the OVRD option.
-

Example

The example below demonstrates how to use Db2I2 REORGCHK to check if a DB2 tablespace requires reorganization.

```
Command ==>> reorgchk                               Scroll ==>> CSR
s00010 TS DBPRIW11.TSDOP                             PRIW1.DRIVER_ON_POLICY
```

The screen below displays the result from previous REORGCHK command.

```
EDIT          DP0022.T41                               REORG is not required
Command ==>>                                         Scroll ==>> CSR
000010 TS DBPRIW11.TSDOP                             PRIW1.DRIVER_ON_POLICY
```

DB2I2 Reference Manual

The example below demonstrates how to use DB2I2 REORG with REORGCHK to generate additional REORGCHK job step to check if REORG is required for selected line object.

```
Command ==> reorg util(reorg) rchk                               Scroll ==> CSR
S00010 TS DBPRIW11.TSDOP                                PRIW1.DRIVER_ON_POLICY
```

Partial result JCL is displayed in the following screen. In STEP001 it invokes REORGCHK command to check if the selected tablespace requires reorganization. DB2I2 sets RC to 0 if it does not require REORG. Subsequence job step check condition code to bypass it if REORGCHK return a 0 return code.

```
000015 //STEP001 EXEC $DB2I2P,REGION=0M,COND=(4,LT)
000016 //SSID DD *
000017 DB2D\ 5 SYSIBM
000018 //DB2I2CMD DD *
000019 REORGCHK RC(0,1)
000020 DISPLAY LOCKS
000021 //LINEOBJ DD *
000022 TS DBPRIW11.TSDOP
000023 //* ----- **
000024 //STEP002 EXEC DSNUPROC,REGION=4M,COND=(0,EQ,STEP001.DB2I2P),
000025 // SYSTEM='DB2D',UID='TSDOP.REORG',UTPROC=''
000026 //* ----- **
000027 //SYSPRINT DD SYSOUT=*
000028 //* ---- ADJUST FOLLOWING SYSREC SPACE ALLOCATION IF COMPRESSION = Y
000029 //SYSREC DD DSN=DP0022.DBPRIW11.TSDOP.REORG.SYSREC,
000030 // DISP=(NEW,DELETE,CATLG),
```

REPAIR

Command Syntax:	REPAIR [[PARMUTIL=]'repair.parmutil.dsname']
Line objects allowed:	DB, TS, TP, IX, IP, IS, ISP TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use DB2I2 REPAIR command to generate DB2 REPAIR utility JCL for selected line objects.
- Specify 'repair.parmutil.dsname' command option to retrieve repair utility options directly from a prepared PARMUTIL REPAIR output file.

Example

The example below demonstrates how to use DB2I2 REPAIR to generate REPAIR SET TABLESPACE NOCOPYPEND for tablespaces DBTST001.STS00001 and DBTST001.STS00002.

```

Command ==> repair                                     Scroll ==> CSR
ss0003  TS  DBTST001.STS00001
ss0004  TS  DBTST001.STS00002
000005  -- Tablespace Process: DBTST001.STS00002
000006  pg jdxxdb2
000007  PL JDXXDB2
000008  SET CURRENT SQLID = `JD00  `;

```

DB2I2 display a repair process options screen for you to select your repair options. The screen below selects option to generate REPAIR utility with NOCOPYPEND option.

```
#REPAIRA ----- DB2I2 REPAIR PROCESS OPTIONS -----
repair Option: 1 (1,2,3,4,5,6)
  1. Set TABLESPACE
     PART _____
     OPTION 1 (1.NOCOPYPEND 2.NORCVRPEND 3.NOCHECKPEND 4.NOAUXWARN
              5.NOAUXCHKP 6.NOAREORPENDSTAR 7.NOAREORPEND)
  2. Set INDEX
     PART _____
     OPTION _ (1.NOCOPYPEND 2.NORCVRPEND 3.NORBDPEND 4.NOCHECKPEND
              5.NOAREORPENDSTAR 6.NOAREORPEND 7.RBDPEND 8.PSRBDPEND)
  3. LOCATE (Please mark a 'X' for selection)
     _ PAGE X'000000'
     _ RID  X'00000000'
     _ KEY _____ For index only
     _ VERIFY _ REPLACE _ DELETE _ DUMP SHRLEVEL CHANGE _ (Y/N)
  4. DBD (Please mark a 'X' for selection)
     _ DROP _ TEST _ DIAGNOSE _ REBUILD
  5. LEVELID
     PART _____
  6. VERSIONS
     CLONE _ (Y/N)

PF3=Exit ENTER=Process Your Selection
```

The partial JCL result from the previous REPAIR command is displayed below.

```
Command ==>
000012 /* ----- Scroll ==> CSR
000013 //STEP001 EXEC DSNUPROC,REGION=4M,COND=(4,LT),
000014 //          SYSTEM='DSN',UID='JD00.REPAIR',UTPROC=''
000015 /* ----- **
000016 //SYSPRINT DD SYSOUT=*
000017 //SYSIN DD *
000018 REPAIR OBJECT LOG YES
000019 SET TABLESPACE DBTST001.STS00001
000020 NOCOPYPEND
000021 REPAIR OBJECT LOG YES
000022 SET TABLESPACE DBTST001.STS00002
000023 NOCOPYPEND
```

REPORT

Command Syntax:	REPORT [[PARMUTIL=] 'report.parmutil.dsname'] ' [LISTDEF=listdef.dsname[(patt*)]] [OPTIONS=options.dsname]
Line objects allowed:	TS, TP IX, IP, IS, ISP for Db2 v6 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use DB2I2 REPORT command to generate DB2 REPORT utility JCL for selected line objects.
- Use REPORT RECOVERY and RECOVERY TO (RBA/LRSN) option to generate additional job step which generates TP line objects from selected input line objects, if it is opened for update since a set RBA point with SETRBA command. You can use this option to narrow down the scope of recovery which a recovery to a previous RBA point is desirable.
- Specify 'report.parmutil.dsname' command option to retrieve report utility options directly from a prepared PARMUTIL REPORT output file.
- For DB2 v7 or above, you can specify LISTDEF option with PDS member pattern match. For example, By specifying LISTDEF=listdef(DB1*) will generate REPORT DB2 utility JCL for all members name prefix with DB1. Because DB2I2 process each LISTDEF one at a time, If you select to use LISTDEF pattern option, it will prompt selection screen one for each LISTDEF member. To avoid this inconvenience, you can use PARMUTIL command to generate PARMUTIL control and then use it as process option.

Example

The example below demonstrates how to use DB2I2 REPORT command to generate DB2 REPORT RECOVERY JCL for selected tablespaces with recovery to RBA option.

```

Command ==> REPORT
S31350 TS DBPRIT01.TSNNI          BP0 0  A N N N 0 36
001351 TS DBPRIT01.TSNNIB        BP0 0  A N N N 64 281
001352 TS DBPRIT01.TSNPD         BP0 0  A N N N 0 38
  
```

DB2I2 displays the following report process options screen so that you select report options to process. The report recovery tablespaces option is used as an example below. A RBA point 014A2E73E1EC is set before issue the DB2I2 REPORT command.

```
#REPORT9 ----- DB2I2 REPORT PROCESS OPTIONS -----
REPORT RECOVERY
  TABLESPACE OR INDEX      1  (1-tablespace 2-index
                                N-Report tablespacset)
  INDEX                     1  (1-NONE 2-ALL)
  RECOVERY TO (RBA/LRSN/)  _  (R/L/ )  014A2E73E1EC
  DSNUM      (####/ALL) ALL
  CURRENT    (Y/N) Y
  SUMMARY    (Y/N) N
  LOCALSITE  (Y/N) Y
  RECOVERYSITE (Y/N) N
  ARCHLOG    (1/2/ALL) 1__

REPORT TABLESPACESET (Y/N) N
SHOWDSNS              (Y/N) N
PF3=Exit  ENTER=Process Your Selection
```

```
#REPORT9 ----- DB2I2 REPORT PROCESS OPTIONS -----
REPORT RECOVERY
  TABLESPACE OR INDEX      1  (1-tablespace 2-index
                                N-Report tablespacset)
  INDEX                     1  (1-NONE 2-ALL)
  RECOVERY TO (RBA/LRSN/)  R  (R/L/ )  014A2E73E1EC
  DSNUM      (####/ALL) ALL
  CURRENT    (Y/N) Y
  SUMMARY    (Y/N) N
  LOCALSITE  (Y/N) Y
  RECOVERYSITE (Y/N) N
  ARCHLOG    (1/2/ALL) 1__

REPORT TABLESPACESET (Y/N) N
SHOWDSNS              (Y/N) N
PF3=Exit  ENTER=Process Your Selection
```

The screen below displays partial JCL result from the previous REPORT command. With RECOVERY TO option, a job step after REPORT step is generated to process the result from DB2 REPORT utility. The result from the post process step contain all the TS line object which have open for update since the specified RBA point.

```
Command ==> SUB                               Scroll ==> CSR
000014 // * ----- **
000015 //STEP001 EXEC DSNUPROC,REGION=4M,COND=(4,LT),
000016 //          SYSTEM='DB2D',UID='TSNNI.REPORT',UTPROC=' '
000017 // * ----- **
000018 //SYSPRINT DD DSN=&&TEMP,DISP=(NEW,PASS),UNIT=SYSDA,
000019 //          SPACE=(TRK,(5,1),RLSE)
000020 //SYSIN DD *
000021 REPORT RECOVERY
000022     TABLESPACE DBPRIT01.TSNNI
000023     DSNUM ALL
000024     CURRENT
000025     LOCALSITE
000026 REPORT RECOVERY
000027     TABLESPACE DBPRIT01.TSNNIB
000028     DSNUM ALL
000029     CURRENT
000030     LOCALSITE
000031 REPORT RECOVERY
000032     TABLESPACE DBPRIT01.TSNPD
000033     DSNUM ALL
000034     CURRENT
000035     LOCALSITE
000036 // * ----- **
000037 //STEP001C EXEC $DB2I2P,REGION=0M,COND=(4,LT)
000038 //DB2I2P.SYSTSIN DD *
000039 P000710 DB2I2.REPORT.RECOVERY RBA=014A2E73E1EC
000040 //RPTRDD DD DSN=&&TEMP,DISP=(OLD,DELETE)
***** ***** Bottom of Data *****
```


DB2I2 Reference Manual

```
** The Report Recovery Analyzer Process is done          DP0022
** Please issue the following DB2I2 command            DP0022
**     ED DB2I2.REPORT.RECOVERY                       DP0022
**     to access the Analysis Result                  DP0022
```

```
Command ==> ed db2i2.report.recovery          Scroll ==> CSR
000034          CURRENT
000035          LOCALSITE
000036 // * ----- **
000037 //STEP001C EXEC $DB2I2P,REGION=0M,COND=(4,LT)
000038 //DB2I2P.SYSTSIN DD *
000039 P000710 DB2I2.REPORT.RECOVERY RBA=014A2E73E1EC
000040 //RPTRDD DD DSN=&&TEMP,DISP=(OLD,DELETE)
***** ***** Bottom of Data *****
```

```
Command ==>          Scroll ==> CSR
***** ***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
000001 -- DB.TS qualifies for RBA=014A2E73E1EC option
000002 -- DBname.TSname      Part Date  Time      Start RBA  End RBA      Star
000003 TP DBPRIT01.TSNNI      0000 120500 08032151 014D2E6F9B8D 014D2E8A5856 014D
000004 TP DBPRIT01.TSNPD      0000 120500 14183503 014D365AE909 014D365AEFA2 014D
***** ***** Bottom of Data *****
```

The result shows only 2 out of three tablespaces which open for update since the specified RBA point. You can then proceed with RECOVER command with INCORE option to recover to a previous RBA point.

RESETG

Command Syntax:	RESETG
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	No

Command Description

- Use DB2I2 RESETG command to reset all Global Variable back to no value status.

Example

```
Command ==> resetg                                Scroll ==> CSR
***** ***** Top of Data *****
000001 tb sysibm.syspack%
```

```
EDIT          DP0022.T41                                Global Variables have be
Command ==>                                           Scroll ==> CSR
***** ***** Top of Data *****
000001 tb sysibm.syspack%
```

REVOKE

Command Syntax:	REVOKE [FROM=from-user] [FROMROLE=from-role] [SQLID=sqid for SET SQLID and By Granter] [INCLUDE=Y N to include dependent privileges]
Line objects allowed:	AL, BP, CL, DB, PG, PL, SG, TS, TB, US, VW (DT, FU, SP, SH for V6 or above) (SQ, MT for V8 or above) (RO, TBC for V9 or above)
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	REVOKE DCL
Reusable:	Yes

Command Description

- Use DB2I2 REVOKE command to generate DB2 REVOKE DCL for selected line objects.
- Use FROM or FROMROLE command option to specify a user ID or Role ID, whose authorization will be revoked. If you do not specify FROM or FROMROLE option, then DB2I2 generates REVOKE statements for all the authorizations the line object has been granted. For example, a TB line object without a FROM or FROMROLE option leads to generating REVOKE for all the authorizations the table has been granted. You can use FROM or FROMROLE option with all line objects listed above except for the US line object.
- Use SQLID option to generate SET SQLID before Revoke and BY GRANTOR.
- Use INCLUDE=Y to include dependent privileges (v10 or above only)

Example

The example below demonstrates how to use DB2I2 REVOKE command to generate DB2 REVOKE DCL to revoke DBADM authorization of database DJDOO from USERA.

```
Command ==> revoke from=usera                               Scroll ==> CSR
S00001 DB DJDOO
```

The screen below displays the result from the previous DB2I2 REVOKE command.

```
Command ==>                                               Scroll ==> CSR
000001 DB DJDOO
000002 REVOKE
000003 DBADM
000004 ON DATABASE
000005 DJDOO
000006 FROM USERA
000007 BY SYSADM;
000007 COMMIT;
```

DB2I2 Reference Manual

The following screens demonstrate a DB2I2 REVOKE command without FROM command option. DB2I2 generates REVOKE statements for all the authorizations the database DJDOO has.

```
Command ==> revoke                               Scroll ==> CSR
S00001 DB DJDOO
```

```
Command ==>                                       Scroll ==> CSR
000002 REVOKE
000003 DBADM
000004   ON DATABASE
000005     DJDOO
000006   FROM USERA
000007   BY SYSADM;
000008 COMMIT;
000009 REVOKE
000010 DBCTRL
000011   ON DATABASE
000012     DJDOO
000013   FROM USERB
000014 COMMIT;
000015   BY SYSADM;
```

REXX

Command Syntax:	REXX [IDD=input-DDNAME IDSN='input.DSNAME'] [DSPRE=workfile.prefix]
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	Anything
Reusable:	Maybe

Command Description

- Use DB2I2 REXX command to invoke a REXX Exec or a CLIST. You can use REXX command to manipulate the input file to a DB2I2 command or output from a DB2I2 command.
- When using DB2I2 REXX command in batch mode, you must specify either IDD (input-DDNAME) or IDSN ('input-DSNAME') command option, which contain your REXX exec or CLIST.
- Use IDD option to process REXX Exec or CLIST from the input-DDNAME specified.
- Use IDSN option to read REXX Exec directly from the input file 'input.dsname'.
- Use DSPRE option to specify work file prefix.

Example

The following example JCL shows how to code inline REXX routine and incorporate it with all the other DB2I2 commands. It shows that the REXX Exec is read in from the DDNAME-REXXIN and the routine does TSO DELETE to delete T1, T2, and T3 datasets and Return a 0 return code back to DB2I2. Because DB2I2 command process each command and verify the return code to decide whether to continue process the rest of the commands in batch, you should set appropriate return code when you code your REXX or CLIST routines.

```

Command ==>
000032 //DB2I2CMD DD *
000035 REXX IDD=REXXIN
000036 .
000037 db2i2 commands
000038 .
000039 //REXXIN DD DATA,DLM=AA
000040 /* REXX */
000041 ADDRESS TSO "DELETE T1"
000042 ADDRESS TSO "DELETE T2"
000043 ADDRESS TSO "DELETE T3"
000044 RETURN 0
000045 AA
000046 //LINEOBJ DD *

```

Scroll ==> CSR

RI

Command Syntax:	RI
Line objects allowed:	TB, DB
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	Yes
Output Type:	line object RI
Reusable:	Yes

Command Description

- Use DB2I2 RI command to generate Referential Integrity line object for a selected TB or DB line object.
- The output from DB2I2 RI command is a RI line, which contains the following format:
RI parent-tbcreator.parent.tbname.child-tbcreator.child.tbname

Example

The example below demonstrates how to use DB2I2 RI command to generate Referential Integrity line objects for a db2 table DSN8410.EMP.

```
Command ==> RI Scroll ==> CSR
***** Top of Data *****
s000058 TB DSN8410.EMP T 18 DSN8D41A DSN8S41E
000065 TB DSN8410.DEPT T 11 DSN8D41A DSN8S41D
000066 TB DSN8410.EACT T 58 DSN8D41A DSN8S41R
```

The screen below displays the result from previous DB2I2 RI command.

```
Command ==> Scroll ==> CSR
***** Top of Data *****
==MSG> /* -----*/
==MSG> /* DB2I2 DB2 Catalog Interface Tool Box */
==MSG> /* By JRH GoldenState Software, Inc. */
==MSG> /* (C) Copyrighted 1997,2011 */
==MSG> /* Licensed to EVALUATION COPY */
==MSG> /* DB2I2 Environment - SSID(DSN) CONNECT( ) SYSIBM(SYSIBM) */
==MSG> /* -----*/
000058 TB DSN8410.EMP T 18 DSN8D41A DSN8S41E
==MSG> Parent_creator.Parent_name.Child_creator.Child_name Relname Delete Rule
000059 RI DSN8410.EMP.DSN8410.DEPT RDE N
000060 RI DSN8410.DEPT.DSN8410.EMP RED N
000061 RI DSN8410.EMP.DSN8410.PROJ RESPEMP R
000062 RI DSN8410.EMP.DSN8410.EMPPROJECT REPAE R
```

Where Delete Rule: N-set Null; R-Restrict; C-Cascade.

RSAUTH

Command Syntax:	RSAUTH [GRANTEE GRANTOR]
Line objects allowed:	BP, CL, SG, TS, US (DT for V6 or above) (RO for V9 or above)
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 RSAUTH command to display resource authorization information for selected line objects.
- Command option GRANTEE or GRANTOR allow you to choose whether you want to display grantor or grantee authorization information. The default, if no option specified, is GRANTEE.

Example

The example below demonstrates how to used Db2I2 RSAUTH command to display resource authorization of Bufferpool BP0.

```
Command ==> RSAUTH                               Scroll ==> CSR
S00001 BP bp0
```

The screen below display the result from previous DB2I2 RSAUTH command. It shows the user PROD and PUBLIC both have been granted usage of the BP0.

```
Command ==>                               Scroll ==> CSR
000001 BP bp0
000002 -- Resource Authorization
000003   GRANTEE  GRANTOR  NAME           TYPE  USEAUTH
000004   -----  -----  -----  -----  -----
000005 US PROD    DB2ADM   BP0                BP    Y
000006 US PUBLIC  DB2ADM   BP0                BP    Y
```

RTAUTH

Command Syntax:	RTAUTH [GRANTEE GRANTOR]
Line objects allowed:	FU, SP, US RO for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 RTAUTH command to display routine authorization information for selected line objects.
- Command option GRANTEE or GRANTOR allow you to choose whether you want to display grantor or grantee authorization information. The default, if no option specified, is GRANTEE.

Example

The example below demonstrates how to used Db2I2 RSAUTH command to display routine authorization of select SP and FU lines.

```
Command ==> RTAUTH                               Scroll ==> CSR
S20274 SP SYSADM.UPDATE_PLAN           UPDATE_PLAN
000275 FU SYSADM.HATSIZE               HATSIZE
```

The screen below display the result from previous DB2I2 RTAUTH command

```
Command ==>                                       Scroll ==> CSR
000276 -- Routine Authorization (SP SYSADM.UPDATE_PLAN)
000277                                           E
000278                                           X C
000279                                           E O
000280                                           T C L
000281                                           Y U L
000282                                           P T I
000283     GRANTEE  GRANTOR  SCHEMA.SPECIFICNAME  E E D
000284 -----
000285 US abc      SYSADM   SYSADM.UPDATE_PLAN   SP Y
000286
000287 -- Routine Authorization (FU SYSADM.HATSIZE)
000288                                           E
000289                                           X C
000290                                           E O
000291                                           T C L
000292                                           Y U L
000293                                           P T I
000294     GRANTEE  GRANTOR  SCHEMA.SPECIFICNAME  E E D
000295 -----
000296 US abc      SYSADM   SYSADM.HATSIZE       FU Y
```


RUN

Command Syntax:	RUN [LIMIT(300 #)] [IN] [T=Y N] [Host-variable options] [DLM=_ ?] [EDIT=N Y] [TRUNC=N Y] [NOTFOUND(CONTINUE SKIP #)] [DURATION=N Y][BR=colname]
Line objects allowed:	SQL block
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes – Multiple line with one SQL SELECT only
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Report or line object
Reusable:	Maybe

Command Description

- Use DB2I2 RUN command to dynamically run a set of SELECT SQL statements.
- The default fetch row limit for Db2I2 RUN command is set to 300 rows. You can use command option LIMIT to override the default fetch row limit. The # specified, must be a positive numeric. Just like SPUFI, **It is not recommended to run a query with very large return set in a TSO environment, which DB2I2 runs.**
- DB2I2 RUN command write query result to an external file, unless you specify IN command option in online mode. IN command option returns query results directly back to your DB2I2 workbench edit session.
- By default, DB2I2 RUN output contains some heading and footing information to describe the result column information. You can use T=N command option to disable printing of both heading and footing. When you process DB2I2 command in batch mode, you can use T=N together with ODSN option to allow you to reuse the result from the RUN output (the output should contain no heading and footing information). The result can then be used by the following DB2I2 commands in sequence with IDSN command option.
- By specifying host variables in your query and dynamically substitute them during the run time, DB2I2 provides you with a query reusable environment. The format of the Host variable is &H-VARNAME-H=value all Upper case. You can code &H-VARNAME-H in your SQL statement, and dynamically substitute them by providing values at RUN time.
- With IDSN command option, you can specify the input to DB2I2 RUN command.
- Specify DLM=? Option to assign output filed delimiter. Default if not specified is an one byte blank.
- You can use –INC within your SQL block to include part of your query from external source. The most common use of –INC is in the WHERE predicates which can usually be generated with QBUILD command.
- In online mode, the result is default to BROWSE mode. Use EDIT=Y to switch to EDIT mode.
- If output length is less than 80 byte, LRECL is set to 80 for ODSN. If the ODSN is an existing sequential file, it is deleted and recreated. If the ODSN is a PDS and the length is less than the result length, then an error occurs.
- Use TRUNC=Y to truncate the result and fit it into work bench size or ODSN size.
- Use DURATION=Y option to display Starting Timestamp, Duration and Ending Timestamp information for the current RUN. This information can be used for Bench Mark purpose.
- Use BREAK=colname to set break based on ONE column. When BREAK is used, there will be a break line which list the total number of rows for each break.
- DB2I2 RUN command DOES NOT support the following datatype: LOB, XML.

Example

The example below demonstrates how to use Db2I2 RUN command to execute a block of SQL statements. Because there is no LIMIT option specified, DB2I2 uses default 300 rows as fetch limit. The WHERE predicates are from an external file named where.query.

```
Command ==> run                               Scroll ==> CSR
000011  TB JD00.T2                               T 326  DBTST001 STS00002
ss0012  SELECT A.COL1, A.COL2
000013  FROM JD00.T2 A
000014  WHERE
ss0015  -INC where.query
```

where.query dataset contains the following: COL1 LIKE 'SDT%'

```
Default fetch records selected: 300
** Number of Rows Returned: 9
***
```

The screen below displays result from previous DB2I2 RUN command with default 300 rows fetch limit.

```
BROWSE      JD00.DB2I2.RUNOUTPUT                Line 00000000 Col 001 049
Command ==>                                       Scroll ==> CSR
***** Top of Data *****
COL1      COL2
-----
SDT001   MASTER TEST
SDT002   TEST DETAIL
SDT003   TEST EXCEPTION
SDT004   EXCEPTION MASTER
SDT006   EXCEPTION SHIP TYPE
SDT007   TEST EXCEPTION STATEMENT
SDT008   TEST EXCEPTION RESPONSIBLE PARTY
SDT009   SUMMARY SETUP GROUP
SDT010   TEST EXCEPTION GROUP MONTHLY REPORT
```

The screen below uses fetch limit 2 rows for the same selected SQL statements to request only return 2 rows from the query result.

```
Command ==> run limit(2)                       Scroll ==> CSR
000011  TB JD00.T2                               T 326  DBTST001 STS00002
ss0012  SELECT A.COL1, A.COL2
ss0013  FROM JD00.T2 A
```

The screen below displays the result from previous DB2I2 RUN command. It shows that DB2I2 only return 2 rows as requested by LIMIT(2) option.

```
Command ==>                                       Scroll ==> CSR
***** Top of Data *****
COL1      COL2
-----
SDT001   MASTER TEST
SDT002   TEST DETAIL
***** Bottom of Data *****
```

The following example uses the IN option to return result from SELECT query directly back to your workbench for immediately reuse. Specify Host variable &H-V1-H to demonstrates how the host variable substitute works. When the following query is running, the host variable &V will be substituted by 'SDT003'.

```
Command ==> run limit(2) &H-V1-H='SDT003' in   Scroll ==> CSR
0ss011  SELECT A.COL1, A.COL2
000012  FROM JD00.T2 A
0ss013  WHERE A.COL1 < &H-V1-H
```

The result from the previous command is displayed below.

```
000011      SELECT  A.COL1, A.COL2
000012      FROM JD00.T2 A
000013      WHERE A.COL1 < &H-V1-H
000014 COL1      COL2
000015 -----
000016 SDT001  MASTER SDTR
000017 SDT002  SDTR  DETAIL
```

You can produce the same query result, by saving the same query in a file, and invoke it with `IDSN` command option. The following example has the query saved in a PDS file `saved.pds(query)`. Issue the following `DB2I2` command and command options to produce the same result as previous example.

```
RUN LIMIT(2) IDSN=saved.pds(query) IN &H-V1-H='SDT003'
```

RUNSTATS

Command Syntax:	RUNSTATS [[PARMUTIL=] 'runstats.parmutil.dsname'] [LISTDEF=listdef.dsname[(patt*)]] [OPTIONS=options.dsname]
Line objects allowed:	TS, TP, IX, IP
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use DB2I2 RUNSTATS command to generate DB2 RUNSTATS JCL for selected line objects.
- Use 'runstats.parmutil.dsname' command option to retrieve prepared DB2I2 RUNSTATS options from a file generated from a DB2I2 PARMUTIL RUNSTATS command.
- For DB2 v7 or above, you can specify LISTDEF option with PDS member pattern match. For example, By specifying LISTDEF=listdef(DB1*) will generate RUNSTATS DB2 utility JCL for all members name prefix with DB1. Because DB2I2 process each LISTDEF one at a time, If you select to use LISTDEF pattern option, it will prompt selection screen one for each LISTDEF member. To avoid this inconvenience, you can use PARMUTIL command to generate PARMUTIL control and then use it as process option.

Example

The example below demonstrates how to use DB2I2 RUNSTATS command to generate DB2 RUNSTATS JCL for Tablespace DBTST001.STS00001.

```
Command ==> RUNSTATS                               Scroll ==> CSR
S00018  TS  DBTST001.STS00001
000019  TS  DBTST001.STS00002
000020  -- Tablespace Process: DBTST001.STS00002
```

DB2I2 display the following RUNSTATS process options screen, so that you select your RUNSTATS options.

```
#RUNSTAA ----- DB2I2 RUNSTATS PROCESS OPTIONS -----
OPTION:  S  (S-TABLESPACE X-INDEX)
For TABLESPACE RUNSTATS
PART
TABLE      ALL_____ ALL/(TABLE-NAME,)
PROFILE    _____ (USE/DELETE/UPDATE/SET/SETFrom Existing Stats)
INCLUDE NPI _____ (Y/N)
TABLESAMPLE _____ (AUTO/0.##)      REPEAT TABLE _____
SAMPLE     25_          (1-100)
COLUMN     _____ ALL/Y-select columns
COLGROUP   _____ Y-Select Columns
INDEX      ALL_____ ALL/(INDEX-NAME PART,)
For INDEX RUNSTATS
INDEX      _____ ALL/(INDEX-NAME PART,)
PART
KEYCARD    Y (Y/N)
FREQUAL    N (Y/N)  NUMCOL  1__  COUNT 10__  FREQ TYPE MOST_ MOST/BOTH/LEAST
HISTOGRAM  N (Y/N)  NUMCOL  1__  NUMQUANTITIES 100
SHRLEVEL   REFERENCE REFERENCE/CHANGE REPORT NO_ NO/YES
UPDATE     ALL_____ ALL/ACCESSPATH/SPACE/NONE
HISTORY    NONE_____ ALL/ACCESSPATH/SPACE/NONE
FORCEROLLUP _____ NO/YES SORTDEV SYSALLDA SORTNUM ____ (2-255)
PF3=Exit  ENTER=Process Your Selection
```

Partial JCL results from the previous RUNSTATS command is displayed below.

```
000021 //          SYSTEM='DBAG',UID='DB2ADM01.STEP001',UTPROC=' '
000022 //* ----- **
000023 //* $LISTDD----- **
000024 //STPRIN01 DD SYSOUT=*
000025 //SYSPRINT DD SYSOUT=*
000026 //* $RUNSTAG----- **
000027 //SYSIN DD *
000028 RUNSTATS TABLESPACE DBTST001.STS00001
000029 TABLE (ALL)
000030 SAMPLE 25
000031 INDEX (ALL)
000032 KEYCARD
000033 )
000034 SHRLEVEL REFERENCE
000035 REPORT NO
000036 UPDATE ALL
000037 HISTORY NONE
000038 SORTDEVT SYSALLDA
```

RXDB2I2** A pricing add-on for JRH-DB2I2

Command Syntax:	RXDB2I2 [idsn=input.dsn] [idd=input.dd] [&Hostvar=hostvar]
Line objects allowed:	Any valid REXX Exec plus DB2I2 line objects between LINEOBJ [DDNAME=DB2I2RXL,DDNAME=ddname] and END_LINEOBJ plus Any valid DB2I2 command prefix with DB2I2REX
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	Process Interface between your REXX Exec and Db2i2 environment
Reusable:	Yes

Command Description

DB2I2 REXX Extender is a **pricing add-on** for JRH-DB2I2. It allows you

- To process your REXX exec directly from your ISPF Edit session within DB2I2 environment.
- To process DB2I2 commands directly from you REXX exec thru **DB2I2REX** interface. The line objects defined between **LINEOBJ** and **END_LINEOBJ** are dynamically allocated thru **DDNAME** option. If no DDNAME spacefied, DDNAME=DB2I2RXL is used as default..
- Stream line process in both ONLINE and BATCH process.
- Use RXDB2I2 to process dynamically generated command. There are only **ONE LEVEL of Nesting** RXDB2I2 is allowed
- Use &H-HOSTVAR-H=hostvar to pass host variables to RXdb2i2 script.
- Rxdb2i2 also allows you to use the FTP, TRACK and MAIL extended db2i2 commands.
- There are 3 extended REXX Functions are added when using Rxdb2i2:
 - RXNOTFND() Return Not Found Status from last command. The vlaue can be 'Y' for Not Found and " or 'N' for others
 - RXERROR() Return Error Status from last command. The vlaue can be 'Y' for Not Found and " or 'N' for others
 - RXERROR(N) To Set Error status to 'N'
 - RXRC() Return The RC or zispfrc from last Command
 - RXRC(?) Set Last RC to ? where ? must be Numeric

Example

The example below, use RXDB2I2 command to

1. Issue DB2I2 QBUILD command against 2 TS line objects defined between LINEOBJ and END_LINEOBJ thru IDD=DB2I2RXL
2. The output of QBUILD then feed into DB2I2 RUN command to select the last quiesce point of the selected TS objects. See next page for the content of the query.
3. The generated DB2I2 SETRBA command then being dynamically executed thru embedded RXDB2I2

4. The DB2I2 RECOVER command then uses the Incore RBA set by SETRBA command to generate DB2 Recovery Job
5. Followed with DB2I2 REBUILD Index command to generate DB2 Rebuild Indexes job steps.
6. The last line just used here to demonstrate a REXX Say inside selected lines

```
Command ==> rxdb2i2                               Scroll ==> CSR
SS0122 LINEOBJ DDNAME=db2i2rxl
000123 TS DDB2ADM.SDB2ADM   BP8K0  0    P N N N  16  2
000124 TS DDB2ADM.SMAP     BP0    0    A N N Y   4  8
000125 END_LINEOBJ
000126 db2i2rex "qbuild f1=dbname f2=tsname odsn=t1 idd=db2i2rxl"
000127 db2i2rex "run t=n idsn=pds.cntl(qq) odsn=t2"
000128 db2i2rex "rxdb2i2 idsn=t2"
000129 db2i2rex "recover idd=db2i2rxl pu(rcvrrbaa) odsn=t3"
000130 db2i2rex "rebuild idd=db2i2rxl pu(rbldall)",
000131 "          odsn=t3 append step#=10 jobcard=n"
SS0132 Say "It is all done in RXDB2I2!!!"
```

The content of query is display below

```
EDIT          DB2ADM.PDS.CNTL(QQ) - 01.04          Columns 00001 00072
Command ==>                                       Scroll ==> CSR
***** Top of Data *****
000001 Select 'db2i2rex setrba', Hex(max("START_RBA"))
000002 From "SYSIBM"."SYSCOPY"
000003 Where
000004 -inc t1
000005 And  ictype = 'Q'
```

Result after QBUILD command

```
EDIT          DB2ADM.T1                          Columns 00001 00072
Command ==>                                       Scroll ==> CSR
***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
000001 -- Begin of QBUILD --
000002 ( (DBNAME = 'DDB2ADM' AND TSNAME = 'SDB2ADM')
000003 OR (DBNAME = 'DDB2ADM' AND TSNAME = 'SMAP')
000004 )
000005 -- End   of QBUILD --
***** Bottom of Data *****
```

Result after RUN command

```
** Default fetch records selected: 300
** Number of Rows Returned: 1
**
BROWSE       DB2ADM.T2                          Line 00000000 Col 001 080
Command ==>                                       Scroll ==> CSR
***** Top of Data *****
db2i2rex setrba 0008042F53D7
** Incore RBA: 0008042F53D7 ***
```

Result after RECOVER command

```
EDIT          DB2ADM.T3                          Columns 00001 00072
Command ==>                                       Scroll ==> CSR
***** Top of Data *****
000001 //DB2ADM01 JOB (ACCT),'DB2I2 AD',
000002 //          NOTIFY=&SYSUID,COND=(0,NE),REGION=4M,
000003 //          CLASS=A,MSGCLASS=H
000004 /** $JOB CARD-----**
000005 /** -----**** DB2I2 DB2 RECOVER JCL GENERATION ****-----+
000006 /** DB2I2 for DB2 OS/390 and UDB zOS Version: 8.0
000007 /**          Release Date: 04/28/2011
000008 /** For - EVALUATION COPY
000009 /** Date - 11/04/29 Time:13:15 Creator:DB2ADM
000010 /** BY - JRH Golden State Software, Inc.
000011 /**          CopyRighted 1997-2011
```

```

000012 //*-----+
000013 //JOBLIB DD DISP=SHR,DSN=DSN810.SDSNLOAD
000014 // DD DISP=SHR,DSN=DSN810.SDSNEXIT
000015 // DD DISP=SHR,DSN=DSN810.RUNLIB.LOAD
000016 //DB2I2P JCLLIB ORDER=(DB2ADM.DB2I280.ISPFLIB,
000017 // DSN810.PROCLIB)
000018 /** $RECOVE$----- **
000019 /** ----- **
000020 //STEP001 EXEC DSNUPROC,COND=(4,LT),
000021 /** LIB='DSN810.SDSNLOAD',
000022 // SYSTEM='DB8G',UID='DB2ADM01.STEP001',UTPROC=' '
000023 /** ----- **
000024 /** $LISTDD----- **
000025 //SYSPRINT DD SYSOUT=*
000026 /** Copy taken at 2008-04-21 12.13.05 with RBA=0008042ED267
000027 //DD0001 DD DSN=DB2ADM1.DDB2ADM.SDB2ADM.P00000.BKUP.G0001V00,
000028 // DISP=OLD
000029 /** $RECOVEG----- **
000030 //SYSIN DD *
000031 RECOVER
000032 TABLESPACE DDB2ADM.SDB2ADM DSNUM ALL
000033 TABLESPACE DDB2ADM.SMAP DSNUM ALL
000034 TORBA X'0008042F53D7'
000035 LOCALSITE
***** ***** Bottom of Data *****

```

Result after REBUILD Index command

```

** No Runstats for TS DDB2ADM.SDB2ADM - (1,1) cylinders are used for all utility work files
** No Runstats for TS DDB2ADM.SMAP - (1,1) cylinders are used for all utility work files ***
.
.
000036 /** $REBUIL$----- **
000037 /** ----- **
000038 //STEP011 EXEC DSNUPROC,COND=(4,LT),
000039 /** LIB='DSN810.SDSNLOAD',
000040 // SYSTEM='DB8G',UID='DB2ADM01.STEP011',UTPROC=' '
000041 /** ----- **
000042 //SYSPRINT DD SYSOUT=*
000043 /** $LISTDD----- **
000044 //SYSUT1 DD DSN=DB2ADM.DB000273.PS000002.RBLD.SYSUT1,
000045 // DISP=(NEW,DELETE,CATLG),
000046 // UNIT=SYSALLDA,
000047 // SPACE=(CYL,(1,1))
000048 //UTPRINT DD SYSOUT=*
000049 /** $DYSORT----- **
000050 //DFSPARM DD *
000051 DYNALLOC=(SYSALLDA,5)
000052 /** $REBUILG----- **
000053 //SYSIN DD *
000054 REBUILD
000055 INDEX (ALL)
000056 TABLESPACE DDB2ADM.SDB2ADM
000057 SORTDEVT SYSALLDA
000058 SORTNUM 5
000059 /** $REBUIL$----- **
000060 /** ----- **
000061 //STEP012 EXEC DSNUPROC,COND=(4,LT),
000062 /** LIB='DSN810.SDSNLOAD',
000063 // SYSTEM='DB8G',UID='DB2ADM01.STEP012',UTPROC=' '
000064 /** ----- **
000065 //SYSPRINT DD SYSOUT=*
000066 /** $LISTDD----- **
000067 //SYSUT1 DD DSN=DB2ADM.DB000273.PS000008.RBLD.SYSUT1,
000068 // DISP=(NEW,DELETE,CATLG),
000069 // UNIT=SYSALLDA,
000070 // SPACE=(CYL,(1,1))
000071 //UTPRINT DD SYSOUT=*
000072 /** $DYSORT----- **
000073 //DFSPARM DD *
000074 DYNALLOC=(SYSALLDA,5)
000075 /** $REBUILG----- **
000076 //SYSIN DD *
000077 REBUILD
000078 INDEX (ALL)
000079 TABLESPACE DDB2ADM.SMAP
000080 SORTDEVT SYSALLDA
000081 SORTNUM 5

```


Result after REXX Say command

It is all done in RXDB2I2!!! ***

The following example, check for quiesce point with RXNOTFND(), if found then issue SETRBA command, otherwise set error off with RXERROR(N) and issue message and set RC=4 with RXRC(4).

```

LINEOBJ DDNAME=DB2I2RXL
Select 'db2i2rex setrba', Hex("START_RBA")
From "SYSIBM"."SYSCOPY"
Where
    DBNAME = '&DB'
And ictype = 'Q'
Order by 2 desc
Fetch first 1 row only
END_LINEOBJ
DB2I2REX "RUN IDD=DB2I2RXL ODSN=T2 &&DB=ABC T=N"
If RxNOTFND() \= 'Y' Then
    DB2I2REX "RXDB2I2 IDSN=T2"
Else
Do
    x = RxError(N)
    Say " ** No Quiesce Point Found for selected Database"
    Say " ** Quiesce Point is Not Set"
    x = Rxrc(4)
    Return 4
End

```

Please See DB2I2.SAMPLIB(@README) for a list of user defined functions which utilize Rxdb2i2.

Member	Description
-----	-----
appl	Application Analysis Demo
copyftss	Build COPY jobs with Real Time Statistics
copyitss	Build Incremental/Merge Copy with Real Time Stats
datamove	Migration & Data Move
diag	Provide a PG or DM with Stmt# to generates Diaganos Information To use this script: 1. SetUp a global variable like the one below GV &DIAG=RXDB2I2 IDSN='your.samplib.lib(DIAG)' 2. Issue the following from command line &DIAG &input=PG loc.cl.pg.ver stmt# &output=diag.output
dscopy	Migrate & DSCOPY to build DSN1COPY with OBID
dscopy2	Build DSCOPY from one location to another
dscopy21	Build DSCOPY from one location to another
flashcpy	Migrate with same OBID, Flash Copy and then
ftpmail	FTP & Mail with Wait Copy Runstats information with SELPATHV & SELPATHU
misc	Misc (Copy , Delete Pds members)
rcvrfull	Recover to Last Full Copy
rcvrdba	Recover to RBA point - Quiesce point
reorgixs	Build Online Reorg IP jobs with Real Time Stats
reorgtss	Build Online Reorg TP jobs with Real Time Stats With Create and Drop Mapping table DDL
rinfo	Real Time Statistics Information
rreorgip	Build Online Reorg IP jobs with Real Time Stats
rreorgtp	Build Online Reorg TP jobs with Real Time Stats
rreorgt2	Build Online Reorg TP jobs with Real Time Stats With Create and Drop Mapping table DDL
runsixs	Build Runstats IX jobs with Real Time Stats
runstss	Build Runstats TS jobs with Real Time Stats
track01	Track Demo with Mail
track02	Track Demo with Mail Attachment
track03	Track Demo with Mail Attachment and Db2 Insert
tracksql	Track Demo sql select
trackddl	Track DB2 Interface DDL
tuning	Application Analysis
util	Real Time Statistics & Recovery

SDSF

Command Syntax:	SDSF SDSF-primary-menu-option SDSF ODSN= [DDNAME=dd1,dd2...]	in online mode to interface to SDSF or to spool job output queue use DDNAME option to spool only selected DDNAME.
Line objects allowed:	JI	
Process Mode:	Online and Batch	
Support		
Multiple line object:	Yes	
Multiple type of line object:	No	
Wild Card % allowed:	No	
Output Type:	Interface to SDSF output queue or spool output queue to a output file	
Reusable:	No	

Command Description

- Use DB2I2 SDSF command in online mode to interface to SDSF. Each Job Information line object contains Job name and Job number information. DB2I2 uses this information to directly interface with SDSF to pull job output queue information.
- SDSF-primary-menu-option is the field indicates how you access your SDSF menu. For example, the example below has S defined on the ISPF primary menu, then use S. If you access your SDSF menu with =U.S, then specify U.S as SDSF primary-menu-option.
- When you generate massive number of jobs to run concurrently, you can use DB2I2LOG TSO command to insert job track steps into you generated JCL.
- You can then run a select query to select only those jobs with status 'A', which indicates it is an ABEND job.
- The output from SQL RUN can then generates JI line objects, which can be used with DB2I2 SDSF command to interface to SDSF and identify and diagnoses problems.
- If you submit DB2I2 SDSF command in batch mode, or specify ODSN command option with SDSF command, DB2I2 then spool all the selected jobs output queue information into a dataset.
- Specify DDNAME=dd1,dd2.. to spool only those selected DDNAME output.
- Specify JI jobnm.* to list all jobnm in job queue.

Example

The example below demonstrates how to use DB2I2 SDSF command to pull job queue information for the selected 2 job information lines. Assuming the ISPF primary screen has SDSF defined as S. (or =S gives you the SDSF menu)

```

ISPF Primary Option Menu
Option ==>
0 Settings      Terminal and user parameters,
1 View          Display source data or listings,
2 Edit          Create or change source data,
3 Utilities     Perform utility functions,
4 Foreground    Interactive language processing,
5 Batch         Submit job for language processing,
6 Command       Enter TSO or Workstation commands,
7 Dialog Test   Perform dialog testing,
8 LM Facility   Library administrator functions,
9 IBM Products  IBM program development products,
10 SCLM         SW Configuration Library Manager,
11 Workplace    ISPF Object/Action Workplace,
S SDSF         System Display and Search Facility

```

DB2I2 Reference Manual

Then issue the following SDSF command with S in online mode, gives you access directly in SDSF output queue.

```
Command ==> sdsf S                               Scroll ==> CSR
S20239 JI JRHJ0001.JOB12345
000240 JI JRHJ0001.JOB12350
```

The screen below displays the result from previous DB2I2 SELECT command.

```
SDSF STATUS DISPLAY ALL CLASSES                   CHARS 'JRHJ0001' FOUND
COMMAND INPUT ==>                                SCROLL ==> CSR
PREFIX=JRHJ*  DEST=(ALL) OWNER=*  SORT=POS/A
NP  JOBNAME  JOBID  OWNER  PRTY  QUEUE  C  POS  SAFF  ASYS  STATUS
   JRHJ0001  JOB12345  JRHJ      1  PRINT      F  5342
```

Press =x or press PF3 few times gives you the next job information.

```
SDSF STATUS DISPLAY ALL CLASSES                   CHARS 'JRHJ0001' FOUND
COMMAND INPUT ==>                                SCROLL ==> CSR
PREFIX=JRHJ*  DEST=(ALL) OWNER=*  SORT=POS/A
NP  JOBNAME  JOBID  OWNER  PRTY  QUEUE  C  POS  SAFF  ASYS  STATUS
   JRHJ0001  JOB12350  JRHJ      1  PRINT      F  5348
```

Specify the following command to spool the job output queue for the selected jobs into a output file R10.

```
Command ==> sdsf odsn=r10                         Scroll ==> CSR
S20239 JI JRHJ0001.JOB12345
000240 JI JRHJ0001.JOB12350
```

```
BROWSE  JRHJ.R10                                Line 00000000 Col 001
Command ==> ,                                  Scroll ==>
***** Top of Data *****
==> JOBNAME: JRHJ0001 JOB Number: JOB12345
      J E S 2  J O B  L O G  --  S Y S T E M  B 0 0 1  --  N

14.35.10 JOB06047 ---- MONDAY,    23 APR 2011 ----
14.35.10 JOB06047 IRR010I  USERID JRHJ    IS ASSIGNED TO THIS JOB.
14.35.10 JOB06047 ICH70001I JRHJ      LAST ACCESS AT 14:33:55 ON MONDAY, A
14.35.10 JOB06047 $HASP373 JRHJ0001 STARTED - INIT 13  - CLASS F - SYS B
14.35.10 JOB06047 IEF403I JRHJ0001 - STARTED - TIME=14.35.10
      .
      .
==> JOBNAME: JRHJ0001 JOB Number: JOB12350
      J E S 2  J O B  L O G  --  S Y S T E M  B 0 0 1  --  N

14.37.10 JOB12350 ---- MONDAY,    23 APR 2011 ----
14.37.10 JOB12350 IRR010I  USERID JRHJ    IS ASSIGNED TO THIS JOB.
14.37.10 JOB12350 ICH70001I JRHJ      LAST ACCESS AT 14:35:55 ON MONDAY, A
14.37.10 JOB12350 $HASP373 JRHJ0001 STARTED - INIT 13  - CLASS F - SYS B
14.37.10 JOB12350 IEF403I JRHJ0001 - STARTED - TIME=14.37.10
```

SELECT

Command Syntax:	SELECT [PRE=corr-name] [JOIN=N Y LEFT RIGHT FULL [TYPE=RI MATCH]]
Line objects allowed:	TB, AL, SY, VW, MT TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	SQL SELECT Skeleton
Reusable:	Yes

Command Description

- Use DB2I2 SELECT command to generate SELECT SQL for the selected line object.
- DB2I2 produces all DB2 column information for the selected line objects.
- Use PRE option to assign correlation name. If not specify, there will be no correlation name used.
- Use JOIN option to assign join type. The option can be N,Y,LEFT,RIGHT,FULL. If not specified, default is N.
- Use TYPE option to choose the type of join. Either join with RI or join with column match. If not specify, default to RI.

Example

The example below demonstrates how to use DB2I2 SELECT command to generate SELECT SQL which join 2 tables with match column join.

```
Command ==> select join=y type=match                               Scroll ==> CSR
S20175 TB DB2ADM.PLAN_TABLE
000176 TB DB2ADM.DSN_STATEMNT_TABLE          T    16   DDB2ADM  SDB2ADM
```

The screen below displays the result from previous DB2I2 SELECT command.

```
000177 SELECT T1."QUERYNO", T1."QBLOCKNO", T1."APPLNAME"
000178         , T1."PROGNAME", T1."PLANNO", T1."METHOD", T1."CREATOR"
.
.
000201         , T2."TOTAL_COST"
000202 FROM
000203     "DB2ADM"."PLAN_TABLE" T1
000204 INNER JOIN "DB2ADM"."DSN_STATEMNT_TABLE" T2
000205     ON T1."QUERYNO" = T2."QUERYNO"
000206     AND T1.APPLNAME = T2.APPLNAME
000207     AND T1.PROGNAME = T2.PROGNAME
000208     AND T1.COLLID = T2.COLLID
000209     AND T1.GROUP_MEMBER = T2.GROUP_MEMBER
000210 WHERE
000211     FETCH FIRST 1 ROWS ONLY
000212     OPTIMIZE FOR 1 ROWS
```

SELPATHU

Command Syntax:	SELPATHU
Line objects allowed:	CU, CI
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	SQL INSERT or UPDATE Skeleton
Reusable:	Yes

Command Description

- Use DB2I2 SELPATHU command to generate update or insert SQL statements from a CU-catalog update line or a CI-catalog insert line. These update or insert SQL statement can be used to update the DB2 catalog statistics for either proto-type your design or use them to influence the access path selection of the DB2 optimizer.
- You can use DB2I2 SELPATHV command to generate CU and CI line objects.
- Input LRECL should be more than 250 bytes long, otherwise an error occurs.

Example

The example below demonstrates how to use DB2I2 SELPATHU command to generate update SQL statement for a CU line.

```
Command ==> SELPATHU                               Scroll ==> CSR
000002 TB JD00.TTBL01
000003 -- - CATALOG TABLE -- DBNAME.TSNAME----- NACTIVE
s000004 CU SYSTABLESPACE      DJDOO.SJDOO      540
```

The screen below displays the result from previous DB2I2 SELPATHU command.

```
Command ==>                                         Scroll ==> CSR
000002 TB JD00.TTBL01
000003 -- - CATALOG TABLE -- DBNAME.TSNAME----- NACTIVE
000004 CU SYSTABLESPACE      DJDOO.SJDOO      540
000005 UPDATE SYSIBM.SYSTABLESPACE SET NACTIVE = 540
000006          WHERE DBNAME = 'DJDOO' AND NAME = 'SJDOO';
```

SELPATHV

Command Syntax:	SELPATHV [<u>OPTION=REPORT</u>][CUCI]
Line objects allowed:	TB, MT
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	N/A
Wild Card % allowed:	Yes
Output Type:	Report or line object CU
Reusable:	Yes

Command Description

- Use DB2I2 SELPATHV command to display access path statistics which optimizer uses to determine access path for a SQL statement..
- By default, DB2I2 SELPATHV command generates access path report with default command option OPTION=REPORT, which displays the access path statistics in report format. Specify command option OPTION=CUCI to request DB2I2 to return CU-catalog table update lines or CI-catalog table insert lines.
- You can make changes to these CI and CU lines and use SELPATHU to generate UPDATE or INSERT SQL to update those DB2 catalog tables.
- Output for SELPATHV need to be an file which has LRECL > 250, otherwise an error occurs.

Example

The example below demonstrates how to use DB2I2 SELPATHV command to generate update SQL statement for a CU line.

```
Command ==> SELPATHV OPTION=CUCI                               Scroll ==> CSR
s00002 TB JD00.TTBL01
```

The screen below displays the result from previous DB2I2 SELPATHV command.

```
Command ==>
***** ***** Top of Data *****
000001 TB JD00.TTBL01
000002 -- - CATALOG TABLE -- DENAME.TSNAME----- NACTIVE
000003 CU SYSTABLESPACE      DJD00.SJD00      540
000004 -- - CATALOG TABLE -- TBCREATOR.TBNAME----- CARD          NPAGES          PCTROWCOMP
000005 CU SYSTABLES          JD00.TTBL          4279          143          64
000006 -- - CATALOG TABLE -- TBCREATOR.TBNAME----- COLNAME          COLCARD          HIGH2KEY  LOW2KEY
000007 CU SYSCOLUMNS        JD00.TTBL          BILL_LADE_NO     336          978-5385  AAH97-33
000008 CU SYSCOLUMNS        JD00.TTBL          CO_CD            1            11010     11010
000009 CU SYSCOLUMNS        JD00.TTBL          CRET_DATE_TIME  1952          q          q          1
000010 -- - CATALOG TABLE -- IXCREATOR.IXNAME  FIRSTKEYCARD     FULLKEYCARD     NLEAF          NLEVELS  CLUSTERRATIO
000011 CU SYSINDEXES         JD00.XN3B005     1            4279          29          2          65
000012 -- - CATALOG TABLE -- TBOWNER.TBNAME   COLNAME          FREQUENCY        COLVALUE
000013 CU SYSCOLDIST         JD00.TTBL          RCV_CMPLT_DT     306          q
000014 CU SYSCOLDIST         JD00.TTBL          RCV_CMPLT_DT     329          q
000015 CU SYSCOLDIST         JD00.TTBL          RCV_CMPLT_DT     334          q
```

SETG

Command Syntax:	SETG
Line objects allowed:	GV
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	N/A
Reusable:	N/A

Command Description

- Use DB2I2 SETG command to set global variables.
- Once you set global variables with SETG command, all the commands executed after the SETG command will be checked to see if there are any global variables, which can be substituted. The commands with global variable defined are substituted with the value defined in the SETG command.
- There are few occasions where this command is useful:
 - In online mode, allows you to run a DB2I2 command with command option exceed the 48 character length limitation on the command line.
 - Allows you to save some of the commonly used command option and reuse it.
 - Allows you to customize your own work bench environment. For example, you can define host variable &Q for your standard query library, whenever you need to run query from you standard query library, you simply just use command option IDSN=&Q(queryname) to access it.

Example

The following example demonstrates how to use DB2I2 SETG to set a global variable &MIGROPT, and use it later with a DB2I2 MIGR command to avoid the length limitation problem:

```

Command ==> SETG                               Scroll ==> CSR
S00001 GV &MIGROPT=AL=Y VW=Y SY=Y BIND=Y GRANT=Y RI=B

Command ==> MIGR &MIGROPT                       Scroll ==> CSR
000001 GV &MIGROPT=AL=Y VW=Y SY=Y BIND=Y GRANT=Y RI=B
S00002 TS TESTDB.TESTTS

```

You can save the GV-global variable lines in a file and retrieve them whenever you need to set these global variables. The following example demonstrates how to use SETG with IDSN option to retrieve a set of pre-saved global variables:

```

Command ==> SETG idsn=global.file               Scroll ==> CSR

```

The following example demonstrates how you can customize your own DB2I2 environment:

```

Command ==> SETG                               Scroll ==> CSR
S40001 GV &Q=my.query.lib
000002 GV &I=my.input.lib
000003 GV &O=my.output.lib
000004 GV &Q=my.query.lib

```

SETRBA

Command Syntax:	SETRBA [12 position hex RBA] [+n -n] [T=I B]
Line objects allowed:	any edit line with valid RBA field
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	N/A

Command Description

- Use DB2I2 SETRBA command to set an INCORE RBA with T=I option or RESTOREBEFORE RBA with T=B. Once the RBA is set, you can use it as the point of recover for the RECOVER TO RBA or RECOVER TO LOGPOINT with INCORE RBA options. Use RECOVER with RESTOREBEFOER to use RESTOREBEFORE RBA. The default is T=I if not specified.
- There are two ways to set an RBA. You can either specify a 12 position Hex RBA as command option or **position the cursor on a valid RBA field** within your workbench edit session and issue the SETRBA command. This RBA field can be from an output of a query run or any sources.
- To set the RBA in batch mode, you have to use 12 position hex RBA command option
- Use +n or -n command option to add or subtract a integer number from the selected RBA.

Example

The following example shows you how to use SETRBA command on a recover to RBA option selection screen. The INCORE RBA is set to X'00E6B559E52D'. when you position your cursor on the X'00E6B559E52D' field and issue DB2I2 SETRBA command.

```

Command ==> SETRBA                               Scroll ==> CSR
=NOTE= T Type
==MSG> T:F:Full Copy          I:Increment Copy          P:Recover tocopy/torba
==MSG> R:load replace log(yes) S:Load Replace log(no) W:Reorg log(no)
==MSG> X:reorg log(yes)       Y:Load Resume log(no)  Z:Load Resume log(yes)
==MSG> T:Term utility         Q:Quiesce
==MSG> S=STYPE
==MSG> :DB2 Image Copy       C:DFSMS Concurrent Copy
==MSG> R:Load Replace(Yes)    S:Load Replace(No)
==MSG> W:Reorg Log(NO)       S:Reorg Log(Yes)
=NOTE= Dsnum
==MSG> 0=TableSpace Level    Otherwise=Partition Level
=NOTE= -Start RBA---TS -date- -time- DSNUM DSNAME-----
000001 00E6B559E52D Q 971222 142519 0 DBTST001.STS00001
000002 00E6B55D4CB1 Q 971222 142814 0 DBTST001.STS00001
000003 00E6B55A3EDA Q 971222 142604 0 DBTST001.STS00001

```


SHAUTH

Command Syntax:	SHAUTH [GRANTEE GRANTOR
Line objects allowed:	SH,US RO for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 SHAUTH command to display authorization information of selected schema or user.
- Use GRANTEE or GRANTOR command option to choose whether you want to display GRANTEE or GRANTOR authorization information of the selected line objects. The default option if not specified is GRANTEE.

SNAPSHOT

Command Syntax:	SNAPSHOT pgno[,pgno] [DSN1PRNT options]
Line objects allowed:	TS, TP, IX, IP, IS, ISP TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Batch ONLY
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 SNAPSHOT command to generate DSN1PRNT snapshot for a select page or a range of pages of selected line objects.
- You can use any valid DSN1PRNT options on the DSN1PRNT options filed except the NUMPARTS option, which is derived from the selected line object.
- Common DSN1PRNT option: FORMAT to format the output
EXPAND to decompress the compressed tablespace
SWONLY to decompress the EDITPROC compressed tablespace

Example

The following example demonstrates how to use DB2I2 SNAPSHOT command, with FORMAT and EXPAND options specified, to print 2 data pages, page 50 and 51, of a compressed tablespace.

```
Command ==> snapshot 50,51 format,expand          Scroll ==> CSR
S00101 TS DJD00.SJD00
```

The screen below displays partial result from previous DB2I2 SNAPSHOT command.

```
Command ==>          Scroll ==> CSR
***** Top of Data *****
DSN1999I START OF DSN1PRNT FOR JOB DB2ADM      AHM$$U1 $$TSU1
DSN1989I DSN1PRNT IS PROCESSED WITH THE FOLLOWING OPTIONS:
          4K/NO IMAGECOPY/NUMPARTS = 0/  FORMAT/  EXPAND/NO SWONLY/  PRINT/
DSN1998I INPUT  DSNAME = TDB2.DSNDBC.DJD00.SJD00.I0001.A001          , VSAM

DATA PAGE:  PGCOMB='00'X  PGLOGRBA='014CD3968E0F'X  PGNUM='00000050'X  PGFLAGS='
            PGFREE='01F8'X  PGFREEP=3622  PGFREEP='0E26'X  PGHOLE1='0014'X  PGMA
            PGNANCH=34
PGTAIL:  PGIDFREE='00'X  PGEND='E'
ID-MAP FOLLOWS:
01 0D56 0C86 0BB6 0AE6 0A16 0946 0876 07A6
09 06D6 0606 0536 0466 0396 02C6 01F6 0126
11 0056

HOLE: XOFFSET='0014'X  LENGTH=66
80004200 0001D850 AA0580E2 0A4A7E80 30400F60 F4F1F8A9 E0ECA87D 503B05E0 .....
32F6CEFC 0C10D2EC 47D9004A F10D9FCA E0FEC20F 49190113 64703011 36470301 .6....
```

SP

Command Syntax:	SP
Line objects allowed:	TB, VW, PL, PG, MT, SQ TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	SP stored procedure line object
Reusable:	Yes

Command Description

- Use DB2I2 SP command to generate DB2 SP – stored procedure line for selected line objects.

Example

The following example demonstrates how to use DB2I2 SP to list all Stored Procedures for collection ID - NULLID.

```

Command ==> sp
s00387 pg .NULLID.%.%

Command ==>
000387 pg .NULLID.%.%
000388 SP SYSPROC.XSR_ADDSCHEMADOC NULLID.SQLC2F0A
000389 SP SYSPROC.XSR_COMPLETE NULLID.SQLC2F0A
000390 SP SYSPROC.XSR_REGISTER NULLID.SQLC2F0A
000391 SP SYSPROC.XSR_ADDSCHEMADOC NULLID.SQLC3F09
000392 SP SYSPROC.XSR_COMPLETE NULLID.SQLC3F09
000393 SP SYSPROC.XSR_REGISTER NULLID.SQLC3F09
000394 SP SYSPROC.XSR_ADDSCHEMADOC NULLID.SQLC4F09
000395 SP SYSPROC.XSR_COMPLETE NULLID.SQLC4F09
000396 SP SYSPROC.XSR_REGISTER NULLID.SQLC4F09
000397 SP SYSPROC.XSR_ADDSCHEMADOC NULLID.SQLC5F09
000398 SP SYSPROC.XSR_COMPLETE NULLID.SQLC5F09
000399 SP SYSPROC.XSR_REGISTER NULLID.SQLC5F09

```

SPACE

Command Syntax:	SPACE
Line objects allowed:	TB, IX or TB tcreator.tbname rowcount %compressed IX ixcreator.ixname rowcount keycard
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 SPACE command to estimate space requirement for selected Tables or Indexes.
- If a regular TB line is selected, DB2I2 displays a table space estimation screen which allows you to change the Segment Size, PCT Free, Free Page, Page Size, PCT Row Compressed and Card(F) for a selected table to see the space requirement for those changes. If the table selected is compressed, a default of 30% will be used in the PCT Row Compressed field. Otherwise, the PCT Row Compressed field is set to 0.
- If a regular IX line is selected, DB2I2 displays a index space estimation screen which allows you to change the Unique, PCT Free, Free Page, Key Card, Row Count for a selected index to see what is the space requirement for those changes.
- If you enter a TB line with row count and %compressed line object option, or a IX line with row count and keycard line object option, DB2I2 bypass the estimation screen display and automatically calculate the space requirement based on the catalog statistics and line object option specified.
- DB2I2 generates space estimation report for each selected line object, which display space requirement in K or cylinders as well as accumulated total number of cylinders for all the selected line objects.

Example

The example below uses DB2I2 SPACE command to estimate a DB2 table and its index.

```
Command ==>> space                               Scroll ==>> CSR
S20066 TB PRIP.NOTEPAD
000067 IX PRIP.IXNPD1
```

Once you select the line objects and press Enter, the following screen is displayed which allows you to change the displayed parameters and estimate the space requirement for those changes.

