HPSS 7.3.1 Metadata Conversion Guide

High Performance Storage System
Release 7.3

May 2010 *(Revision 1.0)*

HPSS Release 7.3
May 2010 (Revision 1.0)

Table of Contents

Chapter 1. UDA Metadata Conversion

1.1. Introduction

The following chapter describes the database schema changes that must be made to convert an HPSS 7.1 system to an HPSS 7.3 system.

The USERATTRS table provides metadata storage space for the User Defined Attributes feature which was added to HPSS in version 7.3.

Conversion of an HPSS 7.1 system to version 7.3 is done by performing the following steps:

1. Add the USERATTRS table to all subsystem databases
2. Adjust schema settings associated with the NSOBJECT table in all subsystem databases
3. Add Relational Integrity constraints associated with the USERATTRS table to all subsystem database schemata.

The schema changes are backward compatible with HPSS 7.1, so these changes can be made well in advance of moving a production system to 7.3. By the same argument it may be possible to move back to a 7.1 system from 7.3, although this has not been tested.

Changes to a subsystem database schema must be performed when HPSS is not running. All system servers including the System Manager and the Startup Daemon must be halted before making changes.

The program that makes the changes is **hpss_managetables**. This program is used by MKHPSS to create database tables and indexes when new systems are created. It is designed to be operated by both MKHPSS and by an HPSS administrator.

It is vitally important to understand that this program, when used from a command line by an administrator, has the ability to destroy an HPSS system. It can create and delete any database table in an HPSS system. It can create and delete any index, view or RI constraint on any database table. It should also be understood that this program alone is the source of the HPSS database schema. Therefore, it will be necessary to use this program from time to time as changes to the schema are required.

hpss_managetables is a keyboard interactive program that operates in a "manual commit" mode. This means that database transactions are committed manually from the keyboard, not automatically. The administrator uses the program to make a connection to the database, then issues commands that make changes to the database. When the connection is made, the program creates an empty database transaction. When any command is issued that makes a change to the database, the change is contained in the transaction and remains uncommitted. Additional commands can be issued which are added to the transaction. The transaction remains uncommitted until a "commit" command is entered at which time the changes are committed to the database. If an "abort" command is entered instead, the changes are rolled back and no net change takes place. If an uncommitted transaction exists when the "quit" command is entered, the program reports the existence of the transaction and will not quit.

If you follow the following instructions carefully and refrain from experimenting with **hpss_managetables**, you won't do any damage to your system. If something unexpected happens, use the "abort" command and "quit" and seek advice.

hpss_managetables must be run by a user ID that has DBADM authority on the databases. We recommend running as "hpssdb". The program can be found in the HPSS "bin" directory.

hpss_managetables recognizes two environment variables, **HPSS_MM_SCHEMA_NAME** and **HPSS_GRP_NAME_SERVER**. Both of these variables have appropriate defaults set by HPSS. **HPSS_MM_SCHEMA_NAME** is set to "HPSS" and **HPSS_GRP_NAME_SERVER** is set to "hpsssrvr". If for some reason these values don't apply to your system, adjust them in the environment in which **hpss_managetables** will be run, or in the *env.conf* file before starting the program.

1.2. Conversion Procedure

This conversion is performed on the subsystem database. These instructions should be repeated for each subsystem database in your HPSS system. The global (or "cfg") database is not changed in this conversion. These are the steps:

Start **hpss_managetables**:

```
$ hpss_managetables
```

```
hmt>
```

Connect to the database to be modified. In this example we will use “subsys1”:

```
hmt> db subsys1
```

The program will connect to the database and show you information about the connection. Take this opportunity to verify that the database and schema names are correct.

Create the **USERATTRS** table:

hmt> add userattrs

The program will create the **USERATTRS** table and show you the SQL commands it executed. If there are no errors, commit this change to the database:

```
hmt> commit
```

Next, we want to add a Relational Integrity constraint to the schema, binding the **NSOBJECT** and **USERATTRS** tables together. But to to do this, it may be necessary to make a small change to the settings on the **NSOBJECT** table. Most systems that were created before HPSS 7.1 will need this change. Make this change with this command:

```
hmt> add indexes nsobject
```

This command will run four SQL statements on the **NSOBJECT** table. Some of these statements will attempt to create indexes that already exist. The program intercepts this error and displays a message telling you that the index was not created and that this situation is not considered to be an error. Any real errors are clearly described.

