developerWorks article template using OpenDocument

Important: Please ensure that your input in this information form is inside the fields. To enter data in a very narrow field , press ctrl-shift-F9 and then use the **Next** button to move through fields.

Type of Submission:

Title: The Scripted Interface for DB2 Advanced Copy Services

Subtitle: Part 4 – Implementation of the Scripted Interface for DB2 ACS Using IBM System Storage DS4800 and Storage Manager

Keywords: DB2 ACS, scripted interface, IBM System Storage DS4800, IBM Storage Manager snapshot, flashcopy, volumecopy

Given: Martin
Family: Jungfer

Job Title: software engineer
Email: jungfer@de.ibm.com
Bio: See below.
Company: IBM Deutschland Research and Development GmbH
Photo filename: jungfer.jpg

Given: Holger
Family: Hellmuth

Job Title: software engineer
Email: holger.hellmuth@de.ibm.com
Bio: See below.
Company: IBM Deutschland Research and Development GmbH
Photo filename:

Abstract: See below.

Bio Martin: Martin Jungfer works at IBM Information Management as a software engineer in the IBM DB2 for Linux, Unix and Windows for SAP development team in Boeblingen, Germany. He has more than ten years of experience with SAP on IBM DB2 for Linux, Unix and Windows. He holds a degree in technical computer science from the University for Applied Science in Albstadt, Germany.

Bio Holger: Holger Hellmuth works at IBM Information Management as a software engineer in the IBM DB2 for Linux, Unix and Windows for SAP development team in Boeblingen, Germany.

He has more than ten years of experience with SAP on IBM DB2 for Linux, Unix and Windows. He holds a degree in engineering physics from the University for Applied Science in Heilbronn, Germany.

Abstract: The IBM® DB2® Advanced Copy Services (DB2 ACS) support taking snapshots for backup purposes in DB2 for Linux®, Unix® and Windows® databases. You can use the DB2 ACS API either through libraries implemented by your storage hardware vendors (wheras until now, only some do) or you can implement this API yourself which however, involve a high effort. This changes with IBM DB2 10.5.

As of IBM DB2 10.5, a new feature called Scripted Interface for DB2 Advanced Copy Services is introduced. It allows you to implement shell scripts instead of C-libraries. These scripts can use the tools provided by the storage vendors to run the snapshot operations. The Scripted Interface can be used independently from your storage hardware. Additionally, DB2 supports every storage hardware as soon as it becomes available on the market.

The feature supports all three architectures of DB2: enterprise server, multi-partitioned database using the database partitioning feature (DPF), and databases using pureScale. The featur is supported on all UNIX and Linux platforms DB2 is certified on.

In this series we will provide an introduction to the Scripted Interface feature and present a real life example. This is the fouth part of the series and demonstrates the usage of the Scripted Interface together with IBM System Storage and IBM Storage Manager. The DB2 database in this example is used by an SAP® NetWeaver® system.

Introduction to IBM System Storage DS4800

The IBM DS4800 storage server is one of several models of the DS4000 series. It delivers breakthrough disk performance and outstanding reliability for demanding applications in compute-intensive environments. Meanwhile its successors series DS5000 and DS5020 are available.

The IBM System Storage DS Storage Manager software is used to configure, manage, and troubleshoot the DS4800 storage server. It is used primarily to configure RAID arrays and logical drives, assign logical drives to hosts, replace and rebuild failed disk drives, expand the size of the arrays and logical drives and convert from one RAID level to another. It allows troubleshooting and management tasks, such as checking the status of the storage server components, updating the firmware of the RAID controllers, and managing the storage server. Finally, it offers advanced functions such as FlashCopy, VolumeCopy, and Enhanced Remote Mirroring. [1]

Hardware and Software Details

The figure1 below shows the hardware and software used in this implementation. The database server is running on a dedicated AIX logical partition (LPAR1). It uses virtual storage for all of its AIX volume groups which are provided by the Virtual I/O Server. The Virtual I/O server is also running on a dedicated AIX LPAR (LPAR2). All filesystems are located on this virtual storage (e.g. AIX rootvg, DB2 tablespace containers, database directories, and transaction log directories).

Physically the storage is located on a IBM DS4800 storage subsystem. This mid range storage subsystem offers a number copy services premium features. The FlashCopy and VolumeCopy feature are utilized in this implementation to create database backups. The AIX LPARs are physically hosted on IBM p550 Express POWER6 hardware.

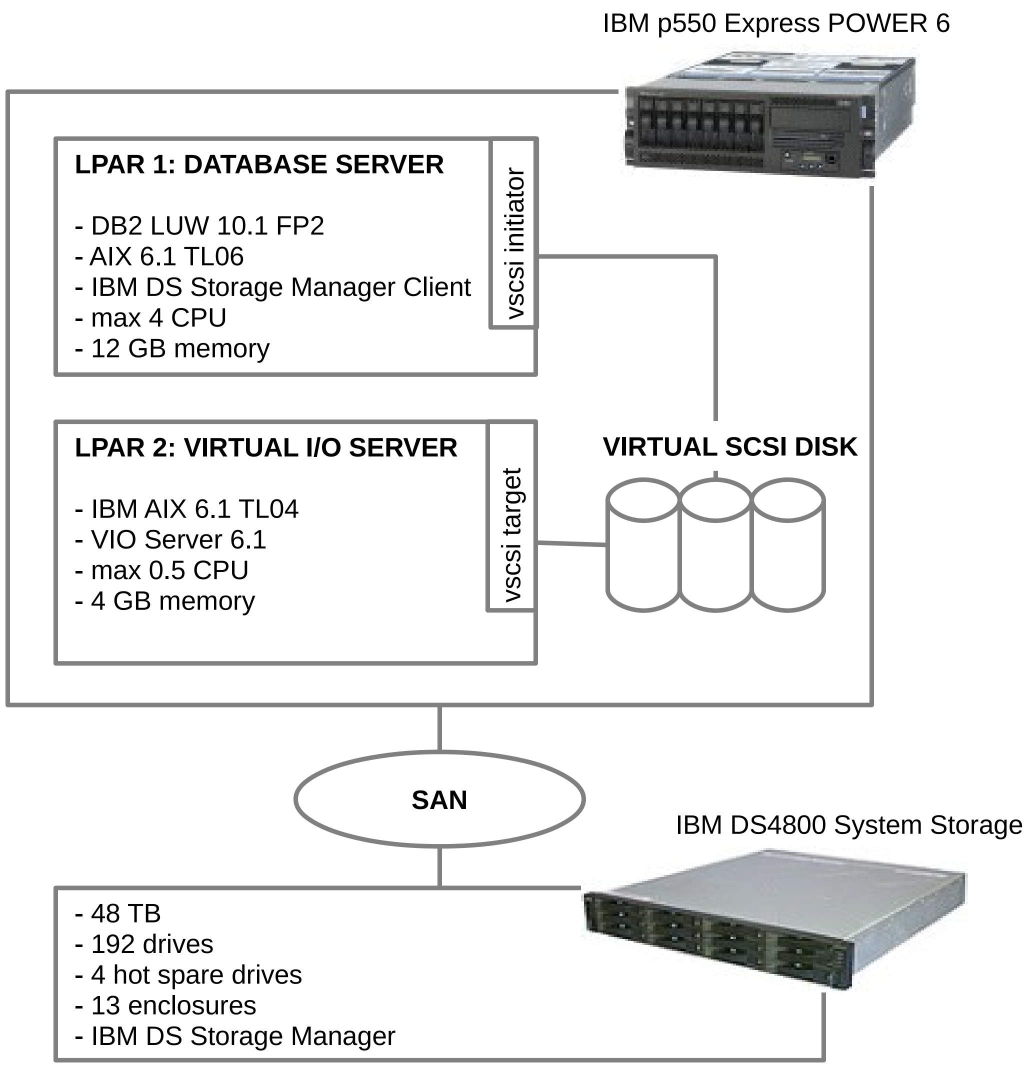

Figure 1: Hardware and Software Overview

Configuration of the virtual storage

This chapter briefly describes the virtual storage configuration used in this scenario. It shows how the IBM Virtual I/O Server is used in combination with the IBM DS 4800 storage subsystem to provide storage to the DB2 database server. For detailed configuration of the IBM Virtual I/O Server and the storage subsystem components refer to the official IBM product documentation [3].

As depicted in figure 2 below the AIX LPAR1 (left) hosts the database server. Local to the database server three hdisks (colored green, yellow and blue) are available through the VIO server in the middle. For simplicity the figure shows only one hdisk per AIX volume group. Usually the AIX volume group consists of many hdisks. Each of the LPAR local hdisks maps to one base logical drive on the IBM DS 4800 storage subsystem. Flashcopy logical drives are configured on the storage subsystem for the base logical drives containing the DB2 files and paths which are part of a DB2 database backup.