DB2I2 Reference Manual

```
#SPACET ----- DB2I2 DB2 TABLE SPACE ESTIMATATION PROCESS OPTIONS -----  
  
Table Name:  PRIP.NOTEPAD  
Rec Lnegth: 135  
  
Segment Size  __          PCT Free          10_  Free Page  0__  
Page Size    4_          Compression Ratio 30_  Card(F)   152161719_____  
  
----- Estimation Results -----  
  
Average Record Size  
Usable Page Size  
Rows Per Page  
Pages Used  
Total Pages  
  
Total KBytes  
Total 3390 Tracks  
Total 3390 Cylinders  
  
PF3=Exit  ENTER=Process Your Selection
```

Make changes for the display options, or simple pressing enter key to accept the displayed options. The screen below displays all the calculated fields in gray color.

```
#SPACET ----- DB2I2 DB2 TABLE SPACE ESTIMATATION PROCESS OPTIONS -----  
  
Table Name:  PRIP.NOTEPAD  
Rec Lnegth: 135  
  
Segment Size  __          PCT Free          10_  Free Page  0__  
Page Size    4_          Compression Ratio 30_  Card(F)   152161719_____  
  
----- Estimation Results -----  
  
Average Record Size 135  
Usable Page Size    3666  
Rows Per Page       27  
Pages Used          5635622  
Total Pages         5635638  
  
Total KBytes        15779786  
Total 3390 Tracks   328746  
Total 3390 Cylinders 21917  
  
PF3=Exit  ENTER=Process Your Selection
```

After you make the changes to the onscreen parameters and presses enter key, the bottom part of the screen displays the estimation results in detail.

Press PF3 key to continue processing next line object in sequence.

DB2I2 Reference Manual

#SPACEI ----- DB2I2 DB2 INDEX SPACE ESTIMATATION PROCESS OPTIONS -----

Index Name: PRIP.IXNPD1
Key Length: 23

Unique Y (Y/N) PCT Free 10_ (0-99) Free Page 0__ (0-99)
Key Card 152161719_____ Row Count 152161719_____

----- Estimation Results -----

Total Leaf Pages	Entries per Leaf page
Total NonLeaf Pages	Entries Per NonLeaf Page
Total Active Pages	Level 2 Pages
Total Free Pages	Level 3 Pages
Space Map Pages	Level 4 Pages
Total Index Pages	Level 5 Pages

Total KBytes
Total 3390 Tracks
Total 3390 Cylinders

PF3=Exit ENTER=Process Your Selection

#SPACEI ----- DB2I2 DB2 INDEX SPACE ESTIMATATION PROCESS OPTIONS -----

Index Name: PRIP.IXNPD1
Key Length: 23

Unique Y (Y/N) PCT Free 10_ (0-99) Free Page 0__ (0-99)
Key Card 152161719_____ Row Count 152161719_____

----- Estimation Results -----

Total Leaf Pages	1257535	Entries per Leaf page	121
Total NonLeaf Pages	10394	Entries Per NonLeaf Page	121
Total Active Pages	1267929	Level 2 Pages	10308
Total Free Pages	0	Level 3 Pages	85
Space Map Pages	156	Level 4 Pages	1
Total Index Pages	1268088	Level 5 Pages	0

Total KBytes 5072352
Total 3390 Tracks 105675
Total 3390 Cylinders 7046

PF3=Exit ENTER=Process Your Selection

Once you finish all the changes, a summary report, which contains space estimation for each selected line object as well as total space requirements are calculated.

Command ==>

Scroll ==> CSR

```

000073      FOR FETCH ONLY
000074 TB  PRIP.NOTEPAD
000075 IX  PRIP.IXNPD1
000076 -- Table Name          Reclen Segsz %Free Frpg PgSize Rowcomp To
000077 TB PRIP.NOTEPAD        135      10    0    4    30    15
000078 -- Index Name          keylen Uniq  %Free Frpg Keycard   Ro
000079 IX PRIP.IXNPD1         23      Y    10    0   152161719  15
000080 -- Total Spaces required: 20852138 K bytes 434421 tracks 28963 Cylinders

```

The following example demonstrates how to **bypass selection screen** and use **row count and %compressed** on a TB line and **row count and key card** on an IX line to estimate space. All the other space calculation parameters are retrieved from DB2 catalog table.

```
Command ==> SPACE                               Scroll ==> CSR
S200081 TB PRIP.NOTEPAD      2000000 40      row count and % compressed
000082 IX PRIP.IXNPD1      2000000 2000000    row count & keycard
```

```
Command ==>                                       Scroll ==> CSR
000081 TB PRIP.NOTEPAD      2000000 40
000082 IX PRIP.IXNPD1      2000000 2000000
000083 -- Table Name          Reclen Segsz %Free Frpg PgSize Rowcomp To
000084 TB PRIP.NOTEPAD          135    0    10    0    4    40    20
000085 -- Index Name          keylen Uniq %Free Frpg Keycard    Ro
000086 IX PRIP.IXNPD1           23     Y    10    0    2000000    20
000087 -- Total Spaces required: 244519 K bytes 5095 tracks 341 Cylinders
```

SPACEADJ

Command Syntax:	SPACEADJ [CYL TRK]
Line objects allowed:	AI,AT
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	TP or IP line with ALLOC line object option
Reusable:	No

Command Description

- Use DB2I2 SPACEADJ command to generate TP or IP lines with ALLOC line object option. This format of TP or IP line object is suitable to use with DB2I2 DSADJ command to actually generate JCL to adjust space allocation. The line object required for DB2I2 SPACEADJ commands are AI for Adjust Index partition line object and AT for Adjust Table partition line object.
- You can specify roundup command option CYL to adjust space allocation to cylinder boundary, or TRK to adjust space allocation to track boundary.
- The format of an AI line is:
AI ixcreator.ixname partno rowcount t keycard
- You can use system UDQ-user defined query **GENAI** to generate AI line object. To invoke GENAI, enter
 RUN IDSN=*GENAI &IXC='ixcreator' &IXN='ixname'
- The format of an AT line is:
AT dbname.tsname partno tcreator.tbname rowcount %compressed
- You can use system UDQ-user defined query **GENAT** to generate AT line object. To invoke GENAT, enter
 RUN IDSN=*GENAT &DB='dbname' &TS='tsname' &COMPRATIO=%compressed
- DB2I2 SPACEADJ command generates TP or IP lines with line option
 ALLOC=(alloc_type,primary,secondary) calculated based on the row count for a AT line and row count and key card for an AI line.

Example

The following example demonstrates how to generate an AT line for a tablespace dbpmit01.tsnpd. Because the command option is longer than the available to issue RUN command with all the options, we use SETG command to set &opt as a global variable &opt.

```

Command ==> setg                               Scroll ==> CSR
***** Top of Data *****
s00001 GV &opt=&db='DBPRIT01' &ts='TSNPD' &compratio=30

EDIT      DB2ADM.T41                             Global Var &OPT is Repla
Command ==>                                       Scroll ==> CSR
***** Top of Data *****
000001 GV &opt=&db='DBPRIT01' &ts='TSNPD'&compratio=30

```


We then issue RUN command to run an UDG-GENAT to generate AT line objects.

```

EDIT          DB2ADM.T41                      Global Var &OPT is Repla
Command ==>  run idsn=*genat in &opt          Scroll ==> CSR
*****      ***** Top of Data *****
000001 GV &opt=&db='DBPRIT01' &ts='TSNPD'
    
```

Result from the previous query run displayed below. It shows row count around 332009 with a default 30% compression ratio.

```

EDIT          DB2ADM.T41                      Columns 00001 00072
Command ==>                                     Scroll ==> CSR
*****      ***** Top of Data *****
000001          PARTITION                      CARD
000002 -----
000003 AT DBPRIT01.TSNPD 0          PRIT.NOTEPAD          332009          30
000004 GV &opt=&db='DBPRIT01' &ts='TSNPD'
    
```

Once you generate the AT line object from Run command, you can then issue SPACEADJ command to generate line objects which can be used with DSADJ to adjust space allocation for the selected line objects.

```

Command ==>  spaceadj                          Scroll ==> CSR
*****      ***** Top of Data *****
000001          PARTITION                      CARD
000002 -----
S00003 AT DBPRIT01.TSNPD 0          PRIT.NOTEPAD          332009          30
000004 GV &opt=&db='DBPRIT01' &ts='TSNPD'
    
```

The output from SPACEADJ command is a TP or an IP line object with line object option ALLOC=(alloc_type,pri,sec) calculated.

```

EDIT          DP0022.T41                      Columns 00001 00072
Command ==>                                     Scroll ==> CSR
*****      ***** Top of Data *****
000001          PARTITION                      CARD
000002 -----
000003 AT DBPRIT01.TSNPD 0          PRIT.NOTEPAD          332009          30
000004 TP DBPRIT01.TSNPD 0 ALLOC=(PAGE,24598,2459)
    
```

The result displayed below shows it requires 24598 4k pages of primary space allocation to accommodate for 332009 rows with 30% compression ratio. You can then use DB2I2 DSADJ command to generate JCL to run the actual space adjustment. The following screen select DSADJ command with MOVE=N option.

```

Command ==>  dsadj move=n                      Scroll ==> CSR
*****      ***** Top of Data *****
000001          PARTITION                      CARD
000002 -----
000003 AT DBPRIT01.TSNPD 0          PRIT.NOTEPAD          332009          30
S00004 TP DBPRIT01.TSNPD 0 ALLOC=(PAGE,24598,2459)
000005 GV &opt=&db='DBPRIT01' &ts='TSNPD'
    
```

The result from DSADJ command displayed below shows the generated ALTER primary and secondary statement.

```

Command ==>
000014 /* ----- Scroll ==> CSR
000015 /** ADJUST PRIMARY/SECONDARY FOR STORAGE GROUP ALLOCATION **/
000016 /** ----- **/
000017 //ALTRD1 EXEC PGM=IKJEFT1B,DYNAMNBR=20,COND=(4,LT)
000018 //SYSPRINT DD SYSOUT=*
000019 //SYSUDUMP DD SYSOUT=*
000020 //SYSTSPRT DD SYSOUT=*
000021 //SYSTSIN DD *
000022 DSN SYSTEM(DB2D)
000023 RUN PROGRAM(DSNTIAD) PLAN(DB2I2A)
000024 //SYSIN DD *
000025 ALTER TABLESPACE DBPRIT01.TSNPD
000026 PRIQTY 98392 SECQTY 9836 ;
000027 /*

```

The following example demonstrates how to process similar step in batch mode to adjust space for index. Use BATCH command to generate batch JCL.

```

Command ==> batch setg
S00001 GV &OPT=&IXC='PRIT' &IXN='IXNPD1'

```

After the JCL is generated, enter the following gray displayed lines:

Line 20 to run UDQ-GENAI to generated AI line. Use T=N RUN command option to disable heading Printing so that you can reuse the output. Use &OPT to substitute global variable &OPT set by SETG command. Save the output in file T42.

Line 21 to run SPACEADJ command use T42 as input as save the output in file T43.

Line 22 to run DSADJ command use T43 as input and generate ALTER DB2 SQL and save the result in File T44.

Line 23 uses FLIST to print the result in T44

```

Command ==>
000014 /* ----- Scroll ==> CSR
000015 //STEP001 EXEC $DB2I2P,REGION=0M,COND=(4,LT)
000016 //SSID DD *
000017 DB2D\ 5 SYSIBM
000018 //DB2I2CMD DD *
000019 SETG
000020 RUN IDSN=*GENAI ODSN=T42 T=N &OPT
000021 SPACEADJ IDSN=T42 ODSN=T43
000022 DSADJ IDSN=T43 ODSN=T44 MOVE=N
000023 FLIST T44 80
000024 //LINEOBJ DD *
000025 GV &OPT=&IXC='PRIT' &IXN='IXNPD1'
000026 //JOB CARD DD DATA,DLM=JJ

```

The following list is the result from previous JCL run output.

```

** ===== DB2I2 Batch Execution Command Summary ===== **
** DB2I2 Command Start/Restart At Command Line: 00001
** (00001) SETG
** (00002) RUN IDSN=*GENAI ODSN=T42 T=N &OPT
** (00003) SPACEADJ IDSN=T42 ODSN=T43
** (00004) DSADJ IDSN=T43 ODSN=T44 MOVE=N
** (00005) FLIST T44 80
** ===== **
** ===== DB2I2 Batch Execution Summary (00001) ===== **
** Global Var &OPT is Added with &IXC='PRIT' &IXN='IXNPD1'
** Execution Output:          OUTDS
** SSID:                      DB2D
** DB2 Catalog Table Prefix: SYSIBM
** Command Requested:(00001) SETG
** Line Object Requested:
**      ...5...1...5...2...5...3...5...4...5...5...5...6...5
**00001 GV &OPT=&IXC='PRIT' &IXN='IXNPD1'
**      OK, Command process successfully
** ===== **
** ===== DB2I2 Batch Execution Summary (00002) ===== **
** Default fetch records selected: 300
** Number of Rows Returned: 1
** Execution Output:          T42
** SSID:                      DB2D
** DB2 Catalog Table Prefix: SYSIBM
** Command Requested:(00002) RUN IDSN=*GENAI ODSN=T42 T=N &OPT
** ----- **
** Line Object Requested:
**00001 SELECT 'AI' ||STRIP(A.IXCREATOR)||'.' ||A.IXNAME, A.PARTITION
**00002      ,A.CARDF, B.FULLKEYCARD
**00003 FROM SYSIBM.SYSINDEXPART A
**00004      ,SYSIBM.SYSINDEXES B
**00005 WHERE A.IXCREATOR = &IXC AND IXNAME LIKE &IXN
**00006 AND A.IXCREATOR = B.CREATOR AND A.IXNAME = B.NAME
**00007 UNION ALL
**00008 SELECT 'AI' ||STRIP(A.OWNER)||'.' ||A.NAME, A.PARTITION
**00009      ,B.CARDF, A.FULLKEYCARD
**00010 FROM SYSIBM.SYSINDEXSTATS A
**00011      ,SYSIBM.SYSINDEXPART B
**00012 WHERE OWNER = &IXC AND NAME LIKE &IXN
**00013 AND A.OWNER = B.IXCREATOR AND A.NAME = B.IXNAME
**      OK, Command process successfully
** ===== **
** ===== DB2I2 Batch Execution Summary (00003) ===== **
** Execution Output:          T43
** SSID:                      DB2D
** DB2 Catalog Table Prefix: SYSIBM
** Command Requested:(00003) SPACEADJ IDSN=T42 ODSN=T43
** ----- **
** Line Object Requested:
**      ...5...1...5...2...5...3...5...4...5...5...5...6...5
**00001 AI PRIT.IXNPD1      0      +8.731150000000000E+05 873115
**      OK, Command process successfully
** ===== **
** ===== DB2I2 Batch Execution Summary (00004) ===== **
** Execution Output:          T44
** SSID:                      DB2D
** DB2 Catalog Table Prefix: SYSIBM
** Command Requested:(00004) DSADJ IDSN=T43 ODSN=T44 MOVE=N
** ----- **
** Line Object Requested:
**      ...5...1...5...2...5...3...5...4...5...5...5...6...5
**00001 IP PRIT.IXNPD1 0 ALLOC=(PAGE,14562,1456)
**      OK, Command process successfully
** ===== **
* FLIST Command Requested: FLIST T44 80
...5...1...5...2...5...3...5...4...5...5...5...6...5...7...5...
/DP002201 JOB 2585810028,'DP0022',NOTIFY=DP0022,
/ CLASS=Y,MSGCLASS=R,REGION=8M
*JOBPARM R=P316
/* CLASS=U PROD CLASS=B,A TEST
/* -----***** DB2I2 DB2 DSADJ JCL GENERATION ****+-----+
/* DB2I2 DB2 WORK BENCH UTILITIES INTERFACE
/** BY:      JRH GOLDENSTATE SOFTWARE, INC.

```

DB2I2 Reference Manual

```
/*          COPYRIGHTED 1997-2006
/* DATE:    00/09/21  TIME:19:01  CREATOR:DP0022
/*-----+
/*JOBPARM S=SYSB
/*JOBLIB  DD DISP=SHR,DSN=SYS1.DB2LOAD
/*DB2I2P  JCLLIB ORDER=DP0022.DB2I2.LIB2T
/*----- *//
/* ADJUST PRIMARY/SECONDARY FOR STORAGE GROUP ALLOCATION *//
/*----- *//
/*ALTRD1 EXEC PGM=IKJEFT1B,DYNAMNBR=20,COND=(4,LT)
/*SYSPRINT DD SYSOUT=*
/*SYSUDUMP DD SYSOUT=*
/*SYSTSPRT DD SYSOUT=*
/*SYSTSIN  DD *
DSN SYSTEM(DB2D)
RUN PROGRAM(DSNTIAD) PLAN(DB2I2A)
/*SYSIN    DD *
ALTER INDEX PRIT.IXNPD1
        PRIQTY 58248  SECQTY 5824 ;
/*
**----- **
SYS00265.T190136.RA000.DP002201.R0255742 was preallocated (no free was done).
READY
END
```

SQ

Command Syntax:	SQ (V8 or above only)
Line objects allowed:	FU
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	SQ sequence line object
Reusable:	Yes

Command Description

- Use DB2I2 SQ command to generate DB2 SQ – sequence line for selected line objects.

SQAUTH

Command Syntax:	SQAUTH [GRANTEE GRANTOR
Line objects allowed:	SQ,US (V8 or above only) RO for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 SQAUTH command to display sequence authorization information of sequence or user.
- Use GRANTEE or GRANTOR command option to choose whether you want to display GRANTEE or GRANTOR authorization information of the selected line objects. The default option if not specified is GRANTEE.

SSID(db2 sub-system ID)

Command Syntax:	SSID(DB2 sub-system ID) or SSID(?)
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	N/A

Command Description

- Use DB2I2 SSID command to switch between two different DB2 sub-systems.
- The SSID specified needed to be defined in the db2i2.clist.library(SSID) file during the installation of DB2I2.
- Use SSID(?) to display all available SSID.

Example

The example below set SSID to DB2 sub-system 'DSN'.

```
EDIT          JD00.DB2I2.WKBENCH
Command ==> SSID(DSN)                               Scroll ==> CSR
***** Top of Data *****
```

The example below display all available SSID with SSID(?) command.

```
Command ==> ssid(?)                               Scroll ==> CSR
```

The screen below display the result from previous SSID(?) command.

```
** ----- SSID Information -----**
SSID: DB2A      Location: ACSCDB2A
SSID: DB2P      Location: ACSCDB2P
SSID: DB2D      Location: ACSCDB2D
SSID: DB2T      Location: ACSCDB2T
SSID: DB2M      Location: ACSCDB2M
**
```

START

Command Syntax:	START [[DB(dbname) [SP(spname)]] [EDIT=Y] [start option]
Line objects allowed:	DB, TS, TP, IX, IP, IS, ISP TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 START command to START a database, a table space or an index space. DB and SP can be coded either on the command line or from line objects selected.
- You can use any valid DB2 START command options on the command line to apply these options to the DB2 START command for the selected line objects.

Example

The following example starts a table space DBTST001.STS00002 with ACCESS(RW) as default.

```
Command ==> start
S00258 TS DBTST001.STS00002 BP0 0 T N N N 64 294
000259 TP 0 1798 178 I SGTST1 TDB2 0 10
```

The result from the previous START command is displayed in the following screen.

```
Command ==> Scroll ==> CSR
***** Top of Data *****
DSNT309I -DSN DSNTDCST TABLESPACE STS00002 IS CURRENTLY START RW.
DSN9022I -DSN DSNTDDIS 'START DATABASE' NORMAL COMPLETION
***** Bottom of Data *****
```

You can use any valid DB2 START command options with START command. For example, START ACCESS(UT).

STATS

Command Syntax:	STATS [TSIX] [OPTION= <u>ALL</u> REORGTSIX REORGTPIP] [T=N]
Line objects allowed:	DB, TS, TP, TB, IX, IP, IS, ISP, PL, PG, IC, CO
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report or line object TP or IP if TSIX option is used
Reusable:	Maybe

Command Description

- Use DB2I2 STATS command to display DB2 catalog statistics information for the selected line objects.
- Output produced from STATS command contains information, which assist you to identify potential performance or integrity problem of your DB2 objects, based on the collected statistics from your DB2 catalog tables.
- Use TSIX command option to return TP or IP lines instead of a report for any of the selected objects. By doing so, you can use the output from STATS command as input to the other DB2I2 commands, such as COPY, REORG or RUNSTSTS command.
- Use OPTION command option to specify output option. ALL is the default which display all statistics result. REORGTSIX returns in TS or IX line object which required REORG. REORGTPIP return in TP or IP line object which required REORG.

Example

The example below demonstrates how to use DB2I2 STATS command to display the statistics of table space DBTST001.STS00001 and DBTST001.STS00002.

```
Command ==> STATS
ss0257 TS DBTST001.STS00001 BP0 0 T N N N 64 294
ss0258 TS DBTST001.STS00002 BP0 0 T N N N 64 294
```

The screen below displays the result from previous DB2I2 STATS command.

```

000257 TS DBTST001.STS00001 BP0 0 T N N N 64 294
==MSG> Tablespace Name..... Stats.Date NACTIVE
==MSG> DBTST001.STS00001 1997-12-23 900
=NOTE= * Last Full Image Copy was made more than 7 days - DBTST001.STS00001
==MSG> PTN. Stats Date CARD..... FARINDREF NEARINDREF %Active %Drop
==MSG> 0 1997-12-23 9287 0 0 17 0
==MSG> Table Name..... Stats Date CARD..... NPAGES....
==MSG> JD00.ELEMENT 1997-12-23 1035 20
==MSG> JD00.ENTITY 1997-12-23 233 6
==MSG> JD00.PLAN_TABLE 1997-12-23 26 4
==MSG> JD00.STD_ABBR 1997-12-23 3386 56
==MSG> JD00.SUBJECT 1997-12-23 49 1
==MSG> JD00.TN0A012 1997-12-23 2741 44
==MSG> JD00.TN0A0121 1997-12-23 55 1
==MSG> JD00.TN0A013 1997-12-23 1081 47
==MSG> JD00.TN0A0131 1997-12-23 676 13
==MSG> NPS3.TRCV004_R 1997-12-23 5 1
==MSG> Index Name..... Stats.Date CG CD CTO FSTKEYCD FULKEYCD NLEAF.
==MSG> JD00.XABR001 1997-12-23 Y Y 100 1930 3386 61
==MSG> JD00.XABR002 1997-12-23 N N 83 2669 3386 61
==MSG> JD00.XENT001 1997-12-23 Y Y 100 233 233 2
==MSG> JD00.XN0A0120 1997-12-23 Y Y 100 235 2741 27
==MSG> JD00.XN0A0130 1997-12-23 Y Y 100 1081 1081 9
==MSG> NPS3.XRCV0040_R 1997-12-23 Y Y 100 5 5 1
==MSG> IPN. Stats Date CARD.... FAROFFPOS NEAROFFPOS LEAFDIST
==MSG> 0 1997-12-23 3386 0 0 0
==MSG> 0 1997-12-23 3386 0 806 0
==MSG> 0 1997-12-23 233 0 0 0
==MSG> 0 1997-12-23 2741 0 0 0
==MSG> 0 1997-12-23 1081 0 0 0
==MSG> 0 1997-12-23 49 0 0 0
==MSG> 0 1997-12-23 5 0 0 0
==MSG> Tablespace Name..... Stats.Date NACTIVE
==MSG> DBTST001.STS00002 1997-12-23 360
==MSG> PTN. Stats Date CARD..... FARINDREF NEARINDREF %Active %Drop
==MSG> 0 1997-12-23 13 0 0 0 0
==MSG> Table Name..... Stats Date CARD..... NPAGES....
==MSG> JD00.T2 1997-12-23 13 1

```

The following details the warning messages:

- **No RUNSTATS for Tablespace**
- **No Full Image Copy EVER taken**
- **Last Full Image Copy > 7 days**
- **No RUNSTATS for Tablespace Part**
- **NEARINDREF+FARINDREF > 10% of CARD**
- **% DROP is > 10**
- **No RUNSTATS for Table**
- **No RUNSTATS for Index**
- **Clustering Index Ratio < 90%**
- **No RUNSTATS for Index Part**
- **FAROFFPOS is > 10% of the CARD**
- **LEAFDIST > 200 & FREEPAGE not = 0**

The following batch run demonstrates how to use STATS together with DB2I2 COPY command to generate full image copy JCL for any table space within database DJDOO which do not have a full image copy exist.

```
000032 //DB2I2CMD DD *
000033 rexx idd=rexxddl
000034 STATS odsn=t1 t=n tsix
000035 flist t1
000036 ed t1 MACRO(IDD=MACRODD)
000037 copy idsn=t1 odsn=t2 'parmutil.cntl(fullcopy)'
000038 //LINEOBJ DD *
000039 TS DJDD00.%
000040 //REXXDD1 DD DATA,DLM=AA
000041 /* rexx */
000042 address tso "delete t1"
000043 address tso "delete t2"
000044 return 0
000045 AA
000046 //MACRODD DD DATA,DLM=MM
000047 x all
000048 f 'NO FULL' all
000049 delete x all
000050 MM
```

The detail description for each line is as follow:

- Line 33 invokes an inline REXX EXEC to delete two work files T1 and T2. The REXXDD1 DDNAME contains a REXX Exec from line 49 through line 52.
- Line 34 invokes the STATS command against a line object TS DJDD.%, which is resided under the default DDNAME LINEOBJ. The output is stored in file T1 (ODSN=T1) with no heading/footer printed (T=N) and the output is in TP or IP format (TSIX).
- Line 35 invoke FLIST to print the file T1
- Line 36 invokes ED to only pick the result line in the file T1 which contain 'NO FULL' string.
- Line 41 invokes COPY command to generate the DB2 COPY utility JCL with a preset option from the option file 'parmutil.cntl(FULLCOPY)'. The input file is T1 and the output file is T2.

STATUS

Command Syntax:	STATUS [pageset-status-abbr, pageset-status-abbr..] Where pageset-status-abbr can be the following: ARBDP AREO* ACHKP AREST AUXW CHKP COPY DEFER GRECP ICOPY INDBT LPL LSTOP OPENF PSRCP PSRBD RBDP RBDP* RECP REFP RELDP REORP REST RESTP RO RW STOP STOPE STOPP UT UTRO UTRW UTUT WEPR
Line objects allowed:	Display output with pageset-status
Process Mode	Online and Batch
Support	
Multiple line object:	No
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 STATUS command to display pageset status explanation.
- In Online mode, Point the cursor to the pageset-status-abbr from display output. Or enter pageset-status-abbr as part of the command option for both online and batch mode to display the explanation of pageset-status-abbr.
- The pageset-status-abbr can be the following seperate with a comma
- ARBDP AREO* ACHKP AREST AUXW CHKP COPY DEFER GRECP ICOPY INDBT LPL LSTOP
OPENF PSRCP PSRBD RBDP RBDP* RECP REFP RELDP REORP REST RESTP RO RW STOP STOPE
STOPP UT UTRO UTRW UTUT WEPR

Example

The following example uses the STATUS UTRO command to display the UTRO status.

```
Command ==> staus utro                               Scroll ==> CSR
***** Top of Data *****
=NOTE= UTRO THE TS, TP, IX, IP, OR LOGICAL IX PARTITION HAS BEEN STARTED
=NOTE= FOR RW PROCESSING, BUT ONLY RO PROCESSING IS ENABLED
=NOTE= BECAUSE A UTILITY IS IN PROGRESS FOR THAT OBJECT.
```

STOP

Command Syntax:	STOP [DB(dbname) [SP(spname)]] [EDIT=Y]
Line objects allowed:	DB, TS, TP, IX, IP, IS, ISP TSC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 STOP command to invoke DB2 STOP command and stop selected DB2 objects. DB and SP can be coded either on the command line or from line objects selected.

Example

The following example uses the STOP Db2I2 command to stop a table space DBTST001.STST0002.

```
Command ==> stop
s00258 TS DBTST001.STS00002 BP0 0 T N N N 64 294
000259 TP 0 1798 178 I SGTST1 TDB2 0 10
000260 TB JD00.ELEMENT T 307 DBTST001 STS00001
000261 TB JD00.ENTITY T 298 DBTST001 STS00001
```

The screen below displays the result from previous STOP command.

```
Command ==> Scroll ==> CSR
***** Top of Data *****
DSN9022I -DSN DSNITDDIS 'STOP DATABASE' NORMAL COMPLETION
***** Bottom of Data *****
```

STOSPCE

Command Syntax:	STOSPCE
Line objects allowed:	SG
Process Mode	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	JCL
Reusable:	Yes

Command Description

- Use STOSPSPACE to generate DB2 STOSPSPACE Utility JCL against selected SG line objects.

Example

The following example uses the STOSPCE DB2I2 command to generate STOSPSPACE utility JCL.

```

Command ==> stospace
s20138 SG DSN8G100 IBMUSER DSNCA10 0 0001-01-01 IBMUSER
000139 SG SYSSMS IBMUSER DSNCA10 0 0001-01-01 IBMUSER

/* $STOSPSPACE----- **
//STEP001 EXEC DSNUPROC,COND=(4,LT),
/* LIB='DSNA10.SDSNLOAD',
// SYSTEM='DBAG',UID='DB2ADM01.STEP001',UTPROC=' '
/* ----- **
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
STOSPSPACE STOGROUP (
    DSN8G100
    ,SYSSMS
)

```

SUPERC

Command Syntax:	SUPERC newds oldds [ODSN='output.dsname'] [superc-process-option]
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 SUPERC command to invoke IBM SUPERC utility to compare two files: newds and oldds.
- You can use this command to compare the outputs from two different DDL or MIGR runs to see if there are any differences between 2 different runs.
- Specify ODSN='output.dsname' command options to direct the SUPERC comparison output to a specific dataset.
- Specify superc-process-option such as CMPCOLM 16:118 to comparing the output of EXPALIN/EXPLAINP

SUPERCS

Command Syntax:	SUPERCS search.dsname search-string [ODSN='output.dsname']
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 SUPERCS command to invoke IBM SUPERC utility SEARCH FOR option to search a string from specified dataset.
- Specify ODSN='output.dsname' command options to direct the SUPERCS search for output to a specific dataset.

SUPPORT

Command Syntax:	SUPPORT
Line objects allowed:	N/A
Process Mode:	Online only
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	Email Message to Db2I2 support team
Reusable:	No

Command Description

- Use DB2I2 SUPPORT command to communicate with DB2I2 Support team thru SMTP mail.

```
#SUPPORT ----- JRH Db2i2 Technical Support Interface Screen -----
```

```
Host Name:  www.jrh-inc.com
From:       your-email.address.com
To:        support@jrh-inc.com
Cc:        jdoong@jrh-inc.com
Subject:    your subject
Body:
  Your message to support team
  _____
  _____
  _____
  _____
  _____
  _____
  _____
  _____
```

```
Please Enter Dataset Name of an Attached File Below
This Dataset should Contains Detail Execution Result of a DB2I2 Run.
Attached DSN: Your Attached File to support team
  _____
```

Enter=Send Mail To JRH Tech Support F3=Exit

Press Enter and the information is delivered to db2i2 support team right away for prompt attention.

SY

Command Syntax:	SY
Line objects allowed:	TB, VW, PL, PG, MT TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object SY
Reusable:	Yes

Command Description

- Use DB2I2 SY command to generate Synonyms line object from selected TB, VW, PG or PL line object.

Example

The following example demonstrates how to use DB2I2 SY command to display Synonyms information for a DB2 package JD00.N0AR006.

```

Command ==> SY                               Scroll ==> CSR
000006 pg JD00.%
==MSG> >>>> Begin of FREE DDL
==MSG> Collid.Name..... Valid. Operative
s00007 PG JD00.N0AR006                       Y      Y
000008 -- FREE Package Name: JD00.N0AR006
    
```

The screen below displays the result from the previous DB2I2 SY command.

```

Command ==>                               Scroll ==> CSR
000006 pg JD00.%
000007 PG JD00.N0AR006                       Y      Y
000008 SY JD00.S1
000009 SY JD00.S2
    
```

SYSIBM

Command Syntax:	SYSIBM(creator)
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	N/A

Command Description

- Use DB2I2 SYSIBM command to select the creator name of your catalog table to be used for all DB2I2 commands. If your shop have installed **mirror catalog tables**, SYSIBM command allows you to switch between mirror catalog tables and the actual DB2 catalog tables.
- To switch to a mirror catalog table, you use SYSIBM(creator of mirror table). For example, if mirror table creator name is DB2ADM, issue the command SYSIBM(DB2ADM) allows you to use a set of tables, with DB2ADM as table creator mirror, as catalog table.
- Use SYSIBM(SYSIBM) command to switch you back to DB2 catalog table.

TAG

Command Syntax:	TAG
Line objects allowed:	syscopy line, AC, AR, CP, RL line
Process Mode:	Online only
Support	
Multiple line object:	No
Multiple type of line object:	No
Wild Card % allowed:	No
Output Type:	Utility JCL
Reusable:	Yes

Command Description

- Use DB2I2 TAG command to tag a SYSCOPY line on the SYSCOPY selection screen or an AC, AR, CP, RL line from DSN1LOGP selection screen.
- When you process DB2I2 RECOVER or DSN1COPY command, if it requires access to SYSIBM.SYSCOPY table to display recovery RBA or image copy file information, DB2I2 displays SYSCOPY selection screen. From the selection screen, you choose a SYSCOPY line with DB2I2 TAG command, and use information on the selected SYSCOPY line to generate RECOVER or DSN1COPY utility JCL.
- You can use DB2I2 TAG command to select a RBA line of AC, AR, CP or RL from a DB2I2 DSN1LOGP command. The information on the selected RBA line will be used to generate DSN1LOGP utility JCL.

Example

The following example demonstrates how to use DB2I2 TAG command to select a RBA point for recover. It starts with a DB2I2 RECOVER command with TORBA option selected. After the selection screen is displayed, you issue DB2I2 TAG command to select a line, which contains the desired RBA point for your recovery job.

```

Command ==>> recover
s00415 TS DDB2ADM.DSNOPTTS BP2 0 R N N N 4 2

#RECOVA ----- DB2I2 RECOVER TABLESPACE/INDEX PROCESS OPTIONS -----

DSNUM          ALL_ (ALL/0-4096)          ---- Performance Enhancement ----
CURRENT        N (Y/N)                    Rebuild IX After TS Rcvry N (Y/N)
TORBA          Y (Y/N/I-Incore RBA)       SORTDEVT SYSALLDA SORTNUM _ (0-9)
TOLOGPOINT    N (Y/N/I-Incore RBA)
LOGONLY       N (Y/N)                    Alloc DD for Last Copy DS Y (Y/N)
TCOPY         N (Y/N/L-LastCopy/F-LastFullCopy)
COPYDSN PREFIX _____

PAGE/CONTINUE  _____                RETAIN for Tape Dataset Y (Y/N)
ERROR RANGE   N (Y/N)
SITE OPTION   A (A-All 1-LOCALSITE 2-RECOVERYSITE)
REUSE        N (Y/N) PARALLEL _ (0-9)
LOGRANGES   _ (Y/N) CURRENTCOPYONLY _ (Y/N)
TAPEUNITS   _ (0-9)
FROMDUMP     _ (Y/N) DUMPCLASS _____
RESTOREBEFORE N (Y-Restore Before RBA/N)
SYSCOPY within 999 Days (1-999) (All SYSCOPY records for last ? days)
One Job per step N (Y/N)
One Recovery Job
for all objects Y (Y/N)
VERIFYSET (Y/N) _ ENFORCE (Y/N) _ BACKOUT (Y/N) _
PF3=Exit ENTER=Process Your Selection

```

After you select TORBA option, DB2I2 displays the screen below for you to choose which RBA point is desirable for your recovery.

```
Command ==>
Scroll ==> CSR
===== -Start RBA-- TSIC ---Date--- --time-- DS# DSNAME-----
000001 00001DA978A3 FTFC 2011-04-11 13.48.15 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1848.P001
000002 00001DA7BC23 FTFC 2011-04-11 13.42.39 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1842.P001
000003 00001D77F1D6 FTFC 2011-04-06 17.09.37 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11096.T2209.P001
000004 00001D60A6A0 FTFC 2011-04-04 18.46.32 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11094.T2346.P001
000005 00001D4821C1 F LB 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.D11091.T0107.P000
000006 00001D4821C1 F 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.L10000.D11090.T1811
000007 00001C573513 FTFC 2011-03-14 19.16.30 0 BKUP.LPF.DDB2ADM.DSNOPTTS.D11074.T0016.P001
000008 00001C515E64 FTFC 2011-03-14 16.56.46 0 BKUP.LPF.DDB2ADM.DSNOPTTS.D11073.T2156.P001
000009 00001A77E312 CL 2011-02-15 19.21.05 0 DDB2ADM.DSNOPTTS
```

You use TAG command here to select a SYSCOPY line for recover. The following example tags a line to select a RBA point 0102947988D8 to be the recover point.

```
Command ==> TAG
Scroll ==> CSR
===== -Start RBA-- TSIC ---Date--- --time-- DS# DSNAME-----
000001 00001DA978A3 FTFC 2011-04-11 13.48.15 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1848.P001
000002 00001DA7BC23 FTFC 2011-04-11 13.42.39 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1842.P001
000003 00001D77F1D6 FTFC 2011-04-06 17.09.37 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11096.T2209.P001
000004 00001D60A6A0 FTFC 2011-04-04 18.46.32 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11094.T2346.P001
000005 00001D4821C1 F LB 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.D11091.T0107.P000
SO0006 00001D4821C1 F 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.L10000.D11090.T1811
000007 00001C573513 FTFC 2011-03-14 19.16.30 0 BKUP.LPF.DDB2ADM.DSNOPTTS.D11074.T0016.P001
000008 00001C515E64 FTFC 2011-03-14 16.56.46 0 BKUP.LPF.DDB2ADM.DSNOPTTS.D11073.T2156.P001
000009 00001A77E312 CL 2011-02-15 19.21.05 0 DDB2ADM.DSNOPTTS

EDIT DB2ADM.DB2I2.RCVR.OUTPUT Record Tagged
Command ==>
Scroll ==> CSR
===== -Start RBA-- TSIC ---Date--- --time-- DS# DSNAME-----
000001 00001DA978A3 FTFC 2011-04-11 13.48.15 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1848.P001
000002 00001DA7BC23 FTFC 2011-04-11 13.42.39 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11101.T1842.P001
000003 00001D77F1D6 FTFC 2011-04-06 17.09.37 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11096.T2209.P001
000004 00001D60A6A0 FTFC 2011-04-04 18.46.32 0 DB2ADM.PF.DDB2ADM.DSNOPTTS.D11094.T2346.P001
000005 00001D4821C1 F LB 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.D11091.T0107.P000
000006 00001D4821C1 F 2011-03-31 18.11.23 0 DB2ADM.DDB2ADM.DSNOPTTS.L10000.D11090.T1811
```

The screen below displays partial JCL generated for RECOVER to RBA command.

```
000019 //STEP001 EXEC DSNUPROC,COND=(4,LT),
000020 //* LIB='DSNA10.SDSNLOAD',
000021 // SYSTEM='DBAG',UID='DB2ADM01.STEP001',UTPROC=' '
000022 //* ----- **
000023 //* $LISTDD----- **
000024 //SYSPRINT DD SYSOUT=*
000025 //DD00001 DD DSN=DB2ADM.DDB2ADM.DSNOPTTS.L10000.D11090.T1811,
000026 // DISP=OLD
000027 //* $RECOVER----- **
000028 //SYSIN DD *
000029 RECOVER
000030 TABLESPACE DDB2ADM.DSNOPTTS DSNUM ALL
000031 TORBA X'00001D4821C1'
000032 LOCALSITE
```

TB

Command Syntax:	TB
Line objects allowed:	DB, TS, TP, IX, IP, IS, ISP, PL, PG, AL, VW, SY (DT, SP, FU, TR for V6 or above) (MT, SQ for V8 or above) IXC, IPC, TSC, TPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object TB
Reusable:	Yes

Command Description

- Use DB2I2 TB command to display DB2 table dependency line objects information for the selected line objects.

Example

The example below demonstrates how to use DB2I2 TB command to display table dependency information of a DB2 package.

```
Command ==> tb
s00087 PG .DSNJAR.DSNX9DPJ.V10R1M0
```

The screen below displays result from previous DB2I2 TB command. It shows S-select, I-Insert, U-Update and D-Delete for all the TB lines.

```
Command ==>
000087 PG .DSNJAR.DSNX9DPJ.V10R1M0
000088 TB SYSIBM.SYSJARCONTENTS .DSNJAR.DSNX9DPJ.V10R1M0 S I - D
000089 TB SYSIBM.SYSJARCONTENTS .DSNJAR.DSNX9DPJ.V10R1M0 S I - D
000090 TB SYSIBM.SYSJARCONTENTS .DSNJAR.DSNX9DPJ.V10R1M0 S I - D
000091 TB SYSIBM.SYSJAROBJECTS .DSNJAR.DSNX9DPJ.V10R1M0 S - U -
000092 TB SYSIBM.SYSJAROBJECTS .DSNJAR.DSNX9DPJ.V10R1M0 S - U -
000093 TB SYSIBM.SYSJAROBJECTS .DSNJAR.DSNX9DPJ.V10R1M0 S - U -
000094 TB SYSIBM.SYSJAVA_OPTS .DSNJAR.DSNX9DPJ.V10R1M0 S I - D
000095 TB SYSIBM.SYSJAVA_OPTS .DSNJAR.DSNX9DPJ.V10R1M0 S I - D
000096 TB SYSIBM.SYSJAVA_OPTS .DSNJAR.DSNX9DPJ.V10R1M0 S I - D
```

TBAUTH

Command Syntax:	TBAUTH [GRANTEE GRANTOR]
Line objects allowed:	AL, TB, VW, MT TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 TBAUTH command to display authorization information of selected AL-Alias, TB-Table or VW-View line objects.
- Use GRANTEE or GRANTOR command option to choose whether you want to display GRANTEE or GRANTOR authorization information of the selected line objects. The default option if not specified is GRANTEE.

Example

The following screens demonstrate how to use DB2I2 TBAUTH command to display GRANTEE information of table/column authorizations for a QMF table Q.OBJECT_DIRECTORY.

```
Command ==>> TBAUTH                               Scroll ==>> CSR
S00004  TB Q.OBJECT_DIRECTORY           T    7    DSQDBCTL DSQTSCT1
```

The screen below displays the results from the previous TBAUTH command.

```
Command ==>>                               Scroll ==>> CSR
000004  TB Q.OBJECT_DIRECTORY           T    7    DSQDBCTL DSQTSCT1
000005                                     U           R
000006                                     P           E
000007                                     D           F
000008                                     A   D   I   S   U   E
000009                                     T   A   E   I   N   E   P   R
000010                                     E   L   L   N   S   L   D   E
000011                                     C   T   E   D   E   E   A   N
000012                                     O   E   T   E   R   C   T   C
000013      GRANTEE  GRANTOR  TCREATOR.TNAME  L   R   E   X   T   T   E   E
000014      -----
000015  US DBXLQMF  DBEXCEL  Q.OBJECT_DIRECTORY           Y
000016  US DBXLQMF  DBEXCEL  Q.OBJECT_DIRECTORY           Y
000017  US DBXLQMF  DBEXCEL  Q.OBJECT_DIRECTORY           Y
000018
000019      GRANTEE  GRANTOR  CREATOR.TNAME           COLNAME
000020      -----
000021  US DSQ9SV   DB2ADM   Q.OBJECT_DIRECTORY           OBJECTLEVEL
```

TBC

Command Syntax:	TBC (for V9 or above)
Line objects allowed:	DB, TS, TP, IX, IP, IS, ISP, PL, PG, AL, VW, SY DT, SP, FU, TR ,MT, SQ, IXC, IPC, TSC, TPC, ISC, ISPC
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object TBC
Reusable:	Yes

Command Description

- Use DB2I2 TBC command to display DB2 table Clone dependency line objects information for the selected line objects.

TEMPLATE

Command Syntax:	TEMPLATE template-option [IDSN=template.indsn] [ODSN=template.outdsn] Where template-option must be one of the following: CHECKDATA CHECKIX CHECKLOB COPY CPY2CPY LOAD MERGECPY REBUILD REORGIX REORGTS UNLOAD
Line objects allowed:	N/A
Process Mode:	Online only
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	No
Output Type:	DB2 TEMPLATE control statements
Reusable:	No

Command Description

- DB2 TEMPLATE utility allows you to specify the characteristic/pattern of work files used in various DB2 utilities by dynamically allocating those files during the running.
- Use DB2I2 TEMPLATE command to create DB2 TEMPLATE utility control statements. The generated template control statements can then be used with the other Db2 utilities to allow you to run DB2 utilities with ease.
- Specify template-option with one of the following:

CHECKDATA	CHECKIX	CHECKLOB	COPY	CPY2CPY
LOAD	MERGECPY	REBUILD	REORGIX	REORGTS
UNLOAD				
- Specify IDSN command option to indicate that the input to the TEMPLATE command is from an existing template file.

Example

The following screens demonstrate how to use DB2I2 TEMPLATE command to generate DB2 COPY utility template control statement information and save the output in template(copy) dataset.

```
Command ==>>  template copy odsn=template(copy)           Scroll ==>>  CSR
S00023  db Dbsysadm
```

Modify the content of the output file to fit your need. Do not change the **highlighted** template name, because it will be used in the other DB2I2 utility commands.

```
EDIT          SYSADM.TEMPLATE(COPY) - 01.00           Columns 00001 00072
Command ==>>                                     Scroll ==>>  CSR
***** ***** Top of Data *****
000001  TEMPLATE TYSCOPY      DSN '<syscopy.dsname>'
000002  TEMPLATE TYSCOPY2    DSN '<syscopy2.dsname>'
000003  TEMPLATE TYSRCPY1   DSN '<sysrcpy1.dsname>'
000004  TEMPLATE TYSRCPY2   DSN '<sysrcpy2.dsname>'
000005  TEMPLATE FILTER     DSN '<filter.dsname>'
000006  -- =====
000007  -- Note:
000008  --   Remove all the -- comment lines
000009  --   Db2I2 displays the first 20 lines of TEMPLATE during utility
000010  --   generation process as comment lines
000011  --
```

```

000012 -- * Do not change the assigned template-name above because they --
000013 -- will be used in the generated DB2I2 utility JCL with --
000014 -- specific template-name assigned. --
000015 -- ----- --
000016 -- Each template contains --
000017 -- TEMPLATE template-name --
000018 -- common-options and --
000019 -- disk-options or tape-options --
000020 -- ----- --
000021 TEMPLATE template-name
000022 -- ----- --
000023 -- common-options --
000024 -- ----- --
000025 UNIT SYSALLDA/name -- device-number device-type
000026 DSN 'name-expression'
000027 -- ----- --
000028 -- Use the following &variables to construct pname-expression:
000029 -- &JO. MVS job name
000030 -- &ST. MVS step name
000031 -- &UT. up to 8 characters utility ID
000032 -- &SS. subsystem ID
000033 -- &IC. F-full copy I-incremental copy C-CHANGELIMIT copy
000034 -- &UN. utility name up to 8 characters
000035 -- CHECKD for check data CHECKI for check index
000036 -- CHECKL for check lob REORGI for reorg index
000037 -- REORGT for reorg tablespace
000038 -- &SQ. sequence number of the list item in the list
000039 -- &LR. L for local copy R for remote copy
000040 -- &PB. P for primary copy B for backup copy
000041 -- &LI. list name for use with COPY FILTERDDN templates
000042 -- &DB. database name
000043 -- &TS. table space name
000044 -- &IS. index space name
000045 -- &SN. space name(table space or index space)
000046 -- &PA. five-digit partition number padded with leading zeros
000047 -- use &PA with PARTLEVEL keyword
000048 -- The following data and time variables are GMT time
000049 -- &DT. YYYYDDD
000050 -- &TI. HHMMSS
000051 -- &JU. YYYYDDD
000052 -- &YE. YYYY
000053 -- &MO. MM
000054 -- &DA. DD
000055 -- &JD. DDD portion of &DT
000056 -- &HO. HH portion of &TI
000057 -- &MI. MM portion of &TI
000058 -- &SC. SS portion of &TI
000059 -- ----- --
000060 MODELDCB dsname
000061 BUFNO 30/integer
000062 DATACLAS data-class-name
000063 MGMTCLAS manage-class-name
000064 STORCLAS storage-class-name
000065 RETPD integer/EXPDL 'YYYYDDD'
000066 VOLUMES (volser volser) VOLCNT integer
000067 GDGLIMIT 99/integer
000068 DISP(NEW/OLD/SHR/MOD DELETE/KEEP/CATLG/UNCATLG
000069 DELETE/KEEP/CATLG/UNCATLG)
000070
000071 -- ----- --
000072 -- disk-options --
000073 -- ----- --
000074 SPACE(primary secondary) CYL/TRK/MB
000075 PCTPRIMT 100/integer
000076 MAXPRIME integer
000077 NBRSECND 10/integer
000078
000079 -- ----- --
000080 -- tape-options --
000081 -- ----- --
000082 UNCNT integer
000083 STACK NO/YES
000084 JES3DD ddname
000085 TRTCH NONE/COMP/NOCOMP

```

TERM

Command Syntax:	TERM
Line objects allowed:	A Display Utility line with UTILID = utility-id
Process Mode:	Online only
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	No
Output Type:	Result from DB2 Terminate Utility command
Reusable:	No

Command Description

- Use TERM command to issue DB2 TERM UTIL(utility-id) against a Display Utility output line with UTILID = utility-id.
- You should use DISPLAY UTILITY command with EDIT=Y to gain access to the output line.

Example

The following screens demonstrate how to use DB2I2 TERM command to terminate a stop utility with utility-id JRHJ0001.STEP001.

```

Command ==> display util(*) edit=y                               Scroll ==> CSR
***** ***** Top of Data *****

Command ==> TERM                                               Scroll ==> CSR
***** ***** Top of Data *****
==MSG> -CAUTION- Profile changed to CAPS ON (from CAPS OFF) because the
==MSG>      data does not contain any lower case characters.
==CHG> DSNU100I  -DSN DSNUGDIS - USERID = JRHJ
000002          MEMBER =
S00003          UTILID = JRHJXXX1.STEP001
000004          PROCESSING UTILITY STATEMENT 1
000005          UTILITY = RUNSTATS
000006          PHASE = UTILINIT   COUNT = 0
000007          NUMBER OF OBJECTS IN LIST = 1
000008          LAST OBJECT STARTED = 1
000009          STATUS = STOPPED
000010 DSN9022I  -DSN DSNUGCCC '-DISPLAY UTIL' NORMAL COMPLETION
***** ***** Bottom of Data *****

Command ==>                                                     Scroll ==> CSR
***** ***** Top of Data *****
DSNU166I  -DSN DSNUGTER - RUNSTATS UTILITY,
          UTILID = JRHJXXX1.STEP001 NOT EXECUTING,
          CLEANUP COMPLETE
DSN9022I  -DSN DSNUGCCC '-TERM UTIL' NORMAL COMPLETION
***** ***** Bottom of Data *****

```

TOKENSCN

Command Syntax:	TOKENSCN 'load.library[member]'
Line objects allowed:	DM, PG
Process Mode:	Online only
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 TOKENSCN command to check consistent token between a DM or PG line object with a load library.
- If you specify a single line object and a 'load.library(member)', the check is done automatically in online mode. Otherwise, a batch job is generated, so that you can submit the job and run the token check in batch mode.

Example

The example below demonstrates how to use DB2I2 TOKENSCN command to check the consistent token in online mode.

```
Command ==> tokenscn db2i2.load(db2i2a)           Scroll ==> CSR
S01364  PG .DB2I2V6.DB2I2A.
```

The screen below displays the result from the previous TOKENSCN command.

```
EDIT          DP0022.T41                          ContOKens Mismatch betwe
Command ==>                                       Scroll ==> CSR
001362  PG .DB2I2V5.DB2I2A.
001363  PG .DB2I2V5T.DB2I2A.
001364  PG .DB2I2V6.DB2I2A.
001365  -- PG .DB2I2A.DB2I2%
```

The example below demonstrates how to use DB2I2 TOKENSCN command to check the consistent token in batch mode. (With no PDS member name specified)

```
Command ==> tokenscn db2i2.load
001362  PG .DB2I2V5.DB2I2A.
S01363  PG .DB2I2V5T.DB2I2A.
001364  PG .DB2I2V6.DB2I2A.
```

DB2I2 Reference Manual

The screen below displays partial result JCL generated from previous TOKENSCN command.

```
Command ==>
000014 //SEARCH EXEC PGM=ISRSUPC,
000015 //          PARM=(SRHCMP,
000016 //          ' ')
000017 //NEWDD DD DSN=DP0022.DB2I2.LOAD,
000018 //          DISP=SHR
000019 //OUTDD DD SYSOUT=*
000020 //SYSIN DD *
000021 SRCHFOR ' |n'
000022 SRCHFOR ' '
```

You can then submit the generated JCL and run the check in batch mode.

TP

Command Syntax:	TP
Line objects allowed:	DB, TS, TB, IX, IP, IS, ISP, MT TSC, TBC, IXC, IPC, ISX, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object TP
Reusable:	Yes

Command Description

- Use DB2I2 TP command to display DB2 Table Partition line objects information for selected line objects.

Example

The example below demonstrates how to use DB2I2 TP command to display Table Partition information for table space start with DJDOO.%.

```
Command ==> TP                                     Scroll ==> CSR
S02514 ts DJDOO.%
```

The screen below displays the result from previous DB2I2 TP command.

```
Command ==>                                     Scroll ==> CSR
002514 ts DJDOO.%
002515 Tp DJDOO.DBXMIGRP 0
002516 Tp DJDOO.DBXXREF 0
002517 Tp DJDOO.DBXENT 0
002518 Tp DJDOO.DBXDLG 0
002519 Tp DJDOO.DBXSCRL 0
002520 Tp DJDOO.DBXMAPD 0
002521 Tp DJDOO.DBXSET 0
002522 Tp DJDOO.DBXUSRP 0
```

TPC

Command Syntax:	TPC (for V9 or above)
Line objects allowed:	DB, TS, TB, IX, IP, IS, ISP, MT TSC, TBC, IXC, IPC, ISX, ISPC
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object TPC
Reusable:	Yes

Command Description

- Use DB2I2 TPC command to display DB2 Table Partition Clone line objects information for selected line objects.

TR

Command Syntax:	TR (for V6 or above only)
Line objects allowed:	DB, TB, PG
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	TR trigger line object
Reusable:	Yes

Command Description

- Use DB2I2 TR command to generate DB2 TR – trigger line for selected line objects.

TRACK** a price added on with RXDB2I2

Command Syntax:	TRACK ODSN=output.dsname ODD=outputDD [DDNAME=stepnm.ddname,stepnm1.ddname1..] [CURRENT]
Line objects allowed:	[JI]
Process Mode:	Batch only
Support	
Multiple line object:	Yes
Multiple type of line object:	No
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use TRACK in Batch Mode to Extract SDSF output.
- Use ODSN or ODD option to store the Extracted output.
- In Batch, use CURRENT option to Extract the Current Running Job information
- Line Object is not needed when you use CURRENT option.
- Use DDNAME option to specify stepname and ddname combination to be extracted.
- All selected Job information will be extracted when no DDNAME option specified.
- Specify DDNAME=JES2.JESMSG LG for Jes2 Job Log Information.
- For DDNAME option, each stepname.ddname are separated by a comma.

Example

The example below demonstrates how to use TRACK to capture the sdsf output Queue, trap and display some job log message. It will also send mail message if there are any job steps with non-zero return code.

The following script is in db2i2.ispfliib(track01s): it invoke track01 and passing the parms to invoke track and mail.

```

-----*
-- Issue Rxdb2i2 Command to Invoke DB2I2 Batch to Build JCL          --*
-- to Extract Job Log information.                                   --*
-- If RC not= 0 Then Send Job logs information thru Mail Body      --*
-- Specify Host variables:                                         --*
--   &H-JNM-H   - JobNM.JobNO both JobNM and JobNo can Ended with * --*
--               For example, ABC*.JOB1* or ABC*. *                --*
--   &H-SJNO-H  - JOB##### Starting after this Job number         --*
--               Use * to get All Job Numbers for &JNM              --*
--   &H-CLASS-H - SMTP output Class                                --*
--   &H-WRITER-H - SMTP Output Writer                              --*
--   &H-PRINT-H - Print e-mail                                     --*
--   &H-HOST-H  - Host Name                                        --*
--   &H-FROM-H  - From Address                                     --*
--   &H-TO-H    - To Addresses                                    --*
--   &H-CC-H    - CC Addresses                                    --*
--               Specify &H-CC-H=Cc= for No CC addresses          --*
-----*
DB2I2REX "Batch ICMD=*track01",
          "Jobdd=N Lineobj=N",
          "&H-JNM-H=DB2ADM*.JOB08* &H-SJNO-H=*",
-- Mail Parns
          "&H-CLASS-H=P",
          "&H-WRITER-H=SMTP",
          "&H-PRINT-H=Y",

```

```
"&H-HOST-H=etjrh.dal-ebid.ihost.com",
"&H-FROM-H=jdoong@jrh-inc.com",
"&H-TO-H=(dsn='jrh.demo(toaddr)')",
"&H-CC-H=Cc="
```

The following script is in db2i2.ispflib(track01):

```
RXDB2I2
//JOBINFO DD *
JI &H-JNM-H &H-SJNO-H
-- Change the Above JI lines to Fit Your Need
//WAITCNTL DD DSN=JRH.DEMO(WAITCNTL),DISP=SHR
/* WAITCNTL contains the information as follows:
/* 04/14/10 19:15:00 00:00:10
//LINEOBJ DD *
-- TEMP used to Hold the output from TRACK
Address Tso "Alloc Fi(TEMP) New Reuse",
            "Space(1,1) Tracks Unit(sysallda)",
            "Recfm(F B) Lrecl(133) Blksize(13300)"
-- JOBINFO2 used to hold the next JI for next TRACK
Address Tso "Alloc Fi(JOBINFO2) New Reuse",
            "Space(1,1) Tracks Unit(sysallda)",
            "Recfm(F B) Lrecl(80) Blksize(800)"
-- NODYMSG Defines The Body of the Mail Text
LINEOBJ DDNAME=BODYMSG
Hi Jerry
```

This script Demonstrates How to use RXDB2I2 with TRACK and MAIL

```
Jerry Doong
12345
```

The Following are Jobs with None-Zero RC

```
END_LINEOBJ
"Execio * Diskr jobinfo(stem jobi. finis"
job_save = word(jobi.1,2)
indd = 'JOBINFO'
Mail.1 = Left("Job-Name",8),
         Left("Job-Num",8),
         Left("Start-Date",10),
         Left("Start-TM",8),
         Left("End-Date",10),
         Left("End-Time",8),
         Left("StepName",8),
         Left("ProcName",8),
         Left("RC",8),
         Left("Clock",8),
         Left("CPU",8),
         Left("SRB",8),
         Left("EXCP",8)
Mail.2 = Copies("-",8),
         Copies("-",8),
         Copies("-",10),
         Copies("-",8),
         Copies("-",10),
         Copies("-",8),
         Copies("-",8),
         Copies("-",8),
         Copies("-",8),
         Copies("-",8),
         Copies("-",8),
         Copies("-",8),
         Copies("-",8),
         Copies("-",8),
         Copies("-",8),
         Copies("-",8),
         Copies("-",8)
LOOP:
DB2I2REX "TRACK idd="indd" odd=TEMP",
         "DDNAME=JES2.JESMSGGLG"
indd = "JOBINFO2"
If RXnotfnd() = 'N' Then
Do
  "Execio * Diskr TEMP(stem reci. finis"
  j = 0
  m = 2
  step_ws = ''
  jobnm = ''
  Do I = 1 to reci.0
    x = word(reci.i,3)
```

```

If x = '----' Then
Do
--  Locate Job_tm, Jobno and format Job_date
  Parse var reci.i job_tm jobno junk junk dd mm yy junk
  If mm = 'JAN' Then
    job_date = yy'-01-'dd
  Else
    If mm = 'FEB' Then
      job_date = yy'-02-'dd
    Else
      If mm = 'MAR' Then
        job_date = yy'-03-'dd
      Else
        If mm = 'APR' Then
          job_date = yy'-04-'dd
        Else
          If mm = 'MAY' Then
            job_date = yy'-06-'dd
          Else
            If mm = 'JUN' Then
              job_date = yy'-06-'dd
            Else
              If mm = 'JUL' Then
                job_date = yy'-07-'dd
              Else
                If mm = 'AUG' Then
                  job_date = yy'-08-'dd
                Else
                  If mm = 'SEP' Then
                    job_date = yy'-09-'dd
                  Else
                    If mm = 'OCT' Then
                      job_date = yy'-10-'dd
                    Else
                      If mm = 'NOV' Then
                        job_date = yy'-11-'dd
                      Else
                        If mm = 'DEC' Then
                          job_date = yy'-12-'dd
                        End
                      Else
--      Locate jobnm
      If x = '$HASP373' Then
        Do
          Parse var reci.i junk junk junk jobnm junk
          jobnm = left(jobnm,8)
        End
      Else
        If substr(x,1,1) = '-' & x \= '-' Then
          Do
            If index(reci.i,'RC') \= 0 & index(reci.i,'EXCP') \= 0 Then
              Do
--      Locate Step Header
              proc_ws2 = Index(reci.i,'PROCSTEP')
              ii = words(reci.i)
              Do iii = 1 to ii
                ws = word(reci.i,iii)
                If ws = 'STEPNAME' | ws = '-STEPNAME' Then
                  step_ws = iii
                Else
                  If ws = 'PROCSTEP' Then
                    proc_ws = iii
                  Else
                    If ws = 'RC' Then
                      rc_ws = iii
                    Else
                      If ws = 'CPU' Then
                        cpu_ws = iii
                      Else
                        If ws = 'TCB' Then
                          cpu_ws = iii
                        Else
                          If ws = 'SRB' Then
                            srb_ws = iii
                          Else
                            If ws = 'EXCP' Then
                              excp_ws = iii
                            Else

```

```

        If ws = 'CLOCK' Then
            clock_ws = iii
        End
    End
    Else
-- Locate Step Execution Line
        If step_ws \= ' ' & word(reci.i,4) \= 'ENDEED.' Then
            Do
-- Locate Step_tm, stepname, proc, cpu, src, clock, RC
                j = j + 1
                jnm.j      = jobnm
                jno.j      = jobno
                step_dt.j  = job_date
                step_tm.j  = word(reci.i,1)
                stepname.j = word(reci.i,step_ws)
                If left(stepname.j,1) = '-' Then
                    stepname.j = right(stepname.j,length(stepname.j)-1)
                stepname.j = left(stepname.j,8)
                If substr(reci.i,proc_ws2,3) = ' ' Then
                    Do
                        proc.j = substr(' ',1,8)
                        wsi = 1
                    End
                    Else
                    Do
                        proc.j      = substr(word(reci.i,proc_ws),1,8)
                        wsi = 0
                    End
                cpu.j      = right(word(reci.i,cpu_ws-wsi),8)
                srb.j      = right(word(reci.i,srb_ws-wsi),8)
                excp.j      = right(word(reci.i,excp_ws-wsi),8)
                clock.j     = right(word(reci.i,clock_ws-wsi),8)
                rc.j        = left(word(reci.i,rc_ws-wsi),8)
-- If same Jno, prev step jj tm = start tm of curr step J
-- If Jno diff, curr step Start Date Time = job_date, job_tm
                jj = j - 1
                If jno.j = jno.jj Then
                    lineo = ,
                        jobnm jobno,
                        step_dt.jj step_tm.jj step_dt.j step_tm.j,
                        stepname.j proc.j,
                        rc.j clock.j cpu.j srb.j excp.j
                Else
                    lineo = ,
                        jobnm jobno job_date job_tm step_dt.j step_tm.j,
                        stepname.j proc.j,
                        rc.j clock.j cpu.j srb.j excp.j
                Say lineo
                If rc.j \= 0 Then
                    Do
                        m = m + 1
                        Mail.m = lineo
                    End
                End
            End
        End
    End
-- Set the next JI line for Next TRACK command
    reco.1 = "JI" job_save jobno
    "Execio 1 Diskw JOBINFO2(stem reco. Finis"
End
-- Reset Error and RC
x = RxError(N)
x = RxRc(0)
-- None-Zero RC
If m > 2 Then
    Do
        "Execio "m" Diskw TEMP(stem mail. Finis"
-- Set Upper Case Off to Send Subject in Upper/Lower Case
    Db2i2rex "UCASE(OFF)"
    Db2i2rex "Mail Class=&H-CLASS-H",
        "Writer=&H-WRITER-H",
        "Print=&H-PRINT-H",
        "Host=&H-HOST-H",
        "From=&H-FROM-H",
        "To=&H-TO-H",
        "&H-CC-H",
        "Subject=Jobs With None-Zero RC@",
        "Body=(dd=bodymsg,temp)"

```

```
      Db2i2rex "UCASE(ON)"
    End
  -- Re-check Wait Control (This can be Modified since last Track)
  "Execio * Diskr WAITCNTL(stem cntli. finis"
  If cntl_ws \= cntli.1 Then
  Do
    Say " ** Wait Control Information ** "
    Say " cntli.1
      cntl_ws = cntli.1
  End
  Parse var cntli.1 cntl_dt cntl_tm wait_tm junk
  -- Check if current Date, Time > Wait Control Stop Time
  cntl_dt = date('S',cntl_dt,'U')
  If date('S') > cntl_dt |,
    (date('S') = cntl_dt & time() > cntl_tm) Then
  Do
    Say " ** >> Track Job End At:" Date() Time()
    Return 0
  End
  -- If not Expired, Wait for wait_tm
  wait_tm = 'PERIOD' wait_tm
  x = WAIT(wait_tm)
  Signal Loop
  Return 0
  -- End of Routine
/*
```

TRAUTH

Command Syntax:	TRAUTH [GRANTEE GRANTOR
Line objects allowed:	TR,US (V6 or above only) RO for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 TRAUTH command to display authorization information of selected trigger or user.
- Use GRANTEE or GRANTOR command option to choose whether you want to display GRANTEE or GRANTOR authorization information of the selected line objects. The default option if not specified is GRANTEE.

