Joern Klauke works as a software developer for SAP on DB2 for Linux, UNIX, and Windows at the IBM Research and Development Lab in Boeblingen (Germany). He has five years of experience with IBM and DB2, assisting customers with best practices, problem analysis and troubleshooting. He holds a degree in Computer Science from the Martin-Luther-University of Halle (Germany).

Part 2 – Implementation of the Scripted Interface for DB2 ACS using Linux LVM

The DB2 Advanced Copy Services support taking snapshots for backup purposes in DB2 LUW databases. Customers can use the DB2 ACS API either through libraries implemented by their storage hardware vendors or implement this API on their own. Until now only some vendors do. Additionally, it is a high effort for customers to implement the C-API of DB2 ACS.

DB2 10.5 introduces a new feature called Scripted Interface for DB2 Advanced Copy Services (DB2 ACS). This makes it possible that customers 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 independent from your storage hardware. Additionally, DB2 supports every storage hardware as soon as it becomes available on the market.

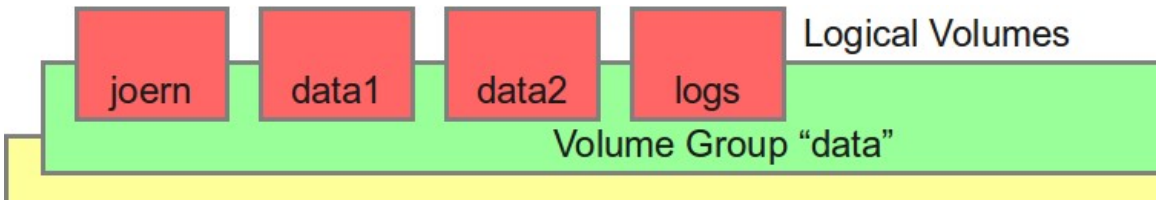
The feature supports all three architectures of DB2, enterprise server, multi-partitioned database using the database partitioning feature (DPF), and databases using pureScale. It is supported on all UNIX and Linux platforms DB2 is certified on.

In this series we will provide an introduction to this feature and present some real life examples in the coming parts. This is the second part of the series and demonstrates the Scripted Interface with the use of Linux LVM.

LVM is the Logical Volume Manager for the Linux Operating System. Among other feature, snapshots of logical volumes are supported. [3] This part of the series demonstrates snapshot backups of DB2 databases with the help of the Scripted Interface for DB2 ACS.

Introduction to LVM

The Linux Logical Volume Manager is a Linux kernel extension that introduces an abstraction layer over storage hardware. This increases the flexibility, e.g. it eases the resizing of partitions. Additionally, it allows to take snapshots of partitions in LVM volume groups.
For this article, the following layout will be used:



In the figure above you can see the physical volume */dev/sdb*. The LVM volume group *data* allocates this complete disk. This volume group contains the logical volumes *joern*, *data1*, *data2*, and *logs*. There is still free space in this volume group that will be used to take the snapshots of the logical volumes. The logical volumes are formatted with the ext4 file system.
The logical volumes are mounted in the following way:

Logical volume	**Mount point**	**Used for**
/dev/data/joern	/db2/joern	Database directory
/dev/data/data1	/db2/data1	Automatic storage path 1
/dev/data/data2	/db2/data2	Automatic storage path 2
/dev/data/logs	/db2/logs	Database logs

The following table shows a short overview which commands of the LVM utilities are used in this article and it gives a short conclusion what the command is used for:

Command	**Usage**
lvdisplay	display information on logical volumes. In particular the option –c is used to get a colon separated list of information including size. For an overview which columns are included take a look on the man page.
lvcreate	Create logical volumes in a volume group. This command is also used to create snapshot volumes for the logical volumes containing the storage paths and the log paths.
lvconvert	will be used to merge the snapshots back into the logical volumes that contain storage and log paths, that is, it will be used for restoring the snapshots.
lvchange	change attributes of a logical volume. In certain cases it will be necessary to deactivate and activate logical volumes to complete merges successfully.
lvremove	remove logical volumes, used during deletion of images.
vgdisplay	Querying information of volume groups. In particular the option –c is used to get a colon separated list of information including size. For an overview which columns are included take a look on the man page.