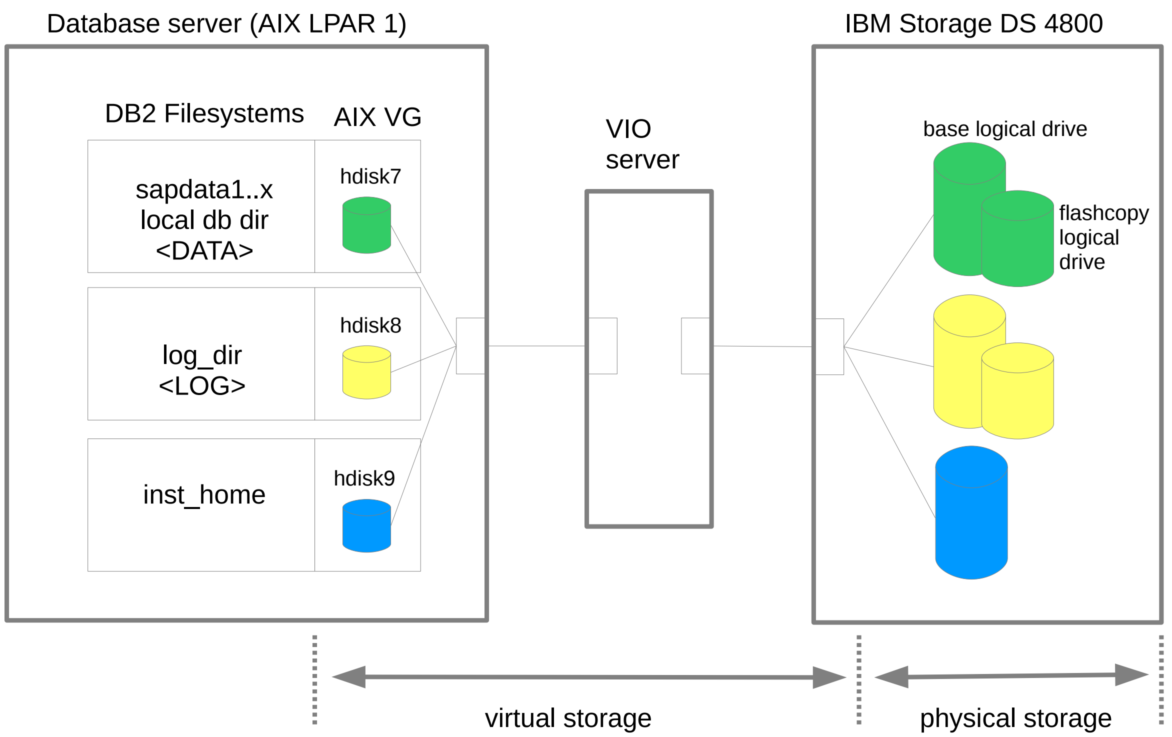

Figure 2: overview of virtual storage configuration

The customer script described in this document refers only to the logical drive names of the storage subsystem, the AIX volume group names and the local hdisk names. These are listed in table 1 below. The VIO server configuration information is not required by the customer script and therefore not listed in the table.

LPAR database server **file system name**	**volume group**	**hdisk**	**DS4800** **base logical drive name** **flashcopy log. drv. name**

`/db2/AMT/sapdata1`
`/db2/AMT/sapdata2`
`/db2/AMT/sapdata3`
`/db2/AMT/sapdata4`
`/db2/AMT/db2amt/NODE0000`
	`sapAMT3vg`	`hdisk7`	`AMT_DATA_MJ`
			`AMT_DATA_MJ_1`[1]

`/db2/db2/log_dir`	`sapAMT4vg`	`hdisk8`	`AMT_LOG_MJ` `AMT_LOG_MJ_1`

`/db2/db2amt`
`/db2/AMT`
`/db2/AMT/log_archive`
`<SAP filesystem>`
	`sapAMT5vg`	`hdisk9`	`AMT_MJ`
			`<no flashcopy>`

Table 1: Mapping of filesystems to volume group, hdisk and logical drive names

e.g.:
The DB2 filesystems `/db2/AMT/sapdata1...4` (storing tablespace container files) and `/db2/AMT/db2amt/NODE0000` (storing the database local directory) are located on AIX Volume Group `sapamt3vg` and `hdisk7` respectively. `hdisk7` is provided to the LPAR by the DS4800 storage subsystem' logical drive `AMT_DATA_MJ`. Logical drive `AMT_DATA_MJ` is flashcopy enabled. The name of the flashcopy logical drive is `AMT_DATA_MJ_1`.

Prerequisite:
The separation of DB2 data and log files into different hdisks and AIX volume groups is a requirement. The sample customer script will terminate with an error in the database restore if this prerequisite is not met.

Database files and transaction log files must be separated on different hdisks of the AIX LPAR and base logical drives of the storage subsystem. This is required to be able to handle flashcopies of storage logical drives independently of each other. In other words if data and logs files reside on a single hdisk it would be possible to do a DB2 snapshot backup including the logfiles, but it would not be possible to do a restore without the logfiles.

The AIX logical volume manager (LVM) allows to create filesystems on single specific local hdisks. The separation of data and log files on separate hdisks mentioned above could be maintained this way. However, a risk of creating a filesystem on a wrong local hdisk exists when the number of involved hdisks increases. To reduce this risk separate AIX volume groups (`sapAMT3vg` and `sapAMT4vg`) are used for DB2 data and log files.

1 FlashCopy logical drives are not mapped to local hdisks on the AIX LPAR

IBM Storage Manager Commands used for backup and restore

The sample customer script uses the FlashCopy feature of the storage subsystem to create snapshot backups of the database. The following overview lists the storage manager script commands that are used to perform the backup and restore work in the customer script. They are called via the storage manager command-line interface (CLI). The IBM Storage Manager client must be installed locally on the database server.

script command	**Usage**
`create Flashcopy LogicalDrive`	This command creates a FlashCopy logical drive of a base logical drive. **Recommendation:** Use the Storage Manager Graphical User Interface to create and initialize the FlashCopy Logical Drives. [2]
`create VolumeCopy`	This command creates a VolumeCopy and starts the VolumeCopy operation. A FlashCopy Logical Drive of the source is required. **Recommendation:** Use the Storage Manager Graphical User Interface to create and initialize the VolumeCopy pairs. See footnote 2.
`recreate Flashcopy LogicalDrive`	Recreates a new flashcopy (snapshot) of a AIX client LPAR local hdisk at a single point in time. The only parameter is the storage system flashcopy logical drive name. The Flashcopy Logical Drive must already exist before this command can run. This is usually done during the setup of the backup environment.
`recopy VolumeCopy`	This command re-initiates a VolumeCopy operation by using an existing VolumeCopy pair. A FlashCopy Logical Drive of the source is required.

Table 2: Storage Manager commands used in customer script

2 IBM recommends using the Storage Manager client GUI to manage your storage subsystems. The command-line interface does not have any mechanisms to prevent you from inadvertently making unwanted changes to the storage subsystem. Because the script commands are capable of damaging a configuration and causing loss of data access if not used correctly.

Backup
The customer script determines which of the local hdisks `(hdisk<x>)` of the database server must be flashcopied in case of a backup. This list of local hdisks is then processed sequentially. If, for example during a DB2 snapshot backup, the customer script determines that local `"hdisk7"` must be flashcopied, it looks up the associated flashcopy logical drive name `"AMT_DATA_MJ_1"` in a separate mapping file. The mapping file is described later in this document and it contains the relation between local AIX hdisks and the logical drive names of the IBM Storage Manager. The customer script prepares and executes the flashcopy with following command via the IBM Storage Manager CLI, assuming the flashcopy drive already exists:

```
            recreate FlashCopy LogicalDrive  ["AMT_DATA_MJ_1"]
```

`"AMT_DATA_MJ_1"` is the default name for the flashcopy logical drive name given by the IBM Storage Manager during its initial creation. Whenever the names regarding flashcopy logical drives are changed on the storage subsystem side, it is absolutely important to reflect these changes in the mapping file, too. Otherwise the backup will fail or wrong logical drives will be flashcopied without notice on the database server side.

Creation of a snapshot backup image with Volume copy
Once the DB2 snapshot backup is complete it is the task of the storage subsystem to create new logical drives out of the flashcopies and their base logical drives. This new logical drive will contain the database snapshot backup image. It is a time and resource consuming process because data will physically be copied from the base and flashcopy logical drives to a target logical drive. This is in contrast to creating a only "logical" flashcopy. It again depends on the sizes of the base and flashcopy logical drives and their copy priority. Figure 3 shows the additionally created "new logical drives" on the storage system. The new logical drive names must be added to the mapping file and can then be used for the database restore.

The sample customer script does not handle the creation of the backup image, because this is primarily the task of the storage subsystem. The commands used for this operation are only mentioned in short. For details please refer to official IBM documentation [1, 4].

The storage manager command `"create VolumeCopy"` creates a snapshot backup image that can be used for restoring the database. On the storage subsystem the command defines and initializes a Volume Copy pair. This procedure is usually done during initial setup or if new base logical drives are added to increase database capacity. It is recommended to use the IBM Storage Manager Graphical User Interface for this task (see table 2).