TS

Command Syntax:	TS
Line objects allowed:	DB, TB, TP, IX, IP, IS, ISP, PL, PG, MT TBC, TPC, IXC, IPC, ISC, ISPC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object TS
Reusable:	Yes

Command Description

- Use DB2I2 TS command to display DB2 Table Space line objects information for selected line objects.

Example

The example below demonstrates how to use DB2I2 TS command to display Table Space information for a selected DB2 package.

```
Command ==> TS                               Scroll ==> CSR
S02514 pg .JDOO.DBEXCEL.
```

The screen below displays all the result from previous DB2I2 TS command.

```
Command ==>                               Scroll ==> CSR
002514 pg .JDOO.DBEXCEL.
002515 TS  DJDOO.DBXMIGRP
002516 TS  DJDOO.DBXXREF
002517 TS  DJDOO.DBXENT
002518 TS  DJDOO.DBXDLG
002519 TS  DJDOO.DBXSCRL
002520 TS  DJDOO.DBXMAPD
002521 TS  DJDOO.DBXSET
002522 TS  DJDOO.DBXUSRP
```

TSC

Command Syntax:	TSC (for V9 or above)
Line objects allowed:	DB, TB, TP, IX, IP, IS, ISP, PL, PG, MT TBC, TPC, IXC, IPC, ISC, ISPC
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object TSC
Reusable:	Yes

Command Description

- Use DB2I2 TSC command to display DB2 Table Space Clone line objects information for selected line objects.

TSIX

Command Syntax:	TSIX
Line objects allowed:	DS
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	N/A
Wild Card % allowed:	Yes
Output Type:	line object TP or IP
Reusable:	Yes

Command Description

- Use DB2I2 TSIX command to display TP or IP line objects information from a DS line object.
- You can use this command to get the Table Part and Index Part information from a DS line object.

Example

The example below demonstrates how to use DB2I2 TSIX command to display all TP and IP lines from a wild card DS line object.

```
Command ==> tsix
S00062      DS DB2ADMA.DSNDBD.DDB2ADM.**
```

The screen below displays all the result from previous DB2I2 TSIX command.

```
Command ==>
000062      DS DB2ADMA.DSNDBD.DDB2ADM.**
000063      TP DDB2ADM.DSNOPTTS      0
000064      IP DB2ADM.DSN_DETTCOST_TABLE_IDX1  0
000065      IP DB2ADM.DSN_FILTER_TABLE_IDX1   0
000066      IP DB2ADM.DSN_PGRANGE_TABLE_IDX1  0
000067      IP DB2ADM.DSN_PREDICAT_TABLE_IDX1 0
000068      IP DB2ADM.DSN_SORTKEY_TABLE_IDX1  0
000069      IP DB2ADM.DSN_STATEMNT_TABLE_IDX1 0
000070      IP DB2ADM.DSN_STRUCT_TABLE_IDX1   0
000071      IP DB2ADM.DSN_VIEWREF_TABLE_IDX1  0
000072      IP DB2ADM.DSN_STATEMENT_CACHE_IDX2 0
000073      IP DB2ADM.DSN_STATEMENT_CACHE_AUXINX 0
000074      IP DB2ADM.DSN_STATEMENT_CACHE_IDX3 0
000075      IP DB2ADM.DSN_SORT_TABLE_IDX1     0
000076      IP DB2ADM.DSN_STATEMENT_CACHE_IDX1 0
000077      IP USER1.DSN_STATEMNT_TABLE_IDX1  0
000078      TP DDB2ADM.DSN8L10X      0
000079      TP DDB2ADM.DSN8S10X      0
000080      TP DDB2ADM.HASHTS        1
.
.
.
```

TSO

Command Syntax:	TSO [TSO command]
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	N/A

Command Description

- Use DB2I2 TSO command to invoke a TSO command.
- Return code is evaluated by DB2I2 process engine after returned from TSO command.
- If a non-zero return code is returned from the specified TSO command, DB2I2 will stop executing the rest of commands in line, unless ERROR(CONTINUE) or ERROR(SKIP #) command option is specified. In which case, DB2I2 will make decision based on the command option specified.
- There is seven built-in TSO command delivered with DB2I2:

FCPY

FCPY TSO command allows you to copy files. You can use FCPY to copy a sequential file with an APPEND option.

The format for FCPY is:

TSO FCPY indsn outdsn [APPEND]

If the outdsn does not exist, FCPY creates it with the same file attributes as the indsn.

APPEND option allows you to append the indsn to the output sequential file outdsn.

FCPY2

FCPY2 TSO command is an enhanced version of FCPY. It has the same feature as FCPY plus it allows you to copy files by specifying IDSN/IDD or ODSN/ODD. It also allows you to specify FREE=N to keep your files allocated after FCPY2 is done.

The format for FCPY2 is:

TSO FCPY2 [IDSN=]indsn[[IDD=idd] [ODSN=]outdsn[[ODD=odd] [APPEND] [FREE=N]

TWAIT

TWAIT TSO command allows you to wait for time event.

The format for TWAIT is:

TSO TWAIT [PERD|ACTL]hhmmssst

Where PERD is used for duration wait and ACTL is used for actual local time wait.

hh-hour

mm-minute

ss-second

t-tenth a second

h-hundredth a second

CAPTBUFR

TSO command allows you to generate SQL insert from DISPLAY BUFFERPOOL DETAIL(*) output to insert the BUFFERPOOL information into a DB2 table. To use this TSO command, you need

- to copy and modify the Create DDL delivered in the db2i2.system.library(DDLBUFR) and
- use EXEC command to generate a DB2 table to host the output from CAPTBUFR.

The format for CAPTBUFR is:

TSO CAPTBUFR Indsn-Outdsn-DB2.BufferTbl-DB2ver

Where

Indsn	contains the output from DISPLAY BUFFERPOOL DETAIL(*)
Outdsn	contains the SQL insert for the detail bufferpool information
DB2.BufferTbl	is the name of the DB2 table generated from the DDLBUFR
DB2ver	is the DB2 version

READBUFR

READBUFR TSO command reads the information from DB2.BufferTbl and generates a BUFFERPOOL detail report.

The format for CAPTBUFR is:

TSO READBUFR ssid\location-DB2.BufferTbl-Bpool-Outdsn

Where

ssid	is the DB2 sub-system ID
location	is the DB2 location name
DB2.BufferTbl	is the name of the DB2 table generated from the DDLBUFR
Bpool	is the specific BUFFERPOOL
Outdsn	is the name of the output dataset

DB2I2LOG

DB2I2LOG TSO command allows you to generate job tracking steps, by invoking TSO DB2I2TRK command, before the first and after the last step for an existing JCL dataset. The job execution tracking information will be stored in a db2 table, which allows you to easily monitor the execution results from massive number of jobs. Comment out COND=(LT,4) from your job card if you want to track batch execution result use this TSO command.

The format for DB2I2LOG is:

TSO DB2I2LOG Jcldsn your.track_table

Where

Jcldsn	contains the JCL job stream to be modified (adding tracking steps before first Job step and after last job step. The format of jcldsn can contain wildcard * to process a group of JCL dataset: MY.*.JCL MY.TEST.* MY.PDS(MEM*)
--------	---

Your.track_table is the name of the DB2 table generated from the DDLTRACK, which contains the execution status for the selected jobs.

DB2I2TRK

DB2I2TRK TSO command allows you to insert job tracking information into the specified db2 tracking table. To use this TSO command, you need

- To copy and modify the Create DDL delivered in the db2i2.system.library(DDLTRACK). You can use HELP *DDLTRACK to find out where the DDL is located. And
- Use EXEC command to generate a DB2 table to host the output from DB2I2TRK.

The format for DB2I2TRK is:

TSO DB2I2TRK your.track_table [S|E|A]

Where

Your.track_table is the name of the DB2 table generated from the DDLTRACK, which contains the execution status for the selected jobs.

[S|E|A] represents the START, END and ABEND status of the tracking job.

The following information will be inserted into the tracking table:

JOBNAME, JOBNUM, STATUS, LAST_UPD_TSTMP

P000710

P000710 TSO command takes the output from DB2 REPORT RECOVERY and generates TS line object, which represents all the table spaces, which have open for update since the specified RBA point.

To invoke P000710-Report Recovery Analysis Routine, you need to specify a DD with the following:

//RPTRDD DD DSN=output.from.report.recovery

Where output.from.report.recovery contains the output from a DB2 REPORT RECOVERY output

Format:

TSO P000710 output.TS.line.object RBA=xxxxxxxxxxxx

or **TSO P000710 output.TS.line.object LRSN=xxxxxxxxxxxx**

Where

output.TS.line.object :

contains the output TS line object from P000710

RBA or LRSN=xxxxxxxxxxxx :

Specify the 12 byte HEX RBA point which is used to decide if a TS line will be generated.

If sysibm.syslograngex contains a log range which after the specified 12 byte Hex, then a TS line object will be generated for the reported table space.

JOBGEN

JOBGEN TSO command reads the input JCL and stripe and replace all inline control parameters with information specified from control file

Format:

TSO JOBGEN 'input.JCL' 'control.file' [REP]

Where

'input.JCL': contains the input JCL to be processed

'control.file' : contains the following:

1st record 'SSID.dataset(SSID)'

used to substitute the inline //SSID DD *

2nd record 'Output.JCL.library'

used to receive the output JCL. JOBNAME is used as the output JCL member name

3rd record 'Output.parameter.library'

optional output parameter library. If specified, all the

//... DD * [\$\$memname]

//... DD DATA,DLM=xx [\$\$memname]

will be output to the library specified.

If [\$\$memname] specified, the specified memname will

Be used as the member of the output parameter, otherwise

Member C##### will be randomly generated as the output

Member name.

The following example of a control dataset:

'my.control(SSID)'

'my.jcllib'

'my.parmlib'

Replacing the following input JCL:

//JOBNAME1 JOB..

..

//SSID DD *

..

//LINEOBJ DD * \$MEMNAME1

..

//DB2I2CMD DD * \$MEMNAME2

..

//JOBNAME2 JOB .

..

//SSID DD *

..

//DB2I2CMD DD * \$MEMNAME3

With the following output JCL

'my.jcllib(JOBNAME1)' contains

//JOBNAME1 JOB..

//SSID DD DISP=SHR,DSN=my.control(SSID)

//LINEOBJ DD DISP=SHR,DSN='my.parmlib(MEMNAME1)

//DB2I2CMD DD DISP=SHR,DSN='my.parmlib(MEMNAME2)

'my.jcllib(JOBNAME2)' contains

//JOBNAME1 JOB..

//SSID DD DISP=SHR,DSN=my.control(SSID)

//DB2I2CMD DD DISP=SHR,DSN='my.parmlib(MEMNAME3)

TRANSFRM

TRANSFRM reads input definition, map the input data file with output definition file to generate an output file which contains transformed output based on the input, output definition files and the input data file.

Format:

TSO TRANSFRM

```

IDEFDD=input.def.ddname|IDEFDSN=input.def.dsname      +
ODEFDD=output.def.ddname|ODEFDSN=output.def.dsname  +
DATADD=input.data.ddname|DATADSN=input.data.dsname  +
ODSN=transform,output.dsname                        +
[COMMIT=(freq,dlm;)] [LRECL=lrecl] [WKSP=1] [pri_alloc] [QUOTE] [SAMPLE=#]

```

Where

'IDEFDD or IDEFDSN - input definition which can contains the following:

varname POS(position) DATATYPE(length,precision) NULLIF(pos)='?'

Example,

CHAR1	POS(40)	CHAR(10)
VARCHAR1	POS(115)	VARCHAR(254)
SMALLINT1	POS(1)	SMALLINT
INTERGER1	POS(3)	INTEGER
DECIMAL1	POS(35)	DEC(7,2)
DATE1	POS(774)	DATE
TIME1	POS(785)	TIME
TIMESTAMP1	POS(774)	TIMESTAMP NULLIF(800)='?'

'ODEFDD or ODEFDSN - definition file used to format output file. Each input variable defined in the IDEFDSN can be used in the ODEFDSN with {varname}:

Example,

```

INSERT INTO MY_TABLE (
  CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DECIMAL, DATE1, TIME1, TIMESTAMP1)
VALUES(
  {CHAR1},
  {VARCHAR1},
  {SMALLINT1},
  {INTERGER1},
  {DECIMAL1},
  {DATE1},
  {TIME1},
  {TIMESTAMP1});

```

COMMIT(freq,dlm) can be used to insert COMMIT statement for each freq records, if the output records are in SQL format. Dlm if not specified default to a semicolon.

LRECL is used to specify the LRECL of the output file. Default if not specified is 80.

WKSP can be used to specify the output file primary allocation in cylinder. Default if not specified is 1 cylinder.

QUOTE option generates single quote for all the CHAR, VARCHAR, DATE, TIME and TIMESTAMP data type. However, if the selected variable is a NULL based on the NULLIF condition, then a NULL is generated without single quote around it.

SAMPLE option processes only the first # of input record and then stop.

The following example demonstrates how the TSO TRANSFRM works:

```

TSO TRANSFRM IDEFDD=IDEFDD ODEFDD=ODEFDD          +
              DATADD=DATADD ODSN=TRANSFRM.OUT     +
              COMMIT=(2,;) LRECL=100 WKSP=2
FLIST TRANSFRM.OUT 79
//DATADD DD DSN=SYSADM.DB00268.TB00015.SYSREC,DISP=SHR
//IDEFDD DD DATA,DLM=AA
CHAR1      POS(40)  CHAR(10)
VARCHAR1   POS(115) VARCHAR(254)
SMALLINT1  POS(1)   SMALLINT
INTERGER1  POS(3)   INTEGER
DATE1     POS(774)  DATE
TIME1     POS(785)  TIME
TIMESTAMP1 POS(774)  TIMESTAMP  NULLIF(800)='?'
AA
//ODEFDD DD DATA,DLM=BB
INSERT INTO MY_TABLE (
  CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DATE1, TIME1, TIMESTAMP1)
VALUES(
  {CHAR1},
  {VARCHAR1},
  {SMALLINT1},
  {INTERGER1},
  {DATE1},
  {TIME1},
  {TIMESTAMP1});
BB

```

The result from the above job output:

```

READY
ISPSTART CMD(DB2I2)
** ===== **
**              DB2I2 Command/Batch Process              **
**              **                                       **
**              By JRH GoldenState Software, Inc.         **
**              COPYRIGHTED 1997-2006 Rev. Date: 06/05/2006 **
**              **                                       **
**              Licensed to EVALUATION COPY**           **
**              UserID: SYSADM **                       **
**              Date: 06/06/03 **                       **
**              Time: 06:16:32 **                       **
** ===== **
** DB2I2 was started in BATCH mode because Online mode can only be **
**              executed with ISPF EDIT environment      **
** ===== DB2I2 Batch Execution Command Summary ===== **
** DB2I2 Command Start/Restart At Command Line: 00001  **
** (00001) TSO TRANSFRM IDEFDD=IDEFDD ODEFDD=ODEFDD DATADD=DATADD ODSN=TRANSFRM.OUT
COMMIT=(2,;)
              LRECL=100 WKSP=2
** (00002) FLIST TRANSFRM.OUT 79
** ===== **
** ===== DB2I2 Batch Execution Summary (00001) ===== **
** ===== **
** Process Input Definition file **
CHAR1      POS(40)  CHAR(10)
VARCHAR1   POS(115) VARCHAR(254)
SMALLINT1  POS(1)   SMALLINT
INTERGER1  POS(3)   INTEGER
DATE1     POS(774)  DATE
TIME1     POS(785)  TIME
TIMESTAMP1 POS(774)  TIMESTAMP
** Process Output Definition file **
INSERT INTO MY_TABLE (
  CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DATE1, TIME1, TIMESTAMP1)
VALUES(
  '{CHAR1}',
  '{VARCHAR1}',
  '{SMALLINT1}',
  '{INTERGER1}',
  '{DATE1}',
  '{TIME1}',
  '{TIMESTAMP1}');

```

DB2I2 Reference Manual

```
** Total Records Processed: 4
** TSO Command processed successfully
** DB2I2 TSO Command processed successfully with RC = 0
** Command Requested:(00001) TSO TRANSFRM IDEFDD=IDEFDD ODEFDD=ODEFDD DATADD=DATADD
ODSN=TRANSFRM.OUT
COMMIT=(2,;) LRECL=100 WKSP=2
** ----- **
**      OK, Command process successfully
** ----- **
** FLIST Command Requested: FLIST TRANSFRM.OUT 79
...5...1...5...2...5...3...5...4...5...5...5...6...5...7...5...
INSERT INTO MY_TABLE (
  CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DATE1, TIME1, TIMESTAMP1)
VALUES(
  '_STATEMNT_',
  '',
  15,
  1,
  '2006-06-28',
  '05.13.12',
  '2006-06-28-05.13.12.330000');
INSERT INTO MY_TABLE (
  CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DATE1, TIME1, TIMESTAMP1)
VALUES(
  'TABLES ',
  '',
  15,
  0,
  '2006-10-02',
  '06.16.44',
  'NULL');
COMMIT;
INSERT INTO MY_TABLE (
  CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DATE1, TIME1, TIMESTAMP1)
VALUES(
  'TABLES ',
  '',
  15,
  1,
  '2006-10-02',
  '06.16.45',
  '2006-10-02-06.16.45.170000');
INSERT INTO MY_TABLE (
  CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DATE1, TIME1, TIMESTAMP1)
VALUES(
  'N_TABLE ',
  '',
  15,
  2,
  '2006-06-28',
  '06.54.46',
  '2006-06-28-06.54.46.460000');
COMMIT;
** ----- **
SYS03157.T061629.RA000.SYSADM1.R0100142 was preallocated (no free was done).
READY
END
```


Another example of TRANSFRM with QUOTE option:

```

TSO TRANSFRM IDEFDD=IDEFDD ODEFDD=ODEFDD          +
      DATADD=DATADD ODSN=TRANSFRM.OUT            +
      COMMIT=(2,;) LRECL=100 WKSP=2 QUOTE
FLIST TRANSFRM.OUT 79
//DATADD DD DSN=SYSADM.DB00268.TB00015.SYSREC,DISP=SHR
//IDEFDD DD DATA,DLM=AA
CHAR1    POS(40) CHAR(10)
VARCHAR1  POS(115) VARCHAR(254)
SMALLINT1 POS(1) SMALLINT
INTERGER1 POS(3) INTEGER
DATE1    POS(774) DATE
TIME1    POS(785) TIME
TIMESTAMP1 POS(774) TIMESTAMP NULLIF(800)='?'
AA
//ODEFDD DD DATA,DLM=BB
INSERT INTO MY_TABLE (
      CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DATE1, TIME1, TIMESTAMP1)
VALUES(
      {CHAR1},
      {VARCHAR1},
      {SMALLINT1},
      {INTERGER1},
      {DATE1},
      {TIME1},
      {TIMESTAMP1});
BB

```

And the result from the about job is list below. The difference of QUOTE option allow single quote generated automatically for all the CHAR, VARCHAR, DATE, TIME and TIMESTAMP data type. However, if the selected variable is a NULL based on the NULLIF condition, then a NULL is generated without single quote around it.

```

READY
ISPSTART CMD(DB2I2)
** ===== **
**                DB2I2 Command/Batch Process                **
**                By JRH GoldenState Software, Inc.           **
**                COPYRIGHTED 1997-2006 Rev. Date: 06/05/2006 **
**                Licensed to EVALUATION COPY**              **
**                UserID: SYSADM **                          **
**                Date: 06/06/03 **                          **
**                Time: 06:12:56 **                          **
** ===== **
** DB2I2 was started in BATCH mode because Online mode can only be **
**                executed with ISPF EDIT environment         **
** ===== DB2I2 Batch Execution Command Summary ===== **
** DB2I2 Command Start/Restart At Command Line: 00001
** (00001) TSO TRANSFRM IDEFDD=IDEFDD ODEFDD=ODEFDD DATADD=DATADD ODSN=TRANSFRM.OUT
COMMIT=(2,;)
      LRECL=100 WKSP=2 QUOTE
** (00002) FLIST TRANSFRM.OUT 79
** ===== **
** ===== DB2I2 Batch Execution Summary (00001) ===== **
** ===== **
** Process Input Definition file **
CHAR1    POS(40) CHAR(10)
VARCHAR1  POS(115) VARCHAR(254)
SMALLINT1 POS(1) SMALLINT
INTERGER1 POS(3) INTEGER
DATE1    POS(774) DATE
TIME1    POS(785) TIME
TIMESTAMP1 POS(774) TIMESTAMP
** Process Output Definition file **
INSERT INTO MY_TABLE (
      CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DATE1, TIME1, TIMESTAMP1)
VALUES(
      {CHAR1},
      {VARCHAR1},
      {SMALLINT1},
      {INTERGER1},

```

```

    {DATE1},
    {TIME1},
    {TIMESTAMP1});
** Total Records Processed: 4
** TSO Command processed successfully
** DB2I2 TSO Command processed successfully with RC = 0
** Command Requested:(00001) TSO TRANSFRM IDEFDD=IDEFDD ODEFDD=ODEFDD
COMMIT=(2,;) LRECL=100 WKSP=2 QUOTE
** ----- **
**      OK, Command process successfully
** ----- **
** FLIST Command Requested: FLIST TRANSFRM.OUT 79
...5....1....5....2....5....3....5....4....5....5....5....6....5....7....5....
INSERT INTO MY_TABLE (
    CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DATE1, TIME1, TIMESTAMP1)
VALUES(
    '_STATEMNT_',
    '',
    15,
    1,
    '2006-06-28',
    '05.13.12',
    '2006-06-28-05.13.12.330000');
INSERT INTO MY_TABLE (
    CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DATE1, TIME1, TIMESTAMP1)
VALUES(
    'TABLES ',
    '',
    15,
    0,
    '2006-10-02',
    '06.16.44',
    NULL);
COMMIT;
INSERT INTO MY_TABLE (
    CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DATE1, TIME1, TIMESTAMP1)
VALUES(
    'TABLES ',
    '',
    15,
    1,
    '2006-10-02',
    '06.16.45',
    '2006-10-02-06.16.45.170000');
INSERT INTO MY_TABLE (
    CHAR1, VARCHAR1, SMALLINT1, INTERGER1, DATE1, TIME1, TIMESTAMP1)
VALUES(
    'N_TABLE ',
    '',
    15,
    2,
    '2006-06-28',
    '06.54.46',
    '2006-06-28-06.54.46.460000');
COMMIT;
** ----- **
SYS03157.T061252.RA000.SYSADML.R0100137 was preallocated (no free was done).
READY
END

```

TO72

TO72 converts and compress DB2I2 script line to 72 bytes long. Use this TSO command

- If you have script line which is > 72 bytes long and you want to process them using DSNTIAD or EXEC db2i2 command, or
- The total length of a single SQL is > 32760 bytes long and there are a lot of waste space before and after each script line.

Format:

TSO TO72 indsn outdsn [dln];

Where

Indsn contains the input dsname, outdsn is the output dsname, and dln is the SQL delimiter, if not specified is default to a semicolon.

The following script will not be compressed by TO72:

- A DB2CMD line – DB2CMD on the first 6 position of the line
- A IDCAMS line – IDCAMS on the first 6 position of the line
- A Blank line

Example

The following examples demonstrate how you can use these built in TSO commands.

- Copy and append file T1 to the end of T2:
TSO FCPY T1 T2 APPEND
- Wait for 10 minutes before resume execution:
TSO TWAIT PERD00100000
- Wait until 4:00 PM before resume execution:
TSO TWAIT ACTL16000000
- Capture DB2 BUFFERPOOL BP0 information once every hour for a total of 8 hours period of time start at 8:00 AM and save the result in MY.BUFFERPOOL table and report from it:

```
TSO TWAIT ACTL08000000
DISPLAY DETAIL(*) ODSN=T1
TSO CAPTBUFR T1-T2-MY.BUFFERPOOL-5
EXEC IDSN=T2
TSO TWAIT PERD01000000
DISPLAY DETAIL(*) ODSN=T1
TSO CAPTBUFR T1-T2-MY.BUFFERPOOL-5
EXEC IDSN=T2 SKIP=(-4,7)
TSO READBUFR DSN\-\MY.BUFFERPOOL-BP0-T3
FLIST T3 80
//LINEOBJ DD *
BP BP0
```

- Issue DB2I2LOG TSO command to add job tracking steps to an existing JCL job stream.

```
TSO DB2I2LOG 'MY.JCL(MYJOB)' MY.TRACK_TABLE
```

The following screen displays the results from the previous command:

```
//MYJOB001 JOB
```

```
./
//STEP000S EXEC $DB2I2P,REGION=0M
//SSID DD *
DSN\ 5 SYSIBM
//DB2I2CMD DD *
TSO DB2I2TRK MY.TABLE S
//LINEOBJ DD DUMMY
```

```
./
//STEP999E EXEC $DB2I2P,REGION=0M,COND=(8,GE)
//SSID DD *
DSN\ 5 SYSIBM
//DB2I2CMD DD *
//LINEOBJ DD DUMMY
TSO DB2I2TRK MY.TABLE E
//STEP998A EXEC $DB2I2P,REGION=0M,COND=(8,LT)
//SSID DD *
DSN\ 5 SYSIBM
//DB2I2CMD DD *
TSO DB2I2TRK MY.TABLE A
//LINEOBJ DD DUMMY
//STEP999A EXEC $DB2I2P,REGION=0M,COND=((8,LT),ONLY)
//SSID DD *
DSN\ 5 SYSIBM
//DB2I2CMD DD *
TSO DB2I2TRK MY.TABLE A
//LINEOBJ DD DUMMY
```

TSSET

Command Syntax:	TSSET
Line objects allowed:	TS, DB
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	TS line object
Reusable:	Yes

Command Description

- Use DB2I2 TSSET command to generate Table Space SET information in TS line object format from selected DB or TS line objects.
- This command can be used to generate all referential integrity related TS line objects.

Example

The following example demonstrates how to use DB2I2 TSSET to generate all referential integrity related TS line objects.

```
Command ==> TSSET
s00552, TS DSNDB06.SYSDBASE
```

The following screen displays the results from previous TSSET command.

```
Command ==> ,
000552, TS DSNDB06.SYSDBASE
000553, TS DSNDB06.SYSDBASE -- RI SET 1
000554, TS DSNDB06.SYSOBJ -- RI SET 1
000555, TS DSNDB06.SYSSTATS -- RI SET 1
000556, TS DSNDB06.SYSSTR -- RI SET 1
```

UCASE

Command Syntax:	UCASE(ON OFF)
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	N/A
Reusable:	No

Command Description

- Use DB2I2 UCASE(ON) to turn on upper case indicator which convert all line objects to Upper Case.
- Use UCASE(OFF) to turn off Upper Case indicator. Use UCASE(OFF) to allow lower case line object.

UNLOAD

Command Syntax:	UNLOAD [[PARMUTIL=]'unload.parmutil.dsn] [HLQ=ImageCopy-high-level-qualifier] [LISTDEF=listdef.dsname[(patt*)]] [TEMPLATE=template.dsname] [OPTIONS=options.dsname]
Line objects allowed:	TS, TP, TB, MT [ICGEN=###] [ICDATE=YMMMDD] for from image copy option [SYSREC=sysrec.dsname] [SYSPUNCH=syspunch.dsname] TSC, TPC, TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	SQL Update Skeleton
Reusable:	Yes

Command Description

- Use DB2I2 UNLOAD command to generate DB2 UNLOAD utility JCL for selected line objects.
- Specify FROM IMAGECOPY option allows unload from image copy. You can not specify FROM IMAGECOPY option when you specify LISTDEF command option because FROM IMAGECOPY option requires specific image copy data set information. Use HLQ to specify High-level-qualifier for the selected Image Copy Dataset.
- Specify ICGEN=### and ICDATE=YMMMDD at end of the selected line object to specify the generation of image copy ### as the input to the UNLOAD, ### if specified, must be <= 0.
- Current generation of image copy is selected if no ICGEN specified.
- Specify ICDATE=YMMMDD to select a specific date of image copy as input to the UNLOAD.
- Specify field specification and WHEN specification if desired.
- Specify SYSREC and SYSPUNCH options to specify the SYSPUNCH and SYSREC dataset information. You can use this information to override the template specified in TEMPLATE command or default unload dataset name.
- TEMPLATE command option is required if you use LISTDEF command option to select your UNLOAD DB2 objects.
- For DB2 v7 or above, you can specify LISTDEF option with PDS member pattern match. For example, By specifying LISTDEF=listdef(DB1*) will generate UNLOAD DB2 utility JCL for all members name prefix with DB1. Because DB2I2 process each LISTDEF one at a time, If you select to use LISTDEF pattern option, it will prompt selection screen one for each LISTDEF member. To avoid this inconvenience, you can use PARMUTIL command to generate PARMUTIL control and then use it as process option.

Example

The following example demonstrates how to use DB2I2 UNLOAD to unload 1 table from last full image copy. Assuming you have created a template file in 'SYSADM.TEMPLATE(UNLOADT) as following:

DB2I2 Reference Manual

```

EDIT          DB2ADM.TEMPLATE(UNLOADT) - 01.01
Command ==>
***** ***** Top of Data *****
000001 TEMPLATE TSYSFNCH   DSN '&US..&TS(2,6)..D1'
000002                   UNIT SYSALDA DISP(NEW,CATLG,CATLG)
000003                   SPACE(1,0) TRK
000004 TEMPLATE TSYSREC   DSN '&US..&TS(2,6)..D2'
000005                   DISP(NEW,CATLG,DELETE)
000006                   UNIT TAPE
000007                   STACK YES
000008                   RETPD 1
000009                   VOLCNT 5
000010                   MODELDCB MODSCB

```

Issue UNLOAD command with the template specified.

```

Command ==> unload template=template(unloadt)
s01599  TB  DB2ADM.PLAN_TABLE          T    3    DDB2ADM  DSNOPTTS

```

The following selection screen is displayed for you to choose the UNLOAD options. Select from Local primary image copy option.

```

#UNLOADA ----- DB2I2 UNLOAD Utility Process          Enter required field
                                                More:      +

Source Specification (For NO LIST option only)
FROM IMAGECOPY              (Y/N) Y          FULL/INCREMENT      (F/I) F
LOCAL/REMOTE COPY (LP/LB/RP/RB) LP (L-Local R-Reomte P-primary B-Backup)
Unload Specification
EBCDIC/ASCII/UNICODE(E/A/U)  _
CCSID                        _____
NOSUBS                       (Y/N) Y
NOPAD                        (Y/N) Y  COLDEL _ , CHARDEL _ " DECPT _ .
SPANNED                       (Y/N) _
FLOAT                        (S390/IEEE) S390
MAXERR                       (1/0-99999999) 1_____
SHARELEVEL (REFERENCE/CHANGE) REFERENCE
  ISOLATION                   (CS/UR) ___ (For SHARELEVEL CHANGE only)
  SKIP LOCKED DATA           (Y/N) N   (For Isolation CS      only)
DECFLOAT-ROUNDMODE           (1-7) _
  1.ROUND-CEILING            2.ROUND-DOWN      3.ROUND-FLOOR 4.ROUND-HALF-DOWN
  5.ROUND-HALF-EVEN          6.ROUND-HALF-UP   7.ROUND-UP
IMPLICIT TIMEZONE            _____
CLONE                        (Y/N) _

FROM TABLE specification      (For no LISTDEF option only)
FROM TABLE                   (M/Y/N) N (M-one unload per table, N-No,

PF3=Exit  ENTER=Process Your Selection

```

DB2I2 generates the following UNLOAD JCL. The FROMCOPY field is populated with the most current local primary full image copy dataset information.

DB2I2 Reference Manual

```

000018 /* $UNLOAD----- **
000019 //STEP001 EXEC DSNUPROC,COND=(4,LT),
000020 /*          LIB='DSNA10.SDSNLOAD',
000021 //          SYSTEM='DBAG',UID='DB2ADM01.STEP001',UTPROC=''
000022 /*----- **
000023 //SYSPRINT DD SYSOUT=*
000024 /* $LISTDD----- **
000025 //SYSTEMPL DD DISP=SHR,DSN=DB2ADM.TEMPLATE(UNLOADT)
000026 //UTPRINT DD SYSOUT=*
000027 /* **-- TEMPLATE detail from 'DB2ADM.TEMPLATE(UNLOADT)'
000028 /* TEMPLATE TSYPNCH   DSN '&US..&TS(2,6)..D1'
000029 /*                   UNIT SYSALLDA DISP(NEW,CATLG,CATLG)
000030 /*                   SPACE(1,0) TRK
000031 /* TEMPLATE TSYSREC   DSN '&US..&TS(2,6)..D2'
000032 /*                   DISP(NEW,CATLG,DELETE)
000033 /*                   UNIT TAPE
000034 /*                   STACK YES
000035 /*                   RETPD 1
000036 /*                   VOLCNT 5
000037 /*                   MODELDCB MODSCB
000038 /* $UNLOADG----- **
000039 //SYSIN DD *
000040 OPTIONS
000041          TEMPLATEDD      SYSTEMPL
000042 UNLOAD TABLESPACE      DDB2ADM.DSNOPTTS
000043          FROMCOPY        DB2ADM.DDB2ADM.DSNOPTTS.LI0000.D11090.T1811
000044          PUNCHDDN        TYPUNCH
000045          UNLDDN          TYSREC
000046          NOSUBS
000047          NOPAD
000048          FLOAT            S390
000049          MAXERR           1
000050          SHRLEVEL         REFERENCE
000051          FROM TABLE      DB2ADM.PLAN_TABLE
000052          HEADER           OBID

```

It is sometime easier to generate TB line object by using the UDF *GENUNLD1. Because the RUN command option is too long to fit in the command line, we use SETG command to set global variable &G1 with the information to be used in the RUN command. The host variables to run *GENUNLD1 are:

- &DEF=##### specify the default row counts if no runstats information collected
- &DS=dataset-prefix dataset-prefix information for SYSREC and SYSPUNCH
- &WHERE=where-predicates contains the WHERE predicates information generated from QBUILD
- T=N to disable the header line displayed
- IN return the RUN result back your online workbench

```

Command = setg
s00026 GV &G1=IDSN=*GENUNLD1 &WHERE=Q1 &DS=XX &DEF=20000 T=N IN

```

```

EDIT          SYSADM.DB2I2.WKBENCH          Global Var &G1 is Replac
Command = run &g1                          Scroll = CSR
000026 GV &G1=IDSN=*GENUNLD1 &WHERE=Q1 &DS=XX &DEF=20000 T=N IN

```

The results from the RUN command are displayed below.

```

Command ==>                               Scroll ==> CSR
000026 GV &G1=IDSN=*GENUNLD1 &WHERE=Q1 &DS=XX &DEF=20000 T=N IN
000027 TB SYSADM.DEPT 20000 SYSPUNCH='XX.SYSPUNCH.DB00268.TB00003.D020506' SYSREC='XX.SYSREC.DB00268.TB00003.D020506'
000028 TB SYSADM.TXNT 20000 SYSPUNCH='XX.SYSPUNCH.DB00268.TB00012.D020506' SYSREC='XX.SYSREC.DB00268.TB00012.D020506'

```

Edit the TB line and add ICGEN=-1 option to the end of the line object to select -1 generation of the image copy to be used for this unload. Issue the same UNLOAD command again.

```

Command ==> UNLOAD TEMPLATE=TEMPLATE(UNLOADT)                               Scroll ==> CSR
000026 GV &G1=IDSN=*GENUNLD1 &WHERE=Q1 &DS=XX &DEF=20000 T=N IN
s20027 TB SYSADM.DEPT 20000 SYSPUNCH='XX.SYSPUNCH.DB00268.TB00003.D020506' SYSREC='XX.SYSREC.DB00268.TB00003.D020506' ICGEN=-1
000028 TB SYSADM.TXNT 20000 SYSPUNCH='XX.SYSPUNCH.DB00268.TB00012.D020506' SYSREC='XX.SYSREC.DB00268.TB00012.D020506' ICGEN=-1

```

DB2/2 Reference Manual

The results is displayed below with appropriate SYSREC and SYSPUNCH TEMPLATE override and appropriate full image copy dataset as input to the generated UNLOAD JCL.

```
//SYSIN DD *
  OPTIONS
    TEMPLATEDD SYSTEMPL
  TEMPLATE TYSREC      UNIT SYSALLDA DSN XX.SYSREC.DB00268.TB00003.D020506
                        DISP (NEW,DELETE,CATLG)
  TEMPLATE TYSPUNCH   UNIT SYSALLDA DSN XX.SYSPUNCH.DB00268.TB00003.D0205
                        DISP (NEW,CATLG,CATLG)
  UNLOAD TABLESPACE DBSYSADM.DSN8S71D
    FROMCOPY          SYSADM.DBSYSADM.DSN8S71D.D2002114.L1030735
    PUNCHDDN          TYSPUNCH
    UNLDDN            TYSREC
    EBCDIC
    NOSUBS
    NOPAD
    FLOAT              S390
    MAXERR             1
    SHRLEVEL           REFERENCE
  FROM TABLE         SYSADM.DEPT
  HEADER              OBID

/* ----- **
//STEP002 EXEC DSNUPROC,REGION=4M,COND=(4,LT),
//          SYSTEM='DSN1',UID='SYSADM1.STEP002',UTPROC=''
/* ----- **
//SYSPRINT DD SYSOUT=*
//SYSTEMPL DD DISP=SHR,DSN=SYSADM.TMP(T2)
//UTPRINT  DD SYSOUT=*
//SYSIN DD *
  OPTIONS
    TEMPLATEDD SYSTEMPL
  TEMPLATE TYSREC      UNIT SYSALLDA DSN XX.SYSREC.DB00268.TB00012.D020506
                        DISP (NEW,DELETE,CATLG)
  TEMPLATE TYSPUNCH   UNIT SYSALLDA DSN XX.SYSPUNCH.DB00268.TB00012.D020506
                        DISP (NEW,CATLG,CATLG)
  UNLOAD TABLESPACE DBSYSADM.SXNT
    FROMCOPY          SYSADM.DBSYSADM.SXNT.D2002114.L1030735
    PUNCHDDN          TYSPUNCH
    UNLDDN            TYSREC
    EBCDIC
    NOSUBS
    NOPAD
    FLOAT              S390
    MAXERR             1
    SHRLEVEL           REFERENCE
  FROM TABLE         SYSADM.TXNT
  HEADER              OBID
```

UPDATE

Command Syntax:	UPDATE [MAP='dclgen Dataset'] [RUN='db2i2.run.output' [COMMIT=#] [SQLTERM(¡?)] [CHAR=HEX]]
Line objects allowed:	TB, AL, SY, VW, MT TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	No
Output Type:	SQL Update Skeleton
Reusable:	Yes

Command Description

- Use DB2I2 UPDATE command to generate DB2 UPDATE SQL statement for selected line objects.
- Use MAP command option to generate embedded static SQL, which uses the output from DCLGEN as a map for host variables matching.
- Use RUN option to read in the output from RUN command and generate UPDATE SQL.
- Use COMMIT=# to specify commit frequency.
- Use SQLTERM(¡?) to specify sql terminator.
- Use CHAR=HEX to generate CHAR or VARCHAR in HEX format if single quote is part of the data from RUN output.

Example

The following example demonstrates how to use DB2I2 UPDATE to generate DB2 UPDATE SQL statements for table Q.PROFILES.

```
Command ==> update                               Scroll ==> CSR
S01006      TBQ.PROFILES                          T
```

The screen below displays the result from previous DB2I2 UPDATE command.

```
Command ==>                                       Scroll ==> CSR
001006      TB Q.PROFILES                          T
001007      UPDATE Q.PROFILES SET
001008          CREATOR                            = --CHAR      8    0
001009          ,CASE                              = --CHAR     18    0
001010          ,DECOPT                            = --CHAR     18    0
001011          ,CONFIRM                           = --CHAR     18    0
001012          ,WIDTH                             = --CHAR     18    0
001013          ,LENGTH                             = --CHAR     18    0
001014          ,LANGUAGE                          = --CHAR     18    0
001015          ,SPACE                              = --CHAR     50    0
001016          ,TRACE                             = --CHAR     18    0
001017          ,PRINTER                            = --CHAR      8    0
001018          ,TRANSLATION                        = --CHAR     18    0
001019          ,PFKEYS                            = --VARCHAR   31    0
001020          ,SYNONYMS                           = --VARCHAR   31    0
001021          ,RESOURCE_GROUP                       = --CHAR     16    0
001022          ,MODEL                               = --CHAR      8    0
001023          ,ENVIRONMENT                         = --CHAR      8    0
001024          WHERE
```

The screen below using the run output as the input to the update.

```

Command ==> run odsn=t1
001599  TB  DB2ADM.PLAN_TABLE          T      3      DDB2ADM  DSNOPTTS
ss1600  SELECT  "QUERYNO", "QBLOCKNO", "APPLNAME", "PROGNAME"
001601      , "PLANNO", "METHOD", "CREATOR", "TNAME", "TABNO"
001602      , "ACCESSTYPE", "MATCHCOLS", "ACCESSCREATOR"
001603      , "ACCESSNAME", "INDEXONLY", "SORTN_UNIQ", "SORTN_JOIN"
001604      , "SORTN_ORDERBY", "SORTN_GROUPBY", "SORTC_UNIQ"
001605      , "SORTC_JOIN", "SORTC_ORDERBY", "SORTC_GROUPBY"
001606      , "TSLOCKMODE", "TIMESTAMP", "REMARKS", "PREFETCH"
001607      , "COLUMN_FN_EVAL", "MIXOPSEQ", "VERSION", "COLLID"
001608      , "ACCESS_DEGREE", "ACCESS_PGROUP_ID", "JOIN_DEGREE"
001609      , "JOIN_PGROUP_ID", "SORTC_PGROUP_ID", "SORTN_PGROUP_ID"
001610      , "PARALLELISM_MODE", "MERGE_JOIN_COLS", "CORRELATION_NAME"
001611      , "PAGE_RANGE", "JOIN_TYPE", "GROUP_MEMBER"
001612      , "IBM_SERVICE_DATA", "WHEN_OPTIMIZE", "QBLOCK_TYPE"
001613      , "BIND_TIME", "OPTHINT", "HINT_USED", "PRIMARY_ACCESSTYPE"
001614      , "PARENT_QBLOCKNO", "TABLE_TYPE", "TABLE_ENCODE"
001615      , "TABLE_SCCSID", "TABLE_MCCSID", "TABLE_DCCSID"
001616      , "ROUTINE_ID", "CTEREF", "STMTTOKEN", "PARENT_PLANNO"
001617      , "BIND_EXPLAIN_ONLY", "SECTNOI", "EXPLAIN_TIME"
001618      , "MERGC", "MERGN"
001619      FROM "DB2ADM"."PLAN_TABLE"
ss1620      WHERE queryno = 0
    
```

```

Command ==> update run=t1 odsn=t1z
s01599  TB  DB2ADM.PLAN_TABLE          T      3      DDB2ADM  DSNOPTTS
    
```

```

Command ==>
***** ***** Top of Data
***** *****
000001 UPDATE "DB2ADM"."PLAN_TABLE" SET
000002     "QUERYNO"           = 0
000003     , "QBLOCKNO"       = 1
000004     , "APPLNAME"       = ' '
000005     , "PROGNAME"       = 'DB2I2A10'
000006     , "PLANNO"        = 1
000007     , "METHOD"        = 0
000008     , "CREATOR"       = 'DB2ADM'
000009     , "TNAME"         = 'PLAN_TABLE'
000010     , "TABNO"         = 1
000011     , "ACCESSTYPE"    = 'I'
000012     , "MATCHCOLS"    = 1
000013     , "ACCESSCREATOR" = 'DB2ADM'
000014     , "ACCESSNAME"    = 'PLAN_TABLE_IDX6'
000015     , "INDEXONLY"    = 'Y'
000016     , "SORTN_UNIQ"   = 'N'
000017     , "SORTN_JOIN"    = 'N'
000018     , "SORTN_ORDERBY" = 'N'
000019     , "SORTN_GROUPBY" = 'N'
000020     , "SORTC_UNIQ"    = 'N'
000021     , "SORTC_JOIN"    = 'N'
000022     , "SORTC_ORDERBY" = 'N'
.
.
000056     , "TABLE_DCCSID" = 1200
000057     , "ROUTINE_ID"    = 0
000058     , "CTEREF"       = 0
000059     , "STMTTOKEN"    = NULL
000060     , "PARENT_PLANNO" = 0
000061     , "BIND_EXPLAIN_ONLY" = 'N'
000062     , "SECTNOI"     = 0
000063     , "EXPLAIN_TIME" = '2011-05-13-19.36.34.670000'
000064     , "MERGC"       = 'N'
000065     , "MERGN"       = 'N'
000066 WHERE
000067 ; COMMIT;
    
```

USAUTH

Command Syntax:	USAUTH [GRANTEE GRANTOR]
Line objects allowed:	US RO for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 USAUTH command to display User Authorization information for selected US line objects.
- You can specify either GRANTOR or GRANTEE command option to display grantor or grantee authorization information for selected user lines. The defaults value for this command option is GRANTEE.

Example

The example below demonstrates how to use DB2I2 USAUTH command with default GRANTEE command option to display all user authorizations have been granted to user PUBLIC.

```
Command ==> USAUTH                               Scroll ==> CSR
S00001 us public
```

The screen below displays the result from previous DB2I2 USAUTH command.

```
Command ==>                                       Scroll ==> CSR
000001 us public
000002 -- User Authorization
000003
000004           C C                               C B
000005           R R C S                           R I C C
000006           A B E E R D R S T                 E S N A A A
000007           L I A A E I E T O S S             A Y D R P P
000008           T N T T A S C O S Y Y T          T S A C T T
000009           E D B E E T P O P P S S R M M E C G H U U
000010           R A S D D E L V A A O A O O A T E I R R
000011           B D D B B S A E L C D P C N N L R N V E E
000012 GRANTEE GRANTOR P D S A C G Y R L E M R E 1 2 I L T E 1 2
000013 -----
000014 US PUBLIC DB2ADM Y N
000015 US PUBLIC DB2ADM Y N
```

VIEWG

Command Syntax:	VIEWG
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	GV line object
Reusable:	Yes

Command Description

- Use DB2I2 VIEWG command to display all global variables.

VIRTUAL

Command Syntax:	VIRTUAL [IX=ixcreator.ixname]
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	SQL INSERT
Reusable:	Yes

Command Description

- Use DB2I2 VIRTUAL to generate SQL INSERT for you DSN_VIRTUAL_INDEXES table.
- DB2 EXPLAIN can use this information to see the impact of WHATIF condition by adding a new index or dropping an existing index.
- Use IX= ixcreator.ixname to populate SQL INSERT with the information from an existing index.

Example

The example below demonstrates how to use DB2I2 VIRTUAL command with IX= command option to generate SQL INSERT statement for the DSN_VIRTUAL_INDEXES table clone from an existing index.

```
Command ==> virtual ix=db2adm.plan_table_idx1
*****
000001
```

The screen below displays the result from previous DB2I2 VIRTUAL command.


```

INSERT INTO "DB2ADM"."DSN_VIRTUAL_INDEXES" (
    "TBcreator"
    , "TBNAME"
    , "IXcreator"
    , "IXNAME"
    , "ENABLE"
    , "MODE"
    , "UNIQUERULE"
    , "COLCOUNT"
    , "CLUSTERING"
    , "NLEAF"
    , "NLEVELS"
    , "INDEXTYPE"
    , "PGSIZE"
    , "FIRSTKEYCARDF"
    , "FULLKEYCARDF"
    , "CLUSTERRATIOF"
    , "PADDED"
    , "COLNO1"
    , "ORDERING1"
    , "COLNO2"
    , "ORDERING2"
)
VALUES (
-- TBcreator TBname
    'DB2ADM'
    , 'PLAN_TABLE'
-- IXcreator IXname
    'DB2ADM'
    , 'PLAN_TABLE_IDX1'
-- ENABLE: Y-Enable N-Disable
    , 'Y'
-- MODE: D-Drop C-Create
    , 'D'
-- UNIQUERULE: U-Unique D-Duplicate
    , 'D'
-- Column Count
    , 2
-- Clustering Y/N
    , 'N'
-- NLEAF
    , 1
-- NLEVELS
    , 2
-- INDEXTYPE 2-NonPartitioned 2nd Index D-Data Partitioned 2nd Index
    , '2'
-- PGSIZE: 4,8,16,32
    , 4
-- FIRSTKEYCARDF
    , +5.100000000000000E+01
-- FULLKEYCARDF
    , +1.360000000000000E+02
-- CLUSTERRATIOF
    , +1.000000000000000E+00
-- PADDED: Y/N
    , ' '
-- Column Name: QUERYNO
    , 1
    , 'A'
-- Column Name: BIND_TIME
    , 46
    , 'A'
)

```

VTOC

Command Syntax:	VTOC [ODSN=output.dsname [SHORT]]
Line objects allowed:	DS
Process Mode:	Online and Batch
Support	
Multiple line object:	Y
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	REPORT
Reusable:	Yes

Command Description

- Use DB2I2 VTOC to invoke IBM IEHLIST LISTVTOC utility to display VTOC information.
- Use ODSN option to specify the output of VTOC.
- Use SHORT option to report Empty Space Information only. SHORT option can only be used with ODSN option.

Example

The example below demonstrates how to use DB2I2 VTOC command with SHORT option to get LISTVTOC summary information.

```
Command ==> vtoc short odsn=t1
S00065      DS DB2ADMA.DSNDBD.DDB2ADM.DSNRDETC.I0001.A001  0000
```

The screen below displays the result from previous DB2I2 VTOC command.

```
BROWSE      DB2ADM.T1
Command ==>
***** Top of Data *****
-DATE: 2011.144  TIME: 13.07.28
CONTENTS OF VTOC ON VOL VPWRKD <THIS VOLUME IS NOT SMS MANAGED>
THERE ARE 2655 EMPTY CYLINDERS PLUS 12 EMPTY TRACKS FROM THE TRACK-MANAGED SPACE
THERE ARE 1448 BLANK DSCBS IN THE VTOC ON THIS VOLUME
THERE ARE 286 UNALLOCATED VIRS IN THE INDEX
```

VW

Command Syntax:	VW
Line objects allowed:	TB, VW, AL, SY, PL, PG FU, SP for V6 or above TBC for V9 or above
Process Mode:	Online and Batch
Support	
Multiple line object:	Yes
Multiple type of line object:	Yes
Wild Card % allowed:	Yes
Output Type:	line object VW
Reusable:	Yes

Command Description

- Use DB2I2 VW command to generate View line object for the selected TB, VW, AL, SY, PG or PL line object.

Example

The example below demonstrates how to use DB2I2 VW command to display View information for a DB2 package JD00.N0AR006.

```

Command ==>> VW                                     Scroll ==>> CSR
000006 pg JD00.%
==MSG> >>>> Begin of FREE DDL
==MSG> Collid.Name..... Valid. Operative
S00007 PG JD00.N0AR006                               Y      Y
000008 -- FREE Package Name: JD00.N0AR006
    
```

The screen below displays the result from previous DB2I2 VW command.

```

Command ==>>                                     Scroll ==>> CSR
000006 pg JD00.%
000007 PG JD00.N0AR006                               Y      Y
000008 VW JD00.V1
000009 VW JD00.V2
    
```

ZPARAM

Command Syntax:	ZPARAM [dsnzparm-name]
Line objects allowed:	N/A
Process Mode:	Online and Batch
Support	
Multiple line object:	N/A
Multiple type of line object:	N/A
Wild Card % allowed:	N/A
Output Type:	Report
Reusable:	No

Command Description

- Use DB2I2 ZPARAM command to display DSNZPARAM information.
- The ZPARAM load module name is determined by the name specified in the Startup.Proclib library with member name ssidMSTR.
- You can specify a ZPARAM name other than the one from the startup proclib to display the content of that particular ZPARAM module.
- For example, the following JCL is a startup procedure for db2 sub-system DSN.
 - Issue ZPARAM will display the content of DSNZPARAM
 - Issue ZPARAM DSNZPAR2 will display the content of zparm module DSNZPAR2

```

VIEW          SYS1.PROCLIB(DSNMSTR) - 01.17          Columns 00001 00072
Command ==>                                     Scroll ==> CSR
***** ***** Top of Data *****
000001 //DSNMSTR  JOB 02006580
000002 //* *****
000003 //*          JCL FOR PROCEDURE FOR THE STARTUP
000004 //*          OF THE DB2 CONTROL ADDRESS SPACE.
000005 //*
000006 //*          INSTALLATION MAY CHANGE PROGRAM LIBRARY
000007 //*          NAME IN STEPLIB DD STATEMENT TO THE
000008 //*          LIBRARY IN WHICH DB2 MODULES ARE
000009 //*          LOADED USING THE PROCEDURE VARIABLE:
000010 //*          LIB
000011 //*
000012 //* *****
000013 //DSNMSTR  PROC
000014 //IEFPROC  EXEC PGM=DSNYASCP,DYNAMNBR=119,REGION=8000K
000015 //          PARM=' ZPARAM(DSNZPARAM) '

```

- The result of the ZPARAM command is display in the following screen.
- Please refer to IBM provided macro DSN6xxxx for detail description of these ZPARAM variables.

Example

```

Command ==>
***** Top of Data ***** Scroll ==> CSR
* -----*
*          DB2I2 DSNZPARM Information          *
* SSID:    DSN                                *
* Source:  SYS1.TEST.SDSNLOAD(DSNZPARM)      *
* -----*

*-----*
*          DSN6SPRM: Create ADMF Initialization Parameters Block          *
*-----*
Level of the PTM                (SPRMLVL ) DSN410
Number of MMRBs                 (SPRMMMRB) 408
No of BM asynchronous write engines (SPRMWREN) 300
MVS/370 or XA                   (SPRMMVS ) XA
SPRM Control Parameters         (SPRMCNTL) X'0000'
BM Log Record All Written       (SPRMIDCK) NO
Write I/O Logrec Written        (SPRMIDCK) YES
Recovery Utility ByPass Reading Syslgng (SPRMIDCK) NO
.
.
.
*-----*
*          DSN6ARVP: Create Archive Data Set Parameters Block          *
*-----*
Archive Dataset Blocksize       (ARVPBKSZ) 28672
Dataset Primary Space Allocation (ARVPRISP) 1234
Dataset Secondary Space Allocation (ARVPSECS) 154
Length Of Supplied UNIT Parameter (ARVPULN1) 4
Type Of Device To Allocate Unit 1 (ARVPUNT1) TAPE
First Prefix Name               (ARVPRE1N) HDB2.DSN.ARCHLOG1
Second Prefix Name              (ARVPRE2N) HDB2.DSN.ARCHLOG2
Archive Log Dataset Cataloging   (ARVPFLG1) YES
.
.
.
*-----*
*          DSN6LOGP: Create LOG Initialization Parameters Block          *
*-----*
Dual Active Log                 (LOGOPT1 ) NO
Offload                         (LOGOPT1 ) YES
Dual BSDS                       (LOGOPT1 ) YES
Dual Archive Log                (LOGOPT2 ) NO
Output Buffer Pool Size          (LOGPOBPS) 400
Input Buffer Pool Size           (LOGPIBPS) 28
Maximum Archive Entries In BSDS (LOGPARCL) 500
Maximum Archive Read TAPE Units (LOGPMRTU) 2
.
.
.
*-----*
*          DSN6SYSP: System Parameters                                *
*-----*
System Msg Routing Code         (SYSPSMRC) X'8000'
NO. Of 4K Segments In Trace Tbl (SYSPTRSZ) 16
NO. Of 4K Segs In Local Trace Tbl (SYSPTLSZ) 16
LOGLOAD Value                   (SYSPLOGL) 50000
Maximum Number Of Background IDS (SYSPIDB ) 30
Maximum Number Of Foreground IDS (SYSPIDF ) 30
Max Number Of Concurrent Threads (SYSPCT ) 100
Max Number Active Remote Threads (SYSPRMT ) 100
Level Of DSN6SYSP              (SYSPLVL ) 99
ADMF MVS Reserve (40K)         (SYSPDB1M) 40960
.
.
.
*-----*
*          DSN6FAC: Start Option For DDF Facility                    *
*-----*
Number Of Facility Entries      (FACNM ) 1
Idle Thread TIMEOUT Interval   (FACTOIN ) 0
Facility Name                   (FACNAME ) DDF
DDF Flag Byte                   (FACSTART) A
RLF Nolimit                     (FACRLFER) YES
RLF Norun                       (FACRLFER) NO
.
.
.
*-----*
*          DSN6GRP: Group Initialization Parameters                    *
*-----*
Level Of Structure              (GRPLVL ) DSN410
Group Name                     (GRPNAME ) DSNCAT
Member Name                    (GRPMNAME) DSN1
Data Sharing                    (GRPDSHR ) NO

```

User Defined Function (UDF)

User Defined Function is a set of predefined DB2I2 commands, which group together to serve as a user defined function. You can specify host variables in your user defined function, and then substitute them at the time when you invoke your UDF.

There are two types of User Defined Functions: system UDF and user UDF. A system UDF located in the DB2I2 system library and shipped with DB2I2 system. A user UDF is created by you and stored in your own library. All UDF, whether it is system UDF or a user UDF, must be invoked with DB2I2 BACTH command together with ICMD command option.

To invoke a system UDF, you use BATCH ICMD=*system.UDF.name. For example ICMD=*ACCCOMP to invoke PLAN_TABLE comparison between two different generations of the same program.

To invoke a user UDF, you use BATCH ICMD=your.UDF.file. Both types of UDF support host variable substitution. The format of the host variable is &hostvar=host-var-value.

The following system User Defined Functions are shipped with the DB2I2 product to demonstrate how to build a user defined function.

ACCCOMP

ACCCOMP is a system UDF, which provides scripts to compare different generations of program from the same DB2 PLAN_TABLE.

The format to run ACCCOMP is:

```
BATCH ICMD=*ACCCOMP &owner=xxxxxxx &pg=xxxxxxx
```

Where &owner is the owner name of plan_table

&pg is the program name

The following DB2I2 command will be included in the generated batch JCL.

```

-----
-- ACCCOMP:  Access comparison report
--           Use EXPLAINP command to generate Exaplin Report to
--           Compare the current and previous generation of the
--           Explain output.
-- Format:
--   BATCH ICMD=*ACCCOMP &owner=xxxxxxx &pg=xxxxxxx
--           Where &owner is the owner name of plan_table
--           &pg is the program name
--
-- Note:  Change the following:
--         PG=pgname for appropriate DB2 package name
--         chnage PG=pgname to PL=plname for PLAN comparison.
-----
REXX  IDD=REXXDEL
EXPLAINP O=&owner PG=&pg ODSN=acccomp.wk1 DET=N GN=0
EXPLAINP O=&owner PG=&pg ODSN=acccomp.wk2 DET=N GN=-1
SUPERF  acccomp.wk1 acccomp.wk2 ODSN=acccomp.output
FLIST  acccomp.output 132
//REXXDEL DD DATA,DLM=AA
/* -- REXX -- */
X=MSG("OFF")
ADDRESS TSO
"DELETE acccomp.wk1"

```

DB212 Reference Manual

```
"DELETE acccomp.wk2"  
"DELETE acccomp.output"  
RETURN 0  
AA
```

ACCCOMPR

ACCCOMPR is a system UDF, which provides scripts to compare the current generation of program from the two different DB2 PLAN_TABLE from different locations.

The format to run ACCCOMPR is:

```
BATCH ICMD=*ACCCOMPR  &ownr1=xxxxxxx &loc1=xxxxxxx  +
                      &ownr2=xxxxxxx &loc2=xxxxxxx  +
                      &pg=xxxxxxx
```

Where &ownr1 is the owner name of the first plan_table
 &loc1 is the location name of the first plan_table
 &ownr2 is the owner name of the 2nd plan_table
 &loc2 is the location name of the 2nd plan_table
 &pg is the program name

The following DB2I2 command will be included in the generated batch JCL.

```
-----
-- ACCCOMPR: Access comparison report                                --
--       Use EXPLAINP command to generate Explain Report to        --
--       Compare the current generation of Explain output form     --
--       2 different locations.                                     --
-- Format:                                                         --
--   BATCH ICMD=*ACCCOMPR &ownr1=xxxxxxx &loc1=xxxxxxx          + --
--                       &ownr2=xxxxxxx &loc2=xxxxxxx          + --
--                       &pg=xxxxxxx                            --
--                                                                 --
--   Note: Please change the following:                             --
--       &ownr1  owner name of the first plan_table              --
--       &loc1   location name of the first plan_table           --
--       &ownr2  owner name of the 2nd plan_table                --
--       &loc2   location name of the 2nd plan_table             --
--       &pg     for appropriate DB2 package name                --
--               change PG=pgname to PL=plname for              --
--               DB2 plan comparison instead of                   --
--               DB2 package comparison.                           --
-----
REXX  IDD=REXXDEL
CONNECT(&loc1)
EXPLAINP O=&ownr1 PG=&pg ODSN=acccomp.wk1 DET=N GN=0
CONNECT(&loc2)
EXPLAINP O=&ownr2 PG=&pg ODSN=acccomp.wk2 DET=N GN=0
CONNECT(RESET)
SUPERF acccomp.wk1 acccomp.wk2 ODSN=acccomp.output
FLIST acccomp.output 132
//REXXDEL DD DATA,DLM=AA
/* -- REXX -- */
  X=MSG("OFF")
  ADDRESS TSO
  "DELETE acccomp.wk1"
  "DELETE acccomp.wk2"
  "DELETE acccomp.output"
RETURN 0
AA
```


GENURLD

GENURLD is a system UDF, which generates DSNTIAUL table unloading JCL, followed by LOAD, REPAIR NOCOPYPENDING and RUNSTATS job steps.

The format to run GENURLD is:

```
BATCH ICMD=*GENURLD &JCLWS='build.jcl' ODSN=gen.jcl
```

Where 'gen.jcl' contains the output JCL to generate the build.jcl. You can keep this file and reuse it in the future.