One of the four SQL commands adds a “primary key” to the **NSOBJECT** table. Adding the primary key to the table is a change we need to make in order to take the next step. In most cases this change will occur with no errors. If there is a problem, it will be clearly spelled out by hpss_managetables. If an error occurs, “abort” the transaction, “quit” the program and get help.

Now we are ready to add the Relational Integrity constraint:

```
hmt> add subsys constraints
```

Again, a number of SQL commands will be issued. Many will be followed by a comment from hpss_managetables indicating that the constraint already existed. The last constraint, **NSOBJECTUSERATTRS1** should be applied without an error or comment.

Commit your work and terminate the program:

```
hmt> commit
....
hmt> quit
```

Repeat these steps for each subsystem database in your HPSS system, altering the "db" command to the appropriate subsystem database name.

Chapter 2. Mover Metadata Conversion

2.1. Introduction

The Multi-Homed Passive Mover Support feature requires a conversion of the Mover specific configuration table. This conversion is only required for 7.3.1 systems and must be performed before starting any Movers.

The procedure for running this conversion has been simplified into a single script.

2.2. Conversion Procedure

To run this conversion you must first compile the 7.3.1 source, ensure that HPSS is not running, and that DB2 is running.

Next, run the conversion script as root by simply doing the following:

```
% /opt/hpss/tools/convert71/hpss_convert_73_mvr_config
```

Run the script and note any errors. If something unexpected happens and you get an error, you will have to seek advice. Or, if you are unsure as to how to interpret the output seek advice from HPSS support.

Chapter 3. Disk Storage Segment Extents Metadata Conversion

3.1. Introduction

PTR 7786 introduced a change to the format of the STORAGESEGDISKEXTENTS table. This PTR is a part of the 7.1.2 and 7.3.1 HPSS releases. To run these systems, the STORAGESEGDISKEXTENTS table must be converted to a new format. This conversion must be performed on all existing systems on which the 7.1.2, 7.3.1 or following releases are installed, even if the system in question does not have a disk cache.

The basics of the conversion are simple. The existing STORAGESEGDISKEXTENTS table is renamed, a new table is created in the new format, the contents of the old table are copied to the new table, and the 7.1.2 or 7.3.1 system is started. If goes well, the old table is removed.

3.2. Conversion Preparation

The STORAGESEGDISKEXTENTS table can be a large table. If the installation in question has a large disk cache, this table will have at least one row for each file in the cache. Installations with a small or non-existant disk cache will have short or empty tables. Before performing this conversion, consider the amount of free space in the tablespace that contains the table. It may be necessary to expand that tablespace. Since expanding a tablespace can be a task that is highly specific to the installation, this document does not describe that process.

The conversion process is designed so that if a problem should occur, the conversion can be aborted and the metadata put back the way it was. The original metadata table is retained. This requires that the system source code be put back to the previous version as well. Should this become necessary, you will have to seek advice – the exact procedure to put things back as they were depends on what failed.

The following conversion procedures can be used on either 7.12 or 7.3.1 systems.

3.3. Conversion Procedure

3.3.1. Verify Prerequisite Space

The first step is to validate the free space in the impacted tablespaces. At a minimum you need at least 50% free space in STORAGESEGDISK and INDEXES tablespaces. The temporary tablespace (usually TEMPSPACE1) should have sufficient space allocated to construct the indexes for the new table (this will vary by site).   The script to verify this is named "check_tablespaces.sh" and it has three arguments:

```
     check_tablespaces.sh [-d subsystem database name] [-s
schema name] [-t Temporary tablespace name]
```

By the default the script will use "HSUBSYS1" as the database name, "HPSS" as the schema name, and "TEMPSPACE1" as the temporary tablespace name. Typical command lines are:

```
     check_tablespaces.sh -d hsubsys1
```

Checks the tablespaces in database "HSUBSYS1", uses the schema "HPSS" to acquire the tablespaces used for the STORAGESEGDISKEXTENTS table data and indexes, and will gather the file system space for the container assigned to tablespace "TEMPSPACE1".

```
     check_tablespaces.sh -d hsubsys2 -t tempspace2
```

Checks the tablespaces in database "HSUBSYS2", uses the schema "HPSS" to acquire the tablespaces used for the STORAGESEGDISKEXTENTS table, and will gather the filesystem space for the container assigned to tablespace "TEMPSPACE2".

```
     check_tablespaces.sh -s hpxx
```

Checks the tablespaces in database "HSUBSYS1", uses the schema "HPXX" to acquire the tablespaces used for the STORAGESEGDISKEXTENTS table, and will gather the filesystem space for the container assigned to tablespace "TEMPSPACE1".