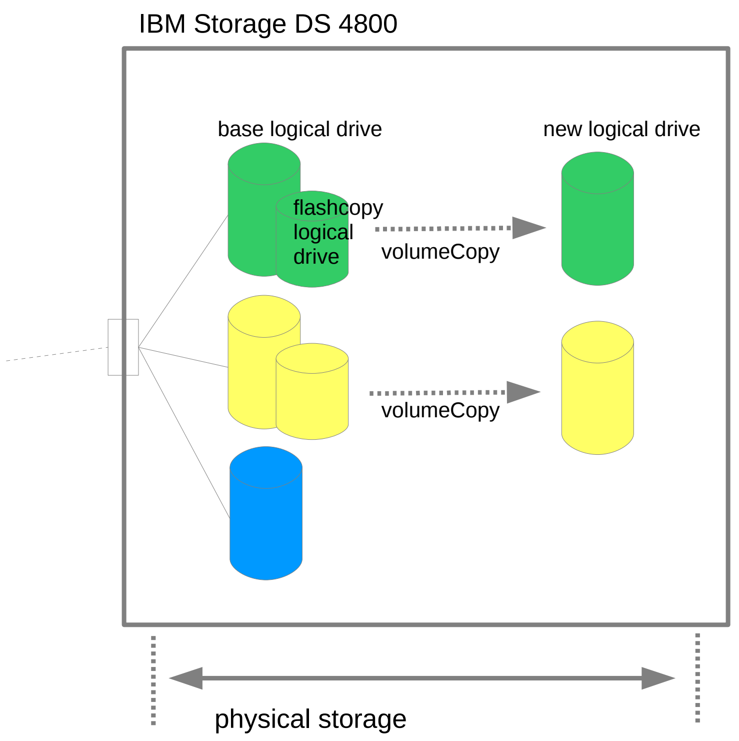

Figure 3: creation of a volume copy

After every snapshot backup the volume copy pair must be updated with the data from the newly taken flashcopy. This is done by the command "`recopy VolumeCopy`". The following lists the storage manager commands used the recreate the logical drives after each DB2 snapshot backup:

```
recopy volumeCopy
      target ["AMT_DATA_MJ-VC"]
      source ["AMT_DATA_MJ-1"]
      copyPriority=medium;

recopy volumeCopy
      target ["AMT_LOG_MJ-VC"]
      source ["AMT_LOG_MJ-1"]
      copyPriority=medium;'
```

The VolumeCopies are created online using the flashcopy logical drives. The base logical drives used by the database are not impacted by this command.

The newly created logical drives on the storage subsystem are mapped to the database server with new local hdisk names (e.g `hdisk6`) but are not included into AIX volume groups. The restore task will recreate the original AIX volume groups with the new local hdisks. This new mapping information is also reflected in table 3 rightmost column "DS4800".

LPAR			**DS4800**
file system name	**volume group**	**hdisk**	**- base logical drive name** **- flashcopy log. drv. name** **- new volcopy log. drv. name**
/db2/AMT/sapdata1 /db2/AMT/sapdata2 /db2/AMT/sapdata3 /db2/AMT/sapdata4 /db2/AMT/db2amt/NODE0000	sapAMT3vg	hdisk7	AMT_DATA_MJ AMT_DATA_MJ_1
	--	**hdisk6**	**AMT_DATA_MJ_VC**
/db2/db2/log_dir	sapAMT4vg	hdisk8	AMT_LOG_MJ AMT_LOG_MJ_1
	--	**hdisk3**	**AMT_LOG_MJ_VC**
/db2/db2amt /db2/AMT /db2/AMT/log_archive <SAP filesystem>	sapAMT5vg	hdisk9	AMT_MJ <no flashcopy>

Table 3: Mapping including the "new base logical drives"

Restore

The database snapshot restore is a very fast process that does not require copying data physically. This is because the image was already prepared previously by the storage subsystem (storage manager command `recopy VolumeCopy`).

The customer script determines by referencing the mapping file which local hdisks must be available to the LPAR hosting the database server. The restore process then recreates the necessary AIX volume groups, mounts the required filesystems and finishes up the restore task by referencing the mapping file. Some of the mentioned steps require root privileges on the database server which are granted by using `sudo`. This is also described in one of the following sections.

Once the restore is complete you can create new flashcopy logical drives (after releasing the VolumeCopy pair on the storage subsystem) on the new base logical drives. This way it is not required to move the database back to its original logical drives in order to have flashcopy backup available again. After adapting the mapping file with the new hdisk names and logical drive names flashcopy backup via Scripted Interface can continue to be used. The mapping after the restore is reflected in the table 4 below.

This restore method is very fast, since the physical copy of the data is done asynchronous and not during the restore process. The disadvantage of this method is the high disk space requirement on the storage subsystem. The volume copy logical drives are of the same size as the base logical drives.

LPAR file system name	volume group	hdisk	DS4800 - base logical drive name - flashcopy log. drv. name - new volcopy log. drv. name

`/db2/AMT/sapdata1`
`/db2/AMT/sapdata2`
`/db2/AMT/sapdata3`
`/db2/AMT/sapdata4`
`/db2/AMT/db2amt/NODE0000`
	`sapAMT3vg`	**`hdisk6`**	**`AMT_DATA_MJ_VC`**

`/db2/db2/log_dir`	`sapAMT4vg`	**`hdisk3`**	**`AMT_LOG_MJ_VC`**

`/db2/db2amt`
`/db2/AMT`
`/db2/AMT/log_archive`
`<SAP filesystem>`
	`sapAMT5vg`	`hdisk9`	`AMT_MJ`
			`<no flashcopy>`

Table 4: Mapping after restore with VolumeCopy

Alternative Restore
Instead of creating new base logical volumes with the VolumeCopy feature, it is also possible to use the flashcopy logical drives directly for the restore. However, then the limitation exists, that no further flashcopy backups can be created. Flashcopies can only be created from a base logical drive, not from a flashcopy logical drive itself.
The advantage of this alternative is its considerably lower disk space requirements compared to the previously described method via VolumeCopy.

Prerequisite for Restore:
The configuration of the VIO server and mapping of the logical drives from the storage system must be complete before the customer script is used for a database restore. It is a prerequisite for the restore that the hdisks (`hdisk3` and `hdisk6`) containing the snapshot backup image are available on the database server. In this case the term "available" means that the hdisk is ready to be included into a AIX volume group.

Overview of Storage Manager Tasks

The following table 5 contains a summary of storage manager tasks described above. Some of the task are handled by this sample customer script, other tasks must be handled externally manually or by script:

Storage Management Task	**handled in customer script**	**handle externally (outside cust. script)**
create flashcopy log. drives on DS4800 (backup)		X
re-create flashcopies on DS4800 (backup)	X	
create VolumeCopy on DS48000		X
recreate VolumeCopy on DS4800		X
make local hdisks available on database server AIX LPAR (restore)		X
recreate AIX volume group in database server AIX LPAR (restore)	X	

Table 5: Summary of Storage Management tasks

Privileges for Running IBM Storage Manager Commands

The customer script that is invoked by the Scripted Interface runs with the privileges of the instance owner. However, the commands listed above run via the IBM Storage Manager Storage command line interface which itself requires root authority. For this part of the series, the privileges to run the commands were granted using sudo.

Furthermore during the restore procedure the customer script makes the logical drives containing the snapshot mirror image available to the database again. This is accomplished by first reimporting the mirrored local hdisks into the original AIX volume groups and then re-mounting the filesystems. These tasks also require root privileges which were granted by sudo, too.

The sudo configuration was updated with the use of the command `visudo` with lines like the following:

```
db2amt ALL=NOPASSWD: /usr/SMclient/SMcli*
db2amt ALL=NOPASSWD: /usr/sbin/mount
db2amt ALL=NOPASSWD: /usr/sbin/umount
db2amt ALL=NOPASSWD: /usr/sbin/varyoffvg sap*
db2amt ALL=NOPASSWD: /usr/sbin/varyonvg sap*
db2amt ALL=NOPASSWD: /usr/sbin/exportvg sap*
db2amt ALL=NOPASSWD: /usr/sbin/recreatevg -y sapAMT*vg -p -Y NA -L / hdisk*
```

For other platforms, there might be other solutions, e.g., adding the instance owner to groups that are allowed to run the appropriate commands. We discourage you to use scripts with the setuid bit set.

Preparation Steps

Before the first call of the customer script several preparation steps must be completed. These steps are listed below.

Preparation step 1: Initialize the FlashCopy logical drive on the storage subsystem
The initialization requires time from several minutes up to hours depending on the size and modification priority of the base logical drive. The initialization is done via the IBM Storage Manager before starting the first snapshot backup. The appropriate storage manager command is "`create FlashCopy LogicalDrive`". It is recommended to use the IBM Storage Manager Graphical User Interface (see recommendation below). The names of the flashcopy logical drives must be made available to the customer script. This is done via the mapping file described in the next step.