'bulid.jcl' contains the output JCL after submit the 'gen.jcl'

```
-----
-- GENURLD:   Generate DSNTIAUL unload and LOAD JCL driver   --
--           Select TB line objects and then issue the following: --
--           -----
--   BATCH ICMD=*GENURLD &JCLWS=your.work.jcl
--           -----
--   Specify ODSN=your.output.jcl option to save the changes
--   you made here so that you can reuse it in the future
--           -----
--   Please replace <?????????> under GLOBDD
--   with appropriate values before submit the job
--   for example,
--   change <GENERATED.TB> to MY.WORK.TB
--           -----
--   Set Global Variables to be used in Unload & Reload process
--           -----
SETG      IDD=GLOBDD
BATCH     ICMD=*GENURLD1 +
          ODSN=&JCLWS +
          &OP1 &OP2 &OP3 &OP4 &OP5 &OP6 &OP7 &OP8 &OP9 +
          &OP10 &OP11 &OP12 &OP13 &OP14
TSO       SUBMIT &JCLWS
//GLOBDD  DD DATA,DLM=GG
-----
-- Change the following Global Variables based on their definitions
--           -----
--   Please replace <?????????> under GLOBDD
--   with appropriate values before submit the job
--   for example,
--   change <GENERATED.TB> to MY.WORK.TB
--           -----
-- <GENERATED.TB>          generated work TB file for DSNTIAUL
-- <OUTPUT.JCL>           generated output JCL
-- <JOB.PARM>              generated job parm work PDS
-- <LOAD.TB>              generated work TB file for LOAD
-- <WHERE.DS>             generated work WHERE predicates file
-- <LOAD.PARMUTIL>        PARMUTIL LOAD input file
-- <REPAIR.NCOPY.PARMUTIL> PARMUTIL REPAIR NOCOPYPEND input file
-- <RUNSTATS.PARMUTIL>    PARMUTIL RUNSTATS input file
-- <fromlocation>         location name for the source table
-- <tolocation>           location name for the traget table
-- <fromclas>             JOB CLASS for source table unload JOB
-- <toclas>               JOB CLASS for target table load JOB
-- <dataset-prefix>      work file and utility file dataset prefix
-- <##>                  Number of cylinders for work files
--           -----
GV &OP1=&TBDSWS=<GENERATED.TB>
GV &OP2=&UNLDJWS=<OUTPUT.JCL>
GV &OP3=&JOBCNTL=<JOB.PARM>
GV &OP4=&LDTBWS=<LOAD.TB>
GV &OP5=&WHEREWS=<WHERE.DS>
GV &OP6=&PUTILL=<LOAD.PARMUTIL>
GV &OP7=&PUTILNC=<REPAIR.NCOPY.PARMUTIL>
GV &OP8=&PUTILRS=<RUNSTATS.PARMUTIL>
GV &OP9=&FROMLOC=<fromlocation>
```

DB2I2 Reference Manual

```
GV &OP10=&TOLOC=<tolocation>  
GV &OP11=&FROMCLAS=<from-job-class>  
GV &OP12=&TOCLAS=<to-job-class>  
GV &OP13=&DSPREWS=<dataset-prefix>  
GV &OP14=&WKSPWS=<##>  
GG
```

OBJCOMP

OBJCOMP is a system UDF, which provides scripts to compare two different DB2 Objects. The format to run OBJCOMP is:

```
BATCH ICMD=*OBJCOMP &loc1=loc1 &loc=loc2 &objt=objt &obj=obj
```

Where &loc1 is the location name of the first object
 &loc2 is the location name of the 2nd object
 &to is the user id to be removed
 &objt is object type DB, TS, TB or IX
 &obj is the name of the object

```
-----
-- OBJCOMP: comparing 2 db2 object DB, TS, TB, IX from same location--
-- or different locations. It contains                               --
-- DDL (or MIGR) db2i2 command to generate ddl scripts             --
-- EDIT exit to remove comment from ddl scripts, and              --
-- SUPERC to compare 2 ddl scripts                                  --
-- Format:                                                         --
-- BATCH ICMD=*OBJCOMP &loc1=loc1 &loc=loc2 &objt=objt &obj=obj --
-- Where &loc1 is the location name of the first object           --
--       &loc2 is the location name of the 2nd object            --
--       &to is the user id to be removed                         --
--       &objt is object type DB, TS, TB or IX                   --
--       &obj is the name of the object                           --
--                                                                 --
-- *****                                                       --
-- FILE USAGE:                                                    --
--   objcomp.wk1&suff - contains DDL or MIGR output form LINEOBJ1 --
--   objcomp.wk2&suff - contains DDL or MIGR output form LINEOBJ2 --
--   objcomp.out&suff - contains SUPERC comparison output         --
-----
-- Issue REXX DELWS to cleanup work files                          (1) --
-----
REXX IDD=DELWS
-----
-- Either issue DDL or MIGR db2i2 command to generate ddl scripts --
-- save the output ddl scripts in work file                        (2-5) --
-----
CONNECT(&loc1)
DDL IDD=LINEOBJ1 ODSN=objcomp.wk1&suff
MIGR IDD=LINEOBJ1 ODSN=objcomp.wk1&suff +
AL=Y VW=Y SY=Y BIND=Y GRANT=Y RI=B +
-- the following for DB2 V6 only
DT=Y LO=Y
CONNECT(&loc2)
DDL IDD=LINEOBJ2 ODSN=objcomp.wk2&suff
MIGR IDD=LINEOBJ2 ODSN=objcomp.wk2&suff +
AL=Y VW=Y SY=Y BIND=Y GRANT=Y RI=B +
-- the following for DB2 V6 only
DT=Y LO=Y
CONNECT(RESET)
-----
-- Remove comment from both wk1 and wk2 files                    (6- ) --
-- >>> add other string to be excluded from comparison          <<<--
-----
EDIT objcomp.wk1&suff
X '--' 1 ALL
DEL X ALL
END_EDIT
EDIT objcomp.wk2&suff
X '--' 1 ALL
DEL X ALL
END_EDIT
-----
-- Issue SUPERC to compare two work files and save the output    --
-- in objcomp.out&suff                                           --
-----
```

```
SUPERC objcomp.wk1&stuff objcomp.wk2&stuff ODSN=objcomp.OUT&stuff
-----
-- Print objcomp.out&stuff with FLIST command (lrecl option 131) --
-----
FLIST objcomp.out&stuff 131
//DELWS DD DATA,DLM=AA
/* ---REXX-----
  DELWS: Delete work files
----- */
  x=MSG('OFF')
  Address TSO
  'Delete objcomp.wk1&stuff'
  'Delete objcomp.wk2&stuff'
  'Delete objcomp.out&stuff'
Return 0
AA
//LINEOBJ1 DD *
&objt &obj
//LINEOBJ2 DD *
&objt &obj
```

REMOVEID

REMOVEID is a system UDF, which provides scripts to help you to remove a user ID from your DB2 sub-system. The format to run REMOVEID is:

```
BATCH ICMD=*REMOVEID &from=xxxxxxx &to=xxxxxxx &diff=
```

Where &from is the user id to be removed
 &to is the user id to be removed
 &diff is the work file suffix which allows you
 to run multiple REMOVEID jobs concurrently

```
-----
-- REMOVEID: DB2I2 udf to remove a user ID from system --
-- Format: --
-- BATCH ICMD=*REMOVEID &from=???????? &to=xxxxxxx &diff= --
-- Where &from is the user id to be removed --
-- &to is the user id to be removed --
-- &diff is the work file suffix which allows you --
-- to run multiple REMOVEID jobs concurrently --
-- -----
-- *****
-- 1. Generate and rebind all packages/plans with owner = '????????' --
-- 2. Copyauth to generate ???????? as grantor information and --
-- Copy the authorization to xxxxxxxx --
-- 3. Revoke to generate revoke DCL from ???????? --
-- *****
-- Note: --
-- *****
-- Change ???????? to actual ID you want to remove from your system --
-- xxxxxxxx to Receiving ID --
-----
Rexx IDD=DELDD
Run Idd=selsql Odsn=removeid.wkl&diff &uid=&from T=N Notfound(SKIP 1)
Rebind Idsn=removeid.wkl&diff Odsn=removeid.out&diff DET=Y O=&to
Copyauth Odsn=removeid.out&diff Append SQLID=&To Grantor
Revoke Odsn=removeid.out&diff Append
Exec Idsn=removeid.out&diff
//DELDD DD DATA,DLM=AA
/* REXX --delete all work files----- */
x=MSG('OFF')
Address TSO
'Delete removeid.wkl&diff'
'Delete removeid.out&diff'
"Alloc fi(ol) ds(removeid.out&diff) new space(1,1) cpls",
"recfm(f b) lrecl(80) blksize(0) reuse unit(sysda)"
"Free fi(ol)"
Return 0
AA
//SELSQL DD DATA,DLM=BB
Select 'PG' ||strip(location)||'.'||strip(collid)||'.'||strip(name)
||'.'||strip(version)
From sysibm.syspackage
Where owner = '&uid'
union all
Select 'PL' ||name
From sysibm.sysplan
Where creator = '&uid'
BB
```

TUNEPG

TUNEPG is a system UDF, which provides scripts to tune DB2 packages.
The format to run TUNEPG is:

```
BATCH ICMD=*TUNEPG &OUTPUT=xxxxxxx &TSOID=xxxxxxx &stuff=
```

Where &OUTPUT is the output file
&TSOID is your TSOID, must contain BIND authority
&stuff is the work file suffix which allows you to run multiple TUNEPG jobs

```
-----
-- TUNEPG: Tuning Assist Using PG Line Objects as Input --
-- Format: --
-- BATCH ICMD=*TUNEPG &OUTPUT=?output? &TSOID=xxxxxxx &stuff= --
-- Where &OUTPUT is the output file --
-- &TSOID is your TSOID, must contain BIND authority--
-- &stuff is the work file suffix which allows you --
-- to run multiple TUNEPG jobs concurrently --
--
-- ***** --
-- File Usage: --
-- &OUTPUT - Output file ..... --
-- tunePg.WS&stuff - work file for TS and IX command ..... --
-- tunePg.W1&stuff - work file for TB command ..... --
-- tunePg.W2&stuff - work file for DELETE FROM PLAN_TABLE... --
-- tunePg.W3&stuff - work file for EXPLAINP..... --
-- tunePg.W4&stuff - work file for BIND COPY..... --
-- tunePg.W5&stuff - work file for BATCH ..... --
-- tunePg.W6&stuff - work file for FREE PACKAGE..... --
-----
-- ***** --
-- Clean up all workfiles (1)
-- *****
REXX IDD=DELWS (2)
-- *****
-- Drill Down from PG line objects
-- *****
ODSN=&output
-- *****
-- Get TB line objects (3)
-- *****
TB ODSN=tunePg.W1&stuff NOTFOUND(SKIP 99)
-- *****
-- Remove Duplicate TB line objects (4-6)
-- *****
EDIT tunePg.W1&stuff
DISTINCT 1,30
END_EDIT
-- *****
-- Generates seletion path statistics (7)
-- *****
SELPATHV IDSN=tunePg.W1&stuff ODSN=&output APPEND
-- *****
-- Get TS line objects (8)
-- *****
TS IDSN=tunePg.W1&stuff ODSN=tunePg.WS&stuff
-- *****
-- Get STATS from TS line objects (9)
-- *****
STATS IDSN=tunePg.WS&stuff ODSN=&output APPEND
-- *****
-- Get IX line objects (10)
-- *****
IX IDSN=tunePg.W1&stuff ODSN=tunePg.WS&stuff NOTFOUND(SKIP 7)
-- *****
-- Drill Down IX line objects (11)
-- *****
IDSN=tunePg.WS&stuff ODSN=&output APPEND
-- *****
```

```

-- Call SQLDEL ddname to generate Delete SQL & EXPLAINP          (12)
-- *****
REXX      IDD=SQLDEL
-- *****
-- EXEC Delete SQL to delete entries from PLAN_TABLE            (13)
-- *****
EXEC      IDSN=tunepg.W2&suff  ERROR(CONTINUE)
-- *****
-- Generate BIND COPY from PG line objects                      (14)
-- *****
BIND COPY O=USER      CL=USER  ODSN=tunepg.W4&suff
-- *****
-- Edit and change BIND COPY options                            (15-18)
-- *****
EDIT      tunepg.W4&suff  ERROR(CONTINUE)
C 'EXPLAIN(NO)' 'EXPLAIN(YES)' ALL
C '(ADD)' '(REPLACE)' ALL
END_EDIT
-- *****
-- EXEC BIND COPY & FREE                                        (19-20)
-- *****
EXEC      IDSN=tunepg.W4&suff
EXEC      IDSN=tunepg.W6&suff
-- *****
-- BATCH command and use EXPLAINP lines as input                (21)
-- *****
BATCH     ICMD=tunepg.W3&suff  ODSN=tunepg.W5&suff
-- *****
-- SUBMIT JOB to run EXPLAINP                                  (28)
-- *****
TSO SUBMIT tunepg.W5&suff
//DELWS DD DATA,DLM=AA
/* --REXX-----
DELWS: Delete work files
----- */
x=MSG('OFF')
Address TSO
'delete tunepg.ws&suff'
'delete tunepg.w1&suff'
'delete tunepg.w2&suff'
'delete tunepg.w3&suff'
'delete tunepg.w4&suff'
'delete tunepg.w5&suff'
'delete tunepg.w6&suff'
'delete &output'
"Alloc fi(o1) ds(&output) new space(1,1) tracks recfm(f b)",
"lrecl(133) blksize(0) reuse unit(sysda)"
"Free fi(o1)"
Return 0
AA
//SQLDEL DD DATA,DLM=BB
/* --REXX-----
SQLDEL: Generates DELETE FROM PLAN_TABLE... in W2
Generates EXPLAINP commands ..... in W3
Generates FREE commands ..... in W6
Input LINEOBJ DD contains PG line objects
----- */
x=MSG('OFF')
Address TSO
"Alloc fi(w2) ds(tunepg.w2&suff) new space(1,1) tracks recfm(f b)",
"lrecl(80) blksize(0) reuse unit(sysda)"
"Alloc fi(w3) ds(tunepg.w3&suff) new space(1,1) tracks recfm(f b)",
"lrecl(80) blksize(0) reuse unit(sysda)"
"Alloc fi(w6) ds(tunepg.w6&suff) new space(1,1) tracks recfm(f b)",
"lrecl(80) blksize(0) reuse unit(sysda)"
outrec.1 = "DELETE FROM PLAN_TABLE"
outrec.2 = " WHERE PROGNAM IN ("
"Execio * Diskr LINEOBJ(Stem inrec. Finis"
j = 2
jj= 0
dlm=' '
Do i = 1 to inrec.0
Parse Value inrec.i with j1 '.' j2 '.' ws '.'
j = j + 1
outrec.j =dlm ""strip(ws)""
jj= jj+ 1
outrec2.JJ ="EXPLAINP ODSN=&output APPEND PG="STRIP(WS)
outrec6.JJ ="DB2CMD FREE PACKAGE(&TSOID."STRIP(WS)""

```

```
        dlm=', '  
End  
j = j + 1  
outrec.j="  );"  
"Execio" j "Diskw w2(Stem outrec. Finis"  
"Execio" jj "Diskw w3(Stem outrec2. Finis"  
"Execio" jj "Diskw w6(Stem outrec6. Finis"  
"Free fi(w2)"  
"Free fi(w3)"  
"Free fi(w6)"  
Return 0  
BB
```


TUNETB

TUNETB is a system UDF, which provides scripts to tune DB2 tables.
The format to run TUNETB is:

```
BATCH ICMD=*TUNETB &OUTPUT=xxxxxxx &TSOID=xxxxxxx &suff=
```

Where &OUTPUT is the output file
&TSOID is your TSOID, must contain BIND authority
&suff is the work file suffix which allows you to run multiple TUNETB jobs

```
-----
-- TUNETB:  Tuning Assist Using TB Line Objects as Input          --
--           sample for ICMD host variable substitute            --
-- Format:  -----                                           --
--   BATCH ICMD=*TUNETB &OUTPUT=?output? &TSOID=xxxxxxx &suff= --
--           Where &OUTPUT is the output file                  --
--           &TSOID is your TSOID, must contain BIND authority--
--           &suff is the work file suffix which allows you   --
--           to run multiple TUNETB jobs concurrently          --
-- -----
-- *****
--   File Usage:
--   &OUTPUT - Output file .....
--   tunetb.WS&suff - work file for TS and IX command .....
--   tunetb.W1&suff - work file for PG command .....
--   tunetb.W2&suff - work file for DELETE FROM PLAN_TABLE....
--   tunetb.W3&suff - work file for EXPLAINP.....
--   tunetb.W4&suff - work file for BIND COPY.....
--   tunetb.W5&suff - work file for BATCH .....
--   tunetb.W6&suff - work file for FREE PACKAGE.....
-----
REXX          IDD=DELWS
REXX          IDD=DELO
REXX          IDD=ALLOCO
SELPATHV     ODSN=&OUTPUT
TS           ODSN=tunetb.WS&suff
STATS       IDSN=tunetb.WS&suff ODSN=&OUTPUT APPEND
IX          ODSN=tunetb.WS&suff NOTFOUND(SKIP 1)
IDSN=WS&suff ODSN=&OUTPUT APPEND
PG          ODSN=tunetb.W1&suff NOTFOUND(SKIP 15)
EDIT        tunetb.W1&suff
DISTINCT    1,30
END_EDIT
IDSN=W1&suff ODSN=&OUTPUT APPEND
REXX        IDD=SQLDEL
EXEC        IDSN=tunetb.W2&suff
BIND COPY   O=USER      CL=USER      IDSN=tunetb.w3&suff      +
            ODSN=tunetb.W4&suff
EDIT        tunetb.W4&suff      ERROR(CONTINUE)
C 'EXPLAIN(NO)' 'EXPLAIN(YES)' ALL
C '(ADD)'      '(REPLACE)' ALL
END_EDIT
EXEC        IDSN=tunetb.W4&suff
EXEC        IDSN=tunetb.W6&suff
BATCH      ICMD=tunetb.W3&suff ODSN=tunetb.W5&suff
TSO SUBMIT  tunetb.W5&suff
REXX       IDD=DELWS
//DELWS DD DATA,DLM=AA
/* ---REXX-----
DELWS: Delete work files
----- */
x=MSG('OFF')
Address TSO
'delete tunetb.ws&suff'
'delete tunetb.w1&suff'
'delete tunetb.w2&suff'
'delete tunetb.w3&suff'
'delete tunetb.w4&suff'
'delete tunetb.w5&suff'
'delete tunetb.w6&suff'
```

```

Return 0
AA
//DELO DD DATA,DLM=AB
/* --REXX-----
DELO: Delete Output file
----- */
x=MSG('OFF')
ADDRESS TSO
'delete &OUTPUT'
Return 0
AB
//ALLOCO DD DATA,DLM=BB
/* --REXX-----
ALLOCO: Allocate Output file
----- */
x=MSG('OFF')
Address TSO
"Alloc fi(ol) ds(&OUTPUT)",
  "new space(1,1) tracks recfm(f b) lrecl(133)",
  "blksize(0) reuse unit(sysda)"
"Free fi(ol)"
Return 0
BB
//SQLDEL DD DATA,DLM=CC
/* --REXX-----
SQLDEL: Generates DELETE FROM PLAN_TABLE... in W2
        Generates EXPLAINP commands ..... in W3
        Generates FREE commands ..... in W6
        Input W1 contains PG line objects
----- */
x=MSG('OFF')
Address TSO
"Alloc fi(w1) ds(tunetb.w1&suff) shr reuse"
"Alloc fi(w2) ds(tunetb.w2&suff)",
  "new space(1,1) tracks recfm(f b) lrecl(80)",
  "blksize(0) reuse unit(sysda)"
"Alloc fi(w3) ds(tunetb.w3&suff)",
  "new space(1,1) tracks recfm(f b) lrecl(80)",
  "blksize(0) reuse unit(sysda)"
"Alloc fi(w6) ds(tunetb.w3&suff)",
  "new space(1,1) tracks recfm(f b) lrecl(80)",
  "blksize(0) reuse unit(sysda)"
outrec.1="DELETE FROM PLAN_TABLE"
outrec.2=" WHERE PROGNAME IN ("
"Execio * Diskr w1(Stem inrec. Finis"
j = 2
jj = 0
dml = ' '
Do i = 1 to inrec.0
  parse value inrec.i with j1 '.' j2 '.' ws '.'
  j = j + 1
  outrec.j =dml "'strip(ws)'"
  jj= jj+ 1
  outrec2.jj = "EXPLAINP ODSN=&OUTPUT APPEND PG="STRIP(WS)
  /* ----- */
  /* Modify the following &TSOID to your TSOID */
  /* ----- */
  outrec6.jj = "DB2CMD FREE PACKAGE(&TSOID."strip(ws)")"
  dml = ','
End
j = j + 1
outrec.j = " );"
"Execio" j "Diskw w2(Stem outrec. Finis"
"Execio" jj "Diskw w3(Stem outrec2. Finis"
"Execio" jj "Diskw w6(Stem outrec6. Finis"
"Free fi(w1)"
"Free fi(w2)"
"Free fi(w3)"
"Free fi(w6)"
Return 0
CC

```

Sample Script

The following sample scripts are shipped with DB2I2, which located in the system library and can be invoked with BATCH command with ICMD=*name command option. They work just like UDF. Except there are no host variable substitute:

COPYSTAT

COPYSTAT is a sample script, which provides scripts to help you to copy catalog statistics from one location to another.

```

-----
-- COPYSTAT: DB2I2 udf to copy catalog statistics from 1 table to --
--          another table from different remote location --
-- Note: A TB line or a group of TB lines can be used as input for --
--       this COPYSTAT process. --
-- -----
-- Detail Description: --
-- -----
-- 1. Connect to source location and issue SELPATHV to get access --
--    path statistics into T1 --
-- 2. Connect(RESET) and use ED exit to change the source info --
--    to target information --
--    Change CU to CI for SYSIBM.SYSCOLDIST information --
-- 3. Use QBUILD and RUN to generate DELETE SQL --
--    to delete old SYSCOLDIST catalog statistics --
-- 4. Use EXEC to execute the generated DELETE SQL and --
--    UPDATE and INSERT SQL generated from SELPATHU --
-- ***** --
-- Note: --
-- ***** --
-- Change location-name to the source location name --
-- databasei to source database name --
-- databaseo to target database name --
-- tablespacei to source tablespace name --
-- tablespaceo to target tablespace name --
-- tablecreatori to source tablecreator name --
-- tablecreatoro to target tablecreator name --
-- tablei to source table name --
-- tableo to target table name --
-- indexcreatori to source indexcreator name --
-- indexcreatoro to target indexcreator name --
-- indexi to source index name --
-- indexo to target index name --
-----
Rexx IDD=DELDD
Connect(location-name)
Selpathv OPTION=CICU odsn=t1
Connect(RESET)
Ed t1 MACRO(IDD=EDITDD)
Selpathu idsn=t1 odsn=t2
Qbuild F1=CREATOR F2=NAME odsn=t3
Run IDD=DELSQLDD &WHERE=t3 odsn=t4 T=N LIMIT(999)
EXEC idsn=t4
EXEC idsn=t2
//DELDD DD DATA,DLM=AA
/* REXX --delete all work files----- */
x=MSG('OFF')
Address TSO 'DELETE T1'
Address TSO 'DELETE T2'
Address TSO 'DELETE T3'
Address TSO 'DELETE T4'
Return 0

```

```
AA
//EDITDD DD DATA,DLM=BB
C 'databasei' 'databaseo' all
C 'tablespacei' 'tablespaceo' all
C 'tablecreatori' 'tablecreatoro' all
C 'tablei' 'tableo' all
C 'indexcreatori' 'indexcreatoro' all
C 'indexi' 'indexo' all
C 'CU SYSCOLDIST ' 'CI SYSCOLDIST ' all
BB
//DELSQLDD DD DATA,DLM=CC
SELECT 'DELETE FROM SYSIBM.SYSCOLDIST |
      ' WHERE TBOWNER = ' ' | CREATOR |
      ' ' AND TBNAME = ' ' | NAME | ' '
FROM SYSIBM.SYSTABLES
WHERE
-INC &WHERE
CC
```

DSCOPYB, DSCOPYBS

DSCOPYB and DSCOPYBS are RXDB2I2 script to generate DSN1COPY from Source TS to Target TS. If there is an error during process, a temp TS is used to copy the structure from source and DSCOPY from source to Temp followed by a CURSOR LOAD from temp to Target. These scripts perform the same function as DSCOPY2 and DSCOPY2S with Batch Interface.

DSCOPYBS

```

-----
-- DSCOPY2S: Sample Rxdb2i2 Script
--          Invoke DSCOPY to copy TS from Source to Target
--          If there is an error:
--            - Generate DDL and Build Temp TS,TB with EXEC
--            - DSCOPY from Source to Temp
--            - LOAD from Temp to Target with CURSOR
-- ** LOADPARAM needed to be defined with PARMUTIL LOAD
-----
Db2i2rex "Rxdb2i2 idsn=*dscopy2 &OUTPUT='output.jcl'",
         "&TSSID=ssid &SLOC=source-location",
         "&HLQ=imagecopy-high-level-qualifier",
         "&SCRE=source-tbcreator &NAME=tbname",
         "&TCRE=target-tbcreator",
--      Temporary DB, TS and TB to be used if Structure is different
         "&TEMPDB=temp-db",
         "&TEMPCRE=temp-tbcreator",
         "&TEMPSG=temp-stogroup",
         "&TEMPID=temp-sqlid",
         "&TEMPVCAT=temp-vcat-name",
         "&LOADPARAM=load-parmutil-with-cursor"

```

DSCOPYB

```

-----
-- Build SC lines for Source DSCOPY
-----
Lineobj DDname=dd0
-----
Select 'SC &TSSID\&SLOC.'||Strip(DBNAME)||'.'||TSNAME
From &SLOC.SYSIBM.SYSTABLES
Where CREATOR = '&SCRE'
And NAME = '&NAME'
End_Lineobj
-----
-- Build SC lines for Target DSCOPY
-----
Lineobj DDname=ddl
-----
Select Strip(DBNAME)||'.'||TSNAME
From SYSIBM.SYSTABLES
Where CREATOR = '&TCRE'
And NAME = '&NAME'
End_Lineobj
-----
-- Build TS and TB lines For Temp DDL
-----
Lineobj DDname=dd2
-----
Select 'TS '||
       Strip(DBNAME)||'.'||Strip(tsname)
From &SLOC.SYSIBM.SYSTABLES
Where CREATOR = '&SCRE'
And NAME = '&NAME'
Union All
SELECT 'TB '||
       Strip(creator)||'.'||Strip(name)
From &SLOC.SYSIBM.SYSTABLES
Where CREATOR = '&SCRE'
And NAME = '&NAME'
End_Lineobj

```

```

-----
-- ED Macro to insert a Temp DROP TABLESPACE
-----
Lineobj DDname=dd3
-----
F 'CREATE TABLESPACE' FIRST
CLINE
REXX tsname = word(CLINE,3)
INSERTB * DROP TABLESPACE &TEMPDB.{TSNAME}; COMMIT;
F 'PRIMARY' FIRST
DELETE * *
TOP
End_Lineobj
-----
-- ED Macro to build SC for Source to Temp DSCOPY
-----
Lineobj DDname=dd4
-----
F 'SC' 1 FIRST
CLINE
REXX Parse var cline v1 v2 v3
REXX Parse value v3 with v31 '.' v32
REXX cline = v1 v2 '&TEMPDB.'v32
CLINE WRITE
End_Lineobj
-----
-- Build TB line with CARD information for LOAD
-----
Lineobj DDname=dd5
-----
SELECT 'TB &TCRE.&NAME', card
From &SLOC.SYSIBM.SYSTABLES
Where CREATOR = '&SCRE'
And NAME = '&NAME'
End_Lineobj
-----
-- ED macro to build Temp EXEC SQL cursor for LOAD
-----
Lineobj DDname=dd6
-----
DELETE * *
INSERTA 1 IDD=DD7
End_Lineobj
-----
-- Temp DECLARE Cursor skelton for dd6
-----
Lineobj DDname=dd7
-----
EXEC SQL
DECLARE C1 CURSOR FOR SELECT * FROM &TEMPCRE.&NAME
ENDEXEC
End_Lineobj
-----
-- Beginning Of Rxdb2i2 Script --
-----
t = time()
suff = ".T"substr(t,1,2)||substr(t,4,2)||substr(t,7,2)
Db2i2rex "Run Idd=dd0 Odsn=t1"suff" t=n trunc=y"
Address Tso "Free fi(dd0)"
If RxNoTFnd() = 'Y' Then
Do
Say " ** No Source Table Found - &SCRE.&NAME"
Call Delete_wk
Exit 16
End
Db2i2rex "Run Idd=ddl Odsn=t1"suff" Append t=n trunc=y"
Address Tso "Free fi(dd1)"
If RxNoTFnd() = 'Y' Then
Do
Say " ** No Target Table Found - &TCRE.&NAME"
Call Delete_wk
Exit 16
End
Address Tso "Alloc Fi(dd1) Ds(t1"suff") Shr Reuse"
"Execio * DiskR ddl(Stem R. Finis"
r.1 = Strip(r.1) Strip(r.2)
"Execio 1 DiskW ddl(Stem R. Finis"
Address Tso "Free fi(dd1)"

```

```

Db2i2rex "Dscopy idsn=t1"suff" Odsn=&OUTPUT",
        "hlq=&HLQ cont=y errdsn=errors" suff
If RxRc() = 4 Then
Do
  Db2i2rex "Run      Idd=dd2 Odsn=t2"suff" t=n trunc=y"
  Address Tso      "Free fi(dd2)"
  Db2i2rex "Connect(&SLOC)"
  Db2i2rex "DDL      idsn=t2"suff" Odsn=t3"suff",
        "sqlid=&TEMPID db=&TEMPDB cre=&TEMPCRE",
        "sg=&TEMPSG vcat=&TEMPVCAT"

  Db2i2rex "Connect(reset)"
  Db2i2rex "ED      t3"suff" Macro(Idd=dd3)"
  Address Tso      "Free fi(dd3)"
  Db2i2rex "Exec     idsn=t3"suff" RC0"
  Db2i2rex "ED      errors" suff" Macro(Idd=dd4)"
  Address Tso      "Free fi(dd4)"
  Db2i2rex "Dscopy  idsn=errors" suff" Odsn=t1"suff",
        "hlq=&HLQ cont=y errdsn=errors2" suff

  Db2i2rex "Run      Idd=dd5 Odsn=t4"suff" t=n"
  Address Tso      "Free fi(dd5)"
  Address Tso      "Fcpy t4"suff" Loadcur"
  Db2i2rex "ED      Loadcur Macro(Idd=dd6)"
  Address Tso      "Free fi(dd6)"
  Db2i2rex "Load     idsn=t4"suff" Odsn=t1"suff" Append",
        "jobcard=n step#=10",
        "parmutil=&LOADPARAM"

  Db2i2rex "Tso     Fcpy t1"suff" &OUTPUT"
  Call Delete_ws
End
Else
  Address Tso "Delete t1"suff
Return
Delete_ws:
  Address Tso "Delete t1"suff
  Address Tso "Delete t2"suff
  Address Tso "Delete t3"suff
  Address Tso "Delete t4"suff
  Address Tso "Delete loadcur"
  Address Tso "Delete errors" suff
Return
-- End Of Script

```

DSCOPY2, DSCOPY2S

DSCOPY2 and DSCOPY2S are RXDB2I2 script to generate DSNICOPY from Source TS to Target TS. If there is an error during process, a temp TS is used to copy the structure from source and DSCOPY from source to Temp followed by a CURSOR LOAD from temp to Target.

DSCOPYBS

```
-----
-- DSCOPYBS: Sample Rxdb2i2 Script(Same as DSCOPY2S with BATCH interface
--           Invoke DSCOPY to copy TS from Source to Target
--           If there is an error:
--             - Generate DDL and Build Temp TS,TB with EXEC
--             - DSCOPY from Source to Temp
--             - LOAD from Temp to Target with CURSOR
-- ** LOADPARM needed to be defined with PARMUTIL LOAD
-----
Db2i2rex "Batch icmd=*dscopyb lineobj=N &OUTPUT='output.jcl'",
         "&TSSID=ssid &SLOC=source-location",
         "&HLQ=imagecopy-high-level-qualifier",
         "&SCRE=source-tbcreator &NAME=tbname",
         "&TCRE=target-tbcreator",
--       Temporary DB, TS and TB to be used if Structure is different
         "&TEMPDB=temp-db",
         "&TEMPCRE=temp-tbcreator",
         "&TEMPSG=temp-stogroup",
         "&TEMPID=temp-sqlid",
         "&TEMPVCAT=temp-vcat-name",
         "&LOADPARM=load-parmutil-with-cursor"
```

DSCOPYB

```
RXDB2I2
//LINEOBJ DD DATA,DLM=LL
-----
-- Build SC lines for Source DSCOPY
-----
Lineobj DDname=dd0
-----
Select 'SC &TSSID&SLOC.'||Strip(DBNAME)||'.'||TSNAME
From &SLOC.SYSIBM.SYSTABLES
Where CREATOR = '&SCRE'
And NAME = '&NAME'
End_Lineobj
-----
-- Build SC lines for Target DSCOPY
-----
Lineobj DDname=ddl
-----
Select Strip(DBNAME)||'.'||TSNAME
From SYSIBM.SYSTABLES
Where CREATOR = '&TCRE'
And NAME = '&NAME'
End_Lineobj
-----
-- Build TS and TB lines For Temp DDL
-----
Lineobj DDname=dd2
-----
Select 'TS '||
       Strip(DBNAME)||'.'||Strip(tsname)
From &SLOC.SYSIBM.SYSTABLES
Where CREATOR = '&SCRE'
And NAME = '&NAME'
Union All
SELECT 'TB '||
       Strip(creator)||'.'||Strip(name)
From &SLOC.SYSIBM.SYSTABLES
Where CREATOR = '&SCRE'
And NAME = '&NAME'
End_Lineobj
-----
```



```

-- ED Macro to insert a Temp DROP TABLESPACE
-----
Lineobj DDname=dd3
-----
F 'CREATE TABLESPACE' FIRST
CLINE
REXX tsname = word(CLINE,3)
INSERTB * DROP TABLESPACE &TEMPDB.{TSNAME}; COMMIT;
F 'PRIMARY' FIRST
DELETE * *
TOP
End_Lineobj
-----
-- ED Macro to build SC for Source to Temp DSCOPY
-----
Lineobj DDname=dd4
-----
F 'SC' 1 FIRST
CLINE
REXX Parse var cline v1 v2 v3
REXX Parse value v3 with v31 '.' v32
REXX cline = v1 v2 '&TEMPDB.'v32
CLINE WRITE
End_Lineobj
-----
-- Build TB line with CARD information for LOAD
-----
Lineobj DDname=dd5
-----
SELECT 'TB &TCRE.&NAME', card
From &SLOC.SYSIBM.SYSTABLES
Where CREATOR = '&SCRE'
And NAME = '&NAME'
End_Lineobj
-----
-- ED macro to build Temp EXEC SQL cursor for LOAD
-----
Lineobj DDname=dd6
-----
DELETE * *
INSERTA 1 IDD=DD7
End_Lineobj
-----
-- Temp DECLARE Cursor skelton for dd6
-----
Lineobj DDname=dd7
-----
EXEC SQL
DECLARE C1 CURSOR FOR SELECT * FROM &TEMPCRE.&NAME
ENDEXEC
End_Lineobj
-----
-- Beginning Of Rxdb2i2 Script --
-----
t = time()
suff = ".T"substr(t,1,2)||substr(t,4,2)||substr(t,7,2)
Db2i2rex "Run Idd=dd0 Odsn=t1"suff" t=n trunc=y"
Address Tso "Free fi(dd0)"
If RxNoTFnd() = 'Y' Then
Do
Say " ** No Source Table Found - &SCRE.&NAME"
Call Delete_wk
Exit 16
End
Db2i2rex "Run Idd=ddl Odsn=t1"suff" Append t=n trunc=y"
Address Tso "Free fi(dd1)"
If RxNoTFnd() = 'Y' Then
Do
Say " ** No Target Table Found - &TCRE.&NAME"
Call Delete_wk
Exit 16
End
Address Tso "Alloc Fi(dd1) Ds(t1"suff") Shr Reuse"
"Execio * DiskR ddl(Stem R. Finis"
r.1 = Strip(r.1) Strip(r.2)
"Execio 1 DiskW ddl(Stem R. Finis"
Address Tso "Free fi(dd1)"
Db2i2rex "Dscopy idsn=t1"suff" Odsn=&OUTPUT",

```

```

                                "hlq=&HLQ cont=y errdsn=errors" suff
If RxRc() = 4 Then
Do
  Db2i2rex "Run      Idd=dd2 Odsn=t2"suff" t=n trunc=y"
  Address Tso      "Free fi(dd2)"
  Db2i2rex "Connect(&SLOC)"
  Db2i2rex "DDL      idsn=t2"suff" Odsn=t3"suff,
                                "sqlid=&TEMPID db=&TEMPDB cre=&TEMPCRE",
                                "sg=&TEMPSG vcat=&TEMPVCAT"
  Db2i2rex "Connect(reset)"
  Db2i2rex "ED      t3"suff" Macro(Idd=dd3)"
  Address Tso      "Free fi(dd3)"
  Db2i2rex "Exec    idsn=t3"suff" RC0"
  Db2i2rex "ED      errorsc"suff" Macro(Idd=dd4)"
  Address Tso      "Free fi(dd4)"
  Db2i2rex "Dscopy  idsn=errorsc"suff" Odsn=t1"suff,
                                "hlq=&HLQ cont=y errdsn=errors2"suff
  Db2i2rex "Run      Idd=dd5 Odsn=t4"suff" t=n"
  Address Tso      "Free fi(dd5)"
  Address Tso      "Fcpy t4"suff" Loadcur"
  Db2i2rex "ED      Loadcur Macro(Idd=dd6)"
  Address Tso      "Free fi(dd6)"
  Db2i2rex "Load    idsn=t4"suff" Odsn=t1"suff" Append",
                                "jobcard=n step#=10",
                                "parmutil=&LOADPARM"
  Db2i2rex "Tso     Fcpy t1"suff" &OUTPUT"
  Call Delete_ws
End
Else
  Address Tso "Delete t1"suff
Return
Delete_ws:
  Address Tso "Delete t1"suff
  Address Tso "Delete t2"suff
  Address Tso "Delete t3"suff
  Address Tso "Delete t4"suff
  Address Tso "Delete loadcur"
  Address Tso "Delete errorsc"suff
Return
-- End Of Script
LL

```

GENSCCMD

Gensccmd is a sample script, which provides scripts to help you to process DSCOPY command.

```

-----
-- GENSCCMD:  Canned Db2I2 commands to generate SC line object  --
--             for DSCOPY command and then generates DSCOPY jobs  --
--             --
-- ** Remove 1 of the 2 of the QBUILD command below to fit your  --
-- ** specific unload need  --
-----
-- for DB line objects
QBUILD ODSN=gensccmd.qbuild F1=DBNAME
-- for TS line objects
QBUILD ODSN=gensccmd.qbuild F1=DBNAME F2=TSNAME
-----
-- Preparing SC line by invoking GENSC user defined query  --
-----
-- Host Variables Usage:
-- =====
--   &FSSID - Source SSID
--   &FLOC  - Source LOCATION
--   &TSSID - Target SSID
--   &TLOC  - Target LOCATION
--   &RCVRIX- Specify &RCVRIX=RCVRIX=Y to generate REBUILD INDEX
--             &RCVRIX=RCVRIX=N No REBUILD INDEX step
--             &RCVRIX=          is the same as RCVRIX=Y
--   &ICDATE- Specify &ICDATE=ICDATE=yymmdd to request specific
--             date of image copy as input
--             &ICDATE=          choose most current full image
--             copy as input
--   &ICGEN - Specify &ICGEN=ICGEN=### to request specific
--             generation of image copy as input
--             ### must be <= 0
--             &ICGEN=          choose most current generation
--             of full image copy as input
--   &WHERE - Specify &WHERE='qbuild.where.predicate.dsname' to
--             include the WHERE predicates generated
--             from QBUILD command
-----
RUN  IDSN=*GENSC          +
     ODSN=scline.output  +
     &FSSID=source_ssid  +
     &FLOC=source_location +
     &TSSID=target_ssid   +
     &TLOC=target_location +
     &RCVRIX=RCVRIX=Y     +
     &ICDATE=            +
     &ICGEN=            +
     &WHERE=gensccmd.qbuild
-----
-- The following DSCOPY use image copy as input to generate  --
-- DSN1COPY jobs with jobname prefixed with DSCPY###  --
-- For every ## number of step, DB2I2 will start a new job  --
-- The generated jobs will be saved in gensccmd.jcl pds with  --
-- member name = dscopy###  --
-----
DSCOPY IDSN=scline.output  +
       ODSN=gensccmd.jcl(*) +
       ICOPY=Y              +
       JOBNM=dscopy###     +
       NEWJOB=##
-----
-- Submit each generated load JCL if needed  --
-----
ED      gensccmd.jcl(dscopy*)  MACRO(IDD=DDSUBMIT)
//DDSUBMIT DD DATA,DLM=BB
SUBMIT
BB

```

MIGRTB

MIGRTB is a sample script, which provides scripts to help you to migrate DB2 tables.

```

-----
-- MIGRTB: Generate MIGRATION JCL to assist you to migrate TB line--
--   File Usage:
--   O1 - Output file contains generated JCL.....
--   W0 - work file contains generated TB line object
--   W1 - work file contains EDIT script for SYSPUNCH
--   W2 - work file contains Migration scripts.....
--   W3 - work file contains TS line objects.....
--   W4 - work file contains PG line objects.....
--   W5 - work file contains REBIND line objects....
--   Usage Note:
--   Modify (4-7)   for appropriate Migration Scripts changes
--   Modify (11),(13),(14) to appropriate PARMUTIL dataset
--   created from PARMUTIL for LOAD, REPAIR NOCOPYPEND,
--   RUNSTATS
-----
-- *****
-- Call SETUP to delete all work files
-- *****
REXX      IDD=SETUP
-- *****
-- Modify the following host variable and run the sqlgen to
-- generate TB line objects for migration process
-- &CREATOR is table creator and
-- &TBNAME is table name
-- *****
RUN      IDD=SQLGEN ODSN=W0 LIMIT(9999) T=N      +
        &CREATOR='CREATOR1'                    +
        &TBNAME='TBNAME1','TBNAME2'
-- *****
-- Generates Migration script with MIGR
-- *****
MIGR     IDSN=W0 ODSN=W2 GRANT=Y
-- *****
-- Edit Migration scripts
-- *****
EDIT     W2
C 'CREATOR1' 'CREATORO' ALL
C 'DBNAME1' 'DBNAMEO' ALL
END_EDIT
-- *****
-- Generate DSNTIAUL JCL
-- *****
DSNTIAUL IDSN=W0 ODSN=O1
-- *****
-- Generate EDIT SYSPUNCH JCL append to O1 start from STEP011
-- *****
REXX     IDD=EDITGEN
BATCH   ICMW=W1 ODSN=O1 APPEND STEP#=10 JOBCARD=N
-- *****
-- Generate EXEC migrationJCL append to O1 start from STEP016
-- *****
BATCH   EXEC IDSN=W2 ODSN=O1 APPEND STEP#=15 JOBCARD=N
-- *****
-- Generate LOAD TABLE JCL append to O1 start from STEP021
-- *****
LOAD    UTIL(LOAD) IDSN=W0 ODSN=O1 APPEND STEP#=20 JOBCARD=N
-- *****
-- Get TS line objects
-- *****
TS      IDSN=W0 ODSN=W3
-- *****
-- Generate REPAIR NOCOPYPEND append to O1 start from STEP031
-- *****
REPAIR  UTIL(REPAIRNC) IDSN=W3 ODSN=O1 APPEND STEP#=30 JOBCARD=N
-- *****
-- Generate RUNSTATS JCL append to O1 start from STEP041
-- *****
RUNSTATS UTIL(RUNSTATS) IDSN=W3 ODSN=O1 APPEND STEP#=40 JOBCARD=N
-- *****
-- Generate PG lines
-- *****

```

```

-- Generate REBIND from PG lines (17)
-- Generate REBIND JCL append to O1 start from STEP051 (18)
-- *****
PG      IDSN=W0      ODSN=W4      NOTFOUND(SKIP 2)
REBIND  IDSN=W4      ODSN=W5
BATCH   EXEC        IDSN=W5 ODSN=O1 APPEND STEP#=50 JOBCARD=N
//SETUP DD DATA,DLM=AA
/* --REXX-----
   SETUP: Delete work & output files
----- */

x=MSG('OFF')
Address TSO
'DELETE W0'
'DELETE W1'
'DELETE W2'
'DELETE W3'
'DELETE W4'
'DELETE W5'
'DELETE O1'
RETURN 0
AA
//EDITGEN DD DATA,DLM=BB
/* --REXX-----
   EDITGEN:Setup work file W1 to hold the EDIT commands to change
           load control statements generated from DSNTIAUL command.
           The standard dataset name for SYSPUNCH from DSNTIAUL is
           TSOID.tbcreator.tbname.SYSPUNCH
           - Only the first 8 position of tbname is used and
           - If there is a '_' in tbname, '_' will be replace by '#'
   Example: TB SYSIBM.PLAN_TABLE for use with DSNTIAUL command
           generate TSOID.SYSIBM.PLAN#TAB.SYSPUNCH dataset name to
           hold the SYSPUNCH dataset.
----- */

x=MSG('OFF')
Address TSO
"Alloc fi(w1) ds(w1) new space(1,1) tracks recfm(f b) lrecl(80)",
  "blksize(0) reuse unit(sysda)"
"Alloc fi(w0) ds(w0) shr reuse"
/* ----- */
/* Generate EDIT statements */
/* ----- */
"Execio * Diskr w0(Stem inrec. Finis"
"Free fi(w0)"
J = 0
Do I = 1 to INREC.0
  Parse value INREC.I with j1 ws_creator '.' ws_name j2
  ws_name = Translate(ws_name,'#','_')
  If Length(ws_name) > 8 Then ws_name = Substr(ws_name,1,8)
  J = J + 1
  OUTREC.J="EDIT "ws_creator"."ws_name".SYSPUNCH"
  J = J + 1
  OUTREC.J="C 'LOG NO' 'LOG NO RESUME NO REPLACE' ALL"
  J = J + 1
  OUTREC.J="C 'SYSREC00' 'SYSREC' ALL"
  J = J + 1
  OUTREC.J="C 'CREATORI' 'CREATORO' ALL"
  J = J + 1
  OUTREC.J="END_EDIT"
End
"Execio "J" Diskw w1(Stem outrec. Finis"
"Free fi(w1)"
Return 0
BB
//SQLGEN DD DATA,DLM=CC
SELECT 'TB '||STRIP(CREATOR)||'.'||NAME, CARDF
FROM SYSIBM.SYSTABLES
WHERE CREATOR IN (&CREATOR) AND NAME IN (&TBNAME)
ORDER BY 1
CC

```

MONBUFR

Monbufr is a sample script, which provides scripts to capture DB2 Bufferpool information and report them.

```
-- Start execution at 8:00 AM
TSO TWAIT ACTL08000000
-- Display Bufferpool detail and save the result in T1
DISPLAY DETAIL(INTERVAL) ODSN=T1
-- Execute TSO CAPTBUFR to generate SQL insert and save the result in T2
TSO CAPTBUFR T1-T2-MY.BUFFERPOOL-5
-- Use EXEC command to execute the SQL insert in T2
EXEC IDSN=T2
-- Use TSO TWAIT to wait for one hour
TSO TWAIT PERD01000000
-- Display Bufferpool detail and save the result in T1
DISPLAY DETAIL(INTERVAL) ODSN=T1
-- Execute TSO CAPTBUFR to generate SQL insert and save the result i
TSO CAPTBUFR T1-T2-MY.BUFFERPOOL-5
-- Use EXEC command to execute the SQL insert in T2 and
-- SKIP back to TSO TWAIT PERD01000000 7 times
EXEC IDSN=T2 SKIP=(-8,7)
-- Execute TSO READBUFR to read the information stored in MY.BUFFERPOOL
-- and save the report in T3
TSO READBUFR DSN\ -MY.BUFFERPOOL-BP0-T3
-- Use FLIST to print the report saved in T3
FLIST T3 80
//LINEOBJ DD *
BP BP0
```

UNLDRELD

UNLDRELD is a sample script, which provides scripts to help you to process UNLOAD and RELOAD.

```

-----
-- UNLDRELD:  Canned Db2I2 commands to generate unload and
--            reload jobs
--
--            The commands have be broken up to 2 parts:
--            UNLOADIT and RELOADIT
--
--            UNLOADIT:
--            You can choose either DSNTIAUL or UNLOAD (V7)
--            for you unload process command.
--            You need to generate and submit all unload JCL
--            and wait until all tables have been unloaded
--            successfully.
--
--            RELOADIT:
--            And then submit the RELOADIT script
--            to generate and submit the LOAD JCL
--
--            Modify name in lower case in the scripts to
--            fit your own need
-----
-- UNLOADIT
-----
-- ** Remove 2 out of 3 of the QBUILD command below to fit your
-- ** specific unload need
-- Host variables definition:
-- &DEF - default number of rows if no runstats information
-- &DS - dataset prefix for SYSREC and SYSPUNCH
-- &WHERE-WHERE predicate from QBUILD output
-----
-- for DB line objects
QBUILD ODSN=unload.qbuild F1=DBNAME
-- for TS line objects
QBUILD ODSN=unload.qbuild F1=DBNAME F2=TSNAME
-- for TB line objects
QBUILD ODSN=unload.qbuild F1=CREATOR F2=NAME
RUN   IDSN=*GENUNLD1      +
      ODSN=unload.tbline  +
      T=N                 +
      &DEF=3000           +
      &DS=dataset-prefix  +
      &WHERE=unload.qbuild
-----
-- Remove either UNLOAD or DSNTIAUL command below
-- IDSN=unload.tbline
-- Input TB lines are generated from previous RUN command
-- ODSN=unload.jcl(*)
-- Output JCL are stored as wild card with member name to be
-- the job name of each generated jobs
-- JOBNM=jobul###
-- The job name have a naming convention of jobul and 3-digit
-- numeric number starting from 1
-- NEWJOB=##
-- Each generated unload job contain ## of unload job steps
-----
-- For Unload process UNLOAD (V7)
-- * Prepared parmutil(UNLOAD) with PARMUTIL UNLOAD command
-- parmutil(UNLOAD)
-- Prepared parmutil(UNLOAD) with PARMUTIL UNLOAD command
-----
UNLOAD IDSN=unload.tbline      +
       ODSN=unload.jcl(*)      +
       JOBNM=jobul###          +
       NEWJOB=##               +
       parmutil(UNLOAD)
-----
-- OR Unload process DSNTIAUL
-----
DSNTIAUL IDSN=unload.tbline      +
         ODSN=unload.jcl(*)      +
         JOBNM=jobul###          +

```

```

NEWJOB=##
-----
-- Make a copy of the unload TB line and use it to prepare the  --
-- load TB lines with an inline edit macro DDEDIT              --
-- DDEDIT: Change SYSPUNCH= to SYSIN=                          --
--           Change stbcreator (source creator) to            --
--           ttbcreator (target creator)                       --
-----
FCPY  unload.tblline  load.tblline
ED    load.tblline   MACRO(IDD=DDEDIT)
-----
-- Submit each generated load JCL if needed                    --
-----
ED    unload.jcl(jobul*)  MACRO(IDD=DDSUBMIT)
//DDEDIT  DD DATA,DLM=AA
C 'SYSPUNCH=' 'SYSIN=' ALL
C 'stbcreator' 'ttbcreator' ALL
AA
//DDSUBMIT DD DATA,DLM=BB
SUBMIT
BB
-- !! ***** !!
-- !! the following scripts need to be processed in a seperated !!
-- !! job after the unload jobs are done                        !!
-- !! ***** !!
-- RELOADIT
-- -----
-- For Reload process
-- * Prepared parmutil(LOAD) with PARMUTIL LOAD command
--   Make sure specify 0 on the estimated rows field when
--   you generate the parmutil for LOAD.
--   If needed, you can also specify COPY and STATS options
-----
LOAD  IDSN=load.tblline      +
      ODSN=load.jcl(*)      +
      JOBNM=jobld###        +
      NEWJOB=##             +
      OVERRIDE              +
      parmutil(LOAD)
ED    unload.jcl(jobld*)  MACRO(IDD=DDSUBMIT)
//DDSUBMIT DD DATA,DLM=BB
SUBMIT
BB

```


USER DEFINED QUERY

User Defined Query is a set of predefined queries, which can be invoked with host variable dynamically substituted at the RUN time.

There are two types of User Defined Queries: system UDQ and user UDQ. A system UDQ located in the DB2I2 system library and shipped with DB2I2 system. A user UDQ is created by you and stored in your own library. All UDQs must be invoked with DB2I2 RUN command together with host variable substitutions.

To invoke a system UDQ, you use RUN IDSN=*system.UDQ.name. For example, use the following command to invoke system UDQ GENAT, which generate AT line object for specified db.ts:

```
RUN IDSN=*GENAT &DB='dbname' &TS='tsname'
```

GENAI

GENAI is a system UDQ, which contains query to build AI line objects from a specified index. The format to run GENAI is:

```
RUN IDSN=*GENAI &IXC='ixcreator' &IXN='ixname'
```

DB2I2 run the following query with host variables substituted.

```
SELECT  'AI '||STRIP(A.IXCREATOR)||'.'||A.IXNAME, A.PARTITION
        ,A.CARDF, B.FULLKEYCARD
FROM    SYSIBM.SYSINDEXPART A
        ,SYSIBM.SYSINDEXES  B
WHERE   A.IXCREATOR = &IXC  AND IXNAME LIKE &IXN
AND     A.IXCREATOR = B.CREATOR AND A.IXNAME = B.NAME
UNION ALL
SELECT  'AI '||STRIP(A.OWNER)||'.'||A.NAME, A.PARTITION
        ,B.CARDF, A.FULLKEYCARD
FROM    SYSIBM.SYSINDEXSTATS A
        ,SYSIBM.SYSINDEXPART B
WHERE   OWNER = &IXC  AND NAME LIKE &IXN
AND     A.OWNER = B.IXCREATOR AND A.NAME = B.IXNAME
```

GENAT

GENAT is a system UDQ, which contains query to build AT line objects from a specified tablespace. The format to run GENAT is:

```
RUN IDSN=*GENAT &DB='dbname' &TS='tsname' &COMPRATIO=%compressed
```

DB2I2 run the following query with host variables substituted.

```
SELECT  'AT ' || STRIP(A.DBNAME) || '.' || A.TSNAME, A.PARTITION,
        STRIP(B.CREATOR) || '.' || B.NAME, A.CARD, '&COMPRATIO'
FROM    SYSIBM.SYSTABLEPART  A
        ,SYSIBM.SYSTABLES    B
        ,SYSIBM.SYSTABLESPACE C
WHERE   B.TYPE = 'T'
AND     A.DBNAME = &DB        AND A.TSNAME = &TS
AND     A.DBNAME = B.DBNAME  AND A.TSNAME = B.TSNAME
AND     A.DBNAME = C.DBNAME  AND A.TSNAME = C.NAME
AND     C.SEGSIZE = 0
UNION  ALL
SELECT  'AT ' || STRIP(A.DBNAME) || '.' || A.TSNAME, A.PARTITION,
        STRIP(B.CREATOR) || '.' || B.NAME, A.CARD, '&COMPRATIO'
FROM    SYSIBM.SYSTABLEPART  A
        ,SYSIBM.SYSTABLES    B
        ,SYSIBM.SYSTABLESPACE C
WHERE   B.TYPE = 'T'
AND     A.DBNAME = &DB        AND A.TSNAME = &TS
AND     A.DBNAME = B.DBNAME  AND A.TSNAME = B.TSNAME
AND     A.DBNAME = C.DBNAME  AND A.TSNAME = C.NAME
AND     C.SEGSIZE ^= 0
```

GENSC

GENSC is a system UDQ, which contains query to build SC line objects to be used with DSCOPY command.
The format to run GENSC is:

```
RUN IDSN=*GENSC &FSSID=source SSID &FLOC=source location &TSSID=Target SSID
&TLOC=Target location &RCVRIX=RCVRIX=Y|N &ICDATE=ICDATE=yyymmdd &ICGEN=###
&WHERE='qbuild.where.dsname'
```

```
-----
-- GENSC: UDQ to build SC line objects for DSCOPY command
--
-- Host Variables Usage:
-- =====
--   &FSSID - Source SSID
--   &FLOC  - Source LOCATION
--   &TSSID - Target SSID
--   &TLOC  - Target LOCATION
--   &RCVRIX- Specify &RCVRIX=RCVRIX=Y to generate REBUILD INDEX
--             &RCVRIX=RCVRIX=N No REBUILD INDEX step
--             &RCVRIX=          is the same as RCVRIX=Y
--   &ICDATE- Specify &ICDATE=ICDATE=yyymmdd to request specific
--             date of image copy as input
--             &ICDATE=          choose most current full image
--             copy as input
--   &ICGEN - Specify &ICGEN=ICGEN=### to request specific
--             generation of image copy as input
--             ### must be <= 0
--             &ICGEN=          choose most current generation
--             of full image copy as input
--   &WHERE - Specify &WHERE='qbuild.where.predicate.dsname' to
--             include the WHERE predicates generated
--             from QBUILD command
-- -----
SELECT 'SC &FSSID\&FLOC.' || SUBSTR((STRIP(DBNAME) || '.' || TSNAME),1,17) ||
      ' &TSSID\&TLOC.' || SUBSTR((STRIP(DBNAME) || '.' || TSNAME),1,17),
      CASE WHEN PARTITION = 0 THEN ' '
            ELSE DIGITS(PARTITION)
            END CASE,
      '&RCVRIX', '&ICDATE', '&ICGEN' FROM SYSIBM.SYSTABLEPART
WHERE
-INC &WHERE
ORDER BY 1,2
```

GENUNLDQ

GENUNLDQ is a system UDQ, which contains query to build TB line objects to be used for DSNTIAUL. The format to run GENUNLDQ is:

```
RUN IDSN=*GENUNLDQ &WHEREWS='where.predicates.from.QBUILD'
Where
    &WHEREWS contains the information from QBUILD output
```

DB2I2 run the following query with host variables substituted.

```
SELECT 'TB ' || STRIP(CREATOR) || '.' || NAME,
CARDF,
'SYSPUNCH=SYSPUNCH.DB' || DIGITS(DBID) ||
'.TB' || DIGITS(OBID) ||
'.D' || SUBSTR(CHAR(CURRENT DATE,ISO),3,2) ||
SUBSTR(CHAR(CURRENT DATE,ISO),6,2) ||
SUBSTR(CHAR(CURRENT DATE,ISO),9,2) ||
'SYSREC=SYSREC.DB' || DIGITS(DBID) ||
'.TB' || DIGITS(OBID) ||
'.D' || SUBSTR(CHAR(CURRENT DATE,ISO),3,2) ||
SUBSTR(CHAR(CURRENT DATE,ISO),6,2) ||
SUBSTR(CHAR(CURRENT DATE,ISO),9,2)
FROM SYSIBM.SYSTABLES
WHERE
-INC &WHEREWS
```

GENUNLD1

GENUNLD1 is a system UDQ, which contains query to build TB line objects to be used for DSNTIAUL or UNLOAD. It is a revision of GENUNLDQ.

The format to run GENUNLD1 is:

```
RUN IDSN=*GENUNLDQ &WHEREWS='where.predicates.from.QBUILD'
```

Where

&WHEREWS contains the information from QBUILD output

DB2I2 run the following query with host variables substituted.

```
-----
-- GENUNLD1: GENERATE UNLOAD TB LINE OBJECT FOR UNLOAD --
-- VARIABLES USED: --
--   &DEF: DEFAULT NUMBER OF ROWS IF NO RUNSTATS INFO --
--   &DS:  SYSPUNCH AND SYSREC DATASET PREFIX --
--   &WHERE: WHERE PREDICATES GENERATED FROM QBUILD --
-----
SELECT 'TB ' || STRIP(CREATOR) || '.' || NAME,
       CASE CARDF
         WHEN -1 THEN &DEF
         ELSE INTEGER(CARDF)
       END,
       'SYSPUNCH=' || &DS.SYSPUNCH.DB' || DIGITS(DBID) ||
         '.TB' || DIGITS(OBID) ||
       '.D' || SUBSTR(CHAR(CURRENT DATE,ISO),3,2) ||
         SUBSTR(CHAR(CURRENT DATE,ISO),6,2) ||
         SUBSTR(CHAR(CURRENT DATE,ISO),9,2) || ' ',
       'SYSREC=' || &DS.SYSREC.DB' || DIGITS(DBID) ||
         '.TB' || DIGITS(OBID) ||
       '.D' || SUBSTR(CHAR(CURRENT DATE,ISO),3,2) ||
         SUBSTR(CHAR(CURRENT DATE,ISO),6,2) ||
         SUBSTR(CHAR(CURRENT DATE,ISO),9,2) || ' ',
       ,
FROM SYSIBM.SYSTABLES
WHERE
-INC &WHERE
```

GENXC

GENXC is a system UDQ, which contains query to build SC line objects to be used with DSCOPY command. The format to run GENXC is:

```
RUN IDSN=*GENXC &FSSID=source SSID &FLOC=source location &TSSID=Target SSID
&TLOC=Target location &ICDATE=ICDATE=yymmdd &ICGEN=###
&WHERE='qbuild.where.dsname'
```

```
-----
-- GENXC: UDQ to build XC line objects for DSCOPY command          --
--                                                                 --
-- Host Variables Usage:                                         --
-- =====                                                       --
--   &FSSID - Source SSID                                         --
--   &FLOC  - Source LOCATION                                     --
--   &TSSID - Target SSID                                         --
--   &TLOC  - Target LOCATION                                     --
--   &ICDATE- Specify &ICDATE=ICDATE=yymmdd to request specific --
--                                     date of image copy as input --
--                                     &ICDATE=                    --
--                                     choose most current full image --
--                                     copy as input                --
--   &ICGEN - Specify &ICGEN=ICGEN=### to request specific        --
--                                     generation of image copy as input --
--                                     ### must be <= 0            --
--                                     &ICGEN=                    --
--                                     choose most current generation --
--                                     of full image copy as input --
--   &WHERE - Specify &WHERE='qbuild.where.predicate.dsname' to --
--                                     include the WHERE predicates generated --
--                                     from QBUILD command         --
-----
SELECT 'XC &FSSID\&FLOC.' || SUBSTR((STRIP(IXCREATOR) || '.' || IXNAME),1,27) ||
' &TSSID\&TLOC.' || SUBSTR((STRIP(IXCREATOR) || '.' || IXNAME),1,27),
CASE WHEN PARTITION = 0 THEN ' '
ELSE DIGITS(PARTITION)
END CASE,
'&ICDATE', '&ICGEN' FROM SYSIBM.SYSINDEXPART
WHERE
-INC &WHERE
ORDER BY 1,2
```

OBIDQ

OBIDQ is a system UDQ, which contains query to build DB, TS, TB line objects in OBID sequence to be used as a preprocessor for DB MIGR with OBID option.