The LVM snapshots are created with *lvcreate* and the option *–s*. LVM works with copy on write, that is, when a block is first changed in the original volume an image of the contents of the block before the change is written to the snapshot. At the end the snapshot contains only these blocks that are changed in the original volume but with the contents that these blocks had during the point in time the snapshot was taken. This can be a small amount of data. But since we also want to be able to restore database that have potentially been dropped we recommend sizing the snapshot volumes in the same size as the original

volumes. If an LVM snapshot volume gets out of space Linux stops writing to it what in turn destroys the snapshot backup image.

Answering the privileges question

The Customer Script that is invoked by the Scripted Interface runs with the privileges of the instance owner. Typically, this user is not allowed to run commands that change the LVM configuration.
For this part of the series the privileges to run the commands were granted using sudo – the commands were added to the sudo configuration with the use of the command visudo with lines like the following:

```
jklauke ALL=NOPASSWD: /sbin/lvdisplay
```

With this the user no longer needs to enter the root password when the command is used.

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

Using LVM for Snapshot Backups with the Scripted Interface

This section will provide a sample implementation of a customer script using the abilities of Linux LVM to create, use and delete snapshots.
The following loop will be needed very often in the customer script:

```
110    for i in `grep "^DATAPATH" $config | \
111              awk -F= '{print $2}' | \
112              xargs -I\{\} df \{\} | \
113              grep '^/dev' | \
114              awk '{print $1;}' | \
115              uniq `
116    do
```

This gives us the set of logical volumes that needs to be backed up during the snapshot.
The following gives a step by step explanation of the command:

1. Taking the list of the paths from the protocol file – this time the data paths are returned (line 110) resulting in:
   ```
   DATAPATH=/db2/joern
   ```
2. Separating the path names from the option names (take from DATAPATH=path only the part of the equal-sign (line 111):
   ```
   /db2/joern
   ```
3. Translating the path to the logical volume on which the path resides (line 112):
   ```
   Filesystem            1K-blocks    Used   Available
   Use%   Mounted on
   /dev/mapper/data-joern   2064208  268980     1690372
    14%   /db2/joern
   ```
4. Taking only the lines that contain the logical volume names (line 113):
   ```
   /dev/mapper/data-joern   2064208  268980     1690372
    14%   /db2/joern
   ```

5. Taking only the name of the logical volume (line 114):
   `/dev/mapper/data-joern`
6. Taking each logical volume only once (line 115):
   `/dev/mapper/data-joern`

Backup

In the rest of the article the operations backup, restore, and delete will be explained in detail. The actions of the operations were already introduced in the first part of this series, that are – in case of backup – prepare, snapshot, verify and depending on the result of the verify call, storeMetaData or rollback.
Again, we recommend sizing the snapshot volumes in the same size as the original volume.

Prepare

The goal of the prepare action in the LVM script is to determine if there is enough free space in each volume group that every logical volume can be backed up in its volume group. For this purpose the function *doPrepare()* was implemented:

```
     65 doPrepare() {
     66 #
     67 # P R E P A R E
     68 #
     69 # ------------------------------------
     70    getSetting "OPERATION"
     71    operation=$_setting
     72    if [ $operation = "SNAPSHOT" ]
     73    then
     74       for group in `sudo vgdisplay -c | awk -F: '{print $1}'`
     75       do
     76          freespace=`sudo vgdisplay -c $group | \
     77                     awk -F":" '{print $16}'`
     78          cmd="egrep '^DATAPATH|^LOGPATH' $config | \
     79               awk -F= '{print \$2}' | \
     80               xargs -I\{\} df \{\} | \
     81               grep '^/dev' | \
     82               awk '{print \$1}' | \
     83               uniq | \
     84               xargs -I\{\} sudo lvdisplay -c \{\} | \
     85               awk -F: -v c=$group '\$2==c { SUM += \$8 } END
{print SUM}'"
     86
     87          echo "# cmd: "$cmd
     88          neededspace=`eval $cmd`
     89
     90          if [ $neededspace -gt $freespace ]
     91          then
     92             echo "# Goup: " $group
     93             echo "# Freespace: " $freespace
     94             echo "# Needed Space: " $neededspace
     95             RC=$RC_NOT_ENOUGH_SPACE
     96             break
     97          fi
     98       done
```

```
 99    fi
100 # ------------------------------------
101 }
```