Preparation step 2: Create the mapping file
The mapping file describes which of the logical drives of the storage subsystem is mapped to which local hdisk name on the database server. An example mapping file is listed below. The first two columns ("original_hdisk" and "FlashCopyLogicalDrive") contain information about the mapping of each local hdisk of the database server to the flashcopy logical drive of the storage subsystem. During backup the customer script reads these first two columns in order to generate the correct storage manager flashcopy commands.

```
# mapping file for Scripted Interface and IBM storage manager
# original_hdisk - FlashCopyLogicalDrive - backup_hdisk - BackuplogicalDrive – original_vg
# ==========================================================================================
hdisk7 AMT_DATA_MJ-1 hdisk6 AMT_DATA_MJ-VC sapAMT3vg
hdisk8 AMT_LOG_MJ-1 hdisk3 AMT_LOG_MJ-VC sapAMT4vg
```

The columns are space separated and comment lines start with a '#' character.

The mapping file contains crucial information for the restore procedure (columns 3, 4 and 5). These columns specify which backup image is being restored if a restore command is issued:

- Column 3 (labeled backup_hdisk) contains the local hdisk name on which the backup image can be accessed on the database server.
- Column 4 (labeled BackuplogicalDrive) indicates the logical drive name which contains the backup image on the storage subsystem.
- Column 5 (labeled original_vg) contains the AIX volume group name which is going to be recreated during the restore.

Attention:
Please note this mapping defines the snapshot backup image that is to be restored. It overrides the timestamp specified with DB2 restore command (TAKEN AT parameter). Therefore the correct logical drive of the storage subsystem must be

mapped to correct local hdisk of the database server. The customer script does not verify if the mapping is semantically correct.[3]

Timestamps of the flashcopies and the database backup are stored in the protocol file which was created during a successful DB2 backup operation. The file should be referenced before starting a restore operation in order to verify that the correct snapshot backup image is available. The following lines of the protocol file store the required information:

- Lines labeled with “TIMESTAMP” represent the database backup timestamp.
- Lines labeled with “USER_FC” contain the timestamp of the flashcopy.

After the flashcopy logical drive is initialized and the mapping file exists (with at least columns 1 and 2) the customer script can be used to create a snapshot backup.

The name of the mapping file is passed to the DB2 backup command via the snapshot OPTIONS parameter. The option name is “FlashCopyTarget”. See the example below:

```
backup database AMT
  use snapshot script "acs_ibmstor.sh"
  options "... FlashCopyTarget=/db2/db2amt/fc_mapping.txt"
```

This mapping file is the only source of information for the customer script about logical drive names on the storage subsystem. It is absolutely important to keep this file up-to-date.

3 This document contains a section “additional virtual storage commands”. It describes the commands used to check the mapping file for correctness.

Implementation of Scripted Interface

This chapter describes the implementation of the customer script using the flashcopy capability of IBM System Storage. All processing steps in the script from the point where it is called from DB2 via the ACS API are included. DB2 ACS API and the general use of the Scripted Interface functionality is documented in the official IBM documentation of DB2 10.5 [1] and in the previous parts of this developerWorks series.

The script coding is kept simple on purpose. Several lines could be saved by combining commands like awk, grep, and output evaluation into one line. However, this would make it harder to read for users who are not used to korn shell syntax.

General Environment Variables

The script calls several Storage Manager and AIX administration commands to perform the flashcopy backups and restores. The Storage Manager Command Line Interface and its generally used options are defined by variables in the beginning of the script as shown in the following excerpt. These variables might need to be adapted to the environment.

The line numbers are only valid in this context. The actual line numbers in the script are different.

```
+28  # IBM Storage Manager path and exes
+29  SMINST="/usr/SMclient"
+30  SMBIN=$SMINST
+31  SMCLI="$SMBIN/SMcli"
+32  SM_HOST="aa.bb.ccc.dd aa.bb.ccc.dd"
+33  SM_PASSWORD="xxxxxx"
```

For example, the Storage Manager Client CLI executable `SMcli` is defined in line 31. Within the script, the variable `$SMCLI` is used to reference the command. Storage Manager commands which execute write operations require that the Storage Manager password is passed along with the command. The password is saved unsecure in the variable `$SM_PASSWORD` in the script. Alternatively it could be stored in an encrypted file and read each time it is needed.

The script writes only the minimal required information into the protocol file, such as the names of the flashcopy logical drives or the filesystems snapped during the backup. Useful information for error analysis of the customer script is written into a separate log file. The location and name are also defined in the script as follows:

```
+47  # Log file
+48  LOGPREFIX="storage_snap"
+49  LOGPOSTFIX="AMT"
+50  LOG=/tmp/${LOGPREFIX}_${LOGPOSTFIX}.log
```

These variables should also be adapted to the environment.

The script uses temporary files to save information needed during processing. The file names are also defined in the script and normally need not be changed. For example, the file defined by $PHYS_VOLUMES (line 70) will store the local hdisk names to be flashcopied. It will be read line by line and the storage system flashcopy command will be executed. In this way, large lists of local hdisks can be handled. See the following lines:

```
# Tempfile names
TMPDIR=/tmp
TMP=${TMPDIR}/${LOGPREFIX}_${LOGPOSTFIX}.tmp
TMP_=${TMPDIR}/${LOGPREFIX}_${LOGPOSTFIX}.tmp_
BGND_RC_FILE=${TMPDIR}/${LOGPREFIX}_BGND_RC_${LOGPOSTFIX}.tmp
SM_CMD_OUT=${TMPDIR}/${LOGPREFIX}_SM_CMD_OUT_${LOGPOSTFIX}.tmp
SM_CMD_ERR=${TMPDIR}/${LOGPREFIX}_SM_CMD_ERR_${LOGPOSTFIX}.tmp
FILESYSTEMS=${TMPDIR}/${LOGPREFIX}_${LOGPOSTFIX}_fs.tmp
PHYS_VOLUMES=${TMPDIR}/${LOGPREFIX}_${LOGPOSTFIX}_hdisk.tmp
VOLGRP=${TMPDIR}/${LOGPREFIX}_${LOGPOSTFIX}_volgrp.tmp
```

During a DB2 backup using the Scripted Interface, it is possible to create a safe copy of the protocol file, which is implemented in this script. The path for the safe copy is also defined in the script. This variable might need to be adapted to the environment. See the following lines:

```
# Protocol Backup directory
PROT_BKP_DIR="/db2/db2amt/scriptACS/prot_bkp/"
```

Data Written to the Protocol File

The following lines are written into the protocol file when the script is called by DB2 ACS API and the action specified by the -a option is “snapshot”. The lines are then inserted by the customer script with the script function `storeSetting`.

Lines starting with the keyword USER_DB2FS save the names of the filesystems and their content type (DATA or LOG) included in the backup.

Lines starting with the keyword USER_FC save the flashcopied local hdisk name, the flashcopy logical drive name, its creation timestamp and the original AIX volumegroup included in the backup.

The following is an example protocol file after a successful flashcopy:

```
# db2ACSPrepare(): BEGIN [Fri Jan 31 15:02:46 2014]
USER_DB2FS=snap_DATA_ /db2/AMT/sapdata1 /dev/lvamtdat1
USER_DB2FS=snap_DATA_ /db2/AMT/sapdata2 /dev/lvamtdat2
USER_DB2FS=snap_DATA_ /db2/AMT/sapdata3 /dev/lvamtdat3
USER_DB2FS=snap_DATA_ /db2/AMT/sapdata4 /dev/lvamtdat4
USER_DB2FS=snap_DATA_ /db2/AMT/sapdata5 /dev/lvamtdat5
USER_DB2FS=snap_DATA_ /db2/AMT/db2amt/NODE0000 /dev/lvamtdbdir
# cmd: /db2/db2amt/acs_ibmstor.sh -a prepare -c
/db2/db2amt/scriptACS/repository/db2acs.AMT.0.db2amt.1391176966.cfg /db2/db2amt/scriptAC
S/repository FlashCopyTarget=/db2/db2amt/fc_mapping.txt
RC_PREPARE=0
# db2ACSPrepare(): END [Fri Jan 31 15:02:47 2014]
# ========================================================================
# db2ACSSnapshot(): BEGIN [Fri Jan 31 15:02:48 2014]
OBJ_ID=0
ACTION=DB2ACS_ACTION_WRITE
USER_FC=snap_DATA_ hdisk7 12_22_db6p2lp02_AMT_DATA_MJ-1 1/31/14_3:08_PM sapAMT3vg
# cmd: /db2/db2amt/acs_ibmstor.sh -a snapshot -c
/db2/db2amt/scriptACS/repository/db2acs.AMT.0.db2amt.1391176966.cfg /db2/db2amt/scriptA
CS/repository FlashCopyTarget=/db2/db2amt/fc_mapping.txt
RC_SNAPSHOT=0
```

```
# db2ACSSnapshot(): END [Fri Jan 31 15:03:06 2014]
# ============================================================================
```

The keywords “USER_DB2FS” and USER_RC” are also defined centrally in the beginning of the script by a variable. This variable might need to be adapted to the environment.