The format to run OBIDQ is:

```
RUN IDSN=*OBIDQ &DB=your.dbname T=N
```

```
-----
-- OBIDQ: Generate all line object in sequence to match OBID info
--       The sequence of generated Line object is:
--           DB
--           TS
--           TB, TBC
-----
-- DataBase
-----
select 'DB ', name, 1, 1
from sysibm.sysdatabase
where name = '&DB'
union all
-----
-- TableSpace
-----
select 'TS ',
       strip(dbname)||'.'||strip(name),
       2, obid
from sysibm.systablespace
where dbname = '&DB'
union all
-----
-- Table
-----
select 'TB ',
       strip(creator)||'.'||strip(name),
       2, obid
FROM   sysibm.systables r
WHERE  r.dbname = '&DB'
and    type = 'T'
order by 3,4
```

VCAT2SGQ

VCAT2SGQ is a system UDF, which query to generate STOP, ALTER and START to convert TP and IP from VCAT to STOGROUP. The format to run VCAT2SGQ is:

```
RUN IDSN=*VCAT2SGQ LIMIT(9999) T=N ODSN=output.dsname
      &DB=Database criteria
      &SG=Storage-Group-Name
```

```
-----
-- VCAT2SGQ: Generate STOP, ALTER, START commands
-- To convert TP and IP from VCAT to STOGROUP
-- Host Variables: &DB - Database Criteria
-- &SG - Stogroup Name
-----
SELECT Substr('IP '||Strip(ixcreator)||'.'||Strip(ixname)||' '||
  Digits(partition),1,24)||'1'||
  'DB2CMD -STOP DB('||Strip(dbname)||') SP('||
  Strip(indexspace)||')'||
  Case When partition = 0 then ''
        Else ' PART('||Digits(partition)||')'
  End
FROM "SYSIBM"."SYSINDEXPART" A,
     "SYSIBM"."SYSINDEXES" B
WHERE stortype = 'E'
And dbname like '&DB'
And creator = ixcreator
And a.ixname = b.name
UNION All
SELECT Substr('IP '||Strip(ixcreator)||'.'||Strip(ixname)||' '||
  Digits(partition),1,24)||'2'||
  'ALTER INDEX '||
  Strip(IXCREATOR)||'.'||Strip(ixname) ||
  Case When partition = 0 then ''
        Else ' ALTER PART('||Digits(partition)||')'
  End ||
  ' USING STOGROUP &SG PRIQTY '||
  Case When a.space > 2097152 Then '2097152'
        Else Digits(a.SPACE)
  End ||
  ' SECQTY '||
  Case When a.SPACE/48/10 < 1 then '48'
        Else Digits(round(a.SPACE/48/10,0)*48)
  End || ';'
FROM "SYSIBM"."SYSINDEXPART" A,
     "SYSIBM"."SYSINDEXES" B
WHERE stortype = 'E'
And dbname like '&DB'
And creator = ixcreator
And a.ixname = b.name
UNION All
SELECT Substr('IP '||Strip(ixcreator)||'.'||Strip(ixname)||' '||
  Digits(partition),1,24)||'3'||
  'DB2CMD -START DB('||Strip(dbname)||') SP('||
  Strip(indexspace)||')'||
  Case When partition = 0 then ''
        Else ' PART('||Digits(partition)||')'
  End
FROM "SYSIBM"."SYSINDEXPART" A,
     "SYSIBM"."SYSINDEXES" B
WHERE stortype = 'E'
And dbname like '&DB'
And creator = ixcreator
And a.ixname = b.name
-----
-- The following are TP lines
-----
UNION All
SELECT Substr('TP '||Strip(dbname)||'.'||Strip(tsname)||' '||
  Digits(partition),1,24)||'1'||
  'DB2CMD -STOP DB('||Strip(dbname)||') SP('||
```



```

        Strip(tsname)||'|')' ||
        Case When partition = 0 then ''
            Else ' PART('||Digits(partition)||'|')'
        End
FROM "SYSIBM"."SYSTABLEPART"
WHERE stortype = 'E'
And dbname like '&DB'
UNION All
SELECT Substr('TP '||Strip(dbname)||'|.'||Strip(tsname)||'|' ' ||
    Digits(partition),1,24)||'2' ||
    'ALTER TABLESPACE ' ||
    Strip(DBNAME)||'|.'||Strip(TSNAME) ||
    Case When partition = 0 then ''
        Else ' ALTER PART('||Digits(partition)||'|')'
    End ||
    ' USING STOGROUP &SG PRIQTY ' ||
    Case When space > 2097152 Then '2097152'
        Else Digits(SPACE)
    End ||
    ' SECQTY ' ||
    Case When SPACE/10/48 < 1 then '48'
        Else Digits(round(SPACE/10/48,0)*48)
    End ||';'
FROM "SYSIBM"."SYSTABLEPART"
WHERE stortype = 'E'
And dbname like '&DB'
UNION All
SELECT Substr('TP '||Strip(dbname)||'|.'||Strip(tsname)||'|' ' ||
    Digits(partition),1,24)||'3' ||
    'DB2CMD -START DB('||Strip(dbname)||') SP(' ||
    Strip(tsname)||'|')' ||
    Case When partition = 0 then ''
        Else ' PART('||Digits(partition)||'|')'
    End
FROM "SYSIBM"."SYSTABLEPART"
WHERE stortype = 'E'
And dbname like '&DB'
Order By 1;

```

Issue the Following ED Macro to remove the first 25 bytes of the output from VCAT2SGQ run.

```
ED vcat2sgq.run.output macro(*VCAT2SGM)
```

```

-----
-- VCAT2SGM: ED Macro to Remove The First 25 Bytes From The Output
-- of VCAT2SGQ
-----
CLINE
LOOP EOF
  REXX CLINE = SUBSTR(CLINE,26,200)
  CLINE WRITE
  CLINE NEXT
LOOP_END

```

REAL TIME STATISTICS RELATED QUERIES (V8+)

The following is a list of all Real Time Statistics User Defined Queries, which can be invoked with host variable dynamically substituted at the RUN time and produce a list of TP or IP lines which requires COPY, REORG or RUNSTATS.

To invoke a system UDQ, you use RUN IDSN=*system.UDQ.name. For example, use the following command to invoke system UDQ COPYFSTS, which generate TP line object from a Database which requires Full Image Copy:

```
RUN IDSN=*COPYFSTS LIMIT(9999) T=N
  &TSSTTB=SYSTABLESPACESTATS or TABLESPACESTATS
  &DB=dbname
  &SCRDAYSNCLASTCOPY=07
  &SCRUPDATEDPAGESPCT=20
  &SCRCHANGESPCT=10
```

COPYFIX, COPYFIX9 and COPYFIXS

contains query to build IP lines for all indexes which requires a Full Image Copy against Real Time Statistics. The criteria are the same as DSNACCOR. Use COPYFIXS rxdb2i2 script to invoke COPYFIX or COPYFIX9. The format to run COPYFIX is:

```
RUN IDSN=*COPYFIX[9] LIMIT(9999) T=N
  &IXSTTB=SYSINDEXSPACESTATS or INDEXSPACESTATS
  &DB=dbname
  &XCRDAYSNCLASTCOPY=7
  &XCRINDEXSIZE=50
  &XCRUPDATEDPAGESPCT=20
  &XCRCHANGESPCT=10
```

*Specify SYSINDEXSPACESTATS or INDEXSPACESTATS for &IXSTTB depends on which SYSIBM table name is used.

COPYFIXS

```
-----
Say " ** COPYFIXS - Generate Full COPY IP with Real Time Statistics"
Say " ** Generating IP lines with Run Command ...."
-----
Address Ispexec "Vget db2ver Profile"
If db2ver = 8 Then
  str = "RUN LIMIT(9999) IDSN=*COPYFIX T=N odsn=w1.@JNM.IP",
        "&IXSTTB=INDEXSPACESTATS"
Else
  str = "RUN LIMIT(9999) IDSN=*COPYFIX9 T=N odsn=w1.@JNM.IP",
        "&IXSTTB=SYSINDEXSPACESTATS"
db2i2rex str,
  &DB=DDB2ADM",
--      &DB=DSN8D91A",
        &XCRDAYSNCLASTCOPY=7",
        &XCRINDEXSIZE=50",
        &XCRUPDATEDPAGESPCT=20",
        &XCRCHANGESPCT=10"
-----
Say " ** Check Not Found to see if any IP need to be Copied ...."
-----
```

```

If RxNotfnd() = 'Y' Then;
Do
  Say " ** No IP Line Objects Meet Full Copy Criteria **"
  x = Rxrc(4)
  db2i2rex "Rxdb2i2 Reset"
  Return 4
End
-----
Say " ** RESIZEIT to 3 jobs ...."
-----
db2i2rex "TSO Fcpy w1.@JNM.IP w2.@JNM.IP"
LINEOBJ ddname=ddl
RESIZEIT 36,45,#JOB=3 LISTNM=CPID91A# LIST#=0
END_LINEOBJ
db2i2rex "ED w1.@JNM.IP macro(idd=ddl)"
-----
Say " ** Removing listdef and genjcl members ...."
-----
LINEOBJ ddname=dd3
IDCAMS delete db2adm.listdef(CPID91A1)
IDCAMS delete db2adm.listdef(CPID91A2)
IDCAMS delete db2adm.listdef(CPID91A3)
IDCAMS delete db2adm.db2i2.genjcl(CPID91A1)
IDCAMS delete db2adm.db2i2.genjcl(CPID91A2)
IDCAMS delete db2adm.db2i2.genjcl(CPID91A3)
IDCAMS If maxcc NE 0 Then set maxcc = 0
END_LINEOBJ
db2i2rex "EXEC IDD=dd3"
-----
Say " ** Generating Listdef information ...."
-----
db2i2rex "LISTDEF parmutil=parmutil(listix)",
         "idsn=w1.@JNM.IP",
         "odsn=listdef(*)"
-----
Say " ** Building COPY Jobs ...."
-----
LINEOBJ ddname=LINEOBJ
dummy
END_LINEOBJ
LINEOBJ ddname=dd4
Rexx Say '*** Change This line to SUBMIT to submit the generated Copy Jobs'
-- Submit
END_LINEOBJ
db2i2rex "COPY",
         "parmutil='db2adm.parmutil(copyix)'",
         "template='db2adm.template(copyix)'",
         "listdef=listdef(CPID91A*)",
         "odsn=db2i2.genjcl(*)",
         "jobnm=CPID91A# job#=0",
         "Macro(idd=dd4)"

```

COPYFIX

```

-----
-- DSNACCOR formula for Full Copy on a IX
-- Default Host Variables
-----
-- &XCRDAYSNCLASTCOPY=7
-- &XCRINDEXSIZE=50
-- &XCRUPDATEDPAGEPCT=20
-- &XCRCHANGESPCT=10
-----
SELECT
  Substr('IP '||Strip(A.CREATOR)||'. '||Strip(A.NAME),1,40)
  ,B.partition
  ,B.NACTIVE
  ,CASE
    When COPYLASTTIME IS NULL
    Then 'COPYLASTTIME IS NULL'
    When REORGLASTTIME > COPYLASTTIME
    Then 'REORGLASTTIME > COPYLASTTIME '||
         'REORGLASTTIME='||Char(REORGLASTTIME)||' '||
         'COPYLASTTIME='||Char(COPYLASTTIME)
    When LOADRLASTTIME > COPYLASTTIME
    Then 'LOADRLASTTIME > COPYLASTTIME '||
         'LOADRLASTTIME='||Char(LOADRLASTTIME)||
         'COPYLASTTIME='||Char(COPYLASTTIME)
    When REBUILDLASTTIME > COPYLASTTIME

```

```

Then 'REBUILDRLASTTIME > COPYLASTTIME' ||
' REBUILDRLASTTIME=' || Char(REBUILDRLASTTIME) ||
' COPYLASTTIME=' || Char(COPYLASTTIME)
When (CURRENT DATE-DATE(COPYLASTTIME)) > &XCRDAYSNCLASTCOPY
Then 'CURRENT DATE-DATE(COPYLASTTIME) > &XCRDAYSNCLASTCOPY ' ||
' COPYLASTTIME=' || Char(COPYLASTTIME)
When (B.NACTIVE > &XCRINDEXSIZE AND
      (((COPYUPDATEDPAGES*100)/NACTIVE > &XCRUPDATEDPAGESPCT) OR
       ((COPYCHANGES*100)/TOTALENTRIES > &XCRCHANGESPCT)))
Then
'NACTIVE=' || Strip(Char(nactive)) || ' > &XCRINDEXSIZE AND' ||
' (COPYUPDATEDPAGES*100)/NACTIVE=' ||
Strip(Char((COPYUPDATEDPAGES*100)/NACTIVE)) ||
' > &XCRUPDATEDPAGESPCT OR' ||
' (COPYCHANGES*100)/TOTALENTRIES=' ||
Strip(Char((COPYCHANGES*100)/TOTALENTRIES)) ||
' > &XCRCHANGESPCT'
END
FROM "SYSIBM"."SYSINDEXES" A INNER JOIN
     "SYSIBM"."&IXSTTB" B
ON A.DBID = B.DBID
AND A.ISOBID = B.ISOBID
WHERE A.DBNAME LIKE '&DB%'
AND A.COPY = 'Y'
AND
      B.NACTIVE > 0 AND
      B.TOTALENTRIES > 0 AND
      (COPYLASTTIME IS NULL OR
       REORGLASTTIME > COPYLASTTIME OR
       LOADRLASTTIME > COPYLASTTIME OR
       REBUILDRLASTTIME > COPYLASTTIME OR
       (CURRENT DATE-DATE(COPYLASTTIME)) > &XCRDAYSNCLASTCOPY OR
       (B.NACTIVE > &XCRINDEXSIZE AND
        ((COPYUPDATEDPAGES*100)/NACTIVE > &XCRUPDATEDPAGESPCT) OR
        ((COPYCHANGES*100)/TOTALENTRIES > &XCRCHANGESPCT)))

```

COPYFIX9 - include Clone table information for V9

```

-----
-- DSNACCOR formula for Full Copy on a IX
-- Default Host Variables
-----
-- &XCRDAYSNCLASTCOPY=7
-- &XCRINDEXSIZE=50
-- &XCRUPDATEDPAGESPCT=20
-- &XCRCHANGESPCT=10
-----
SELECT 'IP' ||
      Case When b.instance = d.instance Then ' '
            Else 'C' End ||
      Substr(Strip(A.CREATOR) || '.' || Strip(A.NAME),1,40)
      ,B.partition
      ,B.NACTIVE
      ,CASE
      When COPYLASTTIME IS NULL
      Then 'COPYLASTTIME IS NULL'
      When REORGLASTTIME > COPYLASTTIME
      Then 'REORGLASTTIME > COPYLASTTIME ' ||
           'REORGLASTTIME=' || Char(REORGLASTTIME) || ' ' ||
           'COPYLASTTIME=' || Char(COPYLASTTIME)
      When LOADRLASTTIME > COPYLASTTIME
      Then 'LOADRLASTTIME > COPYLASTTIME' ||
           'LOADRLASTTIME=' || Char(LOADRLASTTIME) ||
           'COPYLASTTIME=' || Char(COPYLASTTIME)
      When REBUILDRLASTTIME > COPYLASTTIME
      Then 'REBUILDRLASTTIME > COPYLASTTIME' ||
           'REBUILDRLASTTIME=' || Char(REBUILDRLASTTIME) ||
           'COPYLASTTIME=' || Char(COPYLASTTIME)
      When (CURRENT DATE-DATE(COPYLASTTIME)) > &XCRDAYSNCLASTCOPY
      Then 'CURRENT DATE-DATE(COPYLASTTIME) > &XCRDAYSNCLASTCOPY ' ||
           'COPYLASTTIME=' || Char(COPYLASTTIME)
      When (B.NACTIVE > &XCRINDEXSIZE AND
            (((COPYUPDATEDPAGES*100)/NACTIVE > &XCRUPDATEDPAGESPCT) OR
             ((COPYCHANGES*100)/TOTALENTRIES > &XCRCHANGESPCT)))
      Then
           'NACTIVE=' || Strip(Char(nactive)) || ' > &XCRINDEXSIZE AND' ||
           ' (COPYUPDATEDPAGES*100)/NACTIVE=' ||

```

```

Strip(Char((COPYUPDATEDPAGES*100)/NACTIVE))||
' > &XCRUPDATEDPAGESPCT OR' ||
' (COPYCHANGES*100)/TOTALENTRIES=' ||
Strip(Char((COPYCHANGES*100)/TOTALENTRIES))||
' > &XCRCHANGESPCT'
END
FROM "SYSIBM"."SYSINDEXES" A          INNER JOIN
     "SYSIBM"."IXSTTB" B
ON A.DBID = B.DBID
AND A.ISOBID = B.ISOBID              INNER JOIN
     "SYSIBM"."SYSTABLES" C
ON A.TBCREATOR = C.CREATOR
AND A.TBNAME = B.NAME                INNER JOIN
     "SYSIBM"."SYSTABLESPACE" D
ON C.DBNAME = D.DBNAME
AND C.TSNAME = D.NAME
WHERE A.DBNAME LIKE '&DB%'
AND A.COPY = 'Y'
AND
     B.NACTIVE > 0                      AND
     B.TOTALENTRIES > 0                  AND
(COPYLASTTIME IS NULL                  OR
REORGLASTTIME > COPYLASTTIME          OR
LOADRLASTTIME > COPYLASTTIME          OR
REBUILDLASTTIME > COPYLASTTIME        OR
(CURRENT DATE-DATE(COPYLASTTIME)) > &XCRDAYSNCLASTCOPY OR
(B.NACTIVE > &XCRINDEXSIZE            AND
((COPYUPDATEDPAGES*100)/B.NACTIVE > &XCRUPDATEDPAGESPCT) OR
((COPYCHANGES*100)/TOTALENTRIES > &XCRCHANGESPCT)))

```

COPYFTS, COPYFTS9 and COPYFTSS

contains query to build TP lines for all table spaces which requires a Full Image Copy against Real Time Statistics. The criteria are the same as DSNACCOR. Use COPYFTSS rxdb2i2 script to invoke COPYFTS or COPYFTS9. The format to run COPYFTS is:

```

RUN IDSN=*COPYFTS LIMIT(9999) T=N
  &TSSTTB=SYSTABLESPACESTATS or TABLESPACESTATS
  &DB=dbname
  &SCRDAYSNCLASTCOPY=07
  &SCRUPDATEDPAGESPCT=20
  &SCRCHANGESPCT=10

```

*Specify SYSTABLESPACESTATS or TABLESPACESTATS for &TSSTTB depends on which SYSIBM table name is used.

COPYFTSS

```

-----
Say " ** COPYFTSS - Generate Full COPY TP with Real Time Statistics"
Say " ** Generating TP lines with Run Command ...."
-----
Address Ispexec "Vget db2ver Profile"
If db2ver = 8 Then
  str = "RUN LIMIT(9999) IDSN=*COPYFTS T=N odsn=w1.@JNM.TP",
        "&TSSTTB=TABLESPACESTATS"
Else
  str = "RUN LIMIT(9999) IDSN=*COPYFTS9 T=N odsn=w1.@JNM.TP",
        "&TSSTTB=SYSTABLESPACESTATS"
db2i2rex str,
        "&DB=DSN8D91A",
        "&SCRDAYSNCLASTCOPY=07",
        "&SCRUPDATEDPAGESPCT=20",
        "&SCRCHANGESPCT=10"
-----
Say " ** Check Not Found to see if any TP need to be Copied ...."
-----
If RxNotfnd() = 'Y' Then;
Do
  Say " ** No TP Line Objects Meet Incremental Copy Criteria ***"
  Rxrc(4)
  db2i2rex "Rxdb2i2 Reset"

```

```

Return 4
End
-----
Say " ** Removing all TPC lines and SIZEIT to 3 jobs ...."
-----
db2i2rex "TSO Fcopy w1.@JNM.TP w2.@JNM.TP"
LINEOBJ ddname=ddl
X TPC 1 All
Del X all
RESIZEIT 36,45,#JOB=3 LISTNM=CPYD91A# LIST#=0
END_LINEOBJ
db2i2rex "ED w1.@JNM.TP macro(idd=ddl)"
-----
Say " ** Removing listdef and genjcl members ...."
-----
LINEOBJ ddname=dd3
IDCAMS delete db2adm.listdef(CPYD91A1)
IDCAMS delete db2adm.listdef(CPYD91A2)
IDCAMS delete db2adm.listdef(CPYD91A3)
IDCAMS delete db2adm.db2i2.genjcl(CPYD91A1)
IDCAMS delete db2adm.db2i2.genjcl(CPYD91A2)
IDCAMS delete db2adm.db2i2.genjcl(CPYD91A3)
IDCAMS If maxcc NE 0 Then set maxcc = 0
END_LINEOBJ
db2i2rex "EXEC IDD=dd3"
-----
Say " ** Generating Listdef information ...."
-----
db2i2rex "LISTDEF parmutil=parmutil(listts)",
         "idsn=w1.@JNM.TP",
         "odsn=listdef(*)"
-----
Say " ** Building COPY Jobs ...."
-----
LINEOBJ ddname=LINEOBJ
dummy
END_LINEOBJ
LINEOBJ ddname=dd4
Rexx Say '** Change This line to SUBMIT to submit the generated Copy Jobs'
-- Submit
END_LINEOBJ
db2i2rex "COPY",
         "parmutil='db2adm.parmutil(copy)'",
         "template='db2adm.template(copy)'",
         "listdef=listdef(CPYD91A*)",
         "odsn=db2i2.genjcl(*)",
         "jobnm=CPYD91A# job#=0",
         "Macro(idd=dd4)"

```

COPYFTS

```

-----
-- DSNACCOR formula for Full Copy on a TS
-- Default Host Variables
-----
-- &SCRDAYSNCLASTCOPY=7
-- &SCRUPDATEDPAGEPCT=20
-- &SCRCHANGESPCT=10
-----
SELECT
  Substr('TP '||Strip(A.DBNAME)||'. '||Strip(A.NAME),1,30)
  ,B.partition
  ,B.NACTIVE
  ,CASE
    When COPYLASTTIME IS NULL
    Then 'COPYLASTTIME IS NULL'
    When REORGLASTTIME > COPYLASTTIME
    Then 'REORGLASTTIME > COPYLASTTIME '||
         'REORGLASTTIME='||Char(REORGLASTTIME)||' '||
         'COPYLASTTIME='||Char(COPYLASTTIME)
    When LOADRLASTTIME > COPYLASTTIME
    Then 'LOADRLASTTIME > COPYLASTTIME '||
         'LOADRLASTTIME='||Char(LOADRLASTTIME)||' '||
         'COPYLASTTIME='||Char(COPYLASTTIME)
    When (CURRENT DATE-DATE(COPYLASTTIME)) > &SCRDAYSNCLASTCOPY
    Then 'CURRENT DATE-DATE(COPYLASTTIME) > &SCRDAYSNCLASTCOPY '||
         'COPYLASTTIME='||Char(COPYLASTTIME)

```

```

When (COPYUPDATEDPAGES*100)/B.NACTIVE > &SCRUPDATEDPAGESPCT
Then '(COPYUPDATEDPAGES*100)/NACTIVE=' ||
Strip(Char((COPYUPDATEDPAGES*100)/B.NACTIVE)) ||
' And is > &SCRUPDATEDPAGESPCT ' ||
' COPYUPDATEDPAGES=' || Strip(Char(COPYUPDATEDPAGES)) ||
' NACTIVE=' || Strip(Char(B.NACTIVE))
WHEN (COPYCHANGES*100)/TOTALROWS > &SCRCHANGESPCT
Then '(COPYCHANGES*100)/TOTALROWS=' ||
Strip(Char((COPYCHANGES*100)/TOTALROWS)) ||
' And is > &SCRCHANGESPCT ' ||
' COPYCHANGES=' || Strip(Char(COPYCHANGES)) || ' TOTALROWS=' ||
Strip(Char(TOTALROWS))
END
FROM "SYSIBM"."SYSTABLESPACE" A INNER JOIN
"SYSIBM"."&TSSTTB" B
ON A.DBID = B.DBID
AND A.PSID = B.PSID
WHERE A.DBNAME LIKE '&DB%'
AND
B.NACTIVE > 0 AND
B.TOTALROWS > 0 AND
(COPYLASTTIME IS NULL OR
REORGLASTTIME > COPYLASTTIME OR
LOADRLASTTIME > COPYLASTTIME OR
(CURRENT DATE-DATE(COPYLASTTIME)) > &SCRDAYSNCLASTCOPY OR
(COPYUPDATEDPAGES*100)/B.NACTIVE > &SCRUPDATEDPAGESPCT OR
(COPYCHANGES*100)/TOTALROWS > &SCRCHANGESPCT)

```

COPYFTS9 - include Clone table information for V9

```

-----
-- DSNACCOR formula for Full Copy on a TS
-- Default Host Variables
-- =====
-- &SCRDAYSNCLASTCOPY=7
-- &SCRUPDATEDPAGESPCT=20
-- &SCRCHANGESPCT=10
-----
SELECT 'TP' ||
case when a.instance = b.instance Then ' '
else 'C' end ||
Substr(Strip(A.DBNAME)||'.' || Strip(A.NAME),1,20)
,B.partition
,B.NACTIVE
,CASE
When COPYLASTTIME IS NULL
Then 'COPYLASTTIME IS NULL'
When REORGLASTTIME > COPYLASTTIME
Then 'REORGLASTTIME > COPYLASTTIME ' ||
'REORGLASTTIME=' || Char(REORGLASTTIME) || ' ' ||
'COPYLASTTIME=' || Char(COPYLASTTIME)
When LOADRLASTTIME > COPYLASTTIME
Then 'LOADRLASTTIME > COPYLASTTIME ' ||
'LOADRLASTTIME=' || Char(LOADRLASTTIME) || ' ' ||
'COPYLASTTIME=' || Char(COPYLASTTIME)
When (CURRENT DATE-DATE(COPYLASTTIME)) > &SCRDAYSNCLASTCOPY
Then 'CURRENT DATE-DATE(COPYLASTTIME) > &SCRDAYSNCLASTCOPY ' ||
'COPYLASTTIME=' || Char(COPYLASTTIME)
When (COPYUPDATEDPAGES*100)/B.NACTIVE > &SCRUPDATEDPAGESPCT
Then '(COPYUPDATEDPAGES*100)/NACTIVE=' ||
Strip(Char((COPYUPDATEDPAGES*100)/B.NACTIVE)) ||
' And is > &SCRUPDATEDPAGESPCT ' ||
' COPYUPDATEDPAGES=' || Strip(Char(COPYUPDATEDPAGES)) ||
' NACTIVE=' || Strip(Char(B.NACTIVE))
WHEN (COPYCHANGES*100)/TOTALROWS > &SCRCHANGESPCT
Then '(COPYCHANGES*100)/TOTALROWS=' ||
Strip(Char((COPYCHANGES*100)/TOTALROWS)) ||
' And is > &SCRCHANGESPCT ' ||
' COPYCHANGES=' || Strip(Char(COPYCHANGES)) || ' TOTALROWS=' ||
Strip(Char(TOTALROWS))
END
FROM "SYSIBM"."SYSTABLESPACE" A INNER JOIN
"SYSIBM"."&TSSTTB" B
ON A.DBID = B.DBID
AND A.PSID = B.PSID
WHERE A.DBNAME LIKE '&DB%'
AND
B.NACTIVE > 0 AND
B.TOTALROWS > 0 AND

```

```
(COPYLASTTIME IS NULL OR
REORGLASTTIME > COPYLASTTIME OR
LOADRLASTTIME > COPYLASTTIME OR
(CURRENT DATE-DATE(COPYLASTTIME)) > &SCRDAYSNCLASTCOPY OR
(COPYUPDATEDPAGES*100)/B.NACTIVE > &SCRUPDATEDPAGESPCT OR
(COPYCHANGES*100)/TOTALROWS > &SCRCHANGESPCT)
```

COPYITS, COPYITS9 and COPYITSS

contains query to build TP lines for all table spaces which requires an Incremental Image Copy against Real Time Statistics. The criteria are the same as DSNACCOR. Use COPYITSS rxdb2i2 script to invoke COPYITS or COPYITS9. The format to run COPYITS is:

```
RUN IDSN=*COPYITS LIMIT(9999) T=N
  &TSSTTB=SYSTABLESPACESTATS or TABLESPACESTATS
  &DB=dbname
  &ICRUPDATEDPAGESPCT=1
  &ICRCHANGESPCT=1
```

*Specify SYSTABLESPACESTATS or TABLESPACESTATS for &TSSTTB depends on which SYSIBM table name is used.

COPYITSS

```
-----
Say " ** COPYITSS - Incremental & Merge COPY with Real Time Statistics"
Say " ** Deleting listdef and genjcl member"
-----
LINEOBJ ddname=dd3
IDCAMS Delete db2adm.listdef(CPID91A6)
IDCAMS Delete db2adm.db2i2.genjcl(CPID91A6)
IDCAMS If Maxcc NE 0 Then Set Maxcc = 0
END_LINEOBJ
db2i2rex "EXEC IDD=dd3"
-----
Say " ** Run *COPYITS9 to get TP lines requires Incremental Copy"
-----
Address Ispexec "Vget db2ver Profile"
If db2ver = 8 Then
  str = "RUN LIMIT(9999) IDSN=*COPYITS T=N odsn=w1.@JNM.TP",
        "&TSSTTB=TABLESPACESTATS"
Else
  str = "RUN LIMIT(9999) IDSN=*COPYITS9 T=N odsn=w1.@JNM.TP",
        "&TSSTTB=SYSTABLESPACESTATS"
db2i2rex str,
        "&DB=DSN8D91A",
--        "&DB=DDB2ADM",
        "&ICRUPDATEDPAGESPCT=1",
        "&ICRCHANGESPCT=1"
-----
Say " ** Check If No TP Lines Found"
-----
If RxNotfnd() = 'Y' Then;
Do
  Say " ** No TP Line Objects Meet Incremental Copy Criteria **"
  x = Rxrc(4)
  db2i2rex "Rxdb2i2 Reset"
  Return 4
End
-----
Say " ** Removing TPC lines and size to 1 job - CPID91A1"
-----
db2i2rex "TSO Fcpy w1.@JNM.TP w2.@JNM.TP"
LINEOBJ ddname=ddl
x TPC 1 all
del x all
RESIZEIT 36,45,#JOB=1 LISTNM=CPID91A# LIST#=0
END_LINEOBJ
db2i2rex "ED w1.@JNM.TP macro(idd=ddl)"
-----
Say " ** Building Listdef - CPID91A1"
-----
```



```

db2i2rex "LISTDEF parmutil=parmutil(listts)",
         "idsn=w1.@JNM.TP",
         "odsn=listdef(*)"
-----
Say " ** Build Incremental Copy"
-----
LINEOBJ ddname=LINEOBJ
dummy
END_LINEOBJ
db2i2rex "COPY",
         "parmutil='db2adm.parmutil(copyi)'",
         "template='db2adm.template(copy)'",
         "listdef=listdef(CPID91A1)",
         "odsn=w1.@JNM.JCL",
         "jobnm=CPID91A1"
-----
Say " ** Build Merge Copy"
-----
db2i2rex "COPY",
         "parmutil='db2adm.parmutil(copym)'",
         "template='db2adm.template(copy)'",
         "listdef=listdef(CPID91A1)",
         "odsn=w1.@JNM.JCL APPEND",
         "jobcard=n STEP#=10"
-----
Say " ** Copying the Generated JCL to db2i2.genjcl(CPID91A1)"
-----
db2i2rex "TSO fcopy w1.@JNM.jcl db2i2.genjcl(CPID91A1)"

```

COPYIITS

```

-----
-- DSNACCOR formula for Incremental COPY on a TS
-- Default Host Variables
-- =====
-- &ICRUPDATEDPAGESPCT=1
-- &ICRCHANGESPCT=1
-----
SELECT
  Substr('TP '||Strip(A.DBNAME)||'.'||Strip(A.NAME),1,40)
  ,B.partition
  ,B.NACTIVE
  ,CASE
    When REORGLASTTIME > COPYLASTTIME
    Then 'REORGLASTTIME > COPYLASTTIME '||
         'REORGLASTTIME='||Char(REORGLASTTIME)||' '||
         'COPYLASTTIME='||Char(COPYLASTTIME)
    When LOADRLASTTIME > COPYLASTTIME
    Then 'LOADRLASTTIME > COPYLASTTIME '||
         'LOADRLASTTIME='||Char(LOADRLASTTIME)||' '||
         'COPYLASTTIME='||Char(COPYLASTTIME)
    When (COPYUPDATEDPAGES*100)/B.NACTIVE > &ICRUPDATEDPAGESPCT
    Then '(COPYUPDATEDPAGES*100)/NACTIVE='||
         Strip(Char((COPYUPDATEDPAGES*100)/B.NACTIVE))||
         ' And is > &ICRUPDATEDPAGESPCT' ||
         ' COPYUPDATEDPAGES='||Strip(Char(COPYUPDATEDPAGES))||' '||
         'NACTIVE='||Strip(Char(B.NACTIVE))
    WHEN (COPYCHANGES*100)/TOTALROWS > &ICRCHANGESPCT
    Then '(COPYCHANGES*100)/TOTALROWS='||
         Strip(Char((COPYCHANGES*100)/TOTALROWS))||
         ' And is > &ICRCHANGESPCT' ||
         ' COPYCHANGES='||Strip(Char(COPYCHANGES))||' '||
         'TOTALROWS='||Strip(Char(TOTALROWS))
  END
FROM "SYSIBM"."SYSTABLESPACE"  A INNER JOIN
     "SYSIBM"."&TSSTTB" B
  ON A.DBID = B.DBID
  AND A.PSID = B.PSID
WHERE A.DBNAME LIKE '&DB%'
AND
  B.NACTIVE > 0                                AND
  B.TOTALROWS > 0                              AND
  COPYLASTTIME IS NOT NULL                    AND
  (LOADRLASTTIME > COPYLASTTIME              OR
   REORGLASTTIME > COPYLASTTIME              OR
   (COPYUPDATEDPAGES*100)/B.NACTIVE > &ICRUPDATEDPAGESPCT OR
   (COPYCHANGES*100)/TOTALROWS > &ICRCHANGESPCT)

```

```

COPYITS9 - include Clone table information for V9
-----
-- DSNACCOR formula for Incremental COPY on a TS
-- Default Host Variables
-- =====
-- &ICRUPDATEDPAGESPCT=1
-- &ICRCHANGESPCT=1
-----
SELECT 'TP' ||
  Case When a.instance = b.instance Then ' '
        Else 'C ' End ||
  Substr(Strip(A.DBNAME)||'.'||Strip(A.NAME),1,20)
,B.partition
,B.NACTIVE
,CASE
  When REORGLASTTIME > COPYLASTTIME
  Then 'REORGLASTTIME > COPYLASTTIME ' ||
        'REORGLASTTIME=' || Char(REORGLASTTIME) || ' ' ||
        'COPYLASTTIME=' || Char(COPYLASTTIME)
  When LOADRLASTTIME > COPYLASTTIME
  Then 'LOADRLASTTIME > COPYLASTTIME ' ||
        'LOADRLASTTIME=' || Char(LOADRLASTTIME) || ' ' ||
        'COPYLASTTIME=' || Char(COPYLASTTIME)
  When (COPYUPDATEDPAGES*100)/B.NACTIVE > &ICRUPDATEDPAGESPCT
  Then '(COPYUPDATEDPAGES*100)/NACTIVE=' ||
        Strip(Char((COPYUPDATEDPAGES*100)/B.NACTIVE)) ||
        ' And is > &ICRUPDATEDPAGESPCT' ||
        ' COPYUPDATEDPAGES=' || Strip(Char(COPYUPDATEDPAGES)) || ' ' ||
        'NACTIVE=' || Strip(Char(B.NACTIVE))
  WHEN (COPYCHANGES*100)/TOTALROWS > &ICRCHANGESPCT
  Then '(COPYCHANGES*100)/TOTALROWS=' ||
        Strip(Char((COPYCHANGES*100)/TOTALROWS)) ||
        ' And is > &ICRCHANGESPCT' ||
        ' COPYCHANGES=' || Strip(Char(COPYCHANGES)) || ' ' ||
        'TOTALROWS=' || Strip(Char(TOTALROWS))
  END
FROM "SYSIBM"."SYSTABLESPACE" A INNER JOIN
     "SYSIBM"."&TSSTTB" B
  ON A.DBID = B.DBID
  AND A.PSID = B.PSID
WHERE A.DBNAME LIKE '&DB%'
AND
  B.NACTIVE > 0
  B.TOTALROWS > 0
  COPYLASTTIME IS NOT NULL
  (LOADRLASTTIME > COPYLASTTIME
  REORGLASTTIME > COPYLASTTIME
  (COPYUPDATEDPAGES*100)/B.NACTIVE > &ICRUPDATEDPAGESPCT
  (COPYCHANGES*100)/TOTALROWS > &ICRCHANGESPCT)
AND
AND
AND
OR
OR
OR

```

REORGIX, REORGIX9 and REORGIXS

contains query to build IP lines for all table spaces which requires a REORG against Real Time Statistics. The criteria are the same as DSNACCOR. Use REORGIXS rxd2i2 script to invoke REORGIX or REORGIX9. The format to run REORGIX is:

```

RUN IDSN=*REORGIX LIMIT(9999) T=N
  &IXSTTB=SYSINDEXSPACESTATS or INDEXSPACESTATS
  &DB=dbname
  &RRIINSERTDELETEPCT=20
  &RRIAPENDINSERTPCT=10
  &RRIPSEUDODELETEPCT=10
  &RRIMASSDELETELIMIT=0
  &RRILEAFLIMIT=10
  &RRINUMLEVELS=0

```

&EXTENTLIMIT=50

*Specify SYSINDEXSPACESTATS or INDEXSPACESTATS for &IXSTTB depends on which SYSIBM table name is used.

REORGIXS

```
-----
Say " ** REORGIXS - Generate Online Reorg IP with Real Time Statistics"
Say " ** Run *REORGIX9 to generate IP lines needed to be Reorged ...."
-----
Address Ispxexec "Vget db2ver Profile"
If db2ver = 8 Then
  str = "RUN LIMIT(9999) IDSN=*REORGIX T=N EDIT=Y odsn=w1.@JNM.IP",
        "&IXSTTB=INDEXSPACESTATS"
Else
  str = "RUN LIMIT(9999) IDSN=*REORGIX9 T=N EDIT=Y odsn=w1.@JNM.IP",
        "&IXSTTB=SYSINDEXSPACESTATS"
Db2i2rex str,
  "&DB=DDB2ADM",
  "&DB=DSN8D91A",
  "&RRIINSERTDELETEPCT=20",
  "&RRIAPENDINSERTPCT=10",
  "&RRIPSEUDODELETEPCT=10",
  "&RRIMASSDELETELIMIT=0",
  "&RRILEAFLIMIT=10",
  "&RRINUMLEVELS=0",
  "&EXTENTLIMIT=50"
  "&EXTENTLIMIT=1 trace(sql)"
-----
Say " ** Check If No IP lines Found ...."
-----
If RxNotfnd() = 'Y' Then;
Do
  Say " ** No IP Line Objects Meet Reorg Criteria **"
  x = Rxrc(4)
  db2i2rex "rxdb2i2 Reset"
  Return 4
End
-----
Say " ** Remove IPC line and resizeit to 1 Reorg jobs ...."
-----
db2i2rex "TSO Fcopy w1.@JNM.IP w2.@JNM.IP"
LINEOBJ ddname=ddl
x IPC 1 all
del x all
RESIZEIT 56,65,#JOB=1 LISTNM=RGID91A# LIST#=0
END_LINEOBJ
db2i2rex "ED w1.@JNM.IP macro(idd=ddl)"
-----
Say " ** Deleting listdef and db2i2.genjcl members ...."
-----
LINEOBJ ddname=dd3
IDCAMS delete db2adm.listdef(RGID91A1)
IDCAMS delete db2adm.db2i2.genjcl(RGID91A1)
IDCAMS If maxcc NE 0 Then set maxcc = 0
END_LINEOBJ
db2i2rex "EXEC IDD=dd3"
-----
Say " ** Building Listdef members - RGID91A1 ...."
-----
db2i2rex "LISTDEF parmutil=parmutil(listix)",
  "idsn=w1.@JNM.IP",
  "odsn=listdef(*)"
-----
Say " ** Building Online Reorg jobs - RGID91A1 ...."
-----
LINEOBJ ddname=LINEOBJ
dummy
END_LINEOBJ
db2i2rex "REORG",
  "parmutil='db2adm.parmutil(onlyrgix)'",
  "template='db2adm.template(onlyrgix)'",
  "listdef=listdef(RGID91A*)",
  "odsn=db2i2.genjcl(*)",
  "jobnm=RGID91A# job#=0"
```

REORGIX

```
-----
-- DSNACCOR formula for REORG on a IX
```

```

-- Default Host Variables
-- =====
-- &RRIINSERTDELETEPCT=20
-- &RRIAPPENDINSERTPCT=10
-- &RRIPSEUDODELETEPCT=10
-- &RRIMASSDELETELIMIT=0
-- &RRILEAFLIMIT=10
-- &RRINUMLEVELS=0
-- &EXTENTLIMIT=50
-----
SELECT
  Substr('IP '||Strip(A.CREATOR)||'. '||Strip(A.NAME),1,40)
,B.PARTITION
,B.NACTIVE
,CASE
  WHEN ((REORGINSERTS+REORGDELETES)*100)/TOTALENTRIES >
    &RRIINSERTDELETEPCT
  Then
    '((REORGINSERTS+REORGDELETES)*100)/TOTALENTRIES='||
    strip(char(((REORGINSERTS+REORGDELETES)*100)/
      TOTALENTRIES))||
    ' And is > &RRIINSERTDELETEPCT'
  WHEN (REORGAPPENDINSERT*100)/TOTALENTRIES > &RRIAPPENDINSERTPCT
  Then
    '(REORGAPPENDINSERT*100)/TOTALENTRIES='||
    strip(char((REORGAPPENDINSERT*100)/TOTALENTRIES))||
    ' And is > &RRIAPPENDINSERTPCT'
  WHEN (REORGSEUDODELETES*100)/TOTALENTRIES > &RRIPSEUDODELETEPCT
  Then
    '(REORGSEUDODELETES*100)/TOTALENTRIES='||
    strip(char((REORGSEUDODELETES*100)/TOTALENTRIES))||
    ' And is > &RRIPSEUDODELETEPCT'
  WHEN REORGMASSDELETE > &RRIMASSDELETELIMIT
  Then
    'REORGMASSDELETE='||
    strip(char(REORGMASSDELETE))||
    ' And is > &RRIMASSDELETELIMIT'
  WHEN (REORGLAFAFFAR*100)/NACTIVE > &RRILEAFLIMIT
  Then
    '(REORGLAFAFFAR*100)/NACTIVE='||
    strip(char((REORGLAFAFFAR*100)/NACTIVE))||
    ' And is > &RRILEAFLIMIT'
  WHEN REORGNUMLEVELS > &RRINUMLEVELS
  Then
    'REORGNUMLEVELS='||
    strip(char(REORGNUMLEVELS))||
    ' And is > &RRINUMLEVELS'
  WHEN EXTENTS > &EXTENTLIMIT
  Then
    'EXTENTS='||
    strip(char(EXTENTS))||
    ' And is > &EXTENTLIMIT'
  WHEN REORGLASTTIME IS NULL
  Then 'REORGLASTTIME IS NULL'
END
FROM "SYSIBM"."SYSINDEXES" A      INNER JOIN
  "SYSIBM"."IXSTTB" B
  ON A.DBID = B.DBID
AND A.ISOBID = B.ISOBID
WHERE A.DBNAME LIKE '&DB%'
AND A.COPY = 'Y'
AND
  B.TOTALENTRIES > 0
  (REORGLASTTIME IS NULL OR
  ((REORGINSERTS+REORGDELETES)*100)/TOTALENTRIES >
  &RRIINSERTDELETEPCT
  (REORGAPPENDINSERT*100)/TOTALENTRIES > &RRIAPPENDINSERTPCT
  (REORGSEUDODELETES*100)/TOTALENTRIES > &RRIPSEUDODELETEPCT
  REORGMASSDELETE > &RRIMASSDELETELIMIT
  (REORGLAFAFFAR*100)/NACTIVE > &RRILEAFLIMIT
  REORGNUMLEVELS > &RRINUMLEVELS
  EXTENTS > &EXTENTLIMIT)
  AND
  OR
  OR
  OR
  OR
  OR
  OR

```

REORGIX9 - include Clone table information for V9

```

-----
-- DSNACCOR formula for REORG on a IX
-- Default Host Variables
-----
-- &RRIINSERTDELETEPCT=20
-- &RRIAPPENDINSERTPCT=10
-- &RRIPSEUDODELETEPCT=10
-- &RRIMASSDELETELIMIT=0
-- &RRILEAFLIMIT=10
-- &RRINUMLEVELS=0
-- &EXTENTLIMIT=50
-----
SELECT 'IP' ||
  Case When b.instance = d.instance Then ' '
        Else 'C' End ||
  Substr(Strip(A.CREATOR)||'.'||Strip(A.NAME),1,40)
,B.PARTITION
,B.NACTIVE
,CASE
WHEN ((REORGINSERTS+REORGDELETES)*100)/TOTALENTRIES >
  &RRIINSERTDELETEPCT
Then
  '((REORGINSERTS+REORGDELETES)*100)/TOTALENTRIES=' ||
  strip(char(((REORGINSERTS+REORGDELETES)*100)/
    TOTALENTRIES)) ||
  ' And is > &RRIINSERTDELETEPCT'
WHEN (REORGAPPENDINSERT*100)/TOTALENTRIES > &RRIAPPENDINSERTPCT
Then
  '(REORGAPPENDINSERT*100)/TOTALENTRIES=' ||
  strip(char((REORGAPPENDINSERT*100)/TOTALENTRIES)) ||
  ' And is > &RRIAPPENDINSERTPCT'
WHEN (REORGSEUDODELETES*100)/TOTALENTRIES > &RRIPSEUDODELETEPCT
Then
  '(REORGSEUDODELETES*100)/TOTALENTRIES=' ||
  strip(char((REORGSEUDODELETES*100)/TOTALENTRIES)) ||
  ' And is > &RRIPSEUDODELETEPCT'
WHEN REORGMASSDELETE > &RRIMASSDELETELIMIT
Then
  'REORGMASSDELETE=' ||
  strip(char(REORGMASSDELETE)) ||
  ' And is > &RRIMASSDELETELIMIT'
WHEN (REORGLAFFAR*100)/B.NACTIVE > &RRILEAFLIMIT
Then
  '(REORGLAFFAR*100)/NACTIVE=' ||
  strip(char((REORGLAFFAR*100)/B.NACTIVE)) ||
  ' And is > &RRILEAFLIMIT'
WHEN REORGNUMLEVELS > &RRINUMLEVELS
Then
  'REORGNUMLEVELS=' ||
  strip(char(REORGNUMLEVELS)) ||
  ' And is > &RRINUMLEVELS'
WHEN EXTENTS > &EXTENTLIMIT
Then
  'EXTENTS=' ||
  strip(char(EXTENTS)) ||
  ' And is > &EXTENTLIMIT'
WHEN REORGLASTTIME IS NULL
Then 'REORGLASTTIME IS NULL'
END
FROM "SYSIBM"."SYSINDEXES" A      INNER JOIN
     "SYSIBM"."IXSTTB" B
ON A.DBID = B.DBID
AND A.ISOBID = B.ISOBID        INNER JOIN
     "SYSIBM"."SYSTABLES" C
ON A.TBCREATOR = C.CREATOR
AND A.TBNAME = B.NAME          INNER JOIN
     "SYSIBM"."SYSTABLESPACE" D
ON C.DBNAME = D.DBNAME
AND C.TSNAME = D.NAME
WHERE A.DBNAME LIKE '&DB%'
AND A.COPY = 'Y'
AND
  B.TOTALENTRIES > 0
  (REORGLASTTIME IS NULL OR
  ((REORGINSERTS+REORGDELETES)*100)/TOTALENTRIES >
  &RRIINSERTDELETEPCT
  (REORGAPPENDINSERT*100)/TOTALENTRIES > &RRIAPPENDINSERTPCT
  )

```

```
(REORGSEUDODELETES*100)/TOTALENTRIES > &RRIPSEUDODELETEPCT OR
REORGMASSEDELETE > &RRIMASSEDELETELIMIT OR
(REORGLAFAFFAR*100)/B.NACTIVE > &RRILEAFLIMIT OR
REORGNUMLEVELS > &RRINUMLEVELS OR
EXTENTS > &EXTENTLIMIT)
```

REORGTSS, REORGTSS9 and REORGTSS

contains query to build TP lines for all table spaces which requires a REORG against Real Time Statistics. The criteria are the same as DSNACCOR. Use REORGTSS rxd2i2 script to invoke REORGTSS or REORGTSS9. The format to run REORGTSS is:

```
RUN IDSN=*REORGTSS LIMIT(9999) T=N
  &TSSTTB=SYSTABLESPACESTATS or TABLESPACESTATS
  &DB=dbname
  &RRTINSDELUPDPCT=20
  &RRTUNCLUSTINSPCT=10
  &RRTDISORGLLOBPCT=10
  &RRTINDREFLIMIT=10
  &RRTMASSEDELLIMIT=0
  &EXTENTLIMIT=50
```

*Specify SYSTABLESPACESTATS or TABLESPACESTATS for &TSSTTB depends on which SYSIBM table name is used.

REORGTSS

```
-----
Say " ** REORGTSS - Generate REORG TP with Real Time Statistics"
Say " ** Mapping Tbl for Online Reorg is Automatically Built"
Say " ** Processing Run Command to Generate TP lines ...."
-----
Address Ispexec "Vget db2ver Profile"
If db2ver = 8 Then
  str = "RUN idsn=*REORGTSS LIMIT(9999) T=N odsn=w1.@JNM.TP",
        "&TSSTTB=TABLESPACESTATS"
Else
  str = "RUN idsn=*REORGTSS9 LIMIT(9999) T=N odsn=w1.@JNM.TP",
        "&TSSTTB=SYSTABLESPACESTATS"
db2i2rex str,
  "&DB=DSN8D91A",
--  "&DB=DDB2ADM",
  "&RRTINSDELUPDPCT=20",
  "&RRTUNCLUSTINSPCT=10",
  "&RRTDISORGLLOBPCT=10",
  "&RRTINDREFLIMIT=10",
  "&RRTMASSEDELLIMIT=0",
  "&EXTENTLIMIT=50"
-----
Say " ** Check If no TP Lines ...."
-----
If RxNotfnd() = 'Y' Then;
Do
  Say " ** No TP Line Objects Meet Reorg Criteria **"
  x = Rxrc(4)
  db2i2rex "Rxd2i2 Reset"
  Return 4
End
-----
Say " ** Removing TPC line and resizeit to 3 Reorg jobs ...."
-----
db2i2rex "TSO Fcpy w1.@JNM.TP w2.@JNM.TP"
LINEOBJ ddname=ddl
x TPC 1 all
del x all
RESIZEIT 36,45,#JOB=3 LISTNM=ROGD91A# LIST#=0
END_LINEOBJ
db2i2rex "ED w1.@JNM.TP macro(idd=ddl)"
-----
Say " ** Deleting listdef and db2i2.genjcl members ...."
-----
```

```

LINEOBJ ddname=dd3
IDCAMS delete db2adm.listdef(ROGD91A1)
IDCAMS delete db2adm.listdef(ROGD91A2)
IDCAMS delete db2adm.listdef(ROGD91A3)
IDCAMS delete db2adm.db2i2.genjcl(ROGD91A1)
IDCAMS delete db2adm.db2i2.genjcl(ROGD91A2)
IDCAMS delete db2adm.db2i2.genjcl(ROGD91A3)
IDCAMS If Maxcc NE 0 Then Set Maxcc = 0
END_LINEOBJ
db2i2rex "EXEC IDD=dd3"
-----
Say " ** Building Listdef members - ROGD91A# ...."
-----
db2i2rex "LISTDEF parmutil=parmutil(listts)",
         "idsn=w1.@JNM.TP",
         "odsn=listdef(*)"
-----
Say " ** Building Online Reorg Jobs - ROGD91A# ...."
-----
LINEOBJ ddname=LINEOBJ
dummy
END_LINEOBJ
db2i2rex "REORG",
         "parmutil='db2adm.parmutil(reorgoln)'",
         "template='db2adm.template(reorgoln)'",
         "listdef=listdef(ROGD91A*)",
         "mactable=DB2ADM.MAPPING_<jobnm>",
         "odsn=db2i2.genjcl(*)",
         "jobnm=ROGD91A# job#=0 macro(idd=ddisrt)"
-----
-- Mapping Table create DDL
-----
LINEOBJ DDNAME=DDCRE
EXEC SQL
CREATE TABLE "DB2ADM"."MAPPING_{cjob}"
(
  "TYPE"                CHAR(1)                NOT NULL
  ,"SOURCE_RID"         CHAR(5)                NOT NULL
  ,"TARGET_XRID"       CHAR(9)                NOT NULL
  ,"LRSN"              CHAR(6)                NOT NULL
)
                                IN DBMAP.TSMAP
AUDIT NONE
DATA CAPTURE NONE
CCSID EBCDIC
ENDEXEC
EXEC SQL
CREATE TYPE 2 UNIQUE
INDEX "DB2ADM"."MAPPING_{cjob}_INDEX"
ON "DB2ADM"."MAPPING_{cjob}"
( "SOURCE_RID" ASC
  ,"TYPE" ASC
  ,"TARGET_XRID" ASC
  ,"LRSN" ASC
)
USING STOGROUP GDB2ADM
PRIQTY 720
SECQTY 72000
ERASE NO
FREEPAGE 0
PCTFREE 10
GBPCACHE CHANGED
CLOSE YES
ENDEXEC
END_LINEOBJ
-- Drop DDL
LINEOBJ DDNAME=DDDROP
EXEC SQL
DROP TABLE "DB2ADM"."MAPPING_{cjob}"
ENDEXEC
END_LINEOBJ
LINEOBJ DDNAME=DDISRT
F ' JOB ' First
Cline
Rexx Parse Value cline with '/' cjob ' JOB' junk
F '//SYSIN' 1
Inserta * idd=ddcre
bottom

```

```
Inserta * idd=ddd drop
END_LINEOBJ
```

REORGT5

```
-----
-- DSNACCOR formula for REORG on a TS
-- Default Host Variables
-- =====
-- &RRTINSDELUPDPCT=20
-- &RRTUNCLUSTINSPECT=10
-- &RRTDISORGLBPCT=10
-- &RRTINDREFLIMIT=10
-- &RRTMASSEDELLIMIT=0
-- &EXTENTLIMIT=50
-----
SELECT
  Substr('TP '||Strip(A.DBNAME)||'.'||Strip(A.NAME),1,30)
,B.PARTITION
,B.NACTIVE
,CASE
  When ((REORGINSERTS+REORGDELETES+REORGUPDATES)*100)/TOTALROWS >
    &RRTINSDELUPDPCT
  Then '((REORGINSERTS+REORGDELETES+REORGUPDATES)*100)/TOTALROWS='||
    Strip(Char(((REORGINSERTS+REORGDELETES+REORGUPDATES)*100)/
      TOTALROWS))||
    ' And Is > &RRTINSDELUPDPCT'
  WHEN (REORGUNCLUSTINS*100)/TOTALROWS > &RRTUNCLUSTINSPECT
  Then '(REORGUNCLUSTINS*100)/TOTALROWS='||
    Strip(Char((REORGUNCLUSTINS*100)/TOTALROWS))||
    ' And Is > &RRTUNCLUSTINSPECT'
  WHEN (REORGDISORGLBPCT*100)/TOTALROWS > &RRTDISORGLBPCT
  Then '(REORGDISORGLBPCT*100)/TOTALROWS='||
    Strip(Char((REORGDISORGLBPCT*100)/TOTALROWS))||
    ' And Is > &RRTDISORGLBPCT'
  When ((REORGNEARINDREF+REORGFARINDREF)*100)/TOTALROWS >
    &RRTINDREFLIMIT
  Then '((REORGNEARINDREF+REORGFARINDREF)*100)/TOTALROWS='||
    Strip(Char((REORGNEARINDREF+REORGFARINDREF)*100)/
      TOTALROWS))||
    ' And Is > &RRTINDREFLIMIT'
  WHEN REORGMASSEDELETE > &RRTMASSEDELLIMIT
  Then 'REORGMASSEDELETE='||
    Strip(Char(REORGMASSEDELETE))||' And is > &RRTMASSEDELLIMIT'
  WHEN EXTENTS > &EXTENTLIMIT
  Then 'EXTENTS='||
    Strip(Char(EXTENTS))||' And is > &EXTENTLIMIT'
  WHEN REORGLASTTIME IS NULL
  Then 'REORGLASTTIME IS NULL'
END
FROM "SYSIBM"."SYSTABLESPACE" A INNER JOIN
  "SYSIBM"."%TSSTTB" B
  ON A.DBID = B.DBID
  AND A.PSID = B.PSID
WHERE A.DBNAME LIKE '&DB%'
AND TOTALROWS > 0
AND
  (REORGLASTTIME IS NULL OR
  ((REORGINSERTS+REORGDELETES+REORGUPDATES)*100)/TOTALROWS >
  &RRTINSDELUPDPCT OR
  (REORGUNCLUSTINS*100)/TOTALROWS > &RRTUNCLUSTINSPECT OR
  (REORGDISORGLBPCT*100)/TOTALROWS > &RRTDISORGLBPCT OR
  ((REORGNEARINDREF+REORGFARINDREF)*100)/TOTALROWS > &RRTINDREFLIMIT OR
  REORGMASSEDELETE > &RRTMASSEDELLIMIT OR
  EXTENTS > &EXTENTLIMIT)
```

REORGT59 - include Clone table information for V9

```
-----
-- DSNACCOR formula for REORG on a TS
-- Default Host Variables
-- =====
-- &RRTINSDELUPDPCT=20
-- &RRTUNCLUSTINSPECT=10
-- &RRTDISORGLBPCT=10
-- &RRTINDREFLIMIT=10
-- &RRTMASSEDELLIMIT=0
-- &EXTENTLIMIT=50
```



```

-----
SELECT 'TP' ||
  Case When a.instance = b.instance Then ' '
        Else 'C ' End ||
  Substr(Strip(A.DBNAME)||'. '||Strip(A.NAME),1,20)
,B.PARTITION
,B.NACTIVE
,CASE
  When ((REORGINSERTS+REORGDELETES+REORGUPDATES)*100)/TOTALROWS >
    &RRTINSDELUPDPCT
  Then '((REORGINSERTS+REORGDELETES+REORGUPDATES)*100)/TOTALROWS=' ||
    Strip(Char(((REORGINSERTS+REORGDELETES+REORGUPDATES)*100)/
      TOTALROWS)) ||
    ' And Is > &RRTINSDELUPDPCT'
WHEN (REORGUNCLUSTINS*100)/TOTALROWS > &RRTUNCLUSTINSPECT
  Then '(REORGUNCLUSTINS*100)/TOTALROWS=' ||
    Strip(Char((REORGUNCLUSTINS*100)/TOTALROWS)) ||
    ' And Is > &RRTUNCLUSTINSPECT'
WHEN (REORGDISORGL*100)/TOTALROWS > &RRTDISORGLBPCT
  Then '(REORGDISORGL*100)/TOTALROWS=' ||
    Strip(Char((REORGDISORGL*100)/TOTALROWS)) ||
    ' And Is > &RRTDISORGLBPCT'
When ((REORGNEARINDREF+REORGFARINDREF)*100)/TOTALROWS >
  &RRTINDREFLIMIT
  Then '((REORGNEARINDREF+REORGFARINDREF)*100)/TOTALROWS=' ||
    Strip(Char(((REORGNEARINDREF+REORGFARINDREF)*100)/
      TOTALROWS)) ||
    ' And Is > &RRTINDREFLIMIT'
WHEN REORGMASDELETE > &RRTMASDELLIMIT
  Then 'REORGMASDELETE=' ||
    Strip(Char(REORGMASDELETE)) || ' And is > &RRTMASDELLIMIT'
WHEN EXTENTS > &EXTENTLIMIT
  Then 'EXTENTS=' ||
    Strip(Char(EXTENTS)) || ' And is > &EXTENTLIMIT'
WHEN REORGLASTTIME IS NULL
  Then 'REORGLASTTIME IS NULL'
END
FROM "SYSIBM"."SYSTABLESPACE" A INNER JOIN
  "SYSIBM"."&TSSTTB" B
  ON A.DBID = B.DBID
  AND A.PSID = B.PSID
WHERE A.DBNAME LIKE '&DB%'
AND TOTALROWS > 0
AND
  (REORGLASTTIME IS NULL OR
  ((REORGINSERTS+REORGDELETES+REORGUPDATES)*100)/TOTALROWS >
  &RRTINSDELUPDPCT OR
  (REORGUNCLUSTINS*100)/TOTALROWS > &RRTUNCLUSTINSPECT OR
  (REORGDISORGL*100)/TOTALROWS > &RRTDISORGLBPCT OR
  ((REORGNEARINDREF+REORGFARINDREF)*100)/TOTALROWS > &RRTINDREFLIMIT OR
  REORGMASDELETE > &RRTMASDELLIMIT OR
  EXTENTS > &EXTENTLIMIT)

```

RUNSIX , RUNSIX9 and RUNSIXS

contains query to build IP lines for all table spaces which requires a RUNSTATS against Real Time Statistics. The criteria are the same as DSNACCOR. Use RUNSIXS rxdb2i2 script to invoke RUNSIX or RUNSIX9. The format to run RUNSIX is:

```

RUN IDSN=*RUNSIX LIMIT(9999) T=N
  &IXSTTB=SYSINDEXSPACESTATS or INDEXSPACESTATS
  &DB=dbname
  &SRIINDELUPDPCT=20
  &SRIINDELALBS=0
  &SRIMASDELETELIMIT=0

```

*Specify SYSINDEXSPACESTATS or INDEXSPACESTATS for &IXSTTB depends on which SYSIBM table name is used.