Example output:

```
$ /opt/hpss/tools/metadata/db2/check_tablspaces.sh

02/15/2010 15:47:18 connect to HSUBSYS1
02/15/2010 15:47:19 DB20000I  The SQL command completed successfully.

02/15/2010 15:47:19 select tbspace from syscat.tables where tabschema='HPSS' and
tabname='STORAGESEGDISKEXTENTS'
02/15/2010 15:47:19 DB20000I  The SQL command completed successfully.

02/15/2010 15:47:19 select TBSP_UTILIZATION_PERCENT from sysibmadm.tbsp_utilization where
TBSP_NAME='STORAGESEGDISK
'
02/15/2010 15:47:19 DB20000I  The SQL command completed successfully.

02/15/2010 15:47:20 Data tablespace --> STORAGESEGDISK
02/15/2010 15:47:20 utilization percent =                     6.13
```

```
02/15/2010 15:47:20 select distinct substr(b.tbspace,1,20) from syscat.indexes a,
syscat.tablespaces b where a.tabschema='HPSS' and a.tabname='STORAGESEGDISKEXTENTS' and
a.tbspaceid=b.tbspaceid
02/15/2010 15:47:20 DB20000I  The SQL command completed successfully.

02/15/2010 15:47:20 select TBSP_UTILIZATION_PERCENT from sysibmadm.tbsp_utilization where
TBSP_NAME='STORAGESEGDISKIDX    '
02/15/2010 15:47:20 DB20000I  The SQL command completed successfully.

02/15/2010 15:47:20 Index tablespace --> INDEXES
02/15/2010 15:47:20 utilization percent =                    4.59

02/15/2010 15:47:20
02/15/2010 15:47:20 tempspace tablespace --> TEMPSPACE1
02/15/2010 15:47:20
tempspace_cont=/var/hpss/hpssdb/hpssdb/NODE0000/HSUBSYS1/T0000001/C0000000.TMP
02/15/2010 15:47:20 df -khP /var/hpss/hpssdb/hpssdb/NODE0000/HSUBSYS1/T0000001/C0000000.TMP
Filesystem         1024-blocks      Used Available Capacity Mounted on
/dev/mapper/VolGroup00-LogVol00 234476168  36858064 185515244      17%
DB20000I  The SQL command completed successfully.

connect reset
DB20000I  The SQL command completed successfully.

terminate
DB20000I  The TERMINATE command completed successfully.
```

This script is provided as a guide to assist in evaluating the current utilization of the tablespaces being used by the affected table. The script shows you the commands as they are run and will stop if any command gets an error. Run the script and note any errors. If something unexpected happens and you get an error, you will have to seek advice. Or, if you are unsure as to how to interpret the output seek advice from HPSS support.

3.3.2. Perform The Conversion

Before proceeding with the conversion, completely stop HPSS. Stop all HPSS processes and servers. Leave DB2 running. The conversion script will try to connect in “EXCLUSIVE MODE”. The connect step will fail if there are any other connection to the target database. Run the script and note any errors. If something unexpected happens and you get an error, you will have to seek advice. Keep in mind that the original data is preserved and can be set back to it's original state.

The second step constructs the new table and loads it with the data from the original table and makes the table ready for use. This step is performed by a script that runs the DB2 CLP. This script is called “cnvrt_storagesegdiskextents.sh” and it has four arguments:

```
     cnvrt_storagesegdiskextents.sh [-d subsystem database name
(default 'HSUBSYS1')] [-s schema name (default 'HPSS')] [-c
provide before and after row counts] [-i flag to drop original
table indexes]
```

By default, the script converts subsystem 1 in schema “hpss” and will not provide before/after row counts and will not drop the indexes from the original storagesegdiskextents table. The -c option, if specified, will execute “select count(*) from storagesegdiskextents” before and after the conversion. If you have several million rows this may take several minutes each time it is executed. The -i option will drop all of the indexes from the original table. If selected, the recovery time to “undo” the conversion will be increased as the original indexes (along with the foreign key constraint) will have to be recreated. It may be necessary to use the “-i” option if there is not enough additional space in the indexes tablespace to have 2 copies of indexes: one for the original table and one for the new table.