Data Read From Protocol File

The following lines are read from the protocol file when the script is called by the DB2 ACS API. Depending on the action specified by the -a option, some or all of them are evaluated. The lines are not changed by the customer script:

- Lines starting with the keyword DATAPATH_
- Lines starting with the keyword LOGPATH_
- Lines with the keyword DB2BACKUP_MODE, DB2BACKUP_LOGS
- Lines starting with the prefix RC_
- Lines with the keyword RESULT_x_FILE, where x is a variable number
- Line with the keyword OBJ_ID

Depending on how many lines are expected, they are read via the script function `getSetting` for a single line or with the UNIX `awk` command for multiple lines. These keywords are defined by the DB2 ACS API. Please refer to IBM documentation [1] for the detailed description of each keyword.

Keywords specific to this implementation of the customer script start with “USER_”. They are described in the section above

Data Read from Mapping File

The mapping file contains the mappings of logical drive names from the storage subsystem to the local hdisk names of the database server. It must always reflect the current mappings when the customer script is used. The filename is passed to the customer script using the snapshot OPTIONS parameter like described in the previous section of the document.
In the customer script first all options such as -a (for action), -c (for config file) , -o (for Object_id), -t (for timestamp) and their arguments are evaluated with the help of the `getopts` function. After that all remaining arguments are searched for the option labeled “FlashCopyTarget” and the mapping filename (lines 1427 – 1434 below).

```
...
+1418  # OPTIND (getopts) incl. program name, $# does not
+1419  shift `expr $OPTIND - 1`
+1420  debug_info "data: no. of arguments after shift: $#"
+1421
+1422  if [ $# -eq 0 ] # no 2ndary options
+1423  then
+1424    write_log "no external options from DB2 backup command"
+1425  else
+1426    # search in the remaining arguments for "FlashCopyTarget="
+1427    for i in $*
+1428    do
+1429       if [[ "$i" == FlashCopyTarget=* ]]
+1430       then
+1431          FlashCopyTarget=`echo $i | awk -F= '{ print \$2 }'`
```

```
+1432         write_log "Found $FlashCopyTarget as secondary option"
+1433      fi
+1434   done
+1435  fi
+1436  debug_info "data: parsing secondary options done"
...
+1440  # exit if Mapping file was not found
+1441  if [[ -z $FlashCopyTarget ]]
+1442  then
+1443         write_log "Specify mapping file in the USE SNAPSHOT clause"
+1444         write_log "e.g.  ... USE SNAPSHOT …  \
                        OPTIONS \"FlashCopyTarget=<mappingfile.txt>\" "
+1445         write_log "No mapping file found. Exiting."
+1446         exit $RC_MAPPING_ERROR
+1447  fi
...
```

Action Prepare

This chapter describes what the script executes when it is called with the -a prepare parameter by the DB2 ACS API. The main script uses the same coding as already explained in part 1 of the series. The function `doPrepare` handles preparation work for all operations, that are, snapshot, restore and delete.

In case the preparation is to be done for a snapshot (i.e. DB2 database backup), the following steps are performed:

- Check if the Storage Manager CLI executable exists (function `prepare_snapshot_command, line 889 below)`. If it does not exist, the action prepare is terminated with an error, causing the DB2 backup command to fail.
- Generate the list of file systems to be covered in the snapshot (function `get_used_file systems_for_backup, line 893` below). The list is generated according to the following logic:
    - First all lines starting with "DATAPATH_" are read from the protocol file.
    - If database log files should be included in the backup image, all lines starting with "LOGPATH_" are appended.
    - From this list of database paths and files, the associated file systems are retrieved with the AIX command `df -M`.
    - Typically, not every database path has its own file system. Therefore, this intermediate list most likely contains duplicates that must be removed. The final list is saved in a temporary file defined by variable $FILESYSTEMS. This file will then be used by the next function in the program flow "`get_used_pvs_for_backup`"
- Generate the list of local hdisks to be covered in the snapshot (function `get_used_pvs_for_backup, line 896` below) For each local hdisk a separate flashcopy must be taken. The list is generated according to the following logic:
    - For each entry in $FILESYSTEMS:
        - get the device name with the AIX command `df -M` (first field)
        - get the list of local hdisks for the device with the AIX command `lslv -l <device name>`

        - append the local hdisks to the list $TMP preceded by its content type (DATA or LOG)
    - eliminate duplicate entries in the list $TMP (where content type and hdisk name is equal)
    - store final of local hdisks in file $PHYS_VOLUMES
    - check for duplicate local hdisks in final list $PHYS_VOLUMES
        - if duplicates exists it means the prereq to separate DATA and LOG content is not fulfilled. The script will be terminated with error.

In case the preparation is to be done for a restore (i.e. DB2 database restore), the following steps are performed:

- Check if the Storage Manager CLI executable exists (function `prepare_snapshot_command, line 902` below). If it does not exist, the action prepare is terminated with an error, causing the DB2 backup command to fail.
- Next, the backup copies of the protocol files could be restored if needed. However, this is not implemented and should be done with care not to overwrite existing protocol files.
- Check if the target local hdisks (e.g. the hdisks containing the snapshot mirror or the flashcopy) can be used
    - the target local hdisks (column backup_hdisks in the mapping file) must be in state "Available". The output of AIX command `lsdev -l <hdisk>` is evaluated
    - the target local hdisks **must not** be part of a AIX volume group. The output of AIX command `lspv <hdisk>` is evaluated
    - If any of the above checks fail, the script is terminated with error
- The remaining restore logic is covered in the script function `doRestore` .

See the following excerpt (function doPrepare):

```
+865  function doPrepare
+866  ##################
...
+884
+885     case $operation in
+886        "SNAPSHOT")
+887           # prepare for action snapshot
+888           # check needed commands
+889           prepare_snapshot_command
+890           if_error "Error:  prepare_snapshot_command failed."
+891
+892           # get the filesystems and store them in file $FILESYSTEMS
+893           get_used_filesystems_for_backup
+894
+895           # get the used pvs for backup and store them in file $PHYS_VOLUMES
+896           get_used_pvs_for_backup
+897           if_error "Error:  get_used_pvs_for_backup failed."
+898           ;;
+899        "RESTORE")
+900           # prepare for action restore
+901           # check needed commands
+902           prepare_snapshot_command
+903           if_error "Error:  prepare_snapshot_command failed."
+904
+905           # check if target local hdisks are available
```

```
+906          check_hdisks_available
+907          if_error "Error:  check_hdisks_available failed."
+908
+909          # copy backup protocol files into place if the repository is empty
+910          # get_used_filesystems_for_restore, umount, restore, mount in doRestore
function
+911          ;;
+912       "DELETE")
+913          # prepare for deletion of snapshot images
+914          # check needed commands, currently not used
+915          ;;
+916        *)
+917          # default
+918          write_log "   Nothing specific to be prepared."
+919          ;;
+920    esac
```

Action Snapshot

This chapter describes what the script executes when it is called with the -a snapshot parameter by the DB2 ACS API. Function `doSnapshot` is used to handle the work for the action snapshot and uses the temporary file $PHYS_VOLUMES created by function `doPrepare` as input. The storage manager commands executed during the call of the script with action snapshot must complete within a timeout period (specified by variable $SM_TIMEOUT). This is done prevent long times in which the database is in write suspend mode.

- The file $PHYS_VOLUMES is read line by line (lines 1204 - 1274). Each line consists of two fields:
    - The first field is the label “snap_DATA_” or “snap_LOG_:”.
    - The second field is the name of the local hdisk, for example:
      ```
      snap_DATA_ hdisk7
      snap_LOG hdisk8
      ```
- For each local hdisk from $PHYS_VOLUMES the flashcopy logical volume name is looked up in the mapping file ($FlashCopyTarget). If none is found the script will be terminated with error
- The flashcopy command is prepared and executed in a child process (lines 1228 and 1232). The child process must complete within a timeout period, otherwise the child process is terminated.
- The timestamp of the flashcopy must be retrieved explicitly (lines 1239 and 1261)
- The current AIX volumegroup of the local hdisks is looked up via the AIX command `lspv <hdisk>`.
- The following information is written to the protocol file under keyword USER_FC (line 1268):
    - content type (DATA or LOG)
    - local hdisk name
    - flashcopy logical drive name
    - flashcopy timestamp
    - AIX volume group name