RUNSIXS

```

-----
Say " ** REORGTSS - Generate RUNSTATS IP with Real Time Statistics"
Say " ** Run *RUNSIX9 to generate IP lines needed to be Runstats ...."
-----
Address Ispexec "Vget db2ver Profile"
If db2ver = 8 Then
    str = "RUN LIMIT(9999) IDSN=*RUNSIX T=N odsn=w1.@JNM.IP",
        "&IXSTTB=INDEXSPACESTATS"
Else
    str = "RUN LIMIT(9999) IDSN=*RUNSIX9 T=N odsn=w1.@JNM.IP",
        "&IXSTTB=SYSINDEXSPACESTATS"
db2i2rex str,
--      "&DB=DDB2ADM",
--      "&DB=DSN8D91A",
--      "&SRIINSDELUPDPCT=20",
--      "&SRIINSDELABS=0",
--      "&SRIMASSDELETELIMIT=0"
-----
Say " ** Check If No IP lines Found ...."
-----
If RxNotfnd() = 'Y' Then;
Do
    Say " ** No IP Line Objects Meet Runstats Criteria **"
    x = Rxrc(4)
    db2i2rex "Rxdb2i2 Reset"
    Return 4
End
-----
Say " ** Removing IPC line and resizeit to 1 Runstats jobs ...."
-----
db2i2rex "TSO Fcopy w1.@JNM.IP w2.@JNM.IP"
LINEOBJ ddname=ddl
x IPC 1 all
del x all
RESIZEIT 36,45,#JOB=1 LISTNM=RSID91A# LIST#=0
END_LINEOBJ
db2i2rex "ED w1.@JNM.TP macro(idd=ddl)"
-----
Say " ** Deleting listdef and db2i2.genjcl members ...."
-----
LINEOBJ ddname=dd3
IDCAMS delete db2adm.listdef(RSID91A1)
IDCAMS delete db2adm.db2i2.genjcl(RSID91A1)
IDCAMS If maxcc NE 0 Then set maxcc = 0
END_LINEOBJ
db2i2rex "EXEC IDD=dd3"
-----
Say " ** Building Listdef member - RSID91A1 ...."
-----
db2i2rex "LISTDEF parmutil=parmutil(listts)",
        "idsn=w1.@JNM.TP",
        "listnm=RSID91A1",
        "odsn=listdef(*)"
-----
Say " ** Building Runstats with Share Level Change - RSID91A1 ...."
-----
LINEOBJ ddname=LINEOBJ
dummy
END_LINEOBJ
LINEOBJ ddname=dd4
TOP
END_LINEOBJ
db2i2rex "RUNSTATS",
        "parmutil='db2adm.parmutil(runxchg)'",
        "listdef=listdef(RSID91A1)",
        "odsn=db2i2.genjcl(*)",
        "jobnm=RSID91A# job#=0 macro(idd=dd4)"

```

RUNSIX

```

-----
-- DSNACCOR formula for Runstats on a IS
-- Default Host Variables
-- =====
-- &SRIINSDELUPDPCT=20
-- &SRIINSDELABS=0
-- &SRIMASSDELETELIMIT=0

```

```

-----
SELECT
  Substr('IP '||Strip(A.CREATOR)||'.'||Strip(A.NAME),1,40)
,B.partition
,B.NACTIVE
,CASE
  When STATSLASTTIME IS NULL
  Then 'STATSLASTTIME IS NULL'
  When ((STATSINSERTS+STATSDELETES)*100)/TOTALENTRIES > &SRIINSDELUPDPCT
  Then '((STATSINSERTS+STATSDELETES)*100)/TOTALENTRIES='||
    Strip(Char(((STATSINSERTS+STATSDELETES)*100)/
      TOTALENTRIES))||' And is > &SRIINSDELUPDPCT' ||
    ' STATSINSERTS='||Strip(Char(STATSINSERTS))||
    ' STATSDELETES='||Strip(Char(STATSDELETES))||
    ' TOTALENTRIES='||Strip(Char(TOTALENTRIES))
  When STATSINSERTS+STATSDELETES > &SRIINSDELABS
  Then 'STATSINSERTS+STATSDELETES='||
    Strip(Char(STATSINSERTS+STATSDELETES))||
    ' And is > &SRIINSDELABS' ||
    ' STATSINSERTS='||Strip(Char(STATSINSERTS))||
    ' STATSDELETES='||Strip(Char(STATSDELETES))
  When STATSMASDELETE > &SRIMASSDELETELIMIT
  Then 'STATSMASDELETE='||
    Strip(Char(STATSMASDELETE))||' And is > &SRIMASSDELETELIMIT'
END
FROM "SYSIBM"."SYSINDEXES" A      INNER JOIN
     "SYSIBM"."&IXSTTB" B
  ON A.DBID = B.DBID
  AND A.ISOBID = B.ISOBID
WHERE A.DBNAME = 'DB'
AND
  B.TOTALENTRIES > 0 AND
  (STATSLASTTIME IS NULL OR
  (((STATSINSERTS+STATSDELETES)*100)/TOTALENTRIES >
  &SRIINSDELUPDPCT AND
  (STATSINSERTS+STATSDELETES) > &SRIINSDELABS) OR
  STATSMASDELETE > &SRIMASSDELETELIMIT)

```

RUNSIX9 - include Clone table information for v9

```

-----
-- DSNACCOR formula for Runstats on a IS
-- Default Host Variables
-----
-- &SRIINSDELUPDPCT=20
-- &SRIINSDELABS=0
-- &SRIMASSDELETELIMIT=0
-----
SELECT 'IP' ||
  Case When b.instance = d.instance Then ' '
        Else 'C' End ||
  Substr(Strip(A.CREATOR)||'.'||Strip(A.NAME),1,40)
,B.partition
,B.NACTIVE
,CASE
  When STATSLASTTIME IS NULL
  Then 'STATSLASTTIME IS NULL'
  When ((STATSINSERTS+STATSDELETES)*100)/TOTALENTRIES > &SRIINSDELUPDPCT
  Then '((STATSINSERTS+STATSDELETES)*100)/TOTALENTRIES='||
    Strip(Char(((STATSINSERTS+STATSDELETES)*100)/
      TOTALENTRIES))||' And is > &SRIINSDELUPDPCT' ||
    ' STATSINSERTS='||Strip(Char(STATSINSERTS))||
    ' STATSDELETES='||Strip(Char(STATSDELETES))||
    ' TOTALENTRIES='||Strip(Char(TOTALENTRIES))
  When STATSINSERTS+STATSDELETES > &SRIINSDELABS
  Then 'STATSINSERTS+STATSDELETES='||
    Strip(Char(STATSINSERTS+STATSDELETES))||
    ' And is > &SRIINSDELABS' ||
    ' STATSINSERTS='||Strip(Char(STATSINSERTS))||
    ' STATSDELETES='||Strip(Char(STATSDELETES))
  When STATSMASDELETE > &SRIMASSDELETELIMIT
  Then 'STATSMASDELETE='||
    Strip(Char(STATSMASDELETE))||' And is > &SRIMASSDELETELIMIT'
END
FROM "SYSIBM"."SYSINDEXES" A      INNER JOIN
     "SYSIBM"."&IXSTTB" B
  ON A.DBID = B.DBID

```

```

AND A.ISOBID = B.ISOBID          INNER JOIN
   "SYSIBM"."SYSTABLES" C
  ON A.TBCREATOR = C.CREATOR
AND A.TBNAME   = B.NAME          INNER JOIN
   "SYSIBM"."SYSTABLESPACE" D
  ON C.DBNAME  = D.DBNAME
AND C.TSNAME   = D.NAME
WHERE A.DBNAME = '&DB'
AND
  B.TOTALENTRIES > 0 AND
  (STATSLASTTIME IS NULL OR
   (((STATSINSERTS+STATSDELETES)*100)/TOTALENTRIES >
    &SRIINSDELUPDPCT AND
   (STATSINSERTS+STATSDELETES) > &SRIINSDELABS) OR
   STATSMASSEDELETE > &SRIMASSDELETELIMIT)

```

RUNSTS, RUNSTS9 and RUNSTSS

contains query to build TP lines for all table spaces which requires a RUNSTATS against Real Time Statistics. The criteria are the same as DSNACCOR. Use RUNSTSS rxdb2i2 script to invoke RUNSTS or RUNSTS9. The format to run REORGTs is:

```

RUN IDSN=*RUNSTS LIMIT(9999) T=N
   &TSSTTB=SYSTABLESPACESTATS or TABLESPACESTATS
   &DB=dbname
   &SRTINSDELUPDPCT=20
   &SRTINSDELUPDABS=0
   &SRTMASSEDELETELIMIT=0

```

*Specify SYSTABLESPACESTATS or TABLESPACESTATS for &TSSTTB depends on which SYSIBM table name is used.

RUNSTSS

```

-----
Say " ** RUNSTSS - Generate Runstats TP with Real Time Statistics"
Say " ** Run *RUNSTS9 to generate TP lines needed to be Runstats ..."
-----
Address Ispexec "Vget db2ver Profile"
If db2ver = 8 Then
  str = "RUN LIMIT(9999) IDSN=*RUNSTS T=N odsn=w1.@JNM.TP",
        "&TSSTTB=TABLESPACESTATS"
Else
  str = "RUN LIMIT(9999) IDSN=*RUNSTS9 T=N odsn=w1.@JNM.TP",
        "&TSSTTB=SYSTABLESPACESTATS"
db2i2rex str,
--      "&DB=DDB2ADM",
        "&DB=DSN8D91A",
        "&SRTINSDELUPDPCT=20",
        "&SRTINSDELUPDABS=0",
        "&SRTMASSEDELETELIMIT=0"
-----
Say " ** Check If No TP lines Found ...."
-----
If RxNotfnd() = 'Y' Then;
Do
  Say " ** No TP Line Objects Meet Reorg Criteria **"
  x = Rxrc(4)
  db2i2rex "Rxdb2i2 Reset"
  Return 4
End
-----
Say " ** Remove TPC line and resizeit to 1 Runstats Jobs ...."
-----
db2i2rex "TSO Fcopy w1.@JNM.TP w2.@JNM.TP"
LINEOBJ ddname=ddl
x TPC 1 all
del x all
RESIZEIT 36,45,#JOB=1 LISTNM=RSSD91A# LIST#=0
END_LINEOBJ
db2i2rex "ED w1.@JNM.TP macro(idd=ddl)"

```

```

-----
Say " ** Deleteing listdef and db2i2.genjcl members - RSSD91A1 ...."
-----
LINEOBJ ddname=dd3
IDCAMs delete db2adm.listdef(RSSD91A1)
IDCAMs delete db2adm.db2i2.genjcl(RSSD91A1)
IDCAMs If maxcc NE 0 Then set maxcc = 0
END_LINEOBJ
db2i2rex "EXEC IDD=dd3"
-----
Say " ** Building listdef member - RSSD91A1 ...."
-----
db2i2rex "LISTDEF parmutil=parmutil(listts)",
         "idsn=w1.@JNM.TP",
         "listnm=RSSD91A1",
         "odsn=listdef(*)"
-----
Say " ** Building Runstats with Share Level Change - RSSD91A1 ...."
-----
LINEOBJ ddname=LINEOBJ
ts ddb2adm.sdb2adm
END_LINEOBJ
LINEOBJ ddname=dd4
Rexx Say 'Change this line to Submit to submit this generated Job'
-- Submit
END_LINEOBJ
db2i2rex "RUNSTATS",
         "parmutil='db2adm.parmutil(runschg)'",
         "listdef=listdef(RSSD91A1)",
         "odsn=db2i2.genjcl(*)",
         "jobnm=RSSD91A# job#=0 macro(idd=dd4)"

```

RUNSTS

```

-----
-- DSNACCOR formula for Runstats on a TS
-- Default Host Variables
-----
-- &SRTINSDELUPDPCT=20
-- &SRTINSDELUPDABS=0
-- &SRTMASSDELETELIMIT=0
-----
SELECT
  Substr('TP '||Strip(A.DBNAME)||'. '||Strip(A.NAME),1,30)
,B.PARTITION
,B.NACTIVE
,CASE
  When STATSLASTTIME IS NULL
  Then 'STATSLASTTIME IS NULL'
  When ((STATSINSERTS+STATSDELETES+STATSUPDATES)*100)/TOTALROWS >
    &SRTINSDELUPDPCT
  Then '(((STATSINSERTS+STATSDELETES+STATSUPDATES)*100)/TOTALROWS='||
    Strip(Char(((STATSINSERTS+STATSDELETES+STATSUPDATES)*100)/
      TOTALROWS))||' And is > &SRTINSDELUPDPCT' ||
    ' STATSINSERTS='||Strip(Char(STATSINSERTS))||
    ' STATSDELETES='||Strip(Char(STATSDELETES))||
    ' STATSUPDATES='||Strip(Char(STATSUPDATES))||
    ' TOTALROWS='||Strip(Char(TOTALROWS))
  When STATSINSERTS+STATSDELETES+STATSUPDATES > &SRTINSDELUPDABS
  Then 'STATSINSERTS+STATSDELETES+STATSUPDATES='||
    Strip(Char(STATSINSERTS+STATSDELETES+STATSUPDATES))||
    ' And is > &SRTINSDELUPDABS' ||
    ' STATSINSERTS='||Strip(Char(STATSINSERTS))||
    ' STATSDELETES='||Strip(Char(STATSDELETES))||
    ' STATSUPDATES='||Strip(Char(STATSUPDATES))
  When STATSMASSELETE > &SRTMASSDELETELIMIT
  Then 'STATSMASSELETE='||
    Strip(Char(STATSMASSELETE))||' And is > &SRTMASSDELETELIMIT'
END
FROM "SYSIBM"."SYSTABLESPACE" A INNER JOIN
     "SYSIBM"."&TSSTTB" B
  ON A.DBID = B.DBID
  AND A.PSID = B.PSID
WHERE A.DBNAME LIKE '&DB%'
AND
  (STATSLASTTIME IS NULL OR
   (B.TOTALROWS > 0 AND
    ((STATSINSERTS+STATSDELETES+STATSUPDATES)*100)/TOTALROWS >
     &SRTINSDELUPDPCT AND

```

```
(STATSINSERTS+STATSDELETES+STATSUPDATES) > &SRTINSDELUPDABS)) OR
STATSMASDELETE > &SRTMASDELETELIMIT)
```

RUNSTS9 - include Clone table information for V9

```
-----
-- DSNACCOR formula for Runstats on a TS
-- Default Host Variables
-- =====
-- &SRTINSDELUPDPCT=20
-- &SRTINSDELUPDABS=0
-- &SRTMASDELETELIMIT=0
-----
SELECT 'TP' ||
  Case When a.instance = b.instance Then ' '
        Else 'C ' End ||
  Substr(Strip(A.DBNAME)||'.'||Strip(A.NAME),1,20)
,B.PARTITION
,B.NACTIVE
,CASE
  When STATSLASTTIME IS NULL
  Then 'STATSLASTTIME IS NULL'
  When ((STATSINSERTS+STATSDELETES+STATSUPDATES)*100)/TOTALROWS >
&SRTINSDELUPDPCT
  Then '((STATSINSERTS+STATSDELETES+STATSUPDATES)*100)/TOTALROWS=' ||
Strip(Char(((STATSINSERTS+STATSDELETES+STATSUPDATES)*100)/
TOTALROWS))||' And is > &SRTINSDELUPDPCT' ||
' STATSINSERTS=' || Strip(Char(STATSINSERTS)) ||
' STATSDELETES=' || Strip(Char(STATSDELETES)) ||
' STATSUPDATES=' || Strip(Char(STATSUPDATES)) ||
' TOTALROWS=' || Strip(Char(TOTALROWS))
  When STATSINSERTS+STATSDELETES+STATSUPDATES > &SRTINSDELUPDABS
  Then 'STATSINSERTS+STATSDELETES+STATSUPDATES=' ||
Strip(Char(STATSINSERTS+STATSDELETES+STATSUPDATES)) ||
' And is > &SRTINSDELUPDABS' ||
' STATSINSERTS=' || Strip(Char(STATSINSERTS)) ||
' STATSDELETES=' || Strip(Char(STATSDELETES)) ||
' STATSUPDATES=' || Strip(Char(STATSUPDATES))
  When STATSMASDELETE > &SRTMASDELETELIMIT
  Then 'STATSMASDELETE=' ||
Strip(Char(STATSMASDELETE))||' And is > &SRTMASDELETELIMIT'
END
FROM "SYSIBM"."SYSTABLESPACE" A INNER JOIN
"SYSIBM"."%TSSTTB" B
ON A.DBID = B.DBID
AND A.PSID = B.PSID
WHERE A.DBNAME LIKE '&DB%'
AND
(STATSLASTTIME IS NULL OR
(B.TOTALROWS > 0 AND
((STATSINSERTS+STATSDELETES+STATSUPDATES)*100)/TOTALROWS >
&SRTINSDELUPDPCT AND
(STATSINSERTS+STATSDELETES+STATSUPDATES) > &SRTINSDELUPDABS)) OR
STATSMASDELETE > &SRTMASDELETELIMIT)
```

Special Case Study

There are 13 special case studies covered in this section. The purpose of these special case studies is to demonstrate some of the advanced features of DB2I2. They show you how flexible and powerful a DB2I2 workbench can be.

Case Study 1: Database Migration Script Generation

Migrate a DB2 database from DB1 to DB2.

The following are detail migration specifications:

- Changes all the table creator and index creator from TB1 to TB2
- Changes all the Stogroup name from SG1 to SG2
- Changes all the collection ID for packages from CL1 to CL2
- Set the current SQLID to SYSADM2 for all the activities
- Reduce space allocation 50% from the current allocation and rounded it up to cylinder boundary

The following screen is how you code the DB2I2 commands in batch mode to accomplish all the above migration specifications.

```

000001 //DB2I2CMD DD *
000002 MIGR odsn=t1 AL=Y SY=Y VW=Y GRANT=Y BIND=Y +
000003     -50% CYL SQLID=sysadm2
000004 FCPY t1 t2
000005 ED t2 MACRO(IDD=eddd)
000006 SUPERC t1 t2 odsn=t3
000007 FLIST t3
000008 //LINEOBJ DD *
000009 db db1
000010 //EDDD DD *
000011 c 'DB1' 'DB2' all
000012 c 'SG1' 'SG2' all
000013 c 'TB1' 'TB2' all
000014 c 'CL1' 'CL2' all
000015 /*

```

- Line 2-3: specify migration option with MIGR DB2I2 command.
A SET SQLID = 'SYSADM2' is generated for every DDL and DCL with SQLID option
- Line 4: Make a copy of the migration output script
- Line 5: using ED command to change all the change specifications
- Line 6: invoke SUPERC to compare the changed migration script and the original migration script.
- Line 7: print the output from the SUPERC comparison
- Line 9: database name to be migrated
- Line 11-14: ISPF edit commands to be invoked with ED DB2I2 command

Case Study 2: DB2 Dataset Extents Removal

Adjust the primary allocation of all the table spaces in a database DB1 which has more than 10 extents with their base VSAM linear dataset.

The screen below demonstrates how to code the DB2I2 commands in batch mode to remove Db2 dataset extents.

```

000001 //DB2I2CMD DD *
000002 REXX idd=rexxddl
000003 LISTC EXT(10) odsn=t1 t=n tsix
000004 flist t1
000005 DSADJ idsn=t1 odsn=t2 %=100 alloc=(CYL,HIARBA,CYL) move=N maxsz=500
000006 flist t2
000007 //LINEOBJ DD *
000008 DS vcat.dsndbc.DB1.**
000009 //REXXDDL DD DATA,DLM=AA
000010 /* rexx */
000011 Address TSO "delete T1"
000012 Address TSO "delete T2"
000013 Return 0
000014 AA

```

- Line 2 Use DB2I2 REXX command to delete all the work files T1, T2 and set RC to 0. Use command IDD to specify REXXDDL is the DDNAME, which contains the source of your REXX exec. (line 13 – line 17)
- Line 3 Use DB2I2 LISTC command to retrieve dataset information from ICF catalog.
Specify EXT(10) command option to select only for those datasets which has more than 10 extents.
T=N command option is used to suppress printing of report heading and footing.
Specify TSIX command option to produce LISTC output in TP or IP line format.
Saved in output file T2 to be reused in the next step.
- Line 4 Print the output file T1
- Line 5 Use DB2I2 DSADJ command to adjust file space allocation.
Use %=100 command option to adjust the file allocation to the current HI-Allocated-RBA.
Use MOVE=N command option to skip IDCAM DEFINE and DSN1COPY job steps.
Use MAXSZ=500 to specify maximum primary allocation is 500 cylinders.
Input file is from T1 and DSADJ output is saved in file T2
- Line 6 Print output file T2

Case Study 3: Work Load Balancing

When you set up backup or recovery jobs, you need to consider the following:

1. How long it takes to run the image copy backup?
This is usually decided by the size of the objects to be backed up for each image copy jobs.
2. How many initiators, IDBACK value and Tape drives available for the backup jobs?
3. What is the window available to do the image copy backup?

Using the following sample scripts to control the workload of backing up or recovering DB2 objects for a database.

Assuming the following environment:

1. The size of all table spaces for the database to be backed up is 10000 cylinders
2. The image copy output to tape and there are 5 tape drives available for the image copy jobs.
3. Each image copy jobs should back up roughly 2000 cylinders so that they all can finish around the same time period.
4. Specify 2000 for the *resize-parm* to generate 5 image copy jobs each job will back up roughly 2000 cylinders.

```
000001 //DB2I2CMD DD *
000002 LISTC TSIX T=N ODSN='work.dsname'
000003 ED 'work.dsname' MACRO(IDD=RESIZE)
000004 COPY 'parmutil(copy)' IDSN='work.dsname' ODSN='output.dsname'
000005 //RESIZE DD *
000006 RESIZEIT 47,54,resize-parm
000007 //LINEOBJ DD *
000008 TS DB1.%
```

Line 2 Use DB2I2 LISTC command to display DB2 data set space allocation information. The line object are from DDNAME LINEOBJ on line 8.
Specify T=N command option to suppress report heading and footing.
Specify TSIX option to return result in TP or IP Line object format so that you can feed the result to the next DB2I2 command.
The results are saved in output file 'work.dsname'.

Line 3 Use DB2I2 ED command to edit the output from LISTC which invoke an edit macro from DDNAME. Use RESIZEIT edit macro to generate <NEWJOB> token for the line objects generated from LISTC.

Line 4 Use COPY command to generate image copy jobs. The output from COPY command should contain 5 image copy jobs each should back up roughly 2000 cylinders.

You can use the same output data set 'output.dsname' to generate database recovery jobs.

Case Study 4: DB2 Table Migration

Generate a migration job for a table with steps to

- Unload the table,
- Reload the table,
- Repairing table space with NOCOPYPEND option after data is reloaded,
- Executing RUNSTATS utility after REPAIR, and
- REBIND all the packages after RUNSTATS.

The example below is shipped with the product and stored in db2i2.system.library(MIGRTB). . You can invoke this script by BATCH ICMD=*MIGRTB.

The screen below display the source from *MIGRTB which demonstrates how you can code DB2I2 commands in batch mode to accomplish all the above migration specifications.

```

-- -----
-- MIGRTB:  Generate MIGRATION JCL to assist you to migrate TB line--
--   File Usage:
--   O1 - Output file contains generated JCL.....
--   W0 - work file contains generated TB line object
--   W1 - work file contains EDIT script for SYSPUNCH
--   W2 - work file contains Migration scripts.....
--   W3 - work file contains TS line objects.....
--   W4 - work file contains PG line objects.....
--   W5 - work file contains REBIND line objects....
--   Usage Note:
--   Modify (4-7)   for appropriate Migration Scripts changes
--   Modify (11),(13),and (14) to appropriate PARMUTIL Dataset
--   created from PARMUTIL for LOAD, REPAIR NOCOPYPEND,
--   RUNSTATS
-- -----
-- *****
-- Call SETUP to delete all work files (1)
-- *****
REXX      IDD=SETUP
-- *****
-- Modify the following host variable and run query in sqlgen to (2)
--   generate TB line objects for migration process
--   &CREATOR is table creator and
--   &TBNAME is table name
-- *****
RUN      IDD=SQLGEN ODSN=W0 LIMIT(9999) T=N      +
&CREATOR='CREATOR1'
&TBNAME='TBNAME1','TBNAME2'
-- *****
-- Generates Migration script with MIGR (3)
-- *****
MIGR     IDSN=W0 ODSN=W2 GRANT=Y
-- *****
-- Edit Migration scripts (4-7)
-- *****
EDIT     W2
POSTMIGR
C 'CREATOR1' 'CREATORO' ALL
C 'DBNAME1' 'DBNAMEO' ALL
END_EDIT
-- *****
-- Generate DSNTIAUL JCL (8)
-- *****
DSNTIAUL IDSN=W0 ODSN=O1
-- *****
-- Generate EDIT SYSPUNCH JCL appends to O1 start from STEP011 (9-10)
-- *****
REXX     IDD=EDITGEN
BATCH   ICMD=W1      ODSN=O1  APPEND STEP#=10 JOBCARD=N

```

```

-- *****
-- Generate EXEC migration JCL append to 01 start from STEP016      (11)
-- *****
BATCH      EXEC              IDSN=W2   ODSN=01  APPEND STEP#=15  JOBCARD=N

-- *****
-- Generate LOAD TABLE      JCL appends to 01 start from STEP021  (12)
-- *****
LOAD       UTIL(LOAD)       IDSN=W0   ODSN=01  APPEND STEP#=20  JOBCARD=N
-- *****
-- Get TS line objects                                             (13)
-- *****
TS         IDSN=W0          ODSN=W3
-- *****
-- Generate REPAIR NOCOPYPEND append to 01 start from STEP031     (14)
-- *****
REPAIR     UTIL(REPAIRNC)   IDSN=W3   ODSN=01  APPEND STEP#=30  JOBCARD=N
-- *****
-- Generate RUNSTATS        JCL appends to 01 start from STEP041  (15)
-- *****
RUNSTATS   UTIL(RUNSTATS)  IDSN=W3   ODSN=01  APPEND STEP#=40  JOBCARD=N
-- *****
-- Generate PG lines                                             (16)
-- Generate REBIND from PG lines                                  (17)
-- Generate REBIND          JCL appends to 01 start from STEP051  (18)
-- *****
PG         IDSN=W0          ODSN=W4          NOTFOUND(SKIP 2)
REBIND     IDSN=W4          ODSN=W5
BATCH     EXEC              IDSN=W5   ODSN=01  APPEND STEP#=50  JOBCARD=N
//SETUP   DD DATA,DLM=AA
/*  ---REXX-----
      SETUP:  Delete work & output files
----- */

ADDRESS TSO
'DELETE W0'
'DELETE W1'
'DELETE W2'
'DELETE W3'
'DELETE W4'
'DELETE W5'
'DELETE 01'
RETURN 0
AA
//EDITGEN DD DATA,DLM=BB
/*  ---REXX-----
      EDITGEN:Setup work file W1 to hold the EDIT commands to change
              load control statements generated from DSNTIAUL command.
              The standard Dataset name for SYSPUNCH from DSNTIAUL is
              TSOID.tbcreator.tbname.SYSPUNCH
              - Only the first 8 position of tbname is used and
              - If there is a '_' in tbname, '_' will be replace by '#'
      Example: DSNTIAUL generates TSOID.SYSIBM.PLAN#TAB.SYSPUNCH to hold
      SYSPUNCH information for line objects TB SYSIBM.PLAN_TABLE.
----- */

ADDRESS TSO
"ALLOC FI(W1) DS(W1) NEW SPACE(1,1) TRACKS RECFM(F B) LRECL(80)",
"BLKSIZE(0) REUSE UNIT(SYSDA)"
ADDRESS TSO
"ALLOC FI(W0) DS(W0) SHR REUSE"
/* ----- */
/* Generate EDIT statements
/* ----- */
"EXECIO * DISKR W0(STEM INREC. FINIS"
"FREE FI(W0)"
J = 0
Do I = 1 to INREC.0
  Parse value INREC.I with j1 ws_creator '.' ws_name j2
  ws_name = Translate(ws_name,'#','_')
  If Length(ws_name) > 8 Then ws_name = Substr(ws_name,1,8)
  J = J + 1
  OUTREC.J="EDIT "ws_creator"."ws_name".SYSPUNCH"
  J = J + 1
  OUTREC.J="C 'LOG NO' 'LOG NO RESUME NO REPLACE' ALL"
  J = J + 1
  OUTREC.J="C 'SYSREC00' 'SYSREC' ALL"
  J = J + 1
  OUTREC.J="C 'CREATORI' 'CREATORO' ALL"
  J = J + 1

```

```
      OUTREC.J="END_EDIT"  
End  
"EXECIO "J" DISKW W1(STEM OUTREC. FINIS"  
"FREE FI(W1)"  
RETURN 0  
BB  
//SQLGEN DD DATA,DLM=CC  
SELECT 'TB ' ||STRIP(CREATOR)||'.' ||NAME, CARDF  
FROM SYSIBM.SYSTABLES  
WHERE CREATOR IN (&CREATOR) AND NAME IN (&TBNAME)  
ORDER BY 1  
CC
```

Case Study 5: DB2 Package Tuning

Produces all necessary output information to tune a DB2 Package.

The example below is shipped with DB2I2 as a system UDF and stored in db2i2.system.library(TUNEPG). You can invoke it by selecting PG line objects and issuing the following command:

```
BATCH ICMD=*TUNEPG &output=your.output &TSOID=tsoid &SUFF=
```

The following is the detail procedure:

- Drill down from PG line to get all SQL statements in the DB2 package.
- Use DB2I2 SELPATHV command to get all selection path information for all the tables involved.
- Use DB2I2 STATS command to get statistics information for all the table spaces, tables, and indexes involved.
- Use DB2I2 EXEC command to execute DDL to delete entries from your PLAN_TABLE which will be used to keep explain output from the following BIND COPY with EXPLAIN(YES).
- Use DB2I2 BIND COPY command to generate BIND COPY DB2 commands for the selected packages with EXPLAIN option turn on.
- Use FREE and EXEC commands to free the test packages after BIND COPY.
- Use EXPLAINP command to produce DB2 EXPLAIN output.

```
-----
-- TUNEPG:  Tuning Assist Using PG Line Objects as Input  --
-- Format:  --
--   BATCH ICMD=*TUNEPG &OUTPUT=?output? &TSOID=xxxxxxx &suff=  --
--   Where &OUTPUT is the output file  --
--   &TSOID is your TSOID, must contain BIND authority--
--   &suff is the work file suffix which allows you  --
--   to run multiple TUNEPG jobs concurrently  --
--  --
-- *****  --
--   File Usage:  --
--   &OUTPUT - Output file .....  --
--   tunePg.WS&suff - work file for TS and IX command .....  --
--   tunePg.W1&suff - work file for TB command .....  --
--   tunePg.W2&suff - work file for DELETE FROM PLAN_TABLE....  --
--   tunePg.W3&suff - work file for EXPLAINP.....  --
--   tunePg.W4&suff - work file for BIND COPY.....  --
--   tunePg.W5&suff - work file for BATCH .....  --
--   tunePg.W6&suff - work file for FREE PACKAGE.....  --
--  -----
-- *****
-- Clean up all workfiles (1)
-- *****
REXX   IDD=DELWS (2)
-- *****
-- Drill Down from PG line objects
-- *****
ODSN=&output
-- *****
-- Get TB line objects (3)
-- *****
TB   ODSN=tunePg.W1&suff NOTFOUND(SKIP 99)
-- *****
-- Remove Duplicate TB line objects (4-6)
-- *****
EDIT   tunePg.W1&suff
DISTINCT 1,30
END_EDIT
-- *****
-- Generates selection path statistics (7)
-- *****
SELPATHV  IDSN=tunePg.W1&suff ODSN=&output APPEND
-- *****
-- Get TS line objects (8)
-- *****
TS   IDSN=tunePg.W1&suff ODSN=tunePg.WS&suff
-- *****
```

```

-- Get STATS from TS line objects (9)
-- *****
STATS      IDSN=tunepg.WS&suff  ODSN=&output  APPEND
-- *****
-- Get IX line objects (10)
-- *****
IX         IDSN=tunepg.W1&suff  ODSN=tunepg.WS&suff  NOTFOUND(SKIP 7)
-- *****
-- Drill Down IX line objects (11)
-- *****
IDSN=tunepg.WS&suff  ODSN=&output  APPEND
-- *****
-- Call SQLDEL ddname to generate Delete SQL & EXPLAINP (12)
-- *****
REXX      IDD=SQLDEL
-- *****
-- EXEC Delete SQL to delete entries from PLAN_TABLE (13)
-- *****
EXEC      IDSN=tunepg.W2&suff  ERROR(CONTINUE)
-- *****
-- Generate BIND COPY from PG line objects (14)
-- *****
BIND COPY O=USER      CL=USER      ODSN=tunepg.W4&suff
-- *****
-- Edit and change BIND COPY options (15-18)
-- *****
EDIT      tunepg.W4&suff  ERROR(CONTINUE)
C 'EXPLAIN(NO)' 'EXPLAIN(YES)' ALL
C '(ADD)' '(REPLACE)' ALL
END_EDIT
-- *****
-- EXEC BIND COPY & FREE (19-20)
-- *****
EXEC      IDSN=tunepg.W4&suff
EXEC      IDSN=tunepg.W6&suff
-- *****
-- BATCH command and use EXPLAINP lines as input (21)
-- *****
BATCH     ICMD=tynepg.W3&suff  ODSN=tunepg.W5&suff
-- *****
-- SUBMIT JOB to run EXPLAINP (28)
-- *****
TSO SUBMIT tunepg.W5&suff
//DELWS DD DATA,DLM=AA
/* --REXX-----
DELWS: Delete work files
----- */
Address TSO
'delete tunepg.ws&suff'
'delete tunepg.w1&suff'
'delete tunepg.w2&suff'
'delete tunepg.w3&suff'
'delete tunepg.w4&suff'
'delete tunepg.w5&suff'
'delete tunepg.w6&suff'
'delete &output'
"Alloc fi(ol) ds(&output) new space(1,1) tracks recfm(f b)",
"lrecl(133) blksize(0) reuse unit(sysda)"
"Free fi(ol)"
Return 0
AA
//SQLDEL DD DATA,DLM=BB
/* --REXX-----
SQLDEL: Generates DELETE FROM PLAN_TABLE... in W2
Generates EXPLAINP commands ..... in W3
Generates FREE commands ..... in W6
Input LINEOBJ DD contains PG line objects
----- */
Address TSO
"Alloc fi(w2) ds(tunepg.w2&suff) new space(1,1) tracks recfm(f b)",
"lrecl(80) blksize(0) reuse unit(sysda)"
"Alloc fi(w3) ds(tunepg.w3&suff) new space(1,1) tracks recfm(f b)",
"lrecl(80) blksize(0) reuse unit(sysda)"
"Alloc fi(w6) ds(tunepg.w6&suff) new space(1,1) tracks recfm(f b)",
"lrecl(80) blksize(0) reuse unit(sysda)"
outrec.1 = "DELETE FROM PLAN_TABLE"
outrec.2 = " WHERE PROGNAM IN ("
"Execio * Diskr LINEOBJ(Stem inrec. Finis"

```

DB2I2 Reference Manual

```
j = 2
jj= 0
dml=' '
Do i = 1 to inrec.0
  Parse Value inrec.i with j1 '.' j2 '.' ws '.'
  j = j + 1
  outrec.j =dml ""strip(ws)""
  jj= jj+ 1
  outrec2.JJ ="EXPLAINP ODSN=&output APPEND PG="STRIP(WS)
  outrec6.JJ ="DB2CMD FREE PACKAGE(&TSOID."STRIP(WS)""
  dml=', '
End
j = j + 1
outrec.j=" );"
"Execio" j "Diskw w2(Stem outrec. Finis"
"Execio" jj "Diskw w3(Stem outrec2. Finis"
"Execio" jj "Diskw w6(Stem outrec6. Finis"
"Free fi(w2)"
"Free fi(w3)"
"Free fi(w6)"
Return 0
BB
```

Case Study 6: DB2 Table Tuning

Produces all necessary output information to tune a DB2 table.

The example below is shipped with DB2I2 as a system UDF and stored in db2i2.system.library(TUNETB). You can invoke it by selecting TB line objects and issuing the following command:

```
BATCH ICMD=*TUNETB &output=your.output &TSOID=tsoid &SUFF=
```

The following is the detail procedure:

- :
- Use DB2I2 SELPATHV command to get all selection path information for all the tables involved.
 - Use DB2I2 STATS command to get statistics information for all the table spaces, tables, and indexes involved.
 - Use DB2I2 PG command to generate PG lines for all packages involved.
 - Use DB2I2 drill down command to display all package SQL statements.
 - Use DB2I2 EXEC command to execute DDL to delete entries from your PLAN_TABLE which will be used to keep explain output from the following BIND COPY with EXPAIN(YES).
 - Use DB2I2 BIND COPY command to generate BIND COPY DB2 commands for the selected packages with EXPLAIN option turn on.
 - Use FREE and EXEC commands to free the test packages after BIND COPY.
 - Use EXPLAINP command to produce DB2 EXPLAIN output.

```
-----
-- TUNETB: Tuning Assist Using TB Line Objects as Input --
-- sample for ICMD host variable substitute --
-- Format: --
-- BATCH ICMD=*TUNETB &OUTPUT=?output? &TSOID=xxxxxxx &suff= --
-- Where &OUTPUT is the output file --
-- &TSOID is your TSOID, must contain BIND authority--
-- &suff is the work file suffix which allows you --
-- to run multiple REMOVEID jobs concurrently --
--
-- *****
-- File Usage:
-- &OUTPUT - Output file .....
-- tunetb.WS&suff - work file for TS and IX command .....
-- tunetb.W1&suff - work file for PG command .....
-- tunetb.W2&suff - work file for DELETE FROM PLAN_TABLE....
-- tunetb.W3&suff - work file for EXPLAINP.....
-- tunetb.W4&suff - work file for BIND COPY.....
-- tunetb.W5&suff - work file for BATCH .....
-- tunetb.W6&suff - work file for FREE PACKAGE.....
-----
REXX          IDD=DELWS
REXX          IDD=DELO
REXX          IDD=ALLOCO
SELPATHV      ODSN=&OUTPUT
TS            ODSN=tunetb.WS&suff
STATS         IDSN=tunetb.WS&suff ODSN=&OUTPUT APPEND
IX            ODSN=tunetb.WS&suff NOTFOUND(SKIP 1)
IDSN=WS&suff  ODSN=&OUTPUT APPEND
PG            ODSN=tunetb.W1&suff NOTFOUND(SKIP 15)
EDIT          tunetb.W1&suff
DISTINCT     1,30
END_EDIT
IDSN=W1&suff  ODSN=&OUTPUT APPEND
REXX          IDD=SQLDEL
EXEC          IDSN=tunetb.W2&suff
BIND COPY     O=USER          CL=USER          IDSN=tunetb.w3&suff      +
              ODSN=tunetb.W4&suff
EDIT          tunetb.W4&suff          ERROR(CONTINUE)
C 'EXPLAIN(NO)' 'EXPLAIN(YES)' ALL
C '(ADD)'      '(REPLACE)'      ALL
END_EDIT
EXEC          IDSN=tunetb.W4&suff
```



```

EXEC          IDSN=tunetb.W6&suff
BATCH         ICMD=tunetb.W3&suff  ODSN=tunetb.W5&suff
TSO SUBMIT    tunetb.W5&suff
REXX         IDD=DELWS
//DELWS DD DATA,DLM=AA
/* --REXX-----
DELWS:  Delete work files
----- */

Address TSO
'delete tunetb.ws&suff'
'delete tunetb.w1&suff'
'delete tunetb.w2&suff'
'delete tunetb.w3&suff'
'delete tunetb.w4&suff'
'delete tunetb.w5&suff'
'delete tunetb.w6&suff'
Return 0
AA
//DELO DD DATA,DLM=AB
/* --REXX-----
DELO:   Delete Output file
----- */

ADDRESS TSO
'delete &OUTPUT'
Return 0
AB
//ALLOCO DD DATA,DLM=BB
/* --REXX-----
ALLOCO: Allocate Output file
----- */

Address TSO
"Alloc fi(o1) ds(&OUTPUT)",
  "new space(1,1) tracks recfm(f b) lrecl(133)",
  "blksize(0) reuse unit(sysda)"
"Free fi(o1)"
Return 0
BB
//SQLDEL DD DATA,DLM=CC
/* --REXX-----
SQLDEL: Generates DELETE FROM PLAN_TABLE... in W2
        Generates EXPLAINP commands ..... in W3
        Generates FREE commands ..... in W6
        Input W1 contains PG line objects
----- */

Address TSO
"Alloc fi(w1) ds(tunetb.w1&suff) shr reuse"
"Alloc fi(w2) ds(tunetb.w2&suff)",
  "new space(1,1) tracks recfm(f b) lrecl(80)",
  "blksize(0) reuse unit(sysda)"
"Alloc fi(w3) ds(tunetb.w3&suff)",
  "new space(1,1) tracks recfm(f b) lrecl(80)",
  "blksize(0) reuse unit(sysda)"
"Alloc fi(w6) ds(tunetb.w3&suff)",
  "new space(1,1) tracks recfm(f b) lrecl(80)",
  "blksize(0) reuse unit(sysda)"
outrec.1="DELETE FROM PLAN_TABLE"
outrec.2=" WHERE PROGNAME IN ("
"Execio * Diskr w1(Stem inrec. Finis"
j = 2
jj = 0
dlm = ' '
Do i = 1 to inrec.0
  parse value inrec.i with j1 '.' j2 '.' ws '.'
  j = j + 1
  outrec.j =dlm ""strip(ws)""
  jj= jj+ 1
  outrec2.jj = "EXPLAINP ODSN=&OUTPUT APPEND PG="STRIP(WS)
  /* ----- */
  /* Modify the following &TSOID to your TSOID */
  /* ----- */
  outrec6.jj = "DB2CMD FREE PACKAGE(&TSOID."strip(ws)""
  dlm = ','
End
j = j + 1
outrec.j = " );"
"Execio" j "Diskw w2(Stem outrec. Finis"
"Execio" jj "Diskw w3(Stem outrec2. Finis"
"Execio" jj "Diskw w6(Stem outrec6. Finis"

```

DB212 Reference Manual

```
"Free fi(w1)"  
"Free fi(w2)"  
"Free fi(w3)"  
"Free fi(w6)"  
Return 0  
CC
```

Case Study 7: DB2 Object Comparison

Compare two different DB2 objects from the same or different DB2 sub-systems.

The example below is shipped with DB2I2 as a system UDF and stored in db2i2.system.library(OBJCOMP). You can invoke it by issuing the following command:

```
BATCH ICMD=*OBJCOMP &LOC1=location1 &TSOID=tsoid &SUFF=
```

The output information contains SUPERC output from two different DDL or MIGR runs.

```
-----
-- OBJCOMP: comparing 2 db2 object DB, TS, TB, IX from same location--
-- or different locations. It contains                               --
-- DDL (or MIGR) db2i2 command to generate ddl scripts           --
-- EDIT exit to remove comment from ddl scripts, and            --
-- SUPERC to compare 2 ddl scripts                               --
-- Format:                                                         --
--   BATCH ICMD=*OBJCOMP &loc1=loc1 &loc=loc2 &objt=objt &obj=obj --
--                   &suff=                                         --
--   Where &loc1 is the location name of the first object        --
--         &loc2 is the location name of the 2nd object         --
--         &suff is the suffix for work files                    --
--         &objt is object type DB, TS, TB or IX                --
--         &obj  is the name of the object                       --
--
-- *****
-- FILE USAGE:
--   objcomp.wk1&suff - contains DDL or MIGR output form LINEOBJ1 --
--   objcomp.wk2&suff - contains DDL or MIGR output form LINEOBJ2 --
--   objcomp.out&suff - contains SUPERC comparison output        --
-----
-- Issus REXX DELWS to cleanup work files                          (1) --
-----
REXX IDD=DELWS
-----
-- Either issue DDL or MIGR db2i2 command to generate ddl scripts --
-- save the output ddl scripts in work file                        (2-5) --
-----
CONNECT(&loc1)
DDL  IDD=LINEOBJ1  ODSN=objcomp.wk1&suff
MIGR IDD=LINEOBJ1  ODSN=objcomp.wk1&suff          +
      AL=Y VW=Y SY=Y BIND=Y GRANT=Y RI=B          +
-- the following for DB2 V6 only
      DT=Y LO=Y
CONNECT(&loc2)
DDL  IDD=LINEOBJ2  ODSN=objcomp.wk2&suff
MIGR IDD=LINEOBJ2  ODSN=objcomp.wk2&suff          +
      AL=Y VW=Y SY=Y BIND=Y GRANT=Y RI=B          +
-- the following for DB2 V6 only
      DT=Y LO=Y
CONNECT(RESET)
-----
-- Remove comment from both wk1 and wk2 files                    (6- ) --
-- >>> add other string to be excluded from comparison          <<<--
-----
EDIT objcomp.wk1&suff
X '--' 1 ALL
DEL X ALL
END_EDIT
EDIT objcomp.wk2&suff
X '--' 1 ALL
DEL X ALL
END_EDIT
-----
-- Issue SUPERC to comapre two work files and save the output    --
-- in objcomp.out&suff                                           --
-----
SUPERC objcomp.wk1&suff objcomp.wk2&suff ODSN=objcomp.OUT&suff
-----
-- Print objcomp.out&suff with FLIST command (lrecl option 131)  --
```

```
-----  
FLIST objcomp.out&stuff 131  
//DELWS DD DATA,DLM=AA  
/* --REXX-----  
   DELWS: Delete work files  
----- */  
Address TSO  
   'Delete objcomp.wk1&stuff'  
   'Delete objcomp.wk2&stuff'  
   'Delete objcomp.out&stuff'  
Return 0  
AA  
//LINEOBJ1 DD *  
&objt &obj  
//LINEOBJ2 DD *  
&objt &obj
```

Case Study 8: DB2 Bufferpool Capturing and Reporting

The example below demonstrates how to use TSO add-ons (CAPTURE, TWAIT and READBUFR) to collect DB2 bufferpool BPO detail statistics every hour start at 8:00 AM for 8 hours. After the statistics collection, report them use the READBUFR add-on TSO command.

The example script below is shipped with the DB2I2 as sample script and stored in db2i2.system.library(MONBUFR). , you invoke it by issuing BATCH ICMD=*MONBUFR. To use this sample script, you must create a DB2 table – yourid.BUFFERPOOL to hold the capture buffer pool statistics. The DDL for creating your own bufferpool table is in db2i2.system.library(DDLBUFR). You can use HELP *DDLBUFR to get the information about the definition of the db2 table.

```
-- Start execution at 8:00 AM
TSO TWAIT ACTL08000000
-- Display Bufferpool detail and save the result in T1
DISPLAY DETAIL(INTERVAL) ODSN=T1
-- Execute TSO CAPTBUFR to generate SQL INSERT and save the result in T2
TSO CAPTBUFR T1-T2-MY.BUFFERPOOL-5
-- Use EXEC command to execute the SQL insert in T2
EXEC IDSN=T2
-- Use TSO TWAIT to wait for one hour
TSO TWAIT PERD01000000
-- Display Bufferpool detail and save the result in T1
DISPLAY DETAIL(INTERVAL) ODSN=T1
-- Execute TSO CAPTBUFR to generate SQL insert and save the result in T2
TSO CAPTBUFR T1-T2-MY.BUFFERPOOL-5
-- Use EXEC command to execute the SQL insert in T2 and
-- SKIP back to TSO TWAIT PERD01000000 7 times
EXEC IDSN=T2 SKIP=(-8,7)
-- Execute TSO READBUFR to read the information stored in MY.BUFFERPOOL
-- and save the report in T3
TSO READBUFR DSN\ -MY.BUFFERPOOL-BP0-T3
-- Use FLIST to print the report saved in T3
FLIST T3 80
//LINEOBJ DD *
BP BP0
```

Case Study 9: Access Path Comparison

Compares access paths for programs from 2 different versions of the same PLAN_TABLE (ACCCOMP) or current version from different PLAN_TABLE or different location (ACCCOMPR).

The examples below are shipped with DB2I2 as system UDF and stored in db2i2.system.library(ACCCOMP) and db2i2.system.library(ACCCOMPR). , you can invoke it by issuing the following command:

```
BATCH ICMD=*ACCCOMP      &OWNER=planowner &PG=progrname      or
BATCH ICMD=*ACCCOMPR    &LOC1=location1 &OWNR1=planowner1    +
                        &LOC2=location2 &OWNR2=planowner2    +
                        &PG=progrname
```

```
-----
-- ACCCOMP:  Access comparison report                                --
--           Use EXPLAINP command to generate Exaplin Report to   --
--           Compare the current and previous generation of the   --
--           Explain output.                                       --
-- Format:                                           --
--   BATCH ICMD=*ACCCOMP &owner=xxxxxxxx &pg=xxxxxxxx &tsoid=xxxxx --
--           Where &owner is the owner name of plan_table        --
--           &pg is the program name                             --
-- Note:  Change the following:                               --
--       PG=pgname for appropriate DB2 package name             --
--       chnage PG=pgname to PL=plname for PLAN comparison.     --
-----
REXX IDD=REXX1
EXPLAINP O=&owner PG=&pg ODSN=acccomp.wk1 DET=N GN=0
EXPLAINP O=&owner PG=&pg ODSN=acccomp.wk2 DET=N GN=-1
SUPERCC acccomp.wk1 acccomp.wk2 ODSN=acccomp.output
FLIST acccomp.output 132
//REXXDEL DD DATA,DLM=AA
/* -- REXX -- */
X=MSG("OFF")
ADDRESS TSO
"DELETE acccomp.wk1"
"DELETE acccomp.wk2"
"DELETE acccomp.output"
RETURN 0
AA
```

```

-----
-- ACCCOMPR: Access comparison report
--           Use EXPLAINP command to generate Explain Report to
--           Compare the current generation of Explain output form
--           2 different locations.
--
-- Format:
--   BATCH ICMD=*ACCCOMPR &ownr1=xxxxxxx &loc1=xxxxxxx      +
--                       &ownr1=xxxxxxx &loc1=xxxxxxx      +
--                       &pg=xxxxxxx
--
--   Note: Please change the following:
--           &ownr1   owner name of the first plan_table
--           &loc1    location name of the first plan_table
--           &ownr2   owner name of the 2nd  plan_table
--           &loc2    location name of the 2nd  plan_table
--           &pg      for appropriate DB2 package name
--                   change PG=pgname to PL=plname for
--                   DB2 plan comparison instead of
--                   DB2 package comparison.
-----
REXX IDD=REXXDEL
CONNECT(&loc1)
EXPLAINP O=&ownr1 PG=&pg ODSN=acccomp.wk1 DET=N GN=0
CONNECT(&loc2)
EXPLAINP O=&ownr2 PG=&pg ODSN=acccomp.wk2 DET=N GN=0
CONNECT(RESET)
SUPERC acccomp.wk1 acccomp.wk2 ODSN=acccomp.output
FLIST acccomp.output 132
//REXXDEL DD DATA,DLM=AA
/* -- REXX -- */
X=MSG("OFF")
ADDRESS TSO
"DELETE acccomp.wk1"
"DELETE acccomp.wk2"
"DELETE acccomp.output"
RETURN 0
AA

```

Case Study 10: User ID or System Admin ID removal

Produce necessary scripts to safely remove a user ID or System Admin ID from your Db2 sub-system.

The examples below is shipped with DB2I2 as a system UDF and stored in db2i2.system.library(REMOVEID). , you can invoke it by issuing the following command:

```
BATCH ICMD=*REMOVEID &FROM=fromuser &TO=copytouser &SUFF=
```

```
-----
-- REMOVEID: DB2I2 udf to remove a user ID from system --
-- Format: --
-- BATCH ICMD=*REMOVEID &from=???????? &to=xxxxxxx &suff= --
-- Where &from is the user id to be removed --
-- &to is the user id to be removed --
-- &suff is the work file suffix which allows you --
-- to run multiple REMOVEID jobs concurrently --
-- --
-- ***** --
-- 1. Generate and rebind all packages/plans with owner ='????????' --
-- 2. Copyauth to generate ???????? as grantor information and --
-- Copy the authorization to xxxxxxxx --
-- 3. Revoke to generate revoke DCL from ???????? --
-- ***** --
-- Note: --
-- ***** --
-- Change ???????? to actual ID you want to remove from your system --
-- xxxxxxxx to Receiving ID --
-----
Rexx IDD=DELDD
Run Idd=selsql Odsn=removeid.wkl&suff &uid=&from T=N Notfound(SKIP 2)
Rebind Idsn=removeid.out&suff Odsn=removeid.wk2&suff DET=Y O=&to
Copyauth Odsn=removeid.out&suff Append SQLID=&To Grantor
Revoke Odsn=removeid.out&suff Append
Exec Idsn=t4&suff
//DELDD DD DATA,DLM=AA
/* REXX --delete all work files----- */
Address TSO
'Delete removeid.wkl&suff'
'Delete removeid.out&suff'
Return 0
AA
//SELSQL DD DATA,DLM=BB
Select 'PG ' ||strip(location)||'.' ||strip(collid)||'.' ||strip(name)
||'.' ||strip(version)
From sysibm.syspackage
Where owner = '&uid'
union all
Select 'PL ' ||name
From sysibm.sysplan
Where creator = '&uid'
BB
```


Case Study 11: Copy DB2 Catalog Statistics

Produce necessary scripts to copy DB2 catalog statistics from one location to another location.

The example below is shipped with the product as sample script and stored in db2i2.system.library(COPYSTAT)., you invoke it with BATCH ICMD=*COPYSTAT command.

```

-----
-- COPYSTAT: DB2I2 udf to copy catalog statistics from 1 table to --
-- another table from different remote location --
-- Note: A TB line or a group of TB lines can be used as input for --
-- this COPYSTAT process. --
-----
-- Detail Description: --
-----
-- 1. Connect to source location and issue SELPATHV to get access --
-- path statistics into T1 --
-- 2. Connect(RESET) and use ED exit to change the source info --
-- to target information --
-- Change CU to CI for SYSIBM.SYSCOLDIST information --
-- 3. Use QBUILD and RUN to generate DELETE SQL --
-- to delete old SYSCOLDIST catalog statistics --
-- 4. Use EXEC to execute the generated DELETE SQL and --
-- UPDATE and INSERT SQL generated from SELPATHU --
-- ***** --
-- Note: --
-- ***** --
-- Change location-name to the source location name --
-- databasei to source database name --
-- databaseo to target database name --
-- tablespacei to source tablespace name --
-- tablespaceo to target tablespace name --
-- tablecreatori to source tablecreator name --
-- tablecreatoro to target tablecreator name --
-- tablei to source table name --
-- tablei to target table name --
-- indexcreatori to source indexcreator name --
-- indexcreatoro to target indexcreator name --
-- indexi to source index name --
-- indexi to target index name --
-----
Rexx IDD=DELDD
Connect(location-name)
Selpathv OPTION=CICU odsn=t1
Connect(RESET)
Ed t1 MACRO(IDD=EDITDD)
Selpathu idsn=t1 odsn=t2
Qbuild F1=CREATOR F2=NAME odsn=t3
Run IDD=DELSQLDD &WHERE=t3 odsn=t4 T=N LIMIT(999)
EXEC idsn=t4
EXEC idsn=t2
//DELDD DD DATA,DLM=AA
/* REXX --delete all work files----- */
x=MSG('OFF')
Address TSO 'DELETE T1'
Address TSO 'DELETE T2'
Address TSO 'DELETE T3'
Address TSO 'DELETE T4'
Return 0
AA
//EDITDD DD DATA,DLM=BB
C 'databasei' 'databaseo' all
C 'tablespacei' 'tablespaceo' all
C 'tablecreatori' 'tablecreatoro' all
C 'tablei' 'tableo' all
C 'indexcreatori' 'indexcreatoro' all
C 'indexi' 'indexo' all
C 'CU SYSCOLDIST' 'CI SYSCOLDIST' all
BB
//DELSQLDD DD DATA,DLM=CC
SELECT 'DELETE FROM SYSIBM.SYSCOLDIST ||
' WHERE TOWNER = ''||CREATOR||

```

```
      ' ' AND TBNAM = ' ' || NAME | ' '  
FROM SYSIBM.SYSTABLES  
WHERE  
-INC &WHERE  
CC
```

Case Study 12: Callable Interface - MCCLI

DB2I2 support callable interface. You can call DB2I2 directly from your own REXX, CLIST or any program language support ISPF interface.

The following sample REXX routines are delivered with DB2I2 to demonstrate how to code you own routine to invoke DB2I2.

The example below is shipped with the product and stored in db2i2.system.library(MCCLI). , you can use DB2I2 BATCH with CLI=MCCLI option to invoke it.

By default, DB2I2 process each DB2I2 command for all selected line objects. The provided MCCLI routine allows you to **process all DB2I2 commands against all line objects from LINEOBJ one at a time.**

```

/* -REXX-MCCLI-----*/
/*          DB2I2  DB2 CATALOG INTERFACE TOOL BOX          */
/*          BY JRH GOLDENSTATE SOFTWARE, INC.             */
/*          (C) COPYRIGHTED 1997-2011                    */
/*          DB2I2 CALLABLE INTERFACE SAMPLE ROUTINE      */
/*-----*/
/*  1. THE PURPOSE OF THIS MCCLI ROUTINE IS BUILT TO     */
/*  DEMONSTRATE HOW TO CALL DB2I2 FROM YOUR REXX OR CLIST */
/*  2. MCCLI READ LINE OBJECTS IN AND CALL DB2I2 AND PROCESS */
/*  THEM ONE AT A TIME AGAINST A SET OF DB2I2 COMMANDS    */
/*-----*/
/* TRACE ?R */
ADDRESS ISPEXEC "CONTROL ERRORS RETURN"
WS_TIME = TIME()
SAVERC = 0
SUFF = '.T' || SUBSTR(WS_TIME,1,2) || SUBSTR(WS_TIME,4,2) ||
        SUBSTR(WS_TIME,7,2)
WSDSN = 'DB2I2.MCCLI'SUFF
"EXECIO * DISKR LINEOBJ(STEM RECIN. FINIS"
ADDRESS TSO "ALLOC FI(LINEOBJ) DS(WSDSN)",
"SPACE(1) TRACKS " ||
"RECFM(F B) LRECL(80) BLKSIZE(80) UNIT(SYSDA) REUSE"
ADDRESS TSO "FREE FI(LINEOBJ)"
DO I = 1 TO RECIN.0
  RECOLO.1 = RECIN.I
  ADDRESS TSO "ALLOC FI(LINEOBJ) SHR DS(WSDSN) REUSE"
  "EXECIO 1 DISKW LINEOBJ(STEM RECOLO. FINIS"
  CALL DB2I2 when you call DB2I2, LINEOBJ contains only one line object at a time
             DB2I2CMD contains whatever the commands defined from BATCH JCL
  ADDRESS ISPEXEC "VGET (ZISPFRC) SHARED"
  IF ZISPFRC > 4 THEN RETURN ZISPFRC
  ELSE IF ZISPFRC \= 0 THEN
    SAVERC = ZISPFRC
END
ADDRESS TSO "FREE FI(LINEOBJ)"
ADDRESS TSO "DELETE" WSDSN
RETURN SAVERC

```

Case Study 13: Callable Interface - UTILCLI

The example below is shipped with the product and stored in db2i2.system.library(UTILCLI). , you can use BATCH with CLI=UTILCLI option to invoke it.

By default, DB2I2 process each command for all selected line objects. The provided UTILCLI routine reads all line objects from LINEOBJ and process them **one at a time** by calling DB2I2.

```

/* -REXX-UTILCLI-----*/
/*          DB2I2  DB2 CATALOG INTERFACE TOOL BOX          */
/*          BY JRH GOLDENSTATE SOFTWARE, INC.             */
/*          (C) COPYRIGHTED 1997-2011                    */
/*          DB2I2 CALLABLE INTERFACE SAMPLE ROUTINE       */
/*          -----*/
/*  1. The purpose of this UTILCLI routine is to generate any DB2 */
/*  Utility and submit it dynamically                        */
/*  2. UTIL parm contains UTIL ID for this process. This can be */
/*  any valid DB2I2 utility command                        */
/*  3. PARMIN parm contains the DB2I2 utility parm prepared with */
/*  PARMUTIL command                                      */
/*  4. DSPRE parm contains dataset prefix for utility work files */
/*  5. OUTDSN parm contains generated JCL output dataset name */
/*  a PDS without member name specified                   */
/*  6. JOB# contains the start job number in the # replacement on */
/*  the generated jobcard                                  */
/*  7. SUBSW parm contains the option to decide if you want to */
/*  SUBMIT the generated recover job dynamically          */
/* -----*/
/* TRACE ?R */
Parse Upper arg UTIL PARMIN DSPRE OUTDSN JOB# SUBSW JUNK
x=msg('off')
If subsw = ' ' | DATATYPE(JOB#) \= 'NUM' Then
Do
  Say " ** ----- **"
  Say " ** Missing or invalid Processing Parameters"
  Say " ** The format for UTILCLI DB2I2 Callable Interface Routine are:"
  Say " ** ISPSTART +"
  Say " ** CMD(UTILCLI +"
  Say " ** UTIL + (valid DB2I2 Utility command)"
  Say " ** PARMIN + (dataset prepared from PARMUTIL command)"
  Say " ** DSPRE + (Utility work file dataset Prefix)"
  Say " ** OUTDSN + (Output PDS contains the generated utility JCL)"
  Say " ** JOB# + (Numeric starting job number)"
  Say " ** SUBMIT=N + (SUBMIT=Y to submit generated JCL)"
  Say " ** )"
  Say " ** ----- **"
  zispfrc=16
  Address Ispexec "VPUT (ZISPFRC) SHARED"
  Return zispfrc
End
Address Ispexec "CONTROL ERRORS RETURN"
JOB# = JOB# - 1
/* -----*/
/* Prepare temporary dataset names to be used in UTILCLI routine */
/* -----*/
ws_time = TIME()
saverc = 0
suff = '.T' || SUBSTR(WS_TIME,1,2) || SUBSTR(WS_TIME,4,2) || ,
SUBSTR(WS_TIME,7,2)
wsdsn1 = 'DB2I2.UTILCLI.LINEOBJ'SUFF /* Temp LINEOBJ dataset */
wsdsn2 = 'DB2I2.UTILCLI.DB2I2CMD'SUFF /* Temp DB2I2CMD dataset */
wsdsn3 = 'DB2I2.UTILCLI.JOBCARD'SUFF /* Temp JOBCARD dataset */
wsdsn4 = 'DB2I2.UTILCLI'SUFF'.01' /* Temp output dataset */
/* -----*/
/* Allocate and set DB2I2CMD to Call DB2I2 to RUN SQL from LINEOBJ */
/* -----*/
Address Tso "ALLOC FI(DB2I2CMD) DS(wsdsn2)",
"SPACE(1) TRACKS " || ,
"RECFM(F B) LRECL(80) BLKSIZE(8000) UNIT(SYSDA) REUSE"
C.1="RUN LIMIT(99999) ODSN="WSDSN01" T=N"
"EXECIO 1 DISKW DB2I2CMD(STEM C. FINIS"
CALL DB2I2

```

```

/* -----*/
/* Allocate LINEOBJ dataset */
/* -----*/
Address Tso "ALLOC FI(LINEOBJ) DS("wsdsn1)",
"SPACE(1) TRACKS " || ,
"RECFM(F B) LRECL(80) BLKSIZE(80) UNIT(SYSDA) REUSE"
/* -----*/
/* READ in JOBCARD, Allocate and prepare temporary JOBCARD dataset */
/* -----*/
EXECIO * DISKR JOBCARD(STEM Jobin. FINIS"
Address Tso "ALLOC FI(JOBCARD) DS("wsdsnj)",
"SPACE(1) TRACKS " || ,
"RECFM(F B) LRECL(80) BLKSIZE(800) UNIT(SYSDA) REUSE"
jobin_save = jobin.1
Call Jobcard
/* -----*/
/* Read in and process one record at a time from RUN output */
/* -----*/
Address Tso "ALLOC FI(LINEOBJ) SHR DS("wsdsnol") REUSE"
EXECIO * DISKR LINEOBJ(STEM Recin. FINIS"
Do I = 1 TO Recin.0
/* -----*/
/* Prepare LINEOBJ for each RUN output reocrd */
/* -----*/
L.1 = Recin.I
Parse value Recin.I with j1 '.' TS ' ' j2
Address Tso "ALLOC FI(LINEOBJ) SHR DS("wsdsn1") REUSE"
EXECIO 1 DISKW LINEOBJ(STEM L. FINIS"
/* -----*/
/* Prepare DB2I2CMD for the UTIL and OUTDSN to specific member */
/* depends on the 2nd node after . */
/* -----*/
Address Tso "ALLOC FI(DB2I2CMD) SHR DS("wsdsnc") REUSE"
If substr(outdsn,1,1) = "" Then
odswns=substr(outdsn,1,length(outdsn)-1)||"("TS")"
Else
odswns=outdsn||"("TS")"
C.1=UTIL parmin" DSPRE="dspre" +"
C.2=" ODSN="||odswns
EXECIO 2 DISKW DB2I2CMD(STEM C. FINIS"
/* -----*/
/* Call DB2I2 to process prepared LINEOBJ and DB2I2CMD */
/* -----*/
CALL DB2I2
/* -----*/
/* Get return code in ZISPFRC. If it is > 4 then return with abend */
/* other continue and save it in the saverc field. */
/* -----*/
Address Ispexec "VGET (ZISPFRC) SHARED"
If datatype(zispfrc) \= 'NUM' Then zispfrc = 0
If zispfrc > 4 Then RETURN zispfrc
Else If zispfrc \= 0 THEN
saverc = zispfrc
/* -----*/
/* If SUBMIT=Y then submit the generated utility JCL */
/* -----*/
If subsw = 'SUBMIT=Y' Then
Do
Say " ** >>> Submitting "odswns
Address TSO "SUBMIT" odswns
End
/* -----*/
/* If # is specified, set the next job number for next JOBCARD */
/* -----*/
If Jobi \= 0 Then
CALL jobcard2
End
/* -----*/
/* Delete all the work files and Return */
/* -----*/
Address Tso "FREE FI(LINEOBJ)"
Address Tso "DELETE" wsdsnc
Address Tso "DELETE" wsdsn1
Address Tso "DELETE" wsdsnj
Address Tso "DELETE" wsdsnol
Return saverc
/* -----*/
/* Check # and set JOBCARD accordingly */

```

```

/* ----- */
Jobcard:
Parse Var jobin.1 jobname junk
jobi = index(jobname,'#')
If Jobi \= 0 Then
  Call jobcard2
Else
  Call jobcard3
Return
/* ----- */
/* Set job# for the next sequence number */
/* ----- */
Jobcard2:
Job# = Job# + 1
jobws=TRANSLATE(FORMAT(SUBSTR(job#,1,10),10,0),'0',' ')
jobin.1 = substr(jobname,1,jobi-1) ||,
          SUBSTR(jobws,jobi,10-jobi+1) junk
Call Jobcard3
Return
/* ----- */
/* Set JOBCARD */
/* ----- */
Jobcard3:
Address Tso "ALLOC FI(JOBCARD) SHR DS("wsdsnj") REUSE"
"EXECIO "jobin.0" DISKW JOBCARD(STEM Jobin. FINIS"
Return

```

The following Generated JCL demonstrates how to generate multiple recover tablespace/index jobs for database DJRHJ

```

//JOB CARD
//
// *
// * -----+
//JOB LIB DD DISP=SHR,DSN=SYS1.TEST.SDSNLOAD
//DB2I2P JCLLIB ORDER=JRHJ.DB2I2.LIB
// * ----- **
// * THE FORMAT FOR UTILCLI DB2I2 CALLABLE INTERFACE ROUTINE ARE:
// * ISPSTART +
// * CMD(UTILCLI +
// * UTILID + (ANY VALID DB2I2 UTILITY COMMAND)
// * PARMIN + (DATASET PREPARED FROM PARMUTIL COMMAND)
// * DSPRE + (UTILITY WORK FILE DATASET PREFIX)
// * OUTDSN + (OUTPUT PDS CONTAINS GENERATED RECOVER JCL)
// * JOB# + (NUMERIC STARTING JOB NUMBER)
// * SUBMIT=N + (SUBMIT=Y/N)
// * )
// * ----- **
//STEP001 EXEC $DB2I2P,REGION=0M,COND=(4,LT)
//DB2I2P.SYSTSIN DD *
ISPSTART +
CMD(UTILCLI +
RECOVER +
'JRHH.UTIL(RCVRLGON)' +
JRHH +
'JRHH.UTILCLI.DJRHH' +
201 +
SUBMIT=N +
)
//SSID DD *
DSN\ 5 SYSIBM
//DB2I2CMD DD *
//LINEOBJ DD *
SELECT 'TS ' ||STRIP(DBNAME) || '.' ||NAME
FROM SYSIBM.SYSTABLESPACE
WHERE DBNAME = 'DJRHJ'
//JOB CARD DD DATA,DLM=JJ
//JRHHX### JOB 1111111, JRHH, CLASS=F,
// MSGCLASS=R,MSGLEVEL=(1,1),NOTIFY=JRHH
// *
// *
JJ

```

Case Study 14: Previous Point of Time Recovery

The following example shows a series of command to process a point of time recovery with various DB2I2 commands. It demonstrates how to use

- Issue TSSET to get tablespace set information
- Issue RBA command with YYYY-MM-DD to list all the RBA points available on or after the date YYYY-MM-DD.
- Issue SETRBA command to set INCORE RBA
- Issue REPORT with RECOVERY TO option = 'R' to narrow down the tablespaces which have been opened for update after the selected INCORE RBA point.
- Issue ED command to retrieve the result from batch REPORT run.
- Issue RECOVER to recover to the INCORE RBA point only for those TP line object return from REPORT run.