Typical command lines are:

```
     cnvrt_storagesegdiskextents.sh
```

Converts subsystem 1 (database HSUBSYS1) using schema “HPSS” and does not provide before/after row counts and does not drop the indexes from the original table.

```
     cnvrt_storagesegdiskextents.sh -d hsubsys2
```

Converts subsystem 2 (database HSUBSYS2) using schema “HPSS” and **does not** provide before/after row counts and **does not** drop the indexes from the original table.

```
     cnvrt_storagesegdiskextents.sh -i -c
```

Converts subsystem 1 (database HSUBSYS1) using schema “HPSS” and **does** provide before/after row counts and **does** drop the indexes from the original table.

The script shows you the commands as they are run and will stop if any command gets an error. The “load from cursor”, “alter table...add foreign key” and “runstats” commands may take some time to complete in large systems.

When the script completes normally, the metadata tables are ready for HPSS to be started.

Example output from a successful conversion

```
$ /opt/hpss/tools/metadata/db2/cnvrt_storagesegdiskextents.sh -i -c

02/15/2010 16:23:19 Step --> connect

02/15/2010 16:23:17 connect to HSUBSYS1 in exclusive mode
02/15/2010 16:23:19 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:19 Step --> pre_conversion_check

02/15/2010 16:23:19 select count(*) from syscat.columns where colname='ENDING_OFFSET' and
tabschema='HPSS' and tabname='STORAGESEGDISKEXTENTS'
02/15/2010 16:23:19 DB20000I  The SQL command completed successfully.
02/15/2010 16:23:19 Column ENDING_OFFSET does not exist in table HPSS.STORAGESEGDISKEXTENTS,
conversion may proceed.

02/15/2010 16:23:19 Step --> before_conversion

02/15/2010 16:23:19 Step --> describe_table
02/15/2010 16:23:19 db2 describe table HPSS.STORAGESEGDISKEXTENTS

                                Data type                     Column
Column name                     schema    Data type name      Length     Scale Nulls
------------------------------- --------- ------------------- ---------- ----- ------
SSID                            SYSIBM    CHARACTER                   32     0 No
VVID                            SYSIBM    CHARACTER                   32     0 Yes
ORDINAL                         SYSIBM    INTEGER                      4     0 No
OFFSET                          SYSIBM    INTEGER                      4     0 Yes
LENGTH                          SYSIBM    INTEGER                      4     0 Yes

  5 record(s) selected.


02/15/2010 16:23:19 Step --> list_indexes
02/15/2010 16:23:20 db2 select substr(rtrim(indschema) || '.' || rtrim(indname),1,50) as
index_name, substr(colnames,1,60) as columns,uniquerule from syscat.indexes where
tabschema='HPSS' and tabname='STORAGESEGDISKEXTENTS'

INDEX_NAME                                          COLUMNS
UNIQUERULE
--------------------------------------------------
------------------------------------------------------------ ----------
HPSS.STORAGESEGDISKEXTENTS_PKEY                     +SSID+ORDINAL
P
HPSS.VVID_EXTENTS                                   +VVID+OFFSET+LENGTH
D

  2 record(s) selected.


02/15/2010 16:23:20 Step --> list_constraints
02/15/2010 16:23:20 db2 select substr(a.constname,1,30) as
constname,a.type,substr(rtrim(b.tabschema) || '.' || rtrim(b.tabname),1,35) as child_table_name,
substr(rtrim(b.reftabschema) || '.' || rtrim(b.reftabname),1,30) as prnt_table_name from
syscat.tabconst a left outer join  syscat.references b on a.constname=b.constname and
a.tabname=b.tabname where a.tabschema='HPSS' and a.tabname='STORAGESEGDISKEXTENTS'

CONSTNAME                      TYPE CHILD_TABLE_NAME                    PRNT_TABLE_NAME
------------------------------ ---- -----------------------------------
------------------------------
DISKSEGEXTENTSCON1             F    HPSS.STORAGESEGDISKEXTENTS          HPSS.STORAGESEGDISK
STORAGESEGDISKEXTENTS_PKEY     P    -                                   -
```

```
  2 record(s) selected.


02/15/2010 16:23:20 Step --> list_row_counts
02/15/2010 16:23:20 db2 select count(*) as rowcount from HPSS.STORAGESEGDISKEXTENTS

ROWCOUNT
-----------
       2991

  1 record(s) selected.


02/15/2010 16:23:20 Step --> tblspace_data

02/15/2010 16:23:20 select tbspace from syscat.tables where tabschema='HPSS' and
tabname='STORAGESEGDISKEXTENTS'
02/15/2010 16:23:20 DB20000I  The SQL command completed successfully.