See the following lines:

```
+1190  function doSnapshot
```

```
+1191  #################
…
+1204     while read x
+1205     do
…
+1211        # look for $LOCAL_HDISK in file $FlashCopyTarget
+1212        # if FC_LOGICALDRIVE was not found, return with error
...
+1214        CMD="awk ' \$1 ~ /^$LOCAL_HDISK/ { print \$2 }' $FlashCopyTarget"
+1215        FC_LOGICALDRIVE=`eval $CMD`
+1216        if [[ -z $FC_LOGICALDRIVE ]]
+1217        then
+1218          write_log "****  ERROR: no FlashCopyLogicalDrive for $LOCAL_HDISK in
$FlashCopyTarget. Terminating $0."
+1219          RC=$RC_SNAP_ERROR
+1220          return $RC
+1221        fi
+1222
+1223        # recreate the flashcopy on a prepared storage flashcopy logical drive
+1224        # start command in background and wait in timeout_pid for max x seconds
...
+1226        write_log "$S FlashCopy ...."
+1227
+1228        SMCLI_RECREATE="$SMCLI $SM_HOST -c 'recreate Flashcopy LogicalDrive
[\"$FC_LOGICALDRIVE\"];' -p $SM_PASSWORD"
+1229        write_log "   recreate Flashcopy LogicalDrive $FC_LOGICALDRIVE"
+1230        CMD="sudo $SMCLI_RECREATE 1>$SM_CMD_OUT 2>$SM_CMD_ERR; echo \$?
>$BGND_RC_FILE "
...
+1232        eval $CMD &
+1233        timeout_pid $SM_TIMEOUT $!
+1234        if_error "Error: Storage Manager Command recreate FlashCopy exceeded
timeout. Exiting $0."
+1235        # check rc of flashcopy command
+1236        check_sm_result
+1237        if_error "Error: Storage Manager Command recreate FlashCopy failed. Exiting
$0."
+1238
+1239        # request the flashcopy creation timestamp from the storage manager
+1240        # start command in background and wait in timeout_pid for max 10 seconds
...
+1242        write_log "   Request Creation Timestamp ...."
+1243
+1244        SMCLI_VERIFY="$SMCLI $SM_HOST -c 'show LogicalDrive [\"$FC_LOGICALDRIVE\"];'
| awk '/Creation timestamp/ { print \$3 \" \" \$4 \" \" \$5
 }' "
+1245        CMD="sudo $SMCLI_VERIFY 1>$SM_CMD_OUT 2>$SM_CMD_ERR; echo \$?>$BGND_RC_FILE "
+1246        debug_info "data: starting $CMD"
+1247        eval $CMD &
+1248        timeout_pid $SM_TIMEOUT $!
+1249        if_error "Error: Storage Manager Command show LogicalDrive exceeded timeout.
Exiting $0."
+1250        # check result of showLogicalDrive command
+1251        check_sm_result
+1252        if_error "Error: Storage Manager Command show LogicalDrive failed. Exiting
$0."
+1253        # get timestamp saved in tempfile $SM_CMD_DOUT
+1254        CMD_OUT=`awk '{print $1}' $SM_CMD_OUT`
+1255        if ! [[ -z $CMD_OUT ]]
+1256        then
...
+1258           FC_TIMESTAMP=`echo $CMD_OUT | sed 's/ /_/g' `
+1259        else
+1260           FC_TIMESTAMP="unknown"
+1261        fi
+1262
+1263        # lookup current Volume Group name of LOCAL_HDISK
+1264        CMD="$LSPV ${LOCAL_HDISK} | awk -F: '/VOLUME GROUP/ { sub(/ */,\"\", \$3);
print \$3 }'"
+1265        VG_NAME=`eval $CMD`
+1266        if_error "Error: $CMD failed. Exiting $0."
```

```
+1267
+1268        # write local hdisk, current volumegroup name, flashcopy logical drive name
and
+1269        # timestamp to protocol file
…
+1271        storeSetting "$USER_FC=${TYPE} ${LOCAL_HDISK} ${FC_LOGICALDRIVE} $
{FC_TIMESTAMP} ${VG_NAME}"
+1272        write_log $D
+1273
+1274     done < $PHYS_VOLUMES
+1275
...
+1286     debug_info "exit: doSnapshot"
+1287  }
```

Action Restore

This chapter describes what the script executes when it is called with the -a restore parameter by the DB2 ACS API. Function `doRestore` is used to handle the work for the action restore. The mapping file `$FlashCopyTarget` is used retrieve the information which local hdisks to restore.

- As a first step in this function, the correct protocol file for the restore is retrieved by reading the keyword "RESULT_x_FILE from the current protocol file (lines 1013 – 1020).
- The protocol file for the restore is used to generate a list of file systems to be restored (function `get_used_file systems_for_restore, line 765`).
  - First, all filesystem names are stored into a temporary file. This is done by reading all lines starting with "USER_DB2FS" in the protocol file.
  - Afterwards, depending on the keyword "ACTION" in the restore protocol file, snapshot names containing the database log volumes are removed from the list.
  - The final list is stored in the temporary file $FILESYSTEMS again.
- Unmount filesystems retrieved from the restore protocol file (function `umount_filesystems,` line 1024)
- Retrieve the local hdisks that need be restored from the protocol file by looking up line with keyword USER_RC (function `get_used_pvs_for_restore,` line 1028). The list of hdisks to be restored is stored in $PHYS_VOLUMES.
- Reimport the target local hdisks (backup hdisks) into the original AIX volume group (function `restore_pvs`, line 1029)
  - export existing AIX volume groups, in order to recreate them with the target local hdisks
  - recreate the AIX volume groups with the target local hdisks. Look up the new target local hdisks in the mapping file $FlashCopyTarget.
- Mount filesystems listed in file $FILESYSTEMS again (function `mount_filesystems,` line 1034)
- In case of an error in one of the above steps, the script is terminated.

See the following excerpt:

```
 +997  #################
 +998  function doRestore
 +999  ##################
...
+1003  {
...
+1013     getSetting "OBJ_ID"
+1014     result_file_no=$_setting
...
+1018     key="RESULT_"${result_file_no}"_FILE"
+1019     getSetting $key
+1020     restoreConfig=$_setting
+1021
+1022     # unmount all filesystems
+1023     get_used_filesystems_for_restore
+1024     umount_filesystems
+1025     if_error "Error: $CMD failed"
+1026
+1027     # restore flashcopied pvs
+1028     get_used_pvs_for_restore
+1029     restore_pvs
+1030     if_error "Error: $CMD failed"
+1031
+1032     # mount all filesystems
+1033     get_used_filesystems_for_restore
+1034     mount_filesystems
+1035     if_error "Error: $CMD failed"
```

Action Verify

This chapter describes what the script is executing when it is called with the -a verify parameter by the DB2 ACS API. Function `doVerify` is used to handle the work for action verify.

- From the protocol file the lines starting with “USER_FC” are read and stored in a temporary file $TMP (lines 1323 - 1325). “USER_FC” is a user-defined keyword. The lines are written to the protocolfile during the call of the script with -a snapshot action.
- This temporary file is read line by line in a loop (lines 1328 – 1354).
    - Retrieve details of the flashcopy logical drive via the the storage manager command s`how LogicalDrive [FlashCopy Logical Drive”]`
    - The output of the command is parsed for the field “Status”. If the field is not “Optimal”, the variable TMP_RC is incremented by 1 (line 1348).
- After all flashcopy logical drives were verified in the loop, the return code is set. If one of the flashcopy images could not be verified, the return code of the script is set to $RC_VFY_ERROR (lines 1356 – 1364).

See the following excerpt:

```
+1304  ##################
+1305  function doVerify
+1306  ##################
+1307  # verifies the snapshot images created in doSnapshot
+1308  #
+1309  {
...
+1320
+1321     write_log "   verifying all Flashcopies ...."
+1322     debug_info "data: reading key $USER_FC from $config and store in $TMP"
+1323     CMD="awk -F= '\$1 == \"$USER_FC\" { print \$2 }' $config"
+1324     debug_info "data: using command: $CMD"
+1325     eval $CMD > $TMP
+1326
+1327     debug_info "data: reading FlashCopy LogicalDrive from $TMP"
+1328     while read x
+1329     do
+1330       # prepare SMcli show LogicalDrive command paramter Name
+1331       FC_LOGICALDRIVE=`echo $x | awk '{ print $3 }'`
+1332
+1333       SMCLI_VERIFY="$SMCLI $SM_HOST -c 'show LogicalDrive [\"$FC_LOGICALDRIVE\"];'\
          | awk '/Status/ { print \$2 }' "
+1334       CMD="sudo $SMCLI_VERIFY"
+1335       debug_info "data: $CMD"
+1336       # store the output of the command in CMD_OUT
+1337       CMD_OUT=`eval $CMD 2>> $LOG`
+1338       RC=$?
+1339
+1340       if [[ $RC -eq 0 ]]
+1341       then
+1342          if [[ $CMD_OUT == "Optimal" ]];
+1343          then
+1344             write_log "   Verification of Flashcopy $FC_LOGICALDRIVE succeeded."
+1345          fi
+1346       else
+1347          # increment counter, if command did not return "Optimal"
+1348          let "TMP_RC = TMP_RC + 1"
+1349          write_log "   Verification of Flashcopy $FC_LOGICALDRIVE failed."
+1350       fi
+1351
+1352       write_log "   LogicalDrive Status $FC_LOGICALDRIVE = \"$CMD_OUT\"."
+1353
+1354     done < $TMP
+1355
+1356     # set the final return code
+1357     if [[ $TMP_RC -eq 0 ]]
+1358     then
+1359         write_log "   All snapshots successfully verified."
+1360         RC=0
+1361     else
+1362         write_log "   **** ERROR: at least one snapshot image is not in state:
valid. rc=$RC_VFY_ERROR"
+1363         RC=$RC_VFY_ERROR
+1364     fi
+1365
+1366     # test to simulate doVerify failure
+1367     # RC=$RC_VFY_ERROR
+1368
+1369     TIMESTAMP=`date +'%Y%m%d%H%M%S'`
+1370     write_log "\nEnding $0 at $TIMESTAMP ."
...
+1374     return $RC
+1375  }
```

Action StoreMetadata

This chapter describes what the script executes when it is called with the -a store_metadata parameter by the DB2 ACS API. Function `doStoreMetaData` is used to handle the work for this action. In case the snapshot (doSnapshot) and the verification (doVerify) are successful, this is the last call of the script before DB2 will internally process all the remaining backup work. Therefore, in function doStoreMetaData the following tasks are performed:

- cleanup work for a successful snapshot backup (function `cleanup_tempfiles`, line 1064)
- Create the safe copy of the protocol file in a separate directory (lines 1067 – 1071).