1. Since it has first to be determined if currently snapshot is run the option OPERATION is read from the protocol file and checked if it is snapshot (lines 70 – 73).
2. Afterwards each volume group is checked – for every volume group in vgdisplay take the name of the volume group and separate the name from the given string (line 74).
3. Now, determine for the current volume group the free space – taking the vgdisplay output and parsing only the 16th column (line 76 and 77).
4. The next step is to look for all paths of the database that reside on this volume group. At first, the logical volumes are determined like shown in the previous section (lines 78 – 83).
5. For each of the logical volume take the information of lvdisplay (line 84).
6. Sum up the needed space for all logical volumes that reside in the current volume group (line 85).
7. Write the command to the protocol file for debugging purposes and run the command (lines 87 and 88).
8. At the end check if the need space for the snapshot in this volume group is smaller than the free space – if not return an error (lines 90 to 97).

Snapshot

Now that we are sure that we have enough space the snapshot itself is taken in the function *doSnapshot()*:

```
103 doSnapshot() {
104 #
105 # S N A P S H O T
106 #
107 # ------------------------------------
108    getSetting "TIMESTAMP"
109    timestamp=$_setting
110    for i in `grep "^DATAPATH" $config | \
111              awk -F= '{print $2}' | \
112              xargs -I\{\} df \{\} | \
113              grep '^/dev' | \
114              awk '{print $1;}' | \
115              uniq `
116    do
117       vol=`sudo lvdisplay -c $i | awk -F: '{print $1;}'| tr -d
' '`
118       echo "USER_VOLUME_DATA="$vol
119       snapName=`basename $vol`"_snap_"$timestamp
120       sudo lvcreate -s -n $snapName -l100%ORIGIN $vol
121    done
122    getSetting "DB2BACKUP_LOGS"
123    includeLogs=$_setting
124    if [ $includeLogs = "INCLUDE" -a $RC -eq 0 ]
125    then
126       for i in `grep "^LOGPATH" $config | \
127                 awk -F= '{print $2}' | \
```

```
    128                 xargs -I\{\} df \{\} | \
    129                 grep '^/dev' | \
    130                 awk '{print $1;}' | \
    131                 uniq `
    132       do
    133          vol=`sudo lvdisplay -c $i | awk -F: '{print $1;}'| tr
-d ' '`
    134          echo "USER_VOLUME_LOG="$vol
    135          snapName=`basename $vol`"_snap_"$timestamp
    136          sudo lvcreate -s -n $snapName -l100%ORIGIN $vol
    137       done
    138    fi
    139 # ---------------------------------------
    140 }
```

1. For this, take every logical volume that stores data in the first loop that follows the standard command (lines 110 to 116) – after having determined the timestamp of the image (lines 108 and 109).
2. Translate the device mapper to the real name of the logical volume (line 117).
3. For easier restoring write the logical volume that will be snapshot to the protocol file (line 118).
4. Generate the name of the new snapshot by appending the timestamp to the name of the original logical volume (line 119).
5. Run the snapshot (line 120).
6. The options of this command are the following:
   - *-s* – make a snapshot of the logical volume $vol
   - *-n* – gives the name of the new logical volume
   - *-l100%ORIGIN* – size the new logical volume with the same size as the original volume.
7. In the previous steps the snapshots of the data logical volumes were taken. Now it is time to take the snapshots of the log volumes if needed. First determine if it is needed (lines 122 to 125).
8. The following lines do the same for log volumes like was done for the data volumes – determine the logical volumes, store the logical volumes with the prefix USER_VOLUME_LOG for restore purposes, run the snapshot (lines 126 to 137).

Verify

```
    238 doVerify() {
    239 #
    240 # V E R I F Y
    241 #
    242 # ---------------------------------------
    243    mkdir /tmp/verify
    244    getSetting "TIMESTAMP" "" $oldConfig
    245    timestamp=$_setting
    246    for i in `grep "^USER_VOLUME_DATA" $config | \
    247              awk -F= '{print $2}'`
    248    do
    249       vol=$i"_snap_"$timestamp
    250       sudo mount $vol /tmp/verify
```

```
251       $RC=$?
252       sudo umount /tmp/verify
253       if [ $RC -neq 0 ]
254       then
255          echo "# Mounting of $vol failed"
256          break
257       fi
258       echo "# Volume $i checked"
259    done
260    getSetting "DB2BACKUP_LOGS"
261    includeLogs=$_setting
262    if [ $includeLogs = "INCLUDE" -a $RC -eq 0 ]
263    then
264       for i in `grep "^USER_VOLUME_LOG" $config | \
265                 awk -F= '{print $2}'`
266       do
267          vol=$i"_snap_"$timestamp
268          sudo mount $vol /tmp/verify
269          $RC=$?
270          sudo umount /tmp/verify
271          if [ $RC -neq 0 ]
272          then
273             echo "# Mounting of $vol failed"
274             break
275          fi
276          echo "# Volume $i checked"
277       done
278    fi
279    rmdir /tmp/verify
280 # ------------------------------------
281 }
```