02/15/2010 16:23:20 Data tablespace --> STORAGESEGDISK

02/15/2010 16:23:20 Step --> tblspace_index

02/15/2010 16:23:20 select distinct substr(b.tbspace,1,20) from syscat.indexes a,
syscat.tablespaces b where a.tabschema='HPSS' and a.tabname='STORAGESEGDISKEXTENTS' and
a.tbspaceid=b.tbspaceid
02/15/2010 16:23:20 DB20000I  The SQL command completed successfully.
02/15/2010 16:23:20 Index tablespace --> STORAGESEGDISKIDX

02/15/2010 16:23:21 Step --> get_foreign_key_name

02/15/2010 16:23:21 select substr(constname,1,30) from syscat.tabconst where tabschema='HPSS' and
tabname='STORAGESEGDISKEXTENTS' and type='F'
02/15/2010 16:23:21 DB20000I  The SQL command completed successfully.
02/15/2010 16:23:21 Foreign Key constraint name --> DISKSEGEXTENTSCON1

02/15/2010 16:23:21 Step --> drop_foreign_key

02/15/2010 16:23:21 alter table HPSS.storagesegdiskextents drop foreign key DISKSEGEXTENTSCON1
02/15/2010 16:23:21 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:21 Step --> drop_view

02/15/2010 16:23:21 drop view HPSS.STORAGESEGDISKVIEW
02/15/2010 16:23:21 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:21 Step --> rename_orig_table

02/15/2010 16:23:21 rename table HPSS.storagesegdiskextents to storagesegdiskextents_orig
02/15/2010 16:23:21 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:21 Step --> manage_orig_indexes

02/15/2010 16:23:21 alter table HPSS.storagesegdiskextents_orig drop primary key
02/15/2010 16:23:21 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:21 select 'drop index ' || substr(rtrim(indschema) || '.' ||
rtrim(indname),1,50) || ';' from syscat.indexes where tabschema='HPSS' and
tabname='STORAGESEGDISKEXTENTS_ORIG'
02/15/2010 16:23:21 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:21 drop index HPSS.VVID_EXTENTS
02/15/2010 16:23:22 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:22 Step --> create_new_table
```

```
02/15/2010 16:23:22 CREATE TABLE HPSS.STORAGESEGDISKEXTENTS_NEW  (SSID CHAR(32) FOR BIT DATA NOT
NULL , VVID CHAR(32) FOR BIT DATA , ORDINAL INTEGER NOT NULL , STARTING_OFFSET INTEGER ,
ENDING_OFFSET INTEGER ) IN STORAGESEGDISK
INDEX IN STORAGESEGDISKIDX
02/15/2010 16:23:22 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:22 Step --> create_new_indexes

02/15/2010 16:23:22 CREATE UNIQUE INDEX HPSS.SSID_EXTENTS ON HPSS.STORAGESEGDISKEXTENTS_NEW (SSID
ASC, ORDINAL ASC) ALLOW REVERSE SCANS
02/15/2010 16:23:22 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:22 CREATE INDEX HPSS.VVID_EXTENTS_OFFSETS ON HPSS.STORAGESEGDISKEXTENTS_NEW
(VVID ASC, STARTING_OFFSET ASC, ENDING_OFFSET ASC) ALLOW REVERSE SCANS
02/15/2010 16:23:22 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:22 Step --> add_new_primary_key

02/15/2010 16:23:22 ALTER TABLE HPSS.STORAGESEGDISKEXTENTS_NEW ADD CONSTRAINT
STORAGESEGDISKEXTENTS_PKEY PRIMARY KEY (SSID,ORDINAL)
02/15/2010 16:23:22 SQL0598W  Existing index "HPSS.SSID_EXTENTS" is used as the index for the
primary key or a unique key.  SQLSTATE=01550

02/15/2010 16:23:22 Step --> declare_cursor

02/15/2010 16:23:22 declare segcurs cursor for select ssid,vvid,ordinal,offset,offset+length-1
from HPSS.storagesegdiskextents_orig
02/15/2010 16:23:22 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:22 Step --> load_new_table

02/15/2010 16:23:22 load from segcurs of cursor insert into HPSS.storagesegdiskextents_new
nonrecoverable
02/15/2010 16:23:23 DB20000I  The LOAD command completed successfully.

02/15/2010 16:23:23 Step --> rename_new_table

02/15/2010 16:23:23 rename table HPSS.storagesegdiskextents_new to storagesegdiskextents
02/15/2010 16:23:23 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:23 Step --> add_foreign_key

02/15/2010 16:23:23 ALTER TABLE HPSS.STORAGESEGDISKEXTENTS ADD CONSTRAINT DISKSEGEXTENTSCON1
FOREIGN KEY (SSID) REFERENCES HPSS.STORAGESEGDISK (SSID) ON DELETE CASCADE ON UPDATE NO ACTION
ENFORCED ENABLE QUERY OPTIMIZATION
02/15/2010 16:23:23 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:23 Step --> create_insert_trigger

02/15/2010 16:23:23 CREATE TRIGGER HPSS.CHECK_OVERLAP_BEFORE_INSERT NO CASCADE BEFORE INSERT ON
HPSS.STORAGESEGDISKEXTENTS REFERENCING  NEW AS N  FOR EACH ROW  MODE DB2SQL WHEN ( 0 < (select
count(*) from HPSS.storagesegdiskextents where vvid=N.vvid and N.starting_offset <= ending_offset
and N.ending_offset >= starting_offset) ) BEGIN ATOMIC SIGNAL SQLSTATE '99001'  set MESSAGE_TEXT