See the following excerpt:

```
 +1050  ##################
 +1051  function doStoreMetaData
 +1052  ##################
 +1053  # performs post processing after successful backup
 +1054  {
…
 +1063     # cleanup
 +1064     cleanup_tempfiles
 +1065
 +1066     # save the protocol file
 +1067     CMD="cp $config $PROT_BKP_DIR"
 +1068     write_log "   Saving the protocol file $config"
 +1069     write_log "   to $PROT_BKP_DIR"
 +1070     debug_info "data: $CMD"
 +1071     eval $CMD >> $LOG
 +1072     # give a warning instead of ERROR in this phase
 +1073
 +1074     if [[ $? -ne 0 ]]
 +1075     then
 +1076        # copy failed, print a warning in $LOG,
 +1077        write_log "   WARNING **** : protocol file could not be saved"
 +1078     fi
…
 +1086  }
```

Action Rollback

This chapter describes what the script executes when it is called with the -a rollback parameter by the DB2 ACS API. Function `doRollback` is used to handle rollback work for all actions, i. e. snapshot, verify, store_metadata, and restore.

In case the rollback has to be done for a failed snapshot or verify (i.e. DB2 database backup), the following steps are performed:

- From the protocol file the lines starting with "USER_FC" are read and stored in a temporary file $TMP (lines 1222 - 1223). "USER_FC" is a user-defined keyword. The lines are written during the call of the script with -a snapshot action.
- This temporary file is read line by line in a loop (lines 1229 - 1237).
    - The flashcopy logical drive name and its creation timestamp are written to the customer script log file. **The flashcopies must be deleted or invalidated manually on the storage subsystem.**

In case the rollback has to be done for failed action store_metadata (i.e. DB2 database backup), the following step is performed:

- remove the safe copy of the protocol file made in `doStoreMetaData` (line 1252)

In case the rollback is to be done for failed action restore (i.e. DB2 database restore), the following steps are performed:

- try to mount all file systems again (line 1267)
- try to varyoff export the AIX volume group which was restored (line 1274)

At the end function `cleanup_tempfiles` is called to remove the temporary files used (line 1294).

See the following excerpt:

```
+1175  function doRollback
+1176  ##################
…
+1213
+1214     case $CMD_OUT in
+1215        "RC_SNAPSHOT" | "RC_VERIFY")
+1216           # perform clean up of failed flashcopy snapshot images
+1217           # delete all flashcopy snapshot images if doVerify returns an invalid
status
+1218           write_log "   Found: $CMD_OUT is != 0."
+1219           write_log "   Specific Rollback activities for failed SNAPSHOT, VERIFY"
+1220
+1221           write_log $S "listing failed flashcopy snapshot ...."
+1222           CMD="awk -F= '\$1 == \"$USER_FC\" { print \$2 }' $config"
+1223           eval $CMD > $TMP
+1224
+1225           if [[ -s $TMP ]]
+1226           then
+1227             # if the file $TMP exists and is greater than size 0
+1228             # means that at least one flashcopy snapshot is found for deletion
+1229             while read x
+1230             do
+1231                # print name and timestamp of flashcopy snapshot
+1232                FC_NAME=`echo $x | awk '{ print $3 }'`
+1233                FC_TIMESTAMP=`echo $x | awk '{ print $4 }'`
+1234
+1235                write_log "   Manually remove flashcopy snapshot:"
+1236                write_log "   $FC_NAME with timestamp $FC_TIMESTAMP"
+1237             done < $TMP
+1238           else
+1239              # No flashcopy snapshot found for deletion
+1240              write_log "   No flashcopy snapshot to be deleted."
+1241           fi
+1242           ;;
+1243        "RC_STORE_METADATA")
….
+1248           # construct path and filename for protocolfile backup
+1249           write_log "   Removing protocolfile  backup..."
+1250           CMD="`echo $config  |  awk -F/ '{print \$NF }'`"
+1251           write_log "   rm ${PROT_BKP_DIR}$CMD"
+1252           eval "rm ${PROT_BKP_DIR}${CMD} 2>> $LOG"
+1253
+1254           if [[ $? -ne 0 ]]
+1255           then
+1256               write_log "   WARNING: File ${PROT_BKP_DIR}$CMD could not be removed."
+1257           fi
+1258
+1259           write_log $D
+1260           ;;
+1261        "RC_RESTORE")
...
+1266           # get_used_filesystems_for_restore
+1267           umount_filesystems
+1268
+1269           if [[ $? -eq 0 ]]
+1270           then
+1271               debug_info "data: all fs unmounted. "
+1272
+1273               # varyoff and export volume groups
+1274               export_vgs
+1275               if_error "Error: export_vgs failed"
+1276           else
+1277               write_log "   WARNING: Filesystems could not be unmounted "
+1278               RC=$RC_RBCK_ERROR
```

```
+1279          fi
+1280
...
+1283          ;;
+1284        *)
+1285          # default, do nothing
+1286          write_log "   Nothing specific to rollback for failed step: $CMD_OUT"
+1287          ;;
+1288     esac
+1289
….
+1294     cleanup_tempfiles
+1295
…
+1301     return $RC
+1302  }
```

Action Delete

This chapter describes what the script executes when it is called with the -a delete parameter by the DB2 ACS API. Function `doDelete` is used to handle the work for action delete.

- From the protocol file the lines starting with “USER_FC” are read and stored in a temporary file $TMP (lines 968 - 969). “USER_FC” is a user-defined keyword. The lines are written during the call of the script with -a snapshot action.
- This temporary file is read line by line in a loop (lines 972 – 979).
    - The flashcopy logical drive name and its creation timestamp are written to the customer script log file. **The flashcopies must be deleted or invalidated manually on the storage subsystem.**

At the end function `cleanup_tempfiles` is called to remove the temporary files used (line 982).

See the following excerpt:

```
+935  function doDelete
+936  ##################
….
+943  {
…
+963     # look for the snapshot names in the backup protocol file (KEY=USER_FC)
+964     write_log "   Retrieving snapshots from protocol file $deleteConfig"
+965     write_log "   flashcopy snapshot will be removed from repository:"
+966     CMD="awk -F= '\$1 == \"$USER_FC\" {print \$2 }' $deleteConfig"
+967     debug_info "data: executing $CMD > $TMP"
+968     eval $CMD > $TMP
+969     if_error "Error: $CMD failed"
+970
+971     debug_info "data: reading from $TMP"
+972     while read x
+973     do
+974        # print name and timestamp of flashcopy snapshot
+975        FC_NAME=`echo $x | awk '{ print $3 }'`
+976        FC_TIMESTAMP=`echo $x | awk '{ print $4 }'`
+977        write_log "   $FC_NAME with timestamp $FC_TIMESTAMP"
+978
+979     done < $TMP
+980
+981     # cleanup
+982     cleanup_tempfiles
+983
...
+993     return $RC
+994   }
```

Problem Determination

In case the DB2 backup or restore command return SQL errors, the following files contain vital information. They should be analyzed in the following order:

1. The latest protocol file in the specified repository
   It contains the calls of the customer scripts with all parameters.

2. The log file of the customer script
   It is defined in the beginning of the script via variables, e.g. `/tmp/storage_snap_AMT.log`. The script also contains the variable `"DEBUG"`. If active, it writes useful debug information into the log. The log is appended each time the script is called. A call starts and ends with an entry like the following:

   ```
   =====================================
   == Starting customer script =========
   =====================================
   Starting doPrepare at 20130709142032 .
   …
   Ending doPrepare at 20130709142039 .
   =====================================
   == Ending customer skript ===========
   =====================================
   ```

   In most cases where DB2 error SQL2079N is reported during a failing backup or restore operation the log file contains the information about the problem encountered.

3. The DB2 diag log
   Search for strings like `sqluSnapshot` and their following entries.

Performance of Storage Manager Flashcopy

This chapter describes the different runtimes of a IBM Storage Manager flashcopy based backup compared to a traditional DB2 backup to disk. The database had no workload during the online backups.