To find out of the snapshot went right the script tries to mount every snapshot logical volume, checks if this was successful and unmounts it again. The step by step explanation:

1. Create a temporary directory that functions as the mount point in the directory structure of the system (line 243).
2. Read the timestamp needed to generate the snapshot name from the protocol file (lines 244 and 254).
3. For every data volume in the protocol file (lines 246 to 248) generate the name of the snapshot volume (line 249) and try to mount it to the temporary directory (line 250).
4. Check if the mounting was successful (line 251).
5. Unmount the volume (line 252).
6. If mounting the volume failed throw an error (lines 253 to 257).

If log files had to be included the script has to check if the snapshot of these volumes was successful:

1. Read the option for include logs from the protocol file (lines 260 and 261).
2. Check if logs had to be included and if the mounting of the data volumes was successful (line 262).
3. For every log volume in the protocol file (lines 264 and 265), generate the name of the snapshot volume (line 267)

4. Try to mount the volume (line 268)
5. Track the success of mounting the volume (line 269) and unmount it (line 270).
6. Check the result of mounting and throw an error if it failed (lines 271 to 275).

At the end the temporary directory is removed (line 280).

Rollback

If during the verify action an error occurred, that is, if at least one of the snapshot volumes could not be mounted the next action to be called is rolback. This is implemented in the following way:

```
295 doRollback() {
296 #
297 #  R O L L B A C K
298 #
299 # -------------------------------------
300    getSetting "OPERATION"
301    operation=$_setting
302    if [ $operation = "SNAPSHOT" ]
303    then
304       for i in `grep "^VOLUME_DATA" $config | \
305                 awk -F= '{print $2}'`
306       do
307          sudo lvremove $i_snap_$timestamp
308       done
309       getSetting "DB2BACKUP_LOGS"
310       includeLogs=$_setting
311       if [ $includeLogs = "INCLUDE" -a $RC -eq 0 ]
312       then
313          for i in `grep "^VOLUME_LOG" $config | \
314                    awk -F= '{print $2}'`
315          do
316             sudo lvremove $i_snap_$timestamp
317          done
318       fi
319    fi
320 # -------------------------------------
321 }
```

Essentially, it deletes all snapshot volumes that should be existent by running the following steps:

1. Check if this is a rollback of a snapshot operation (lines 300 to 303).
2. Loop across all data volumes given in the protocol file (lines 304 to 306).
3. Remove the logical volume (line 307) with lvremove. This takes only the volume name as a parameter.
4. Check if there should be snapshot volumes of the logs (lines 309 to 312).
5. Loop across the log volumes and remove them (line 316).

StoreMetaData does not run any command. It could save the protocol file to any other backup infrastructure. As already stated in the first article the protocol file is essential to be able to restore the image. At the point when storeMetaData is called no needed information is added to the protocol file anymore, that is, it can be used in this state.