Issue the DB2I2 TSSET command to list all Referential Integrity related tablespaces.

```
Command ===> TSSET Scroll ===> CSR
002025 TS DBPRIT01.ZICS PRIT.ZIP_CODE
```

The following screen displays the results from the previous DB2I2 TSSET command. It shows that there are 7 additional tablespaces, which have referential integrity relationship with the selected TS line object.

```
Command ===> Scroll ===> CSR
002025 TS DBPRIT01.ZICS PRIT.ZIP_CODE
002026 TS DBPRIT01.AEZS -- TS SET 1 <NEWJOB>
002027 TS DBPRIT01.ARZS -- TS SET 1
002028 TS DBPRIT01.CIZS -- TS SET 1
002029 TS DBPRIT01.LOZS -- TS SET 1
002030 TS DBPRIT01.ZICS -- TS SET 1
002031 TS DBPRIT01.AEGS -- TS SET 1
002032 TS DBPRIT01.ATES -- TS SET 1
002033 TS DBPRIT01.CITS -- TS SET 1
```

Issue DB2I2 RBA command with a desired date in the format of YYYY-MM-DD to list all RBA lines with ICDATE on or after the specified date from SYSIBM.SYSCOPY table.

```
Command ===> RBA 1999-06-01 Scroll ===> CSR
002025 TS DBPRIT01.ZICS PRIT.ZIP_CODE
002026 TS DBPRIT01.AEZS -- TS SET 1 <NEWJOB>
002027 TS DBPRIT01.ARZS -- TS SET 1
002028 TS DBPRIT01.CIZS -- TS SET 1
002029 TS DBPRIT01.LOZS -- TS SET 1
002030 TS DBPRIT01.ZICS -- TS SET 1
002031 TS DBPRIT01.AEGS -- TS SET 1
002032 TS DBPRIT01.ATES -- TS SET 1
002033 TS DBPRIT01.CITS -- TS SET 1
```

The following screen display all the available RBA points from the previous DB2I2 RBA command.

```
Command ===> Scroll ===> CSR
=NOTE= RP:RecoverySite Primary RB:RecoverySite Backup
=NOTE= DS#=DSNUM
=NOTE= 0:TableSpace Level Other:Partition Level
===== -Start RBA-- TSIC -Date & time- DS# DSNAME-----
000001 010FCDE85EFB S 990601 225227 0 DBPRIT01.LOZS
000002 010FCAEF4EC9 S 990601 185403 0 DBPRIT01.ZICS
000003 010FCAC8AEA0 S 990601 183703 0 DBPRIT01.CITS
000004 010FCA916AC2 S 990601 181922 0 DBPRIT01.ARZS
000005 010FCA80662E S 990601 181527 0 DBPRIT01.ATES
000006 010FCA5C4C69 S 990601 181117 0 DBPRIT01.AEZS
000007 010FCA4AD9B4 S 990601 180048 0 DBPRIT01.AEGS
000008 010FC9B67BB4 S 990601 170550 0 DBPRIT01.CIZS
***** Bottom of Data *****
```


DB2I2 Reference Manual

The following screen displays the result JCL generated from the previous REPORT command.

```
Command ==> Scroll ==> CSR
000015 //STEP001 EXEC DSNUPROC,REGION=4M,COND=(4,LT),
000016 //          SYSTEM='DB2D',UID='AEZS.REPORT',UTPROC=' '
000017 //* ----- **
000018 //SYSPRINT DD DSN=&&TEMP,DISP=(NEW,PASS),UNIT=SYSDA,
000019 //          SPACE=(TRK,(5,1),RLSE)
000020 //SYSIN DD *
000021 REPORT RECOVERY
000022          TABLESPACE DBPRIT01.AEZS
000023          DSNUM ALL
000024          CURRENT
000025          LOCALSITE
000026 REPORT RECOVERY
000027          TABLESPACE DBPRIT01.ARZS
000028          DSNUM ALL
000029          CURRENT
000030          LOCALSITE
000031 REPORT RECOVERY
000032          TABLESPACE DBPRIT01.CIZS
000033          DSNUM ALL
000034          CURRENT
000035          LOCALSITE
000036 REPORT RECOVERY
000037          TABLESPACE DBPRIT01.LOZS
000038          DSNUM ALL
000039          CURRENT
000040          LOCALSITE
000041 REPORT RECOVERY
000042          TABLESPACE DBPRIT01.ZICS
000043          DSNUM ALL
000044          CURRENT
000045          LOCALSITE
000046 REPORT RECOVERY
000047          TABLESPACE DBPRIT01.AEGS
000048          DSNUM ALL
000049          CURRENT
000050          LOCALSITE
000051 REPORT RECOVERY
000052          TABLESPACE DBPRIT01.ATES
000053          DSNUM ALL
000054          CURRENT
000055          LOCALSITE
000056 REPORT RECOVERY
000057          TABLESPACE DBPRIT01.CITS
000058          DSNUM ALL
000059          CURRENT
000060          LOCALSITE
000061 //* ----- **
000062 //STEP001C EXEC $DB2I2P,REGION=0M,COND=(4,LT)
000063 //DB2I2P.SYSTSIN DD *
000064 P000710 DB2I2.REPORT.RECOVERY RBA=010FCAEF4EC9
000065 //RPTRDD DD DSN=&&TEMP,DISP=(OLD,DELETE)
***** Bottom of Data *****
```

Submit the generated REPORT JCL to run it in batch. Once the REPORT batch job is finished, you should see the following message displayed.

```
** The Report Recovery Analyzer Process is done
** Please issue the following DB2I2 command
**   ED DB2I2.REPORT.RECOVERY
**   to access the Analysis Result
17.36.23 JOB03119 $HASP165 DB2ADM01 ENDED AT ACSC MAXCC=0 CN(INTERNAL)
***
```

Issue the DB2I2 ED command to edit the result from REPORT batch run.

```
Command ==> ED DB2I2.REPORT.RECOVERY
000055          LOCALSITE
000056 REPORT RECOVERY
000057          TABLESPACE DBPRIT01.CITS
000058          DSNUM ALL
```

The following screen displays the result from the previous DB2I2 ED command.

```

Command ==>
***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG> your edit profile using the command RECOVERY ON.
000001 -- DB.TS qualifies for RBA=010FCAEF4EC9 option
000002 -- DBname.TSname Part Date Time Start RBA End RBA Star
000003 TS DBPRIT01.AEVS 0000 110800 15160578 014BEDF7A426 014BEDF82C0A 014B
000004 TS DBPRIT01.CIZS 0000 100200 11334042 01488AA8302A 01488AA868F3 0148
000005 TS DBPRIT01.LOZS 0000 112700 10362277 014CBB819AEB 014CBB81C000 014C
000006 TS DBPRIT01.ZICS 0000 092500 10390078 01482DA76000 01482DAB1CBE 0148
000007 TS DBPRIT01.AEGS 0000 032400 09155202 01287F48436C 0128805A6354 0128
000008 TS DBPRIT01.CITS 0000 092500 10390040 01482DA73532 01482DAB1CBE 0148
***** Bottom of Data *****
    
```

Issue DB2I2 RECOVER command against the result from REPORT batch run to recover only those tablespace which open for update after the selected INCORE RBA point.

```

Command ==> RECOVER
***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG> your edit profile using the command RECOVERY ON.
000001 -- DB.TS qualifies for RBA=010FCAEF4EC9 option
000002 -- DBname.TSname Part Date Time Start RBA End RBA Star
SS0003 TS DBPRIT01.AEVS 0000 110800 15160578 014BEDF7A426 014BEDF82C0A 014B
000004 TS DBPRIT01.CIZS 0000 100200 11334042 01488AA8302A 01488AA868F3 0148
000005 TS DBPRIT01.LOZS 0000 112700 10362277 014CBB819AEB 014CBB81C000 014C
000006 TS DBPRIT01.ZICS 0000 092500 10390078 01482DA76000 01482DAB1CBE 0148
000007 TS DBPRIT01.AEGS 0000 032400 09155202 01287F48436C 0128805A6354 0128
SS0008 TS DBPRIT01.CITS 0000 092500 10390040 01482DA73532 01482DAB1CBE 0148
***** Bottom of Data *****
    
```

Select TORBA = 'I' INCORE recovery to generate DB2 RECOVER utility JCL with TORBA option.

```

#RECOVA ----- DB2I2 RECOVER TABLESPACE/INDEX PROCESS OPTIONS -----
DSNUM          ALL_ (ALL/0-4096)          ---- Performance Enhancement ----
CURRENT        Y (Y/N)                   Rebuild IX After TS Rcvry Y (Y/N)
TORBA          I (Y/N/I-Incore RBA)      SORTDEVT SYSALLDA SORTNUM _ (0-9)
TOLOGPOINT    N (Y/N/I-Incore RBA)
LOGONLY       N (Y/N)                   Alloc DD for Last Copy DS Y (Y/N)
TOCOPY        N (Y/N/L-LastCopy/F-LastFullCopy)
COPYDSN PREFIX _____

PAGE/CONTINUE _____ RETAIN for Tape Dataset Y (Y/N)
ERROR RANGE   N (Y/N)
SITE OPTION   A (A-All 1-LOCALSITE 2-RECOVERYSITE)
REUSE         N (Y/N) PARALLEL _ (0-9)
LOGRANGES    _ (Y/N) CURRENTCOPYONLY _ (Y/N)
TAPEUNITS    _ (0-9)
FROMDUMP      _ (Y/N) DUMPCLASS _____
RESTOREBEFORE N (Y-Restore Before RBA/N)
SYSCOPY within 007 Days (1-999) (All SYSCOPY records for last ? days)
One Job per step N (Y/N)
One Recovery Job
for all objects _ (Y/N)
VERIFYSET (Y/N) _ ENFORCE (Y/N) _ BACKOUT (Y/N) _
PF3=Exit ENTER=Process Your Selection
    
```

The following screen displays partial result from the previous RECOVER command.

```

000014 // * ----- **
000015 //STEP001 EXEC DSNUPROC,REGION=4M,COND=( 4,LT) ,
000016 //          SYSTEM='DB2D' ,UID='AEZS.RCVR1' ,UTPROC=' '
000017 // * ----- **
000018 //SYSPRINT DD SYSOUT=*
000019 //SYSUT1  DD DSN=DB2ADM.DBPRIT01.AEZS.RCVR1.SYSUT1 ,
000020 //          DISP=(NEW,DELETE,CATLG) ,
000021 //          UNIT=SYSDA ,
000022 //          SPACE=(CYL,(00003,00001))
000023 //UTPRINT  DD SYSOUT=*
000024 //SORTPARM DD DSN=DB2ADM.PROD.CONTROL(DYNALLOC) ,DISP=SHR
000025 //SYSIN    DD *
000026 RECOVER
000027     TABLESPACE DBPRIT01.AEZS                DSNUM ALL
000028     TORBA X'010FCAEF4EC9'
000029     LOCALSITE
000030     REBUILD INDEX(ALL) TABLESPACE DBPRIT01.AEZS
000031     SORTDEVT SYSDA SORTNUM 3
000032 // * ----- **
000033 //STEP002 EXEC DSNUPROC,REGION=4M,COND=( 4,LT) ,
000034 //          SYSTEM='DB2D' ,UID='AEZS.RCVR1' ,UTPROC=' '
000035 // * ----- **
000036 //SYSPRINT DD SYSOUT=*
000037 //SYSUT1  DD DSN=DB2ADM.DBPRIT01.CIZS.RCVR1.SYSUT1 ,
000038 //          DISP=(NEW,DELETE,CATLG) ,
000039 //          UNIT=SYSDA ,
000040 //          SPACE=(CYL,(00001,00001))
000041 //UTPRINT  DD SYSOUT=*
000042 //SORTPARM DD DSN=DB2ADM.PROD.CONTROL(DYNALLOC) ,DISP=SHR
000043 //SYSIN    DD *
000044 RECOVER
000045     TABLESPACE DBPRIT01.CIZS                DSNUM ALL
000046     TORBA X'010FCAEF4EC9'
000047     LOCALSITE
000048     REBUILD INDEX(ALL) TABLESPACE DBPRIT01.CIZS
000049     SORTDEVT SYSDA SORTNUM 3
000050 // * ----- **
000051 //STEP003 EXEC DSNUPROC,REGION=4M,COND=( 4,LT) ,
000052 //          SYSTEM='DB2D' ,UID='AEZS.RCVR1' ,UTPROC=' '
000053 // * ----- **
000054 //SYSPRINT DD SYSOUT=*
000055 //SYSUT1  DD DSN=DB2ADM.DBPRIT01.LOZS.RCVR1.SYSUT1 ,
000056 //          DISP=(NEW,DELETE,CATLG) ,
000057 //          UNIT=SYSDA ,
000058 //          SPACE=(CYL,(00001,00001))
000059 //UTPRINT  DD SYSOUT=*
000060 //SORTPARM DD DSN=DB2ADM.PROD.CONTROL(DYNALLOC) ,DISP=SHR
000061 //SYSIN    DD *
000062 RECOVER
000063     TABLESPACE DBPRIT01.LOZS                DSNUM ALL
000064     TORBA X'010FCAEF4EC9'
000065     LOCALSITE
000066     REBUILD INDEX(ALL) TABLESPACE DBPRIT01.LOZS
000067     SORTDEVT SYSDA SORTNUM 3
000068 // * ----- **
000069 //STEP004 EXEC DSNUPROC,REGION=4M,COND=( 4,LT) ,
000070 //          SYSTEM='DB2D' ,UID='AEZS.RCVR1' ,UTPROC=' '
000071 // * ----- **
000072 //SYSPRINT DD SYSOUT=*
000073 //SYSUT1  DD DSN=DB2ADM.DBPRIT01.ZICS.RCVR1.SYSUT1 ,
000074 //          DISP=(NEW,DELETE,CATLG) ,
000075 //          UNIT=SYSDA ,
000076 //          SPACE=(CYL,(00001,00001))
000077 //UTPRINT  DD SYSOUT=*
000078 //SORTPARM DD DSN=DB2ADM.PROD.CONTROL(DYNALLOC) ,DISP=SHR
000079 //SYSIN    DD *
000080 RECOVER
000081     TABLESPACE DBPRIT01.ZICS                DSNUM ALL
000082     TORBA X'010FCAEF4EC9'
000083     LOCALSITE
000084     REBUILD INDEX(ALL) TABLESPACE DBPRIT01.ZICS
000085     SORTDEVT SYSDA SORTNUM 3
000086 // * ----- **
000087 //STEP005 EXEC DSNUPROC,REGION=4M,COND=( 4,LT) ,
000088 //          SYSTEM='DB2D' ,UID='AEZS.RCVR1' ,UTPROC=' '
000089 // * ----- **

```

DB212 Reference Manual

```
000090 //SYSPRINT DD SYSOUT=*
000091 //SYSUT1 DD DSN=DB2ADM.DBPRIT01.AEGS.RCVR1.SYSUT1,
000092 // DISP=(NEW,DELETE,CATLG),
000093 // UNIT=SYSDA,
000094 // SPACE=(CYL,(00001,00001))
000095 //UTPRINT DD SYSOUT=*
000096 //SORTPARM DD DSN=DB2ADM.PROD.CONTROL(DYNALLOC),DISP=SHR
000097 //SYSIN DD *
000098 RECOVER
000099 TABLESPACE DBPRIT01.AEGS DSNUM ALL
000100 TORBA X'010FCAEF4EC9'
000101 LOCALSITE
000102 REBUILD INDEX(ALL) TABLESPACE DBPRIT01.AEGS
000103 SORTDEVT SYSDA SORTNUM 3
000104 //* ----- **
000105 //STEP006 EXEC DSNUPROC,REGION=4M,COND=(4,LT),
000106 // SYSTEM='DB2D',UID='AEZS.RCVR1',UTPROC=' '
000107 //* ----- **
000108 //SYSPRINT DD SYSOUT=*
000109 //SYSUT1 DD DSN=DB2ADM.DBPRIT01.CITS.RCVR1.SYSUT1,
000110 // DISP=(NEW,DELETE,CATLG),
000111 // UNIT=SYSDA,
000112 // SPACE=(CYL,(00001,00001))
000114 //SORTPARM DD DSN=DB2ADM.PROD.CONTROL(DYNALLOC),DISP=SHR
000115 //SYSIN DD *
000116 RECOVER
000117 TABLESPACE DBPRIT01.CITS DSNUM ALL
000118 TORBA X'010FCAEF4EC9'
000119 LOCALSITE
000120 REBUILD INDEX(ALL) TABLESPACE DBPRIT01.CITS
000121 SORTDEVT SYSDA SORTNUM 3
***** Bottom of Data *****
```

Submit the generated JCL and you have completed your point of time recovery.

The same procedure can be prepared in batch mode with the routine described below:

A system delivered ED macros POTRCVR can be used to generate recovery to last quiesce point jobs. To use it, simply select a **DB line** and then issue **BATCH ICMD=*POTRCVR**. Modify the red field if desirable.

The routine is best fit if you have your save your image copy to DASD and recovery can be done with one recovery per job.

```

-----
-- SAMPLE ED MACRO (POINT OF TIME RECOVERY TO LAST QUIESCE POINT) --
-----
-- ** BEFORE YOU CAN USE THIS MACRO, YOU NEED TO --
-- USE PARMUTIL COMMAND TO PREPARE PARMUTIL FOR --
-- REPORT RECOVERY TO INCORE RBA --
-- RECOVER TO INCORE RBA --
-- . SPECIFY ONE JOB PER STEP TO ALLOW PARALLEL --
-- RECOVERY JOB PROCESS --
-- . SPECIFY REBUILD INDEX AFTER RECOVER TS OPTION --
-----
-- THE LINE OBJECT FOR THIS MACRO IS A DB LINE OBJECT --
-----
-- . LIST ALL THE RBA POINT --
-- . FIND THE FIRST QUIESCE POINT --
-- . SET GLOBAL VARIABLE &RBA WITH MACRO1 --
-- . SETRBA TO &RBA --
-- . REPORT RECOVERY TO INCORE RBA &RBA --
-- . THE OUTPUT OF REPORT RECOVERY IS EDITED WITH MACRO2 --
-- . CHANGE DEFAULT REPORT RECOVERY OUTPUT (IF NEEDED) --
-- . FIND RBA= LINE AND GET INCORE RBA INFO INTO ED_INCORE_RBA --
-- . INSERT RECOVER TO INCORE RBA JOB STEP FROM RCVRDD DDNAME --
-- . WITH {RBA} SUBSTITUTE --
-- . SUBMIT JOB TO PROCESS REPORT RECOVERY AND RECOVERY JOBS GEN --
-----
TS ODSN=TS.LINE
RBA IDSN=TS.LINE ODSN=RBA.LINE
ED RBA.LINE MACRO(IDD=MACRO1)
SETRBA &RBA
REPORT IDSN=TS.LINE ODSN=DB2I2.UTIL.JCL(REPORT) PARMUTIL(RPTRBA) +
      JOBNM=REPORT
ED DB2I2.UTIL.JCL(REPORT) MACRO(IDD=MACRO2)
TSO SUBMIT DB2I2.UTIL.JCL(REPORT)
//MACRO1 DD DATA,DLM=AA
F 'Q' 17 FIRST find first quiesce line
SETG &RBA ' ' set global variable &RBA with first RBA value
AA
//MACRO2 DD DATA,DLM=BB
C 'DB2I2.REPORT.RECOVERY' 'MY.REPORT.RECOVERY' change default output dataset name
F 'RBA=' FIRST find RBA= line
CLINE get RBA= line info into CLINE
REXX PARSE VALUE CLINE WITH ED_V1 'RBA=' ED_INCORE_RBA ' ' ED_V2 set ED_INCORE_RBA
BOTTOM go to the bottom of JCL
INSERTA * IDD=RCVRDD insert a recovery job step
BB
//RCVRDD DD DATA,DLM=CC
//STEPRCVR EXEC $DB2I2P
//SSID DD *
-->>!!SSID INFORMATION!!<<-- specify SSID information
//DB2I2CMD DD DATA,DLM=DD
SETRBA {ED_INCORE_RBA} set incore RBA with SETRBA command
RECOVER PARMUTIL(RCVRIRBA) + generate recovery jobs with wild card
      IDSN=MY.REPORT.RECOVERY ODSN=DB2I2.UTIL.JCL(*) + output and jobname RCVR001, RCVR002..
      JOBNM=RCVR###
DD
//JOB CARD DD DATA,DLM=EE
//JOBNM## JOB -->>!!RECOVERY JOB CARD INFORMATION!!<<-- specify JOB CARD information
EE
CC

```

A system delivered ED macros POTRCVRD can be used to generate recovery to last quiesce point jobs with DASD image copy. To use it, simply select a **DB line** and then issue **BATCH ICMD=*POTRCVRD**. Modify the red field if desirable.

The routine is best fit if you have your save your image copy to DASD and recovery can be done with one recovery for all objects and then build indexes concurrently.

```

-----
-- SAMPLE ED MACRO (POINT OF TIME RECOVERY TO LAST QUIESCE POINT) --
-----
-- ** BEFORE YOU CAN USE THIS MACRO, YOU NEED TO --
-- USE PARMUTIL COMMAND TO PREPARE PARMUTIL FOR --
-- REPORT RECOVERY TO INCORE RBA (RPTRBA) --
-- RECOVER TO INCORE RBA FOR DASD (RCVIRBAD) --
-- . SPECIFY ONE JOB PER STEP = 'N' --
-- . SPECIFY ONE JOB FOR ALL OBJECTS = 'Y' --
-- . SPECIFY REBUILD INDEX AFTER RECOVER TS = 'N' --
-- . ALLOC DD FOR LAST FULL IMAGE COPY = 'N' --
-- . RETAIN MULTIPLE DATASET TAPE AFTER RCVR= 'N' --
-- REBUILD INDEX ORE RBA FOR DASD (RBLDIDX) --
-- . ONE JOB PER STEP = Y --
-- . SORTKEYS = Y --
-- . REUSE = Y --
-- . STATISTICS = Y --
-----
-- THE LINE OBJECT FOR THIS MACRO IS A DB LINE OBJECT --
-----
-- . LIST ALL THE RBA POINT --
-- . FIND THE FIRST QUIESCE POINT --
-- . SET GLOBAL VARIABLE &RBA WITH MACRO1 --
-- . SETRBA TO &RBA --
-- . REPORT RECOVERY TO INCORE RBA &RBA --
-- . THE OUTPUT OF REPORT RECOVERY IS EDITED WITH MACRO2 --
-- . CHANGE DEFAULT REPORT RECOVERY OUTPUT (IF NEEDED) --
-- . FIND RBA= LINE AND GET INCORE RBA INFO INTO ED_INCORE_RBA --
-- . INSERT RECOVER TO INCORE RBA JOB STEP FROM RCVRDD DDNAME --
-- . WITH {RBA} SUBSTITUTE --
-- . SUBMIT JOB TO PROCESS REPORT RECOVERY AND RECOVERY JOBS GEN --
-----
TS ODSN=TS.LINE
RBA IDSN=TS.LINE ODSN=RBA.LINE
ED RBA.LINE MACRO(IDD=MACRO1)
SETRBA &RBA
REPORT IDSN=TS.LINE ODSN=DB2I2.UTIL.JCL(REPORT) PARMUTIL(RPTRBA) +
      JOBNM=REPORT
ED DB2I2.UTIL.JCL(REPORT) MACRO(IDD=MACRO2)
TSO SUBMIT DB2I2.UTIL.JCL(REPORT)
//MACRO1 DD DATA,DLM=AA
F 'Q' 17 FIRST find first Quiesce line
SETG &RBA ' ' set global variable &RBA with first RBA value
AA
//MACRO2 DD DATA,DLM=BB
C 'DB2I2.REPORT.RECOVERY' 'MY.REPORT.RECOVERY' change default output dataset name
F 'RBA=' FIRST find RBA= line
CLINE get RBA= line info into CLINE
REXX PARSE VALUE CLINE WITH ED_V1 'RBA=' ED_INCORE_RBA ' ' ED_V2 set ED_INCORE_RBA
BOTTOM go to the bottom of JCL
INSERTA * IDD=RCVRDD insert a recovery job step
BB
//RCVRDD DD DATA,DLM=CC
//STEPRCVR EXEC $DB2I2P
//SSID DD *
-->>!!SSID INFORMATION!!<<-- specify SSID information
//DB2I2CMD DD DATA,DLM=DD
SETRBA {ED_INCORE_RBA} set incore RBA with SETRBA command
RECOVER PARMUTIL(RCVIRBAD) + generate recovery jobs with wild card
      IDSN=MY.REPORT.RECOVERY ODSN=RCVIRBAD.JCL + output and jobname RCVR001, RCVR002..
      JOBNM=RCVIRBAD
ED RCVIRBAD.JCL MACRO(IDD=MACRO3)
REBUILD PARMUTIL(RBLDIDX) +
      IDSN=MY.REPORT.RECOVERY +
      ODSN=RCVIRBAD.JCL APPEND JOBNM=RBLD### NEWJOB=1

```


For PARMUTIL(RBLDIDX): multiple rebuild index jobs after table space recovery

```
#RBLDA ----- DB2I2 REBUILD INDEX PROCESS OPTIONS -----
INDEX          (ALL)      (ALL) for Rebuild TS INDEX All option
PART           _____ otherwise leave with space
SORTDEVT      SYSALLDA
SORTNUM       _____ (2-255)
One Job per step N        (Y/N)
SORTKEYS      N          (Y/N)
REUSE         N          (Y/N)
STATISTICS    N          (Y/N)
SCOPE        ALL_____ (ALL/PENDING)
CLONE        _____ (Y/N)
SHRLEVEL     1          (1-REFERENCE 2-CHANGE)
DRAIN Specification
  DRAIN-WAIT  _____ RETRY _____ RETRY-DELAY _____
FLASHCOPY    N          (N/Y/C-Consistent)
FCCOPYDDN    FYSCOPY_
For SHRLEVEL CHANGE
  MAXRO      _____ (MAX RO Log Apply)
  LONGLOG    1          (1-CONTINUE 2-TERM 3-DRAIN)
  DELAY      1200_____ (In Seconds)

PF3=Exit  ENTER=Process Your Selection
```

A modified version of POTRCVR name POTRCVRT can be used specifically to tape image copy.

The routine requires you to

- o Create a DB2 table **COPY_SEQ_TABLE**: to store image copy sequence information. Use **FGET *DDLCOPYS** to copy the information into your edit session, modify it and then use EXEC to create the copy sequence table.

```
SET CURRENT SQLID = '????????';
CREATE TABLESPACE TS??????
  IN DB??????
  USING STOGROUP SG??????
  PRIQTY 12
  SECQTY 12
  ERASE NO
  FREEPAGE 0
  PCTFREE 5
  COMPRESS NO
  GBPCACHE CHANGED
  BUFFERPOOL BP0
  LOCKSIZE PAGE
  LOCKMAX SYSTEM
  CLOSE NO
  CCSID EBCDIC
  MAXROWS 1
; COMMIT;
CREATE TABLE COPY_SEQ_TABLE
(
  DBNAME          CHAR(8)          NOT NULL WITH DEFAULT
  ,TSNAME         CHAR(8)          NOT NULL WITH DEFAULT
  ,PARTNO        CHAR(3)          NOT NULL WITH DEFAULT
  ,JOBNAME       CHAR(8)          NOT NULL WITH DEFAULT
  ,SEQNO         SMALLINT         NOT NULL WITH DEFAULT
)
  IN DB?????? .TS??????

AUDIT NONE
DATA CAPTURE NONE
CCSID EBCDIC
; COMMIT;
CREATE TYPE 2 UNIQUE
  INDEX XCOPIES1
ON COPY_SEQ_TABLE
( JOBNAME ASC
  , SEQNO ASC
)
  USING STOGROUP SG??????
```


DB2/2 Reference Manual

```

        PRIQTY      12
        SECQTY      12
        ERASE       NO
        FREEPAGE    0
        PCTFREE     10
        GBPCACHE    CHANGED
BUFFERPOOL  BP0
CLOSE      NO
; COMMIT;
CREATE TYPE 2 UNIQUE
        INDEX XCOPYS2
ON        COPY_SEQ_TABLE
( DBNAME  ASC
, TSNAME  ASC
, PARTNO  ASC
)
        USING STOGROUP      SG??????
        PRIQTY      12
        SECQTY      12
        ERASE       NO
        FREEPAGE    0
        PCTFREE     10
        GBPCACHE    CHANGED
BUFFERPOOL  BP0
CLOSE      NO
; COMMIT;
```

- Translates TP line objects to a series of DELETE and INSERT SQL to store the relation of image job name information and table space partition information with a system defined ED macro **TPTOSQL**. For each <NEWJOB> token on TP line object, job no is incremented by 1.
 Use **FGET *TPTOSQL** to make a copy of this macro. Modify the field in red below to meet your requirement.
 Use **FPUT MACRO(TPTOSQL)** to create a new copy in your MACRO library.
 Use **FGET *DELSKEL** to make a copy of this macro. Modify the field in red below to meet your requirement.
 Use **FPUT MACRO(DELSKEL)** to create a new copy in your MACRO library.
 Use **FGET *INSSKEL** to make a copy of this macro. Modify the field in red below to meet your requirement.
 Use **FPUT MACRO(INSSKEL)** to create a new copy in your MACRO library.
 Use **ED xxxxxx MACRO(MACRO(TPTOSQL))** to process the DELETE/INSERT SQL generation.
 Where xxxxxx should be an empty existing dataset will hold the generated DELETE and INSERT SQL

With the example below:

Assuming TP.LINE contains the following TP line objects:

```
TP DB00001.TS00001 0
TP DB00001.TS00002 0
TP DB00001.TS00003 0    <NEWJOB>
TP DB00002.TS00001 0
```

After you issue **ED xxxxxx MACRO(MACRO(TPTOSQL))**, xxxxxx should contains the following:

```
DELETE FROM XXXX.COPY_SEQ_TABLE
WHERE JOBNAME = 'MMMM0001'
;
INSERT TO XXXX.COPY_SEQ_TABLE
(DBNAME, TSNAME, PARTNO, JOBNAME, SEQNO) VALUE (
'DB00001', 'TS00001', '0', 'MMMM0001', 1)
;
INSERT TO XXXX.COPY_SEQ_TABLE
(DBNAME, TSNAME, PARTNO, JOBNAME, SEQNO) VALUE (
'DB00001', 'TS00002', '0', 'MMMM0001', 2)
;
DELETE FROM XXXX.COPY_SEQ_TABLE
WHERE JOBNAME = 'MMMM0002'
;
INSERT TO XXXX.COPY_SEQ_TABLE
(DBNAME, TSNAME, PARTNO, JOBNAME, SEQNO) VALUE (
'DB00001', 'TS00003', '0', 'MMMM0002', 1)
;
INSERT TO XXXX.COPY_SEQ_TABLE
(DBNAME, TSNAME, PARTNO, JOBNAME, SEQNO) VALUE (
'DB00002', 'TS00001', '0', 'MMMM0002', 2)
;
```

```

-----
-- SAMPLE ED MACRO DESCRIPTION: (TPTOSQL)
-- READ A DATASET WHICH CONTAIN TP LINE OBJECTS
-- PRODUCT DELETE SQL FOR EACH NEW JOB
-- PRODUCT INSERT SQL FOR EACH TP LINE OBJECT
-- FOR EACH TP LINE OBJECT:
-- IF THERE IS A <NEWJOB> TOKEN THEN
-- INCREMENT ED_JOBNO BY 1
-- PREPARE ED_JOBNM WITH ED_JOBNM_PRE AND ED_JOBNO
-- SET ED_SEQNO TO 0
-- INSERT A DELETE SQL SKELTON TO THE END
-- INCREMENT ED_SEQNO BY 1
-- INSERT A INSERT SQL SKELTON TO THE END
-----
-- INFORMATION FROM DATAIN
-- TP DBNAME.TSNAME PARTNO
-- TP DBNAME.TSNAME PARTNO <NEWJOB>
-----
-- INFORMATION IN MACRO(DELKEL):
-- DELETE FROM XXXX.COPY_SEQ_TABLE
-- WHERE JOBNAME = '{ED_JOBNM}'
-- ;
-----
-- INFORMATION IN MACRO(INSSKEL):
-- INSERT TO XXXX.COPY_SEQ_TABLE
-- (DBNAME, TSNAME, PARTNO, JOBNAME, SEQNO) VALUE (
-- '{ED_DB}', '{ED_TS}', '{ED_PARTNO}', '{ED_JOBNM}', '{ED_SEQNO}')
-- ;
-----
REXX ED_JOBNM_PRE = 'MMMM' /* JOB NAME PREFIX !!CHANGE THIS */
REXX ED_JOBNO = 0001 /* START JOB NUMBER !!CHANGE THIS */
REXX ED_JOBNO_LEN = 4 /* LENGTH OF ED_JOBNO !!CHANGE THIS */
REXX ED_SEQNO = 0 /* JOB OBJECT SEQ NO */
REXX ED_JOBNM = 'MMMM0001' /* FIRST JOB NAME !!CHANGE THIS */
-----
-- !!CHANGE TP.LINE TO A DSNNAME WHICH CONTAINS TP LINE OBJECTS
-----
REXX ADDRESS TSO "ALLOC FI(DATAIN) DS(TP.LINE) SHR REUSE"
REXX "EXECIO 1 DISKR DATAIN"
REXX IF LAST_RC \= 0 THEN SIGNAL FINISH
REXX PULL EDVAR
-----
-- !!CHANGE 99 TO A NUMBER LARGE ENOUGH TO PROCESS ALL OF YOU DATAIN
-----
LOOP 99
REXX PARSE VAR EDVAR EDV1 EDV2 ED_PARTNO EDV3
REXX PARSE VALUE EDV2 WITH ED_DB '.' ED_TS
REXX IF INDEX(EDV3, '<NEWJOB>') \= 0 THEN +
REXX DO; +
REXX ED_JOBNO = ED_JOBNO + 1; +
REXX ED_WS = SUBSTR(ED_JOBNO, 1, ED_JOBNO_LEN); +
REXX ED_JOBNM = ED_JOBNM_PRE || +
REXX TRANSLATE(FORMAT(ED_WS, ED_JOBNO_LEN, 0), '0', ' '); +
REXX ED_SEQNO = 0; +
REXX END;
REXX ED_SEQNO = ED_SEQNO + 1
REXX IF INDEX(EDV3, '<NEWJOB>') = 0 & LP \= 1 THEN SKIP 2
BOTTOM
INSERTA * IDSN=MACRO(DELKEL)
BOTTOM
INSERTA * IDSN=MACRO(INSSKEL)
REXX "EXECIO 1 DISKR DATAIN"
REXX IF LAST_RC \= 0 THEN SIGNAL FINISH
REXX PULL EDVAR
LOOP_END
REXX ADDRESS TSO "FREE FI(DATAIN)"

```

- o Use system delivered ED macros POTRCVRT to generate recovery to last quiesce point jobs with tape image copy and copy sequence consideration. To use it, simply select a [DB line](#) and then issue [BATCH ICMD=*POTRCVRT](#). Modify the red field if desirable.

```

-----
-- SAMPLE ED MACRO (POINT OF TIME RECOVERY TO LAST QUIESCE POINT) --
-- *** MODIFY VERSION TO INTERFACE TO COPY_SEQ_TABLE WHICH ALLOW --
-- GROUP ALL TS NEEDED TO BE RECOVERED IN THEIR BACKUP SEQUENCE--
-----
-- ** BEFORE YOU CAN USE THIS MACRO, YOU NEED TO --
-- USE PARMUTIL COMMAND TO PREPARE PARMUTIL FOR --
-- REPORT RECOVERY TO INCORE RBA --
-- RECOVER TO INCORE RBA --
-- . SPECIFY ONE JOB PER RECOVERY = 'N' --
-- . SPECIFY ONE JOB FOR ALL OBJECTS = 'N' --
-- . SPECIFY RETAIN MULTIPLE DATASET TAPE AFTER RECVR = 'Y' --
-- . SPECIFY REBUILD INDEX AFTER RECOVER TS OPTION --
-----
-- THE LINE OBJECT FOR THIS MACRO IS A DB LINE OBJECT --
-----
-- . LIST ALL THE RBA POINT --
-- . FIND THE FIRST QUIESCE POINT --
-- . SET GLOBAL VARIABLE &RBA WITH MACRO1 --
-- . SETRBA TO &RBA --
-- . REPORT RECOVERY TO INCORE RBA &RBA --
-- . THE OUTPUT OF REPORT RECOVERY IS EDITED WITH MACRO2 --
-- . CHANGE DEFAULT REPORT RECOVERY OUTPUT (IF NEEDED) --
-- . FIND RBA= LINE AND GET INCORE RBA INFO INTO ED_INCORE_RBA --
-- . INSERT RECOVER TO INCORE RBA JOB STEP FROM RCVRDD DDNAME --
-- . WITH {RBA} SUBSTITUTE --
-- . SUBMIT JOB TO PROCESS REPORT RECOVERY AND RECOVERY JOBS GEN --
-----
TS ODSN=TS.LINE
RBA IDSN=TS.LINE ODSN=RBA.LINE
ED RBA.LINE MACRO(IDD=MACRO1)
SETRBA &RBA
REPORT IDSN=TS.LINE ODSN=DB2I2.UTIL.JCL(REPORT) PARMUTIL(RPTRBA) +
JOBNM=REPORT
ED DB2I2.UTIL.JCL(REPORT) MACRO(IDD=MACRO2)
TSO SUBMIT DB2I2.UTIL.JCL(REPORT)
//MACRO1 DD DATA,DLM=AA
F 'Q' 17 FIRST
SETG &RBA ' '
AA
//MACRO2 DD DATA,DLM=BB
C 'DB2I2.REPORT.RECOVERY' 'MY.REPORT.RECOVERY'
F 'RBA=' FIRST
CLINE
REXX PARSE VALUE CLINE WITH EDV1 'RBA=' ED_INCORE_RBA ' ' EDV2
BOTTOM
INSERTA * IDD=RCVRDD
BB

```

DB2I2 Reference Manual

```
//RCVRDD DD DATA,DLM=CC
//STEPRCVR EXEC $DB2I2P
//SSID DD *
-->>!!SSID INFORMATION!!<<--
//DB2I2CMD DD DATA,DLM=DD
QBUILD  IDSN=MY.REPORT.RECOVERY.ODSN=MY.REPORT.RECOVERY.WHERE + use QBUILD to generate WHERE
        F1=DBNAME F2=TSNAME F3=PARTNO                          predicates and use RUN to
RUN      IDD=QUERYDD ODSN=MY.REPORT.RECOVERY.OUTPUT T=N LIMIT(9999) to re-sequence to backup seq
ED       MY.REPORT.RECOVERY.OUTPUT MACRO(IDD=MACRO3)           use MACRO3 to add <NEWJOB>
SETRBA  {ED_INCORE_RBA}
RECOVER PARMUTIL(RCVIRBAT) +
        IDSN=MY.REPORT.RECOVERY.OUTPUT ODSN=DB2I2.UTIL.JCL(*) +
        JOBNM=RCVR###
DD
//MACRO3 DD DATA,DLM=FF          MACRO3 process each RUN output line for each new job add <NEWJOB>
REXX ED_V4_SAVE = ''
CLINE NEXT
LOOP EOF
  REXX PARSE VAR CLINE ED_V1 ED_V2 ED_V3 ED_V4 ED_V5
  REXX IF ED_V4 \= ED_V4_SAVE THEN +
  REXX CLINE = STRIP(CLINE,T) <NEWJOB>
  REXX ED_V4_SAVE = ED_V4
  CLINE WRITE
  CLINE NEXT
LOOP_END
FF
//QUERYDD DD DATA,DLM=QQ        QUERYDD to re-sequence to original image copy sequence
SELECT 'TP '||STRIP(DBNAME)||'.'||TSNAME, PARTNO, JOBNAME, SEQNO
      , ' '
FROM XXXX.COPY_SEQ_TABLE
WHERE
-INC MY.REPORT.RECOVERY.WHERE
ORDER BY JOBNAME, SEQNO
QQ
//JOB CARD DD DATA,DLM=EE
//JOBNM## JOB --->>!!RECOVERY JOB CARD INFORMATION!!<<---
EE
CC
```

* The green lines are the difference between tape vs DASD backup/recovery.

For PARMUTIL(RCVIRBAT): recover with tape backup sequence consideration

```
#RECOVA ----- DB2I2 RECOVER TABLESPACE/INDEX PROCESS OPTIONS -----
DSNUM          ALL_ (ALL/0-4096)          ---- Performance Enhancement ----
CURRENT        Y (Y/N)                    Rebuild IX After TS Rcvry Y (Y/N)
TORBA         I (Y/N/I-Incore RBA)        SORTDEVT SYSALLDA SORTNUM _ (0-9)
TOLOGPOINT    N (Y/N/I-Incore RBA)
LOGONLY       N (Y/N)                    Alloc DD for Last Copy DS Y (Y/N)
TOCOPY        N (Y/N/L-LastCopy/F-LastFullCopy)
COPYDSN PREFIX _____

PAGE/CONTINUE _____                RETAIN for Tape Dataset Y (Y/N)
ERROR RANGE   N (Y/N)
SITE OPTION   A (A-All 1-LOCALSITE 2-RECOVERYSITE)
REUSE         N (Y/N) PARALLEL _ (0-9)
LOGRANGES    _ (Y/N) CURRENTCOPYONLY _ (Y/N)
TAPEUNITS    _ (0-9)
FROMDUMP     _ (Y/N) DUMPCLASS _____
RESTOREBEFORE N (Y-Restore Before RBA/N)
SYSCOPY within 007 Days (1-999) (All SYSCOPY records for last ? days)
One Job per step N (Y/N)
One Recovery Job
for all objects N (Y/N)
VERIFYSET (Y/N) _ ENFORCE (Y/N) _ BACKOUT (Y/N) _
PF3=Exit ENTER=Process Your Selection
```

Case Study 15: Disaster Recovery

DB2I2 implements disaster recovery process with a TSO command call DR. The same recovery strategy as point of time recovery is used in the DB2I2 TSO DR command, which is if the selected tablespace does not have any updates since the consistent point, then there is no need to recover those tablespaces. The DR consistent point is the point when the full volume dump is performed.

Before you can use DR command, you should have the following procedures in place:

1. Make sure there is an archive log db2 command before your scheduled full volume dump. This establishes a consistent point for recovery checking.
2. Registers all databases to be recovered during the disaster. The format of this register-database-file is a sequential or PDS file contains the following:
 - 1st record of the file contains the PARMUTIL output generated for REPORT RECOVERY with RECOVERY TO option. The format is
DRPRMRPT='your.report.recoveryto.parmutil.file'
 - 2nd record contains the PARMUTIL output generated for RECOVER to current option. The format is
DRPRMRCV='your.recover.to.current.file'
 - The rest of the records contain the database name, job name for REPORT job, and job name for RECOVERY jobs.
DBNAME job-name-for-report job-name-for-recovery
The job-name-for-report and job-name-for-recovery are job names defined to run which REPORT or RECOVERY jobs are generated during the process of DR command. Specify # as suffix of the job Name allows you to run multiple recovery jobs concurrently.
The following is an example of a register file:

```
DRPRMRPT='DB2I2.PARMUTIL(REPORTDR)'  
DRPRMRCV='DB2I2.PARMUTIL(RECOVRDR)'  
MYDB1 MYRPT01 MYRCV1##  
MYDB2 MYRPT02 MYRCV2##
```

During the disaster recovery, after you have recover you DB2 catalog and directory and restore your full volume dump, issues the following commands in sequence:

- Issues DB2I2 DSJU004 command to display all the RBA points from BSDS.
- Issues DB2I2 SETRBA command to select the RL line, which represents the archive log command before
- Issue **DB2I2 TSO DR 'register-database-file' output-file-prefix Y|N [Track_table_name]**

Where

'register-data-file' contains all the database required to be recovered with the format as described above

output-file-prefix represents the prefix for all the output PDS files.

For example,

DB2I2 generates the following PDS when MY.DR is used as output-file-prefix:

'MY.DR.JCL' contains DR JCL with database as the member name

'MY.DR.TS' contains TS line objects with database as the member name

'MY.DR.TS.REPORT' contains DB2 REPORT JCL with database as the member name

'MY.DR.REPORT.TS' contains TS line open for update after the RL RBA point, with database as the member name

'MY.DR.REPORT.TS.RECOVERY' contains RECOVER TO CURRENT jobs for all TS line open for update after the RL RBA point, with database as the member name

Y|N specify Y to submit recovery jobs after they are generated. Specify N to generate recovery jobs only.

Track_table_name specify optional job tracking db2 table name if you want DB2I2 to generate job execution status tracking steps with DB2I2LOG command. Please refer to TSO command for detail.

During the execution of DR command, DB2I2 generates and process the following routines for each database:

- DB2I2 generates one DR job for each specified database (in DR.JCL)
- Submits each generated DR JCL
- For each DR JCL process
 - Issues SETRBA command to set the in-core RBA to the consistent point RBA
 - Issues DB2I2 TS command to get all the TS lines
 - Issues DB2I2 REPORT command with RECOVER TO option to generate report JCL
 - Submits the generated JCL
 - For each submitted JCL:
 - Process DB2 REPORT utility for all selected TS lines
 - Issues SETRBA command to set the in-core RBA to the consistent point RBA
 - Issues P000710 to select only those TS lines, which have been opened for update since the consistent RBA.
 - Issues RECOVER to recover to current only for those TP lines returned from P000710
 - If submit option Y is selected, DB2I2 submits all the generated Recovery jobs.

Case Study 16: Unload and Reload

GENURLD is a system defined UDF which does the DSNTIAUL unload and DB2 LOAD, REPAIR and RUNSTATS against a set of TB line object on table at a time. The source table and target table can come from the same or different locations.

To invoke this process, first, select a set of TB lines and issue the following command

```
BATCH ICMD=*GENURLD &JCLWS='your.build.jcl' ODSN='your.gen.jcl'
```

Where 'your.gen.jcl' is the generate process output JCL
'your.build.jcl' is the build process output JCL

Because there are a serial of parameters needed to be modified in 'your.gen.jcl', you should save this output file for later usage.

The following screen will be displayed in 'your.gen.jcl':

```
SETG      IDD=GLOBDD
BATCH     ICMD=*GENURLD1 +
          ODSN='your.gen.jcl' +
          &OP1 &OP2 &OP3 &OP4 &OP5 &OP6 &OP7 &OP8 &OP9 +
          &OP10 &OP11 &OP12 &OP13 &OP14
TSO       SUBMIT 'your.gen.jcl'
//GLOBDD  DD DATA,DLM=GG
GV &OP1=&TBDSWS=<GENERATED.TB>
GV &OP2=&UNLDJWS=<OUTPUT.JCL>
GV &OP3=&JOBCTL=<JOB.PARM>
GV &OP4=&LDTBWS=<LOAD.TB>
GV &OP5=&WHEREWS=<WHERE.DS>
GV &OP6=&PUTILL=<LOAD.PARMUTIL>
GV &OP7=&PUTILNC=<REPAIR.NCOPY.PARMUTIL>
GV &OP8=&PUTILRS=<RUNSTATS.PARMUTIL>
GV &OP9=&FROMLOC=<fromlocation>
GV &OP10=&TOLOC=<tolocation>
GV &OP11=&FROMCLAS=<from-job-class>
GV &OP12=&TOCLAS=<to-job-class>
GV &OP13=&DSPREWS=<dataset-prefix>
GV &OP14=&WKSPWS=<###>
GG
```

Modify all the information within <>. The following are the definition for each of these fields:

<GENERATED.TB>	generated work TB sequential file for DSNTIAUL
<OUTPUT.JCL>	generated output JCL (created as a new sequential file)
<JOB.PARM>	generated job parm work PDS file
<LOAD.TB>	generated work TB sequential file for LOAD
<WHERE.DS>	generated work WHERE predicates sequential file
<LOAD.PARMUTIL>	PARMUTIL LOAD input file
<REPAIR.NCOPY.PARMUTIL>	PARMUTIL REPAIR NOCOPYPEND input file
<RUNSTATS.PARMUTIL>	PARMUTIL RUNSTATS input file
<fromlocation>	location name for the source table
<tolocation>	location name for the target table
<fromclas>	JOB CLASS for source table unload JOB
<toclas>	JOB CLASS for target table load JOB
<dataset-prefix>	work file and utility file dataset prefix
<###>	Number of cylinders for work files

After you modify these fields and submit the job, you should have the output from generate process kept in 'your.build.jcl'. You should have similar result like the one display below:

```

-----
-- GENURLD1:  Invoked by GENURLD                               --
--              to prepare the DSNTIAUL unload JCL             --
--                      LOAD JCL                               --
--                      REPAIR NO COPYEND JCL                  --
-----
-- QBUILD to generate WHERE predicates for the query RUN
-----
QBUILD  F1=CREATOR F2=NAME ODSN=&WHEREWS
-----
-- RUN the query to build the TB line with SYSPUNCH and SYSREC info
-----
RUN      IDSN=*GENUNLDQ T=N ODSN=&TBDSWS NOTFOUND(SKIP 99)
-----
-- FCPY to make a copy of the generated load TB
-----
TSO FCPY &TBDSWS +
        &LDTBWS
-----
-- ED to invoke instream edit macro in CHNGIDD
--   to modify the SYSPUNCH= to SYSIN=
--   which is the format for the LOAD TB line
-- **You can also change the other information such as
--   tbcreator or tbname information under CHNGIDD
-----
ED      &LDTBWS +
        MACRO(IDD=CHNGIDD)
REXX  IDD=PROCDD
//PROCDD DD DATA,DLM=RR
/* REXX ----- */
/* This inline REXX routine:                                  */
/* 0. Connect to source system                                */
/* 1. Generate DSNTIAUL to unload table for source tables    */
/*    the Job Class will be substituted with FROMCLAS       */
/* 2. Edit the SYSPUNCH from DSNTIAUL to change the following: */
/*    - source.tbname to target.tbname                       */
/*    - LOG NO to LOG NO RESUME NO REPLACE                   */
/*    - SYSREC00 to SYSREC                                    */
/* 3. Connect to target system                                */
/* 4. Generate LOAD to load table for target tables          */
/*    the Job Class will be substituted with TOCLAS         */
/* 5. If REPAIR NOCOPYPEND and RUNSTATS is desired          */
/*    the REPAIR NOCOPYPEND and RUNSTATS jobs will be generated */
/* ----- */
x = msg('off')
Address Ispexec "control errors Return"
ws_time = time()
saverc = 0
Job# = 0
suff = '.t' || substr(ws_time,1,2) || substr(ws_time,4,2) ||
       substr(ws_time,7,2)
x=listdsi(&JOBCNTL)
If x \= 0 Then
Do
  Address Tso "Alloc fi(jobcntl)",
             "ds(&JOBCNTL)",
             "space(15,5) tracks " ||
             "dir(20) " ||
             "recfm(f b) lrecl(250) blksize(0) unit(sysda) reuse"
  Address Tso "Free fi(jobcntl)"
End
Else
If sysdsorg \= 'PO' | syslrecl \= 250 Then
Do
  If sysdsorg \= 'PO' Then
    Say &JOBCNTL 'is not a PDS'
  Else
    Say &JOBCNTL 'LRECL is not 250'
  Say 'Job Abended with Condition Code 16'
  saverc = 16
  zispfrc = saverc
  Address Ispexec "vput (zispfrc) shared"
  signal RT_End
End
End

```

```

Address TSO "Delete "&UNLDJWS
Address TSO "Alloc fi(lb1) ds(&TBDSWS) SHR REUSE"
"Execio * Diskr lb1(stem recinl1. finis"
Address Tso "Free fi(lb1)"
Address TSO "Alloc fi(lb2) ds(&LDTBWS) SHR REUSE"
"Execio * Diskr lb2(stem recinl2. finis"
Address Tso "Free fi(lb2)"
"Execio * Diskr jobcard(stem recinj. finis"
ws = recinj.1
Parse Var ws jobname junk
jobi = index(jobname,'#')
If Jobi \= 0 Then
Do
  jobl = length(jobname)
  jobil = 0
  Do jobii = jobi to jobl
    If Substr(jobname,jobii,1) \= '#' Then
      Leave
    jobil = jobil + 1
  End
  Address Tso "Alloc fi(jobcard)",
    "space(1 1) tracks " ||,
    "recfm(f b) lrecl(80) blksize(0) unit(sysda) reuse"
End
cl = 0
cp = 0
Do i = 1 to recinj.0
  jobc = index(recinj.i,' CLASS=')
  If jobc \= 0 Then
  Do
    cl = i
    cp = jobc + 7
    leave
  End
  Else
  Do
    jobc = index(recinj.i,',CLASS=')
    If jobc \= 0 Then
    Do
      cl = i
      cp = jobc + 7
      leave
    End
  End
End
"Execio "recinj.0" Diskw jobcard(stem recinj. finis"
Address Tso "Alloc fi(db2i2cmd)",
"space(1 1) tracks " ||,
"recfm(f b) lrecl(250) blksize(0) unit(sysda) reuse"
Do i = 1 to recinl1.0
  ii = 0
  Call Job_Process
  Call Job_Class "&FROMCLAS"
  parse var recinl1.i j1 tbl1 j2
  parse value j2 with j1 'SYSPUNCH=' editds ' ' j3
  parse var recinl2.i j1 tbl2 j2
  parse value editds with j1 '.' j2 '.' j3 '.' j4
  wso = substr(j2,6,2) || substr(j3,6,2) || substr(j4,6,2)
  wss = "&JOBCTRL"
  If substr(wss,1,1) = " " Then
  Do
    wsoul = substr(wss,1,length(wss)-1) || "(UL" ||,
      wso || " )"
    wsold = substr(wss,1,length(wss)-1) || "(LD" ||,
      wso || " )"
    wsots = substr(wss,1,length(wss)-1) || "(TS" ||,
      wso || " )"
  End
  Else
  Do
    wsoul = wss || "(UL" || wso || " )"
    wsold = wss || "(LD" || wso || " )"
    wsots = wss || "(TS" || wso || " )"
  End
  Address Tso "Alloc fi(dd1) DS("wsoul") shr reuse"
  recol.1 = recinl1.i
  "Execio 1 Diskw ddl(stem recol. finis"
  Address Tso "Alloc fi(dd1) DS("wsold") shr reuse"

```

```

recol.1 = recinl2.i
"Execio 1 Diskw ddl(stem recol. finis"
/* ----- */
/* build DB2I2 command for unload */
/* ----- */
Call add_a_cmd "CONNECT("&FROMLOC")"
Call add_a_cmd "DSNTIAUL +"
Call add_a_cmd " IDSN="wsoul "+"
Call add_a_cmd " ODSN=&UNLDJWS +"
Call add_a_cmd " DSPRE=&DSPREWS +"
Call add_a_cmd " WKSP=&WKSPWS +"
Call add_a_cmd " STEP#=10 APPEND"
/* ----- */
/* Change *GENURLDE to your own */
/* EDIT routine if desirable */
/* ----- */
Call add_a_cmd "BATCH ICMD=*GENURLDE +"
Call add_a_cmd " IDSN=*DUMMY +"
Call add_a_cmd " ODSN=&UNLDJWS +"
Call add_a_cmd " JOBCARD=N APPEND STEP#=20 +"
Call add_a_cmd " &EDITDS="editds "+"
Call add_a_cmd " &TBL1="tbl1" +"
Call add_a_cmd " &TBL2="tbl2
/* ----- */
/* build DB2I2 command for load */
/* ----- */
If &FROMLOC = &TOLOC Then
Do
  stp#=30
  ws = 'JOBCARD=N STEP#='stp#
End
Else
Do
  Call Job_Class "&TOCLAS"
  Call add_a_cmd "CONNECT("&TOLOC")"
  stp#=10
  ws = 'STEP#='stp#
End
/* ----- */
/* Load target table */
/* ----- */
Call add_a_cmd "LOAD &PUTILL +"
Call add_a_cmd " IDSN="wsold "+"
Call add_a_cmd " ODSN=&UNLDJWS +"
Call add_a_cmd " DSPRE=&DSPREWS +"
Call add_a_cmd " WKSP=&WKSPWS +"
Call add_a_cmd " APPEND "ws
If "&PUTILNC" \= '' |,
"&PUTILRS" \= '' Then
Do
  /* ----- */
  /* get TS line object */
  /* ----- */
  Call add_a_cmd "TS ODSN="wsots "+"
  Call add_a_cmd " IDSN="wsold
  If "&PUTILNC" \= '' Then
  Do
    /* ----- */
    /* setup REPAIR NOCOPYPEND */
    /* ----- */
    stp#=stp#+10
    ws = 'JOBCARD=N APPEND STEP#='stp#
    Call add_a_cmd "REPAIR &PUTILNC +"
    Call add_a_cmd " IDSN="wsots "+"
    Call add_a_cmd " ODSN=&UNLDJWS +"
    Call add_a_cmd " "ws
  End
  If "&PUTILRS" \= '' Then
  Do
    /* ----- */
    /* setup RUNSTATS job step */
    /* ----- */
    stp#=stp#+10
    ws = 'JOBCARD=N APPEND STEP#='stp#
    Call add_a_cmd "RUNSTATS &PUTILRS +"
    Call add_a_cmd " IDSN="wsots "+"
    Call add_a_cmd " ODSN=&UNLDJWS +"
    Call add_a_cmd " "ws
  End

```

```

        End
    End
    "Execio "ii" Diskw db2i2cmd(stem cmd. finis"
    Call DB2I2
    Address Ispexec "vget (zispfrc) shared"
    If zispfrc > 4 Then
    Do
        saverc = zispfrc
        signal RT_End
    End
    Else
        If zispfrc \= 0 Then
            saverc = zispfrc
        End
    End
    Address Tso "Free fi(lineobj)"
    zispfrc = saverc
    Address Ispexec "vput (zispfrc) shared"
RT_End:
    Exit saverc
ADD_a_cmd: ARG wscmd
    ii = ii + 1
    cmd.ii = wscmd
Return
JOB_Process:
    If jobi \= 0 Then
    Do
        job# = job# + 1
        If length(job#) > jobil Then
            job# = Substr(job#,length(job#)-jobil+1,jobil)
            jobws=Translate(format(Substr(job#,1,jobil),jobil,0),'0',' ')
            If jobi + jobil - 1 = jobl Then
                recinj.1 = Substr(jobname,1,jobi-1) ||,
                    jobws junk
            Else
                recinj.1 = Substr(jobname,1,jobi-1) ||,
                    jobws ||,
                    Substr(jobname,(jobi+jobil),(jobl-jobi-jobil+1)) junk
            End
        "Execio "recinj.0" Diskw jobcard(stem recinj. finis"
    Return
JOB_Class: ARG jclass
    If cl \= 0 Then
        recinj.cl = substr(recinj.cl,1,cp-1) || jclass ||,
            substr(recinj.cl,cp+1,80-cp)
        "Execio "recinj.0" Diskw jobcard(stem recinj. finis"
    Return
RR
/* ----- **
/* Insert more C line below to change the source tb to target tb **
/* ----- **
//CHNGIDD DD DATA,DLM=AA
C 'SYSPUNCH=' 'SYSIN=' ALL
AA

```

You can make desired changes to those **bold print fields** and submit the JCL to generate the JCL for the table unload and reload process.

Case Study 17: CODEGEN with ED macro

CODEGEN is a system defined ED macro which does generates COBOL code from a DELETE or UPDATE query.