The DB2 total backup size is approximately 60 GB.

Backup Type	**Runtime (in minutes)**
DB2 offline backup compressed	Approx. 60 mins
DB2 online backup compressed	Approx. 60 mins
DB2 offline backup with flashcopy	< 2 min
DB2 online backup with flashcopy	< 2 min

Restore Type	**Runtime**
DB2 restore	Approx. 60 mins
DB2 restore with flashcopy	Approx. 2 mins

The advantage of using the Scripted Interface with IBM Storage Manager flashcopy feature is obviously the faster runtime of both backup and restore compared to a traditional DB2 backup and restore. However, the flashcopy requires additional work on the storage subsystem side to build the backup images out of the flashcopy logical drive and base logical drive. This is time consuming an not counted into the backup or restore runtime.

Conclusion

This part of the series demonstrated the use of IBM Storage Manager Flashcopy feature with the Scripted Interface and DB2 ACS to back up and restore DB2 databases. It explained the actions of all operations in detail.

Additional virtual storage commands

This section describes the virtual storage commands used to prepare and manage DB2 backups via Scripted Interface. Tasks which are typically in responsibility of storage administration such as creating new logical drives or mapping of LUN ids to virtual or logical disks are not covered. These tasks must be completed before the customer script can be used.

By being able to display the mapping information you can validate if the mapping file used by the customer script reflects the correct mapping and flashcopy names. Storage Logical drive names, LUN ids, virtual and logical hdisk names are usually provided by storage administration.

Display the relationship between virtual hdisk, logical hdisk and storage logical drive

First list the LUN mappings from the storage subsystem to the VIO server. This is a storage manager command and executed on the database server. The LUN id of the command output is used to identify the logical hdisk on the VIO server which in turn represents the logical drive of the storage subsystem. In the example below storage LUN ID 22 (hex 16) is mapped to VIO logical disk hdisk25. This is reflected in the two rightmost columns in table 6 below.

```
Storage Manager command:

sudo /usr/SMclient/SMcli <primary_ip> <secondary_ip> -c 'show StorageSubsystem
lunMappings host ["<vio_server"];'

SAMPLE OUTPUT:

MAPPINGS (Storage Partitioning - Enabled (18 of 64 used))-------------------

   Logical Drive Name              LUN  Controller  Accessible by     Logical Drive status

   Access Logical Drive            31   A,B         Host vio_server  Optimal
...
   12_22_db_server_AMT_DATA_MJ     22   A           Host vio_server  Optimal
   12_22_db_server_AMT_DATA_MJ-1   26   A           Host vio_server  Optimal
   12_23_db_server_AMT_LOG_MJ-VC   23   B           Host vio_server  Optimal
   12_24_db_server_AMT_DATA_MJ-VC  25   B           Host vio_server  Optimal
   12_25_db_server_AMT_LOG_MJ      27   B           Host vio_server  Optimal
   12_25_db_server_AMT_LOG_MJ-1    24   B           Host vio_server  Optimal
   12_26_db_server_AMT_MJ          28   B           Host vio_server  Optimal
```

Next display the mapping between physical, logical, and virtual devices (VTD) on the VIO server. The command is issued on the vio server and its output contains

- the LUN id L16 of the storage logical drive in hexadecimal notation as substring of "Physloc".
- the logical hdisk name hdisk25 (backing device)
- the virtual target device name (VTD) AMT_DATA_MJ
- the LUN id of the virtual target device in hexadecimal notation (0x87)

```
VIO COMMAND:
```

```
lsmap -all

SAMPLE OUTPUT:

SVSA            Physloc                                          Client Partition ID
--------------- -------------------------------------------- ------------------
vhost1          U9117.MMA.10D359D-V1-C21                     0x00000004

VTD                     AMT_DATA_MJ
Status                  Available
LUN                     0x8700000000000000
Backing device          hdisk25
Physloc                 U789D.001.DQD52MY-P1-C4-T1-W204400A0B8472BAA-L16000000000000
```

Next display the mapping of virtual target device VTD of the VIO server to the virtual disks on the database server. The command is issued on the database server and its output contains

- the virtual disk name hdisk7
- the LUN id L87 in hexadecimal notation of the VTD of the VIO server as substring in the physical location code.

```
AIX COMMAND:

lscfg -l hdisk7

SAMPLE OUTPUT:

hdisk7           U9117.MMA.10D359D-V4-C21-T1-L8700000000000000  Virtual SCSI Disk Drive
```

LPAR **- virtual disk** **- LUN ID (hex)**	**VIO Server** **- virtual target device (VTD)** **- logical disk** **- LUN ID (hex)** **- physical location code LUN (hex)**	**DS4800** **- Logical Drive Name** **- LUN ID**
- hdisk7 - 0x87	- AMT_DATA_MJ - hdisk25 - 0x87 - L16 (hex 16 = dec 22)	- 12_22_db_server_AMT_DATA_MJ - 22
- hdisk8 - 0x89	- AMT_LOG_MJ - hdisk30 - 0x89 - L1B (hex 1B = dec 27)	- 12_25_db_server_AMT_LOG_MJ - 27
- hdisk9 - 0x8A	- AMT_MJ - hdisk31 - 0x8A - L1C (hex 1C = dec 28)	- 12_26_db_server_AMT_MJ - 28

Table 6: LUN – virtual – logical - disk mapping

Display the flashcopy logical drive names

The flashcopy logical drives are normally not mapped to the database server. However they can be displayed using the following storage manager command. The command output contains in section “FLASHCOPY LOGICAL DRIVES”

- the flashcopy logical drive name (e.g. `12_22_db_server_AMT_DATA_MJ-1`)
- the base logical drive name (e.g. `12_22_db_server_AMT_DATA_MJ`)

```
Storage Manager Command:

sudo /usr/SMclient/SMcli <primary_ip> <secondary_ip> -c 'show allLogicalDrives ;'

Sample Output:
...

FLASHCOPY LOGICAL DRIVES------------------------------

SUMMARY

   Number of flashcopy logical drives: 2
   NAME                          STATUS   CREATION TIMESTAMP
   12_22_db_server_AMT_DATA_MJ-1  Optimal  2/21/14 3:56 PM
   12_25_db_server_AMT_LOG_MJ-1   Optimal  2/21/14 3:56 PM

DETAILS

   FLASHCOPY LOGICAL DRIVE NAME: 12_22_db_server_AMT_DATA_MJ-1


      FlashCopy status:                              Optimal
      Creation timestamp:                            2/21/14 3:56 PM
      Associated base logical drive (standard):      12_22_db_server_AMT_DATA_MJ
      Associated flashcopy repository logical drive:  12_22_db_server_AMT_DATA_MJ-R1
...
```

Display the VolumeCopy target logical drive

The VolumeCopy target logical drive which can be used for a database restore can be displayed using the following storage manager command. The command output contains

- the source logical drive name (e.g. `12_22_db_server_AMT_DATA_MJ-1`)
- the target logical drive name (e.g. `12_22_db_server_AMT_DATA_MJ-VC`)

```
Storage Manager command:

sudo /usr/SMclient/SMcli <primary_ip> <secondary_ip> -c 'show VolumeCopy
allLogicalDrives;'

Sample output:

Copy pair: 12_22_db_server_AMT_DATA_MJ-1 and 12_24_db_server_AMT_DATA_MJ-VC

   Copy status:           Completed

   Start timestamp:       2/28/14 12:04:24 PM
   Completion timestamp:  2/28/14 12:07:10 PM
   Copy priority:         Highest
   Source logical drive:  12_22_db_server_AMT_DATA_MJ-1
       Logical Drive ID:  60:0a:0b:80:00:47:2c:a0:00:00:f5:b2:53:06:fe:f3
   Target logical drive:  12_24_db_server_AMT_DATA_MJ-VC
       Logical Drive ID:  60:0a:0b:80:00:47:2c:a0:00:00:f4:c0:52:9d:68:d0
       Read-only:         Enabled
```

Using this information and the commands explained above you can check if the mapping file used for a database restore reflects the correct mapping. In the example the volume copy was created by using the flashcopy logical drive `12_22_db_server_AMT_DATA_MJ-1` as the source. The target logical drive `12_24_db_server_AMT_DATA_MJ-VC` can be used for a database restore. It must be mapped to the database server as a virtual disk. Typically this is in responsibility of storage administration.

Literature

[1] IBM Redbook SG24-7010-06 : "IBM System Storage DS4000 and Storage Manager V10.30"

[2] "IBM System Storage DS3000, DS4000, and DS5000 Command Line Interface and Script Commands Programming Guide" GA32-0961-05

[3] IBM Power Systems Hardware documentation → Power Systems information → POWER6 Systems → Managing System Resources → Virtualization technologies → Virtual I/O Server

[4] IBM Redbook SG24-7822-01 "System Storage DS Storage Manager Copy Services Guide"

Disclaimer and Copyrights