= 'segment overlap fault'; END

02/15/2010 16:23:23 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:23 Step --> create_update_trigger

02/15/2010 16:23:23 CREATE TRIGGER HPSS.CHECK_OVERLAP_BEFORE_UPDATE NO CASCADE BEFORE UPDATE OF
SSID,VVID,ORDINAL,starting_OFFSET,ENDing_OFFSET ON HPSS.STORAGESEGDISKEXTENTS REFERENCING  NEW AS
N  FOR EACH ROW  MODE DB2SQL WHEN ( 0 < (select count(*) from HPSS.storagesegdiskextents where
vvid=N.vvid and N.starting_offset <= ending_offset and N.ending_offset >= starting_offset) )
BEGIN ATOMIC SIGNAL SQLSTATE '99001'  set MESSAGE_TEXT = 'segment overlap fault'; END
```

```
02/15/2010 16:23:23 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:23 Step --> recreate_view

02/15/2010 16:23:24 CREATE VIEW HPSS.STORAGESEGDISKVIEW AS SELECT   S.SSID, S.VVID,
S.ALLOCATED_LENGTH, S.OWNER, S.CREATION,   S.UPDATE, M.SCLASS_ID, S.FILE_SYSTEM_ID,
M.BLOCK_SIZE, M.CLUSTER_LENGTH, E.ORDINAL, E.starting_OFFSET, E.ENDing_OFFSET   FROM
(HPSS.STORAGESEGDISK AS S INNER JOIN HPSS.STORAGEMAPDISK AS M   ON S.VVID = M.VVID) INNER JOIN
HPSS.STORAGESEGDISKEXTENTS AS E   ON S.SSID = E.SSID
02/15/2010 16:23:24 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:24 Step --> rename_orig_table_complete

02/15/2010 16:23:24 rename table HPSS.storagesegdiskextents_orig to
storagesegdiskextents_orig_complete
02/15/2010 16:23:24 DB20000I  The SQL command completed successfully.

02/15/2010 16:23:24 Step --> runstats

02/15/2010 16:23:24 runstats on table HPSS.storagesegdiskextents with distribution and detailed
indexes all
02/15/2010 16:23:24 DB20000I  The RUNSTATS command completed successfully.
02/15/2010 16:23:24 Step --> after_conversion

02/15/2010 16:23:24 Step --> describe_table
02/15/2010 16:23:24 db2 describe table HPSS.STORAGESEGDISKEXTENTS

                                Data type                     Column
Column name                     schema    Data type name      Length     Scale Nulls
------------------------------- --------- ------------------- ---------- ----- ------
SSID                            SYSIBM    CHARACTER                   32     0 No
VVID                            SYSIBM    CHARACTER                   32     0 Yes
ORDINAL                         SYSIBM    INTEGER                      4     0 No
STARTING_OFFSET                 SYSIBM    INTEGER                      4     0 Yes
ENDING_OFFSET                   SYSIBM    INTEGER                      4     0 Yes

  5 record(s) selected.


02/15/2010 16:23:24 Step --> list_indexes
02/15/2010 16:23:24 db2 select substr(rtrim(indschema) || '.' || rtrim(indname),1,50) as
index_name, substr(colnames,1,60) as columns,uniquerule from syscat.indexes where
tabschema='HPSS' and tabname='STORAGESEGDISKEXTENTS'

INDEX_NAME                                          COLUMNS
UNIQUERULE
--------------------------------------------------
------------------------------------------------------------ ----------
HPSS.SSID_EXTENTS                                   +SSID+ORDINAL
P
HPSS.VVID_EXTENTS_OFFSETS                           +VVID+STARTING_OFFSET+ENDING_OFFSET
D

  2 record(s) selected.


02/15/2010 16:23:24 Step --> list_constraints
02/15/2010 16:23:24 db2 select substr(a.constname,1,30) as
constname,a.type,substr(rtrim(b.tabschema) || '.' || rtrim(b.tabname),1,35) as child_table_name,
substr(rtrim(b.reftabschema) || '.' || rtrim(b.reftabname),1,30) as prnt_table_name from
syscat.tabconst a left outer join  syscat.references b on a.constname=b.constname and
a.tabname=b.tabname where a.tabschema='HPSS' and a.tabname='STORAGESEGDISKEXTENTS'

CONSTNAME                      TYPE CHILD_TABLE_NAME                    PRNT_TABLE_NAME
------------------------------ ---- -----------------------------------
```

```
------------------------------
DISKSEGEXTENTSCON1             F    HPSS.STORAGESEGDISKEXTENTS          HPSS.STORAGESEGDISK
STORAGESEGDISKEXTENTS_PKEY     P    -                                   -

  2 record(s) selected.


02/15/2010 16:23:24 Step --> list_row_counts
02/15/2010 16:23:25 db2 select count(*) as rowcount from HPSS.STORAGESEGDISKEXTENTS

ROWCOUNT
-----------
       2991

  1 record(s) selected.



exiting
commit
DB20000I  The SQL command completed successfully.

connect reset
DB20000I  The SQL command completed successfully.

terminate
DB20000I  The TERMINATE command completed successfully.
```