To invoke CODEGEN, select a set of control card lines together with UPDATE or DELETE SQL and then issue the following commad (See CODEGENS for sample of control statements)

```
ED your.output.file MACRO(*CODEGENM) PASS=Y
```

Where 'your.output.file' contains the output generated from CODEGEN. The control statements can be entered with a SS block select, a IDSN=input.control,stmt.dsn or IDD=input.control.stmt,ddname

The following is the ED Macro CODEGENM:

```
-----
-- CodegenM: Cobol Code Generator for UPDATE and DELETE SQL --
-----
NUM OFF
-- DELETE all lines
DELETE * *
-----
-- Check lnoobj.0 to see if any line objects are selected --
-----
REXX Say '**>> Total number of Records Passed to Your ED macro:' lnoobj.0
REXX If lnoobj.0 = 0 Then; +
REXX Do; Say '**>> No Query Lines Selected'; +
REXX LastRC = 16; +
REXX Signal Finish; End;
-----
-- Initialize all variables --
-- EdQuery - DELETE or UPDATE SQL --
-- Edwi - line number of WHERE --
-- Edmi - Number of Messages --
-----
REXX EdQuery = ''
REXX Edwi = 0
REXX Edmi = 0
REXX Userid = userid()
REXX Edprocsw = ''
REXX Edssid = ''
REXX Edcollid = ''
REXX Edplan = ''
REXX Edpgmid = ''
REXX EdCommit = ''
REXX EdLoad = ''
REXX Edcobsys = ''
REXX Edle = ''
REXX Eddbrm = ''
REXX Edjob = ''
REXX edcompj = ''
REXX Edcobskl = ''
REXX Edgoskel = ''
REXX Edwkdsn = ''
-----
-- Looping thru all line objects to setup control variable and query --
-----
REXX Do Edii = 1 to lnoobj.0; +
REXX Edline = lnoobj.edii; +
REXX Edln.edii = strip(Edline,T); +
REXX Parse value edln.edii with edj1 '=' edj2; +
REXX If Edj1 = 'PROCESS' Then; +
REXX Edprocsw = Word(edj2,1); +
REXX Else; +
REXX If Edj1 = 'PGMID' Then; +
REXX Edpgmid = Word(edj2,1); +
REXX Else; +
REXX If Edj1 = 'SSID' Then; +
REXX Edssid = Word(edj2,1); +
REXX Else; +
```

```

REXX      If Edj1 = 'COLLID' Then;                                +
REXX          Edcollid = Word(edj2,1);                            +
REXX      Else;                                                  +
REXX      If Edj1 = 'PLAN' Then;                                  +
REXX          Edplan   = Word(edj2,1);                            +
REXX      Else;                                                  +
REXX      If edj1 = 'COMMIT' Then;                                +
REXX          EdCommit = Word(edj2,1);                            +
REXX      Else;                                                  +
REXX      If edj1 = 'LOAD' Then;                                  +
REXX      Do;                                                    +
REXX          EdLoad = Word(edj2,1);                              +
REXX          If Substr(EdLoad,1,1) = "" Then;                    +
REXX              EdLoad = Substr(EdLoad,2,length(EdLoad)-2);    +
REXX          Else;                                              +
REXX              EdLoad = Userid"."Edload;                       +
REXX      End;                                                  +
REXX      Else;                                                  +
REXX      If edj1 = 'LE' Then;                                    +
REXX      Do;                                                    +
REXX          EdLe = Word(edj2,1);                                +
REXX          If Substr(EdLe,1,1) = "" Then;                      +
REXX              EdLe = Substr(EdLe,2,length(EdLe)-2);          +
REXX          Else;                                              +
REXX              EdLe  = Userid"."Edle;                          +
REXX      End;                                                  +
REXX      Else;                                                  +
REXX      If edj1 = 'COBSYSLB' Then;                              +
REXX      Do;                                                    +
REXX          Edcobsys = Word(edj2,1);                            +
REXX          If Substr(Edcobsys,1,1) = "" Then;                  +
REXX              Edcobsys = Substr(Edcobsys,2,length(Edcobsys)-2); +
REXX          Else;                                              +
REXX              Edcobsys = Userid"."Edcobsys;                   +
REXX      End;                                                  +
REXX      Else;                                                  +
REXX      If edj1 = 'DBRM' Then;                                  +
REXX      Do;                                                    +
REXX          Eddbrm = Word(edj2,1);                              +
REXX          If Substr(Eddbrm,1,1) = "" Then;                    +
REXX              Eddbrm = Substr(Eddbrm,2,length(Eddbrm)-2);    +
REXX          Else;                                              +
REXX              Eddbrm = Userid"."Eddbrm;                       +
REXX      End;                                                  +
REXX      Else;                                                  +
REXX      If edj1 = 'WKDSN' Then;                                  +
REXX          Edwkdsn = Word(edj2,1);                              +
REXX      Else;                                                  +
REXX      If edj1 = 'JOB CARD' Then;                              +
REXX          Edjob = Word(edj2,1);                                +
REXX      Else;                                                  +
REXX      If edj1 = 'COMPJCL' Then;                              +
REXX          edcompj = Word(edj2,1);                              +
REXX      Else;                                                  +
REXX      If edj1 = 'COBSKEL' Then;                              +
REXX          Edcobskl = Word(edj2,1);                            +
REXX      Else;                                                  +
REXX      If edj1 = 'GOSKEL' Then;                              +
REXX          Edgoskel = Word(edj2,1);                            +
REXX      Else;                                                  +
REXX      Do;                                                    +
REXX          If Word(edln.edii,1) = 'WHERE' Then edwi = edii;    +
REXX          If Substr(edln.edii,1,2) \= '--' Then;              +
REXX              EdQuery = EdQuery Edln.edii;                   +
REXX      End;                                                  +
REXX End;

-----
-- Check for the valid combinations --
-----

REXX If Edprocsw = '' Then;                                        +
REXX Do; Edprocsw = 'SOURCE';                                    +
REXX     Edmi = Edmi + 1;                                        +
REXX     Edm.edmi = '** No PROCESS Entered. Use PROCESS=SOURCE'; +
REXX     Say Edm.edmi;                                          +
REXX End;

REXX If Edpgmid = '' Then;                                       +
REXX Do; Edpgmid = 'CODEGEN';                                    +
REXX     Edmi = Edmi + 1;                                       +

```

```

REXX      Edm.edmi = '** No EDPGMID      Entered. Use EDPGMID=CODEGEN'; +
REXX      Say Edm.edmi;                                                    +
REXX End;
REXX If EdCommit = '' Then;                                               +
REXX Do; EdCommit = '50';                                                 +
REXX      Edmi = Edmi + 1;                                                 +
REXX      Edm.edmi = '** No EdCommit Entered. Use EdCommit=50'; +
REXX      Say Edm.edmi;                                                    +
REXX End;
REXX If Edjob = '' Then;                                                  +
REXX Do; Edjob = '*CODEGJOB';                                             +
REXX      Edmi = Edmi + 1;                                                 +
REXX      Edm.edmi = '** No JOBCARD Entered. Use JOBCARD=*CODEGJOB'; +
REXX      Say Edm.edmi;                                                    +
REXX End;
REXX If edcompj = '' Then;                                               +
REXX Do; edcompj = '*CODEGENJ';                                           +
REXX      Edmi = Edmi + 1;                                                 +
REXX      Edm.edmi = '** No COMPJCL Entered. Use COMPJCL=*CODEGENJ'; +
REXX      Say Edm.edmi;                                                    +
REXX End;
REXX If Edwkdsn = '' Then;                                               +
REXX Do; Edwkdsn = 'CODEGEN.WKDSN';                                       +
REXX      Edmi = Edmi + 1;                                                 +
REXX      Edm.edmi = '** No WKDSN Entered. Use WKDSN=CODEGEN.WKDSN'; +
REXX      Say Edm.edmi;                                                    +
REXX End;
REXX If Edle = '' Then;                                                  +
REXX Do; Edle = 'SYS1.SCEELKED';                                           +
REXX      Edmi = Edmi + 1;                                                 +
REXX      Edm.edmi = '** No LE Entered. Use LE="edle"'; +
REXX      Say Edm.edmi;                                                    +
REXX End;
REXX If Edcobsys = '' Then;                                              +
REXX Do; Edcobsys = 'SYS1.SIGYCOMP';                                       +
REXX      Edmi = Edmi + 1;                                                 +
REXX      Edm.edmi = '** No COBSYSLB Entered. Use COBSYSLB="Edcobsys"'; +
REXX      Say Edm.edmi;                                                    +
REXX End;
REXX If Edcobskl = '' Then;                                              +
REXX Do; Edcobskl = '*CODEGEN';                                           +
REXX      Edmi = Edmi + 1;                                                 +
REXX      Edm.edmi = '** No COBSKEL Entered. Use COBSKEL=*CODEGEN'; +
REXX      Say Edm.edmi;                                                    +
REXX End;
REXX If Edgoskel = '' Then;                                              +
REXX Do; Edgoskel = '*CODEGENG';                                           +
REXX      Edmi = Edmi + 1;                                                 +
REXX      Edm.edmi = '** No GOSKEL Entered. Use GOSKEL=*CODEGENG'; +
REXX      Say Edm.edmi;                                                    +
REXX End;
REXX If (Edprocsw = 'COMPILE' | Edprocsw = 'GO') & (EdLoad = '' | +
REXX      Eddbrm = '' | Edssid = '' | Edcollid = '') Then; +
REXX Do; If EdLoad = '' Then; +
REXX      Say '**>> Missing LOAD='; +
REXX      If Eddbrm = '' Then; +
REXX      Say '**>> Missing DBRM='; +
REXX      If Edssid = '' Then; +
REXX      Say '**>> Missing SSID='; +
REXX      If Edcollid = '' Then; +
REXX      Say '**>> Missing COLLID='; +
REXX      LastRC = 16 ; +
REXX      Signal Finish; End;
REXX If Edprocsw = 'GO' & Edplan = '' Then; +
REXX Do; If Edplan = '' Then; +
REXX      Say '**>> Missing PLAN='; +
REXX      LastRC = 16 ; +
REXX      Signal Finish; End;
-----
-- Process UPDATE or DELETE Query --
-----
REXX Say '**>> Query Used for CODEGEN <<*'
REXX Say Strip(Edquery)
REXX Say '**>> ----- <<*'
REXX Edws = word(EdQuery,1)
REXX Edvi = 0
REXX edwss= ''
-----

```

```

-- Process UPDATE Query      --
-----
REXX If Edws = 'UPDATE' Then;          +
REXX Do; Parse Var EdQuery ed1 Edtblnm ed2;          +
REXX   Parse Value Edtblnm with edcrtr '.' edtbnm;    +
REXX   edcol = '';          +
REXX   Parse Value ed2 with ed3 'SET' ed4 'WHERE' ed5;  +
-----
-- Build column name between SET and WHERE      --
-- Edva. - DB2 Column Name                      --
-- Edvaa. - COBOL working Storage variable Name --
-- Edvl. - The assigned value of each column    --
-- Edvi - Number of columns                    --
-- Edcol - String of all columns                --
-----
REXX   Do While (Index(ed4,'=') \= 0);          +
REXX   Parse Value ed4 with edvl '=' ed4;        +
REXX   If ed4 \= '' Then;          +
REXX   Do;          +
REXX     edvi = edvi + 1;          +
REXX     edva.edvi = Word(edvl,words(edvl));    +
REXX     Replace _ with -          +
REXX     If Index(edva.edvi,'_') \= 0 Then;      +
REXX     edvaa.edvi = Translate(edva.edvi,'-','_'); +
REXX     Else edvaa.edvi = edva.edvi;          +
REXX     edcol = edcol||edwss"edva.edvi";      +
REXX     Parse Value ed4 with edvl.edvi ',' ed4; +
REXX     edvl.edvi = Strip(edvl.edvi);          +
REXX     edwss = ',';          +
REXX   End;          +
REXX End;          +
REXX   edg = "&TBC="Edcrtr" &TBN="Edtbnm" &COL="Edcol"; +
REXX   edsw = 'U';          +
REXX End;          +
REXX Else;          +
-----
-- Process DELETE Query      --
-----
REXX If Edws = 'DELETE' Then;          +
REXX Do; Parse Value EdQuery with ed1 'FROM' ed2;    +
REXX   Parse Var ed2 Edtblnm ed2;          +
REXX   Parse Value Edtblnm with edcrtr '.' edtbnm;    +
REXX   edcol = '';          +
REXX   Parse Value ed2 with ed3 'WHERE' ed4;          +
REXX   If ed4 = '' Then;          +
REXX   Do;          +
REXX     Say '**>> Missing WHERE for DELETE ' ;    +
REXX     Lastrc = 16 ;          +
REXX     Signal Finish;          +
REXX   End;          +
REXX   edvi = edvi + 1;          +
-----
-- Pick Only the First Column name after WHERE  --
-----
REXX   edva.edvi = Word(ed4,1);          +
REXX   If Index(edva.edvi,'_') \= 0 Then;          +
REXX     edvaa.edvi = Translate(edva.edvi,'-','_'); +
REXX     Else edvaa.edvi = edva.edvi;          +
REXX     edcol = edcol||edwss"edva.edvi";      +
REXX     edvl.edvi = Strip(edvl.edvi);          +
REXX   edg = "&TBC="Edcrtr" &TBN="Edtbnm" &COL="Edcol"; +
REXX   edsw = 'D';          +
REXX End;          +
-----
-- Prepare Global Variable &G1 for DB2I2 RUN command --
-----
REXX Edg = Edg 'ODSN='Edwkdsn
SETG2 &G1=Edg
DB2I2 RUN IDSN=*CODEGENQ &G1 T=N
REXX Address Tso "Alloc fi(fl) ds("Edwkdsn") shr reuse"
REXX "Execio * Diskr fl(Stem edrin. Finis"
REXX Address Tso "Free fi(fl)"
-----
-- Get Information from RUN output from Edwkdsn --
-----
REXX Do edii = 1 to edvi;          +
REXX Do ediii = 1 to edrin.0;          +
REXX   If index(edrin.ediii,edva.edii) \= 0 Then;  +

```



```

REXX Do;
REXX Parse var edrin.ediii edval edva2 edva3 edva4 edva5 edva6;
REXX ed01.edii = edval;
REXX ed02.edii = edva2; /* type */
REXX ed03.edii = edva3;
REXX ed04.edii = edva4;
REXX ed05.edii = edva5;
REXX ed06.edii = Strip(edva6);
REXX End;
REXX End;
REXX End;
REXX Parse Value wdsdn with edj1 '(' edmem ')' edj2
-----
-- Insert Codegen Skeltens:
-- For SOURCE Edcobskl
-- COMPILE Edjob Edcompj Edcobski
-- GO Edjob Edcompj Edcobski Edgoskel
-----

REXX edlnpos = 1
REXX If Edprocsw = 'SOURCE' Then SKIP 5
INSERTA 1 IDSN={Edjob}
BOTTOM
INSERTA * IDSN={Edcompj}
F AA FIRST
REXX edlnpos = '*'
INSERTA {edlnpos} IDSN={Edcobskl}
REXX If Edprocsw \= 'GO' Then SKIP 2
BOTTOM
INSERTA * IDSN={Edgoskel}
REXX eds6 = Substr(' ',1,6)
REXX eds11= Substr(' ',1,11)
REXX eds15= Substr(' ',1,15)
REXX eds19= Substr(' ',1,19)
-----
-- Insert Default Messages Edm. and Line Objects Lnobj. as COB comment--
-----

F '>>COMMENT' First
DELETE * 1
REXX Wsjunk = eds6 || SUBSTR('* DB2I2 CODEGEN for Query',1,39)|| +
REXX 'Date: 'Date(U) Time() '*'
INSERTB * {Wsjunk}
-- Looping thru Default Messages Edm.
REXX Edii = 1
LOOP {edmi}
REXX edmo = eds6 || Substr(edm.edii,1,62) '*'
INSERTB * {edmo}
REXX Edii = Edii + 1
LOOP_END
-- Looping thru Line Objects Lnobj.
REXX Edii = 1
LOOP {lnobj.0}
REXX Edln2 = eds6 || '*' Substr(Edln.edii,1,60) '*'
INSERTB * {Edln2}
REXX Edii = Edii + 1
LOOP_END
REXX Eoloop = 'N'
-----
-- Insert Column Information for Declare Cursor Process --
-----

F '>>CURSOR' First
DELETE * 1
REXX Edii = 1
REXX Edws = ' '
LOOP {edvi}
REXX Edln2 = eds19 || edws edva.edii
INSERTB * {Edln2}
REXX Edii = Edii + 1
REXX Edws = ','
LOOP_END
-----
-- Insert Table Information for Declare Cursor Process --
-----

REXX Edln2 = eds15 || 'From ' Edtblnm
INSERTB * {Edln2}
REXX Edif = lnobj.0 - edwi + 1
REXX Edii = edwi
-----
-- Insert WHERE Information for Declare Cursor Process --
-----

```

```

-----
REXX Eoloop = 'N'
LOOP {edif}
  REXX Edln2 = eds15 || Strip(Substr(edln.edii,1,60))
  INSERTB * {Edln2}
  REXX Edii = edii + 1
LOOP_END
REXX Edln2 = eds15 || 'For Update Of'
INSERTB * {Edln2}
REXX Edii = 1
REXX Edws = ' '
-----
-- Insert Column Information for FOR UPDATE OF Process --
-----
REXX Eoloop = 'N'
LOOP {edvi}
  REXX Edln2 = eds19 || edws edva.edii
  INSERTB * {Edln2}
  REXX Edii = Edii + 1
  REXX Edws = ','
LOOP_END
-----
-- Insert Working Storage Variable Process --
-----
F '>>COLUMN' First
DELETE * 1
REXX Edii = 1
REXX Eoloop = 'N'
LOOP {edvi}
  REXX If ed02.edii \= 'VARCHAR' Then SKIP 9
  REXX Edvaag.edii = edvaa.edii||'-G'
  REXX Edln2 = eds11 || '05 ' edvaag.edii..'
  INSERTB * {Edln2}
  REXX Edvaal.edii = edvaa.edii||'-L'
  REXX Edln2 = eds15 || '49 ' substr(edvaal.edii,1,18) 'Pic s9(4) Comp.'
  INSERTB * {Edln2}
  REXX Edln2 = eds15 || '49 ' substr(edvaa.edii||' ',1,18) ed06.edii
  INSERTB * {Edln2}
  REXX SKIP 2
  REXX Edln2 = eds11 || '05 ' substr(edvaa.edii,1,22) ed06.edii
  INSERTB * {Edln2}
  REXX Edii = Edii + 1
LOOP_END
-----
-- Insert MOVE statement for Fetch Process --
-----
F '>>FETCH' FIRST
DELETE * 1
REXX If Edsw = 'D' Then SKIP 11
REXX Edii = 1
REXX Eoloop = 'N'
LOOP {edvi}
  REXX If word(edvl.edii,1) = 'CURRENT' Then SKIP 5
  REXX If ed02.edii \= 'VARCHAR' Then SKIP 2
  REXX Edln2 = eds11 || 'MOVE' length(edvl.edii) 'TO' edvaal.edii..'
  INSERTB * {Edln2}
  REXX Edln2 = eds11 || 'MOVE' edvl.edii 'TO' edvaa.edii..'
  INSERTB * {Edln2}
  REXX Edii = Edii + 1
LOOP_END
-----
-- Insert UPDATE SET or DELETE FROM EXEC SQL --
-----
REXX Edln2 = eds11 || 'EXEC SQL'
INSERTB * {Edln2}
REXX If edsw = 'U' Then; +
REXX Edln2 = eds15 || 'Update 'Edtblnm; +
REXX Else If edsw = 'D' Then; +
REXX Edln2 = eds15 || 'Delete From 'Edtblnm;
INSERTB * {Edln2}
REXX If edsw = 'D' Then SKIP 20
-----
-- UPDATE SET column = value Process --
-----
REXX Edln2 = eds15 || 'Set'
INSERTB * {Edln2}
REXX Edii = 1
REXX Edws = ' '

```

```

REXX Eoloop = 'N'
LOOP {edvi}
  REXX If Word(edvl.edii,1) \= 'CURRENT' Then SKIP 2
  REXX Edln2 = eds15 || edws edva.edii '= 'edvl.edii
  REXX SKIP 4
  REXX If ed02.edii \= 'VARCHAR' Then SKIP 2
  REXX Edln2 = eds15 || edws edva.edii '= : 'edvaag.edii
  REXX SKIP 1
  REXX Edln2 = eds15 || edws edva.edii '= : 'edvaa.edii
  INSERTB * {Edln2}
  REXX Edii = Edii + 1
  REXX Edws = ', '
LOOP_END
REXX edln2 = eds15 || 'Where current of PROC-CS'
INSERTB * {Edln2}
REXX edln2 = eds11 || 'END-EXEC.'
INSERTB * {Edln2}
TOP

```

The following is the [CODEGJOB jobcard](#) skeleton:

```

//{USERID}A JOB job card information,
//      NOTIFY={USERID}

```

The following is the [CODEGENJ compiling JCL](#) skeleton:

```

//CODEGENC PROC
//*-----*** D B 2 P R E C O M P I L E ***-----+
//PC      EXEC PGM=DSNHPC,PARM='HOST(COBOL),APOST'
//*-----+
//DBRMLIB DD DISP=SHR,DSN=&DBRM(&MEM)
//SYSCIN  DD DSN=&&DSNHOUT,DISP=(MOD,PASS),UNIT=VIO,
//        SPACE=(CYL,(1,1))
//SYSPRINT DD SYSOUT=*
//SYSTEM  DD SYSOUT=*
//SYSUT1  DD SPACE=(CYL,(5,2)),UNIT=VIO
//SYSUT2  DD SPACE=(CYL,(5,2)),UNIT=VIO
//*
//*-----*** C O M P I L I N G ***-----+
//COB     EXEC PGM=IGYCRCTL,COND=(4,LT),
//        PARM=(DYN,'TRUNC(BIN)',SSR,'')
//*-----+
//STEPLIB DD DSN={EDCOBSYS},DISP=SHR
//SYSIN   DD DSN=&&DSNHOUT,DISP=(OLD,PASS)
//SYSLIN  DD DSN=&&SYSLIN,
//        UNIT=SYSALLDA,SPACE=(TRK,(15,15),RLSE),DISP=(NEW,PASS)
//SYSUT1  DD UNIT=SYSALLDA,SPACE=(TRK,(15,15),RLSE)
//SYSUT2  DD UNIT=SYSALLDA,SPACE=(TRK,(15,15),RLSE)
//SYSUT3  DD UNIT=SYSALLDA,SPACE=(TRK,(15,15),RLSE)
//SYSUT4  DD UNIT=SYSALLDA,SPACE=(TRK,(15,15),RLSE)
//SYSUT5  DD UNIT=SYSALLDA,SPACE=(TRK,(15,15),RLSE)
//SYSUT6  DD UNIT=SYSALLDA,SPACE=(TRK,(15,15),RLSE)
//SYSUT7  DD UNIT=SYSALLDA,SPACE=(TRK,(15,15),RLSE)
//SYSPRINT DD SYSOUT=*
//SYSPUNCH DD DUMMY
//SYSTEM  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*-----*** L I N K - E D I T ***-----+
//LKED   EXEC PGM=HEWL,COND=(4,LT),
//        PARM='LIST,XREF,AMODE(31),RMODE(ANY)'
//*-----+
//SYSLIB  DD DSN={EDLE},DISP=SHR
//SYSLIN  DD DSN=&&SYSLIN,DISP=(OLD,DELETE)
//        DD DDNAME=SYSIN
//SYSLMOD DD DSN=&&SYSLMOD,DISP=SHR
//SYSUT1  DD UNIT=SYSALLDA,SPACE=(TRK,(30,30))
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//        PEND
//STEP01  EXEC CODEGENC,
//        DBRM='{EDDBRM}',
//        MEM='{EDMEM}',
//        SYSLMOD='{EDLOAD}({EDMEM})'
//PC.SYSIN DD DATA,DLM=AA

```

```

AA
/*-----***   BIND PACKAGE                               ***-----+
//STEPBIND EXEC PGM=IKJEFT01,DYNAMNBR=100,REGION=4M
/*-----+
//SYSABOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN S({EDSSID})
  BIND PACKAGE({EDCOLLID}) -
  OWNER({USERID}) QUALIFIER({USERID}) -
  MEMBER({EDMEM}) LIBRARY('{EDDBRM}') -
  ACTION(REPLACE) CURRENTDATA(NO) DEGREE(1) ENABLE(*) -
  EXPLAIN(NO) FLAG(I) ISOLATION(CS) -
  SQLERROR(NOPACKAGE) VALIDATE(BIND)
END

```

The following is the [CODEGENS Sample Control Line Object](#):

```

-----
-- NOTE: All Control Information Entered needed to be UPPER CASE
--PROCESS = SOURCE COMPILE GO      Default PROCESS = SOURCE
--COMPJCL = COB2.SOURCE(CODEGENJ)  Default COMPJCL = *codegenj
--COBSKEL = COB2.SOURCE(CODEGEN)   Default COBSKEL = *codegen
--GOSEKL  = COB2.SOURCE(CODEGENG)  Default GOSKEL  = *codegeng
--JOB CARD = MY.JOB CARD           Default JOB CARD = *codegjob
--WKDSN   = CODEGEN.WKDSN         Default WKDSN   = CODEGEN.WKDSN
--PGMID   = CODEGEN              Default PGMID   = CODEGEN
--COMMIT  = 1                    Default COMMIT  = 50
--LE      = 'SYS1.SCEELKED'      Default LE      = 'SYS1.SCEELKED'
--COBSYSLB= 'SYS1.SIGYCOMP'      Default COBSYSLB= 'SYS1.SIGYCOMP'
--Require Control Card For
--  COMPILE: SSID, COLLID, LOAD, DBRM
--  GO      : SSID, COLLID, LOAD, DBRM, PLAN
-----
JOB CARD = TEST.CNTL(JOB CARD)
PROCESS  = GO
PGMID    = CODEGEN
COMMIT   = 50
SSID     = DSN
DBRM     = TEST.DBRMLIB
COLLID   = JRHJ
PLAN     = VJRHJ
LOAD     = TEST.LOAD
DELETE FROM JRHJ.PLAN_TABLE
WHERE
  PROGRAM LIKE 'T1AA005%'

```

The following is the [CODEGENQ Query](#) invoked by CODEGENM:

```

SELECT  A.NAME, COLTYPE, LENGTH, SCALE, NULLS,
        CASE WHEN COLTYPE = 'SMALLINT' THEN 'Pic S9(4) Comp.'
              WHEN COLTYPE = 'INTEGER'   THEN 'Pic S9(9) Comp.'
              WHEN COLTYPE = 'DECIMAL'   THEN
                'Pic S9(' || STRIP(CHAR(LENGTH)) ||
                'V9(' || STRIP(CHAR(SCALE)) || ') Comp-3.'
              WHEN COLTYPE = 'CHAR'      THEN
                'Pic X(' || STRIP(CHAR(LENGTH)) || ') .'
              WHEN COLTYPE = 'VARCHAR'   THEN
                'Pic X(' || STRIP(CHAR(LENGTH)) || ') .'
              WHEN COLTYPE = 'DATE'      THEN
                'Pic X(10) .'
              WHEN COLTYPE = 'TIME'      THEN
                'Pic X(08) .'

```

```

                WHEN COLTYPE = 'TIMESTAMP' THEN
                    'Pic X(26).'
                ELSE
                    ' '
                END CASE
FROM SYSIBM.SYSCOLUMNS A
WHERE TBCREATOR = &TBC
AND     TBNAME = &TBN
AND     NAME IN (&COL)

```

The following is the **CODEGENG** insert skeleton for GO step:

```

//STEPGO      EXEC PGM=IKJEFT1B,DYNAMNBR=100
//STEPLIB    DD DSN={EDLOAD},DISP=SHR
//SYSOUT     DD SYSOUT=*
//SYSPRINT   DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
DSN S({EDSSID})
RUN PROGRAM ({EDMEM})          -
PLAN      ({EDPLAN})

```

The following is the **CODEGEN COBOL** source skeleton:

```

Identification Division.
Program-ID.      {EDPGMID}.
*-----*
>>COMMENT
*-----*
Environment Division.
Input-Output Section.
File-Control.
Data Division.
File Section.
***-----***
*** Working Storage Area ***
***-----***
Working-Storage Section.
01 WS-Area.
   05 Filler                               Pic X(20) Value
                                           '***WORKING STORAGE***'.
   05 Commit-freq                           Pic S9(9) Comp Value {EDCOMMIT}.
   05 Commit-cnt                             Pic S9(9) Comp Value 0.
   05 No-Commit-cnt                         Pic S9(9) Comp Value 0.
   05 Total-Cnt                             Pic S9(9) Comp Value 0.

01 SQL-Message-Area.
   05 SQL-Text-Len                          Pic S9(9) Comp Value +79.
   05 SQL-Msg.
       10 SQL-Len                           Pic S9(4) Comp Value +632.
       10 SQL-Text                          Pic X(79) Occurs 8.

***-----***
*** Fetch Structure      ***
***-----***
01 WS-columns-in.
>>COLUMN
***-----***
*** Declare Cursor      ***
***-----***
EXEC SQL
    Declare PROC-CS Cursor With Hold For
    Select
>>CURSOR
END-EXEC.

***-----***
*** SQLCA                ***
***-----***
EXEC SQL Include SQLCA      END-EXEC.

```

```

***-----***
Procedure Division.
***-----***
    EXEC SQL
        Whenever NOT FOUND Continue
    END-EXEC.

    EXEC SQL
        Whenever SQLERROR Goto 900-SQLCODE-Error
    END-EXEC.

PERFORM 100-Process-Cursor THRU 100-Process-Cursor-Exit.

Display '** Total Number of Rows Processed: ' Total-cnt.
Display '** Number of Commits Taken:      ' No-Commit-cnt.

***-----***
** Program Exit Point
***-----***
000-Finish.
Goback.

***-----***
100-Process-Cursor.
***-----***
    EXEC SQL
        Open Proc-CS
    END-EXEC.

    EXEC SQL Fetch PROC-CS
        Into :WS-columns-in
    END-EXEC.

PERFORM 200-Fetch-Rows THRU 200-Fetch-Rows-Exit
    UNTIL SQLCode NOT = ZERO.

    EXEC SQL
        Close Proc-CS
    END-EXEC.

100-Process-Cursor-Exit.
Exit.

***-----***
200-Fetch-Rows.
***-----***

>>FETCH

Add 1 to Total-Cnt.
Add 1 to Commit-Cnt.
If commit-cnt > commit-freq
    EXEC SQL
        Commit
    END-EXEC
    Move 1 To Commit-cnt
    Add 1 to No-Commit-Cnt
End-If.

EXEC SQL Fetch PROC-CS
    Into :WS-columns-in
END-EXEC.

200-Fetch-Rows-Exit.
Exit.

***-----***
900-SQLCODE-Error.
***-----***
    Call 'DSNTIAR' Using SQLCA
                        SQL-Msg
                        SQL-Text-Len.

    Display SQL-Text(1).
    Display SQL-Text(2).
    Display SQL-Text(3).
    Display SQL-Text(4).
    Display SQL-Text(5).
    Display SQL-Text(6).
    Display SQL-Text(7).
    Display SQL-Text(8).
    Move 16 To Return-Code.

```

Goback.

APPENDIX

DB2I2 Line Object to DB2I2 Command Cross Reference

The information in *ITALIC* is for Db2 V6 or above only. And the information in *ITALIC* is for Db2 V7 or above only.

DB2I2 Line Object	Available DB2I2 Commands			
AC-Active log line	SETRBA	TAG		
AI-Adjust Index Part line	QBUILD	SPACEADJ		
AL-Alias	AUTH	COAUTH	COPYAUTH	CREATE
	CURSORD	drilldown	DCLGEN	DDL
	DELETE	DSNTIAUL	FETCH	GRANT
	IMPACT	INSERT	MIGR	MT
	PG	PL	QBUILD	REVOKE
	SELECT	TB	TBAUTH	UPDATE
	VW			
AR-Archive Log line	HMIGRATE	HRECALL	SETRBA	TAG
AT-Adjust Table Part line	QBUILD	SPACEADJ		
BP-BufferPool	AUTH	COPYAUTH	DISPLAY	GRANT
	QBUILD	REVOKE	RSAUTH	
CI-Catalog Insert line	SELPATHU			
CL-Collection	AUTH	COPYAUTH	GRANT	QBUILD
	REVOKE	RSAUTH		
CO-Column	ALTER	drilldown	GRANT	PG
	PL	QBUILD	STATS	
CP-system Check Point line	TAG			
CU-Catalog Update line	SELPATHU			
DB-DataBase	ALTER	AUTH	COPYAUTH	CREATE
	Drilldown	DBAUTH	DBDSIZE	DDL
	DISPLAY	GRANT	IMPACT	IP
	IPC	IS	ISC	ISP
	ISPC	IX	IXC	LISTDEF
	MIGR	MT	OI	PG
	PL	QBUILD	REPAIR	REVOKE
	RI	START	STATS	TB
	TBC	TP	TPC	TR
	TS	TSC	TSSET	STOP
DM-DbrM	Drilldown	EXPLAINP	PACKIT	QBUILD
	TOKENSCN			
DS-DataSet	GENVCAT	HMIGRATE	HRECALL	LISTC
	QBUILD	TSIX	VTOC	
DT-Distinct Type	AUTH	COPYAUTH	CREATE	Drilldown
	DDL	GRANT	MIGR	QBUILD
	REVOKE	RSAUTH	TB	
FL-Flashcopy Line	FLASH			
FU-Function	AUTH	COPYAUTH	CREATE	DISPLAY
	Drilldown	DDL	GRANT	IMPACT
	MIGR	PG	PL	QBUILD
	REVOKE	RTAUTH	SQ	START
	TB	VW		
GV-Global Variable	SETG			
IC-Index Column	QBUILD	STATS		

IP-IndexPart	CHECK Drilldown DSN1LOGP ISC IXC QBUILD REORG RUNSTATS TB TS	CPY2CPY_ DISPLAY DSN1PRNT ISP LISTC RBA REORGCHK SNAPSHOT TBC TSC	COPY DSADJ GENVCAT ISPC LISTDEF REBUILD REPAIR START TP	DB DSN1COPY IS IX MT RECOVER REPORT STOP TPC
IPC-IndexPart Clone	CHECK Drilldown DSN1LOGP ISC IXC QBUILD REORG RUNSTATS TB TS	CPY2CPY DISPLAY DSN1PRNT ISP LISTC RBA REORGCHK SNAPSHOT TBC TSC	COPY DSADJ GENVCAT ISPC LISTDEF REBUILD REPAIR START TP	DB DSN1COPY IS IX MT RECOVER REPORT STOP TPC
IS-Index Space	CPY2CPY DISPLAY GENVCAT ISPC LISTDEF REBUILD REPAIR STATS TP	COPY DSN1COPY IP IX MT RECOVER REPORT STOP TPC	DB DSN1LOGP IPC IXC QBUILD REORG SNAPSHOT TB TS	Drilldown DSN1PRNT LISTC RBA REORGCHK START TBC TSC
ISC-Index Space Clone	CPY2CPY DISPLAY GENVCAT ISPC LISTDEF REBUILD REPAIR STATS TP	COPY DSN1COPY IP IX MT RECOVER REPORT STOP TPC	DB DSN1LOGP IPC IXC QBUILD REORG SNAPSHOT TB TS	Drilldown DSN1PRNT LISTC RBA REORGCHK START TBC TSC
ISP-Index Space Part	CPY2CPY DISPLAY DSN1PRNT IS LISTC RBA REORGCHK START TBC TSC	COPY DSADJ GENVCAT ISPC LISTDEF REBUILD REPAIR STATS TP	DB DSN1COPY IP IX MT RECOVER REPORT STOP TPC	Drilldown DSN1LOGP IPC IXC QBUILD REORG SNAPSHOT TB TS
ISPC-Index Space Part Clone	CPY2CPY DISPLAY DSN1PRNT IS LISTC RBA REORGCHK START TBC TSC	COPY DSADJ GENVCAT ISPC LISTDEF REBUILD REPAIR STATS TP	DB DSN1COPY IP IX MT RECOVER REPORT STOP TPC	Drilldown DSN1LOGP IPC IXC QBUILD REORG SNAPSHOT TB TS

IX-Index	ALTER CREATE DISPLAY GENVCAT IS LISTC PG REBUILD REPAIR SPACE TB TS	CHECK Drilldown DSN1COPY IMPACT ISC LISTDEF PL RECOVER REPORT START TBC TSC	CPY2CPY DB DSN1LOGP IP ISP MIGR QBUILD REORG RUNSTATS STATS TP	COPY DDL DSN1PRNT IPC ISPC MT RBA REORGCHK SNAPSHOT STOP TPC
IXC-Index Clone	ALTER CREATE DISPLAY GENVCAT IS LISTC PG REBUILD REPAIR SPACE TB TS	CHECK Drilldown DSN1COPY IMPACT ISC LISTDEF PL RECOVER REPORT START TBC TSC	CPY2CPY DB DSN1LOGP IP ISP MIGR QBUILD REORG RUNSTATS STATS TP	COPY DDL DSN1PRNT IPC ISPC MT RBA REORGCHK SNAPSHOT STOP TPC
JA-Java	Drilldown	GRANT	REVOKE	
JI-Job Information	SDSF			
MS-Column Mask	ALTER MIGR	CREATE	DDL	Drilldown
MT-Material query Table	AL CREATE DCLGEN FETCH INSERT IX PL SP TP VW	AUTH CURSORD DDL FU IP MIGR QBUILD SY TS	COAUTH Drilldown DELETE GRANT IS MT REVOKE TB UNLOAD	COPYAUTH DB DSNTIAUL IMPACT ISP PG SELECT TBAUTH UPDATE
OI-Object ID	Drilldown			
PG-PackaGe	AL COPYAUTH GRANT PL SP TB VW	AUTH Drilldown IX QBUILD SQ TOKENSCN EXPLORE	BIND FREE MT REBIND STATS TR	BIND COPY FU PGAUTH REVOKE SY TS
PL-Plan	AL Drilldown IX PLAUTH SP TB	AUTH FREE MT QBUILD SQ TS	BIND FU PACKIT REBIND STATS VW	COPYAUTH GRANT PG REVOKE SY
PM-Row Permission	ALTER MIGR	CREATE	DDL	Drilldown
RI-Referential Integrity line	Drilldown	DDL	MIGR	QBUILD
RL-Archive Log line	SETRBA	TAG		

RO-Role	AUTH DBAUTH PLAUTH RTAUTH TRAUTH	COAUTH DrillDown QBUILD SHAUTH USAUTH	COPYAUTH GRANT REVOKE SQAUTH	CREATE PGAUTH RSAUTH TBAUTH
SC-tableSpace Copy line	DSCOPY	QBUILD		
SG-StoGroup	ALTER Drilldown QBUILD	AUTH DDL REVOKE	COPYAUTH GRANT RSAUTH	CREATE MIGR STOSPACE
SH-schema line	AUTH QBUILD	COPYAUTH REVOKE	Drilldown SHAUTH	GRANT
SP-stored procedure line	AUTH DDL MIGR REVOKE	COPYAUTH DISPLAY PG RTAUTH	CREATE GRANT PL START	Drilldown IMPACT QBUILD STOP
SQ-sequence line	AUTH DDL PG SQAUTH	COPYAUTH GRANT PL	CREATE IMPACT QBUILD	Drilldown MIGR REVOKE
SY-Synonyms	CREATE DDL IMPACT PG TB	CURSORD DELETE INSERT PL UPDATE	Drilldown DSNTIAUL MIGR QBUILD VW	DCLGEN FETCH MT SELECT
TB-TaBle	AL COPYAUTH DB DSNTIAUL GRANT IPC ISPC LOAD PG RI SPACE TP TSC	ALTER CREATE DCLGEN DT IMPACT IS IX MIGR PL SELECT STATS TPC UNLOAD	AUTH CURSORD DDL FETCH INSERT ISC IXC MT QBUILD SELPATHV SY TR UPDATE	COAUTH Drilldown DELETE FU IP ISP LISTDEF OI REVOKE SP TBAUTH TS VW
TBC-TaBle Clone	AL COPYAUTH DB DSNTIAUL GRANT IPC ISPC LOAD PG RI SPACE TP TSC	ALTER CREATE DCLGEN DT IMPACT IS IX MIGR PL SELECT STATS TPC UNLOAD	AUTH CURSORD DDL FETCH INSERT ISC IXC MT QBUILD SELPATHV SY TR UPDATE	COAUTH Drilldown DELETE FU IP ISP LISTDEF OI REVOKE SP TBAUTH TS VW
TC-Trusted Context	CREATE	DrillDown	DDL	MIGR

TP-TablePart	CHECK DB DSN1LOGP IPC ISPC MT RBA REPAIR START TBC	COPY DISPLAY DSN1PRNT IS LISTC OI RECOVER REPORT STOP TS	CPY2CPY DSADJ GENVCAT ISC LISTDEF QBUILD REORG RUNSTATS STATS TSC	Drilldown DSN1COPY IP ISP MODIFY QUIESCE REORGCHK SNAPSHOT TB UNLOAD
TPC-TablePart Clone	CHECK DB DSN1LOGP IPC ISPC MT RBA REPAIR START TBC	COPY DISPLAY DSN1PRNT IS LISTC OI RECOVER REPORT STOP TS	CPY2CPY DSADJ GENVCAT ISC LISTDEF QBUILD REORG RUNSTATS STATS TSC	Drilldown DSN1COPY IP ISP MODIFY QUIESCE REORGCHK SNAPSHOT TB UNLOAD
TR-Trigger line	AUTH IMPACT TB	CREATE MIGR TRAUTH	Drilldown PG	DDL QBUILD
TS-Tablespace	ALTER CPY2CPY DDL DSN1PRNT IP ISP LISTC MT QBUILD REORG REVOKE START TBC UNLOAD	AUTH CREATE DISPLAY GENVCAT IPC ISPC LISTDEF OI QUIESCE REORGCHK RSAUTH STATS TP	CHECK Drilldown DSN1COPY GRANT IS IX MIGR PG RBA REPAIR RUNSTATS STOP TPC	COPY DB DSN1LOGP IMPACT ISC IXC MODIFY PL RECOVER REPORT SNAPSHOT TB TSSET
TSC-Tablespace Clone	ALTER CPY2CPY DDL DSN1PRNT IP ISP LISTC MT QBUILD REORG REVOKE START TBC UNLOAD	AUTH CREATE DISPLAY GENVCAT IPC ISPC LISTDEF OI QUIESCE REORGCHK RSAUTH STATS TP	CHECK Drilldown DSN1COPY GRANT IS IX MIGR PG RBA REPAIR RUNSTATS STOP TPC	COPY DB DSN1LOGP IMPACT ISC IXC MODIFY PL RECOVER REPORT SNAPSHOT TB TSSET
US-User	AUTH GRANT REVOKE SQAUTH	COAUTH PGAUTH RSAUTH TBAUTH	COPYAUTH PLAUTH RTAUTH TRAUTH	DBAUTH QBUILD SHAUTH USAUTH
VL-VoLume	QBUILD			

VW-View	AL	AUTH	COAUTH	CREATE
	CURSORD	Drilldown	DCLGEN	DDL
	DELETE	DSNTIAUL	FETCH	FU
	GRANT	IMPACT	INSERT	MIGR
	MT	PG	PL	QBUILD
	REVOKE	SELECT	SP	SY
	TB	TBAUTH	UPDATE	VW
XC-index Copy line	DSCOPY	QBUILD		

DB2I2 Command to Line Objects Cross Reference

DB2I2 Command	Input Line Object	Output Type ³
Drill down	DB,TS, TB, AL, SY, VW, IX, PG, PL, SG, DM, TP, IP, DS, RI, DT,TR, SP, FU SH, IS, ISP, MT, OI, SQ, TSC, TPC, TBC, IXC, ISC, IPC, ISPC, TC, RO	Line Object
AL	TB, VW, PG, PL, MT, TBC	Line Object - AL
ALTER	DB,TS, TB, IX, SG, SP, FU, SQ, CO, MS, PM, A display UTILITY line with UTILID = utility-id	DDL - ALTER
AUTH	AL, BP, CL, DB, PG, PL, SG, TS, TB, US, VW, DT, SH, FU, SP, TR, SQ, MT, RO, TBC	Report
BATCH		JCL
BIND	PG, PL	DB2 Command - BIND
BIND COPY	PG	DB2 Command - BIND COPY
CANCEL [DDF] THREAD	A display thread line with Numeric Token as the last field on the line	
CHECK	TS, IX, TP, IP, TSC, IXC, TPC, IPC	JCL
COAUTH	AL, TB, US, VW, MT, RO	Report
CONNECT		
CONNECT(RESET)		
COPY	TS, TP, IX, IP, IS, ISP, TSC, TPC, IXC, IPC, ISC, ISPC	JCL
COPYAUTH	US, RO, AL, BP, CL, DB, DT, FU, MT, PG, PL, SG, SH, SP, SQ, TB, TBC, TS, VW	DCL - GRANT
<u>CPY2CPY</u>	TS, TP, IX, IP, IS, ISP, TSC, TPC, IXC, IPC, ISC, ISPC	JCL
CREATE	AL, DB, DT, FU, IX, SG, SP, SY, TB, TR, TS, VW, MT, SQ, RO, TC	DDL
CURSORD	TB, AL, SY, VW, MT, TBC	Cobol
DB	TS, TP, IX, IP, IS, ISP, TB, MT, TSC, TPC, IXC, IPC, ISC, ISPC, TBC	
DBAUTH	DB, US, RO	Report
DBDSIZE	DB	Report
DB2CMD	DB2 command lines	Report
DCLGEN	TB,AL,VW, SY, MT, TBC	DB2 Command - DCLGEN
DDL	AL, DB, TS, TB, IX, SY, VW, SG, RI, DT, SP, FU, TR, SQ, MT, TBC, TC, MS, PM	DDL - CREATE
DELETE	TB,AL,SY,VW, MT, TBC	SQL - DELETE
DISPLAY	DB, TS, IX, TP, IP, BP, SP, FU, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC	Report
DSADJ	TP, IP, ISP, TPC, IPC, ISPC	JCL
DSCOPY	SC,XC	JCL
DSNJU004		Report
DSNTEP2	SQL blocks	Report
DSNTIAD	SQL blocks	Report
DSNTIAUL	TB,VW, AL, SY, SQL blocks, MT, TBC	JCL
DSN1COPY	TS, IX, TP, IP, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC	JCL
DSN1LOGP	TS, IX, TP, IP, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC	JCL
DSN1PRNT	TS, IX, TP, IP, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC	JCL
DT	TB, TBC	Line object DT
ED		
EDIT		
EXEC	DB2I2 scripts	Report
EXPLAIN	SQL statements block or DBRM statements block	Report
EXPLAINP	PL, PG, DM	Report

⁵. Only Line object Output Type can be Reusable

<u>EXPLORE</u>	PG	DB2 command – REBIND or reusable explain report
FETCH	TB, AL, SY, VW, MT, TBC	Cobol
FLASH	FC	JCL with dfsmsDSS fast replication
FLIST		Report
FREE	PG, PL	DB2 Command – FREE
FTP		Process FTP
FU	TB,VW,PL,PG, MT, SQ	Line object FU
GENVCAT	TS, TP, IX, IP, DS, IS, ISP, TSC, TPC, IXC, IPC, ISC, ISPC	IDCAM command - CREATE
GRANT	AL, BP, CL, CO, DB, PG, PL, SG, TS, TB, US, VW, DT, FU, SP, SH, TR, MT, SQ, RO, TBC	DCL – GRANT
HELP		
HELPLO		
HMIGRATE	DS, AR	
HRECALL	DS, AR	
IMPACT	AL, DB, TS, TB, IX, SY, VW, DT, FU, SP, TR, MT, SQ, TBC	Line Object
INFO		
INSERT	TB, AL, SY, VW, MT, TBC	SQL - Insert
IP	DB, TS, TB, MT, IX, IS, ISP, TP, TSC, TBC, IXC, ISC, ISPC, TPC	Line Object - IP
IPC	DB, TS, TB, MT, IX, IS, ISP, TP, TSC, TBC, IXC, ISC, ISPC, TPC	Line Object - IPC
IS	DB, TS, TB, MT, IX, IP, ISP, TP, PL, PG, TSC, TBC, IXC, IPC, IPSC, TPC	Line object IS
ISC	DB, TS, TB, MT, IX, IP, ISP, TP, PL, PG, TSC, TBC, IXC, IPC, ISPC, TPC	Line Object - ISC
ISP	DB, TS, TB, MT, IX, IP, IS, TP, TSC, TBC, IXC, IPC, ISC, TPC	Line object ISP
ISPC	DB, TS, TB, MT, IX, IP, IS, TP, TSC, TBC, IXC, IPC, ISC, TPC	Line object ISPC
IX	DB, TS, TB, PL, PG, IS, ISP, TSC, TBC, ISC, ISPC	Line Object - IX
IXC	DB, TS, TB, PL, PG, IS, ISP, TSC, TBC, ISC, ISPC	Line Object - IXC
JOBCARD		
LISTC	TS, IX, TP, IP, DS, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC	Report or Line Object
LISTDEF	DB, TS, TP, IX, IP, TB, IS, ISP, MT, TSC, TPC, IXC, IPC, TBC, ISC, ISPC	JCL
LOAD	TB, MT, TBC	JCL
MAIL		Send Email thru SMTP
MIGR	AL, DB, TS, TB, IX, SY, VW, SG, RI, DT, FU, SP, TR, MT, SQ, TBC, TC, MS, PM	DDL, DCL, IDCAM commands, DB2 commands
MODIFY	TS, TP, IX, IS, TSC, TPC	JCL
MS	TB	MS line object
MT	DB, TS, TP, TB, IX, IP, MT, AL, VW, SY, PL, PG, DT, FU, SP, TR, SQ, IS, ISP	MT line object
OI	DB, TS, TB, IX, MT, IX	Dbid,obid
OPTIONS		Utility Option file
PACKIT	DM, PL	DB2 Commands - BIND
PARMUTIL		DB2I2 Parmutil File
PG	TS, AL, TB, SY, VW, IX, PL, DB, SP, FU, TR, MT, SQ, PG, CO, TSC, TBC, IXC	Line Object - PG
PGAUTH	PG, US, RO	Report
PL	TS, AL, TB, SY, VW, IX, PG, DB, SP, FU, TR, MT, SQ, PL, CO, TSC, TBC, IXC	Line Object - PL
PLAUTH	PL, US, RO	Report
PM	TB	PM line object
QBUILD	any valid line objects except ,AC, AR, CI, CO, CP, CU, GV, IC, RL	SQL WHERE predicates
QUIESCE	TS, TP, TSC, TPC	JCL
RBA	TS, TP, IX, IP, IS, ISP, TSC, TPC, IXC, IPC, ISC, ISPC	SYSCOPY RBA line object
REBIND	PG, PL	DB2 Command - REBIND
REBUILD	TS, IX, TP, IP, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC	JCL
RECOVER	TS, IX, TP, IP, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC	JCL

REORG	TS, IX, TP, IP, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC	JCL
REORGCHK	TS, IX, TP, IP, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC	
REPAIR	DB,TS, IX, TP, IP, IS, ISP, TSC, IXC, TPC, IPC, ISC, ISPC	JCL
REPORT	TS, TP, IX, IP, IS, ISP	JCL
RESETG		
REVOKE	AL, BP, CL, DB, PG, PL, SG, TS, TB, US,VW, DT, FU, SP, SH, MT, SQ, RO, TBC, JA	DCL - REVOKE
REXX		
RI	TB, DB	Line Object - RI
RSAUTH	BP, CL, SG, TS, US, DT, RO	Report
RTAUTH	FU, SP, US, RO	Report
RUN	SQL SELECT statement	Report or Line Object
RUNSTATS	TS, IX, TP, IP	JCL
<i>RXDB2I2</i>	<i>Any valid rexx or db2i2 command together with LINEOBJ and END_LINEOBJ</i>	
SDSF	JI	
SELECT	TB, AL, SY, VW, MT, TBC	SQL - SELECT
SELPATHU	CI, CU	SQL - UPDATE or INSERT
SELPATHV	TB, MT	Line object - TB
SETG	GV	
SETRBA	Any edit line with valid RBA field (12 position Hex)	
SHAUTH	SH,US, RO	Report
SNAPSHOT	TS, TP, IX, IP, IS, ISP, TSC, TPC, IXC, IPC, ISC, ISPC	Report
SP	TB,VW, PL, PG, MT, SQ, TBC	Line object SP
SPACE	TB, IX	Report
SPACEADJ	AI, AT	TP or IP line object with ALLOC option
SQ	FU	SQ Line object
SQAUTH	SQ,US, RO	Report
SSID(ssid)		
START	DB, SP, FU, TS, IX, TP, IP, IS, ISP, TSC, TPC, IXC, IPC, ISC, ISPC	Report
STATS	DB, TS, TB, IX, TP, IP, SP, ISP, PL, PG, CO, IC	Report or Line object
STATUS	Display output with pageset-status	PageSet Status information
STOP	DB, TS, IX, TP, IP, IS, ISP, TSC, TPC, IXC, IPC, ISC, ISPC	Report
STOSPACE	SG	JCL for DB2 STOSPACE
SUPERC		Report
SUPERCS		Report
SUPPORT		Communicate with DB2I2 Support team
SY	TB, VW, PL, PG, MT, TBC	Line Object - SY
SYSIBM(creator)		
TAG	syscopy_line, AC,AR,CP,RL log dataset/RBA range line	Tag the line
TB	DB, TS, IX, IP, TP, IS, ISP, PL, PG, AL, VW, SY, DT, FU, TR, SP, MT, SQ, IXC, IPC, TSC, TPC, ISC, ISPC	Line Object - TB
TBAUTH	AL,VW,TB, MT, TBC	Report
TBC	DB, TS, IX, IP, TP, IS, ISP, PL, PG, AL, VW, SY, DT, FU, TR, SP, MT, SQ, IXC, IPC, TSC, TPC, ISC, ISPC	Line Object - TBC
TEMPLATE		Utility Template file
TERM	A display utility line with UTILID=	
TOKENSCN	PG, DM	Report
TP	DB, TB, TS, IP, IX, IS, ISP, MT, TBC, TSC, IPC, IXC, ISC, ISPC	Line Object - TP
TPC	DB, TB, TS, IP, IX, IS, ISP, MT, TBC, TSC, IPC, IXC, ISC, ISPC	Line Object - TPC
TR	DB, TB, PG	Line object - TR
TRACK	JI	Extract SDSF output
TRAUTH	TR, US, RO	Report
TS	DB, TB, TP, IP, IX, IS, ISP, PL, PG, MT, TBC, TPC, IPC, IXC, ISC, ISPC	Line Object - TS

TSC	DB, TB, TP, IP, IX, IS, ISP, PL, PG, MT, TBC, TPC, IPC, IXC, ISC, ISPC	Line Object - TSC
TSIX	DS	Line Object - TP, IP
TSO		
TSSET	DB, TS	Line Object - TS
UCASE		Upper Case
UNLOAD	TS, TP, TB, MT, TSC, TPC, TBC	JCL
UPDATE	TB, AL, SY, VW, MT, TBC	SQL UPDATE
USAUTH	US, RO	Report
VIEWG		GV line object
VIRTUAL		SQL INSERT
VTOC	DS	IEHLIST LISTVTOC
VW	TB, VW, AL, SY, PL, PG, FU, SP, MT, TBC	Line Object - VW
ZPARM		Report

INDEX

- #
- #PART191, 193
- %
- % Global option.....22
- %=###22
- %=###84
- %=###89
- %=###147
- %=###190
- &
- &hostvar115
- *
- *MYMENU119
- ?
- ? 41, 44, 151
- {
- { substitute111
- {REXX_VAR}111
- }
- } substitute111
- <
- <%=###>21
- <ALLOC>21
- <NEWJOB>116, 124, 177
- A**
- ACCCOMP.....334, 406
- ACCCOMPR336, 406
- Access Path Comparison406
- ACTL.....306
- AL27, 46
- ALLOC.....147, 190
- alloc_type84, 89, 147
- ALLOC=.....29, 84, 89, 147
- ALTER27, 47
- ALTER Utility47
- APPEND.....22
- Assign different user in batch mode.....52
- AUTH.....27, 49
- B**
- BATCH.....27, 50
- BATCH ADVANCED FUNCTION52
- BATCH DDNAME description.....52
- BIND27, 56
- BIND COPY.....27, 58
- BOTTOM**114
- Bufferpool Statistics Capturing and reporting405
- C**
- Callable Interface.....51, 411
- CANCEL [DDF] THREAD.....60
- CAPTBUFR307, 405
- Capture bufferpool Information.....307
- Case Study391
- CHECK.....28, 61
- CLI.....27, 51
- CLINE112
- CLINE NEXT112
- CLINE PREV112
- CLINE WRITE.....112
- COAUTH28, 63
- CODEGEN437
- Comment on Command Line18
- Comment on Line Object.....18
- CONNECT28, 64
- COPY28, 66
- Copy DB2 Catalog Statistics409
- COPYAUTH28, 70, 71
- COPYSTAT347, 349, 352, 409
- CPOS112
- CPY2CPY.....72
- CREATE.....28, 75
- CURRENT31
- CURSORD28, 76
- Customize Global Variable.....25
- CYL22
- CYL147
- D**
- DB28, 77
- DB2CMD28, 80
- DB2I2 Architect Flow12
- DB2I2 Command Summary with Line Object and
Output Type.....453
- DB2I2 ED Macro interface.....113
- DB2I2LOG307, 431
- DB2I2REX246
- DB2I2RXL**246
- DB2I2TRK308
- DBAUTH28, 78
- DBDSIZE28, 79
- DCLGEN28, 82
- DDL29, 84
- DDLBUFR.....405
- DDLCOPYS424

DDLREORG227
 DDLTRACK308
 DELETE112
 DELSKEL426
 DFLTSP.....22, 61, 91, 182, 217, 219, 223
 Disaster Recovery430
 DISPLAY87
 DISTINCT114, 124
 DLM=241
 DR430
 Drill Down.....41
 DSADJ.....29, 89
 DSCOPY29, 91
 DSN1COPY29, 101
 DSN1LOGP.....29, 105
 DSN1PRNT.....30, 108
 DSNJU004.....29, 95
 DSNTDP2.....29, 97
 DSNTIAD.....29, 98
 DSNTIAUL29, 99, 100
 DT.....30, 109
 Dynamic select SQL.....36

E

ED.....30, 110
 EDIT30, 124
END_LINEOBJ246
 EOF112, 113
 EOLOOP113
 ERROR(CONTINUE).....22
 ERROR(SKIP #).....22
 ERROROFF.....113
 EXEC.....126
 EXPLAIN128
 EXPLAINP132
 EXPLORE135
 EXT177

F

FCPY306
 FETCH30, 136
 FGET115, 117
 File Extracting310
 File Transform310
 FLIST137, 140
 format of GENAI.....264
 format of GENAT264
 FPUT115
 FREE31, 141
 FU31

G

GENAI.....264
 GENAI.....361
 GENAT.....264
 GENAT.....362

GENSC363
 GENSCCMD355
 GENUNLD1365
 GENUNLDQ364, 433
 GENURLD337, 432
 GENVCAT31, 146, 147
 GENXC .366, 368, 370, 373, 376, 378, 382, 385, 388
 GETG113
 Global Command Options22
 Global Line Object option21
 Global Variable113, 114, 234, 255, 264, 327, 328, 330
 Global Variable and Host Variable Usage.....25
 Global Variable Definition25
 GRANT31, 149

H

HELP32, 151
 HELPLO.....32, 161
 HIARBA.....89, 147
 HIURBA.....89, 147
 HMIGRATE32, 162
 Host Variable Definition.....25
 Host Variable Substitution.....27, 36, 50, 241
 HOSTVAR41
 How to invoke DB2I2.....14
 HRECALL.....32, 95, 163

I

ICDATE92
 ICGEN92
 ICMD.....27, 50, 52, 394
 ICMD Limitation52
 ICOPI92
 IDD22
 IDSN.....22
 IDSN*.....50
 IDSN=*DUMMY50
 IMPACT32, 164
 IN.....241
 -INC.....241
 INCORE RBA37, 219, 256
 INFO.....32, 165
 INSERT32, 166
 INSERTA111
 INSERTB111
 INSSKEL.....426
 IP 32, 168, 169
 IS 32, 170, 171
 ISP32, 172, 173
 IX.....32, 174, 175

J

JCL=N27, 50
 Job Tracking431
 JOB Tracking facility308

JOB#.....22
 JOBCARD.....16, 32, 176
 JOBCARD=N.....22
 JOBGEN.....309
 JOBNM.....22

L

LAST_RC.....113
 LIMIT.....241
 line object.....15
 Line Object.18, 20, 27, 28, 29, 30, 31, 32, 33, 34, 35,
 36, 37, 38, 39, 40
 Line Object to DB2I2 Command Cross Reference 447
LINEOBJ.....246
 LISTC.....32, 177
 LISTDEF.....32, 180
 LNOBJ.....110
 LOAD.....33, 182
 LOAD cache-name.....111
 LOOP.....113
 LOOP_END.....113
 LP.....113

M

MACRO.....30
 MACRO(*MYMENU).....110
 MAPTABLE=.....223
 MAXSZ.....89
 MC.....51
 MCCLI.....411
 MIGR.....190
 Migration scripts.....190
 MIGRTB.....356, 394
 MODIFY.....33, 195
 MONBUFR.....358, 405
 MT.....33, 197, 198
 MYMENU.....110, 119

N

NEWJOB.....21, 23, 114
 NEXTLINE.....113
 NOREC=.....115
 NOSTATS.....41
 NOSTOPS.....92
 NOTFOUND(CONTINUE).....23
 NOTFOUND(SKIP #).....23

O

OBJCOMP.....339, 403
 ODSN.....23
 ODSN*.....50
 OI.....33, 199
 Optional Add-ons.....405
 Output Type.....453
 OVERRIDE.....33, 182
 OVRD.....89, 227

OVRD=.....223

P

P000710.....308, 417
 PACKIT.....33, 201
 PARMUTIL.....34, 203
 Partition Assist.....193
 Partition Assistance.....191
 PERD.....306
 PG.....34, 205
 PGAUTH.....34, 206
 PL.....34, 207
 PLAUTH.....34, 208, 209
 Point of Time Recovery.....415
 POS=.....115
 POSTMIGR.....114, 191
 POTRCVR.....421
 POTRCVRD.....422
 POTRCVRT.....424
 Preparing JCL for Production.....309
 Prototype.....409
 PTX=partition-index-name.....191, 193

Q

QBUILD.....34, 210
 QUIESCE.....34, 212

R

RBA.....34, 213
 RBLDIX.....219
 RCHK.....223
 RCVRIX=N.....92
 READBUFR.....307
 REBIND.....34, 215
 REBUILD.....34, 217
 RECOVER.....35, 219
 REMOVEID.....341, 408
 REORG.....35, 223
 REORGCHK.....35, 223, 227
 REORGCHK criteria.....227
 REPAIR.....35, 229
 REPORT.....35, 231
 Report to Recovery.....308, 417
 RESETG.....35, 234
 RESIZEIT.....116, 124
 Restart after EXEC Abend.....126
 Restart in Batch.....52
 RESUME.....182
 REVOKE.....35
 REXX.....35, 112, 237
 rexx Variable.....113
 RI.....36, 238
 RPTRDD.....308, 417
 RSAUTH.....36
 RTAUTH.....36, 240
 RUN.....36, 241

RUNSTATS36, 244
RXDB2I2.....246

S

Sample Scripts347
SDSF.....36, 250
SELECT36, 252
SELPATHU.....36, 253
SELPATHV.....36, 254
SET GLOBAL.....36
SETG.....36, 113, 114, 255, 264
SETG2.....114
SETRBA.....37, 95, 117, 219, 256
Setup.....14
SHAUTH.....37, 257
SIGNAL FINISH.....113
SIZE.....177
SKIP23
SKIP #113
SNAPSHOT37, 258
SORTKEYS92
SP.....37, 259
SPACE.....37, 260
Space Adjustment264
Space Estimation260
SPACEADJ37, 264
Spool Job Output Queue.....250
SQ.....37, 269
SQAUTH.....37, 270
SQL length > 32760 bytes long315
SQLID23
SQLTERM30
SSID271
SSID(ssid).....37
START37, 272
STATS37, 273
STEP#23
STOP37, 276, 277, 278
SUPERC.....37, 38, 279, 280, 281
SY38, 282
Sysadm ID Removal408
SYSIBM38, 283
SYSIN=182
SYSPUNCH=99, 223
SYSREC=99, 182, 223

T

T=N23
TAG.....38, 95, 219, 284
TB.....38, 286, 288
TBAUTH.....38, 287
TEMPLATE38, 289
TERM.....291
Time Event Wait.....306
TO72.....315

TOKENSCN.....38, 292
TOP114
TP38, 294, 295
TPTOSQL426
TR.....38, 296
TRANSFRM.....310
TRAUTH.....297, 302
TRK22
TRK147
TS39, 303, 304
TSIX39, 177, 305
TSO39, 306
TSO CAPTBUFR307
TSO DB2I2LOG307
TSO DB2I2TRK.....308
TSO FCPY.....306
TSO JOBGEN309
TSO P000710308
TSO RAEDBUFR307
TSO TO72315
TSO TWAIT.....306
TSSET39, 318
TUNEPG342, 397
TUNETB345, 400
TWAIT306, 405

U

UCASE39, 319
UDF52, 334
UDQ361
UNLDRELD.....359
UNLOAD39, 320
Unload and Reload432
UPDATE39, 324
USAUTH.....39, 326
User defined function.....50
User Defined Functions334
User Defined Query361
User ID Removal408
user setup14
UTILCLI.....412

V

VIEWG.....39, 327, 328, 330
VW40, 331

W

Wildcard Line Object20
WKSP24
Work Load Balancing.....393
WSDSN112

Z

ZPARAM40, 332