Example output with errors:

```
$ /opt/hpss/tools/metadata/db2/cnvrt_storagesegdiskextents.sh -i -c
02/15/2010 16:28:06 connect to HSUBSYS1 in exclusive mode
02/15/2010 16:28:06 SQL1035N  The database is currently in use.  SQLSTATE=57019
ERROR: There was at least one unexpected SQLCODE
```

3.3.3. Subsystems

If the installation has multiple subsystems, you must convert each of the STORAGESEGDISKEXTENTS tables in each of the subsystems. Repeat the appropriate procedure above modifying the "-d hsubsys1" commands parameter on the invocations of the check_tablespaces.sh and cnvrt_storagesegdiskextents.sh scripts appropriately.

3.4. Using The New Table

Once you have performed the conversion described above, the next step should be to verify the conversion is correct. You can do this by starting the HPSS system and noting if any errors are reported by the Core Server. Avoid starting MPS until you have verified all is well.

The Core Server checks the elements of the database schema as it is starting. If it detects any errors, it will not start. If the installation has any files in its disk cache, the Core Server will build the in-memory space maps associated with those files. If there are errors associated with this process, the server will send appropriate alarm messages. Plan on not opening the system to its users until the Core Server has informed you that all of the disk space maps have been built without error.

If anything is amiss, stop the system and seek advice. Do not attempt to correct errors associated with this conversion yourself. This conversion process has been carefully tested and we don't expect problems with it, but don't attempt to correct problems yourself.

When you have established that the system is running properly, you will need to decide when to remove the old metadata table, “storagesegdiskextents_orig_complete”. Once disk files are created or destroyed using the new metadata table, the old table cannot be used as a “fall back”. Installations with a large number of disk files will probably want to recover the space occupied by the old table.

Remove the old table by using the DB2 CLP. Give the following commands:

```
connect to subsysx
drop table hpss.storagesegdiskextents_orig_complete
terminate
```

Since the name “storagesegdiskextents_orig_complete” is not a recognized HPSS metadata table name, you can't harm the system by dropping it, even when the system is running.