Restore

The purpose of the restore is obvious, merge the snapshot volumes back to the original volumes, that is, move every original block from the snapshot over the changed in the original volume. Let us take a look on the implementation:

```
142 doRestore() {
143 #
144 # R E S T O R E
145 #
146 # -------------------------------------
147
148    getSetting "OBJ_ID"
149    id=$_setting
150    # Construct key to search for in currenct protocol file
151    key="RESULT_"$id"_FILE"
152    getSetting $key
153    oldConfig=$_setting
154
155    getSetting "TIMESTAMP" "" $oldConfig
156    timestamp=$_setting
157    for i in `grep "^USER_VOLUME_DATA" $oldConfig | \
158              awk -F= '{print $2}'`
159    do
160       vol=$i"_snap_"$timestamp
161       echo "# Unmounting volume $vol"
162       sudo umount -f $i
163       echo "# Merging volume $vol"
164       sudo lvconvert --merge --background $vol
165       if [ $? -neq 0 ]
166       then
167          echo "# Deactivating volume $vol"
168          sudo lvchange -an $i
169          echo "# Activating volume $vol"
170          sudo lvchange -ay $i
171       fi
172       echo "# Mounting volume $vol"
173       sudo mount $i
174       echo "# Take the backup of volume $vol again"
175       sudo lvcreate -s -n $vol -l100%ORIGIN $i
176    done
177    # if logs included
178    getSetting "ACTION"
179    readAction=$_setting
180    if [ $readAction = "DB2ACS_ACTION_READ_BY_OBJECT" ]
181    then
182       for i in `grep "^USER_VOLUME_LOG" $oldConfig | \
183                 awk -F= '{print $2}'`
184       do
185          vol=$i"_snap_"${timestamp}
186          echo "# Umounting volume $vol"
187          sudo umount -f $i
188          echo "# Merging volume $vol"
189          sudo lvconvert --merge --background $vol
190          if [ $? -neq 0 ]
191          then
192             echo "# Deactivating volume $vol"
```

```
193             sudo lvchange -an $i
194             echo "# Activating volume $vol"
195             sudo lvchange -ay $i
196          fi
197          echo "# Mounting volume $vol"
198          sudo mount $i
199          echo "# Take the backup of volume $vol again"
200          sudo lvcreate -s -n $vol -l100%ORIGIN $i
201       done
202    fi
203 # ------------------------------------
204 }
```

During the restore there are two protocol files opened – the protocol file of the current restore operation that is still changed and the protocol file of the snapshot backup operation, opened read only. The second one has first to be opened. To do this, the object ID is read from the restore protocol file (lines 148 and 149), the option name is generated (line 151), and the name of the backup protocol file is read (lines 152 and 153). Now, the restore from this protocol file and the corresponding image:

1. Read the timestamp needed to generate logical volume names from the protocol file (lines 155 and 156).
2. Loop across every data volume that can be found in the protocol file (lines 157 to 176).
3. Generate the snapshot volume name (line 160).
4. Unmount the original logical volume (line 162).
5. Merge the snapshot volume (line 164).
6. If this was not successful it might be necessary to deactivate and activate the volume again (lines 165 to 171).
7. Mount the merged original volume (line 173).
8. Since merging removes the snapshot volume it is necessary to recreate right away (line 175).

Now it is time to take care of log volumes:

1. Read the option action from the restore protocol file (line 178 and 179). Depending on this the log volumes have to be restored – if its value is DB2ACS_ACTION_READ_BY_OBJECT the restore is needed, if its value is DB2ACS_ACTION_READ_BY_GROUP it is not needed.
2. In the following steps the log volumes are restored taking the same steps as for restoring the data volumes (lines 182 to 201).

Delete

The action delete removes the snapshot volumes from the corresponding volume groups by running the following steps.

```
206 doDelete() {
207 #
208 # D E L E T E
209 #
210 # ------------------------------------
211    getSetting "RESULT_"${objectId}"_FILE"
212    oldConfig=$_setting
213    getSetting "TIMESTAMP" "" $oldConfig
```

```
214    timestamp=$_setting
215    for i in `grep "^USER_VOLUME_DATA" $oldConfig | \
216              awk -F= '{print $2}'`
217    do
218       vol=$i"_snap_"${timestamp}
219       echo "# Volume $vol"
220       echo "# "`sudo lvremove -f $vol`
221    done
222    getSetting "DB2BACKUP_LOGS" "" $oldConfig
223    includeLogs=$_setting
224    if [ $includeLogs = "INCLUDE" ]
225    then
226       for i in `grep "^USER_VOLUME_LOG" $oldConfig | \
227                 awk -F= '{print $2}'`
228       do
229          vol=$i"_snap_"${timestamp}
230          echo "# Volume $vol"
231          echo "# "`sudo lvremove -f $vol`
232       done
233    fi
235 # ------------------------------------
236 }
```

1. The object IDs are now given by the command that invoked the script by providing the option –o. The name of the option in the delete restore protocol file of the backup protocol file can be generated this way (line 211) and read (line 212).
2. Now, the timestamp can be read that is needed to generate the names of the snapshot logical volumes (lines 213 and 214).
3. Loop over all data volumes (lines 215 to 221), generate the name of the snapshot volume (line 218) and remove the volume (line 220).
4. Check if the backup image contained log files, that is, check if the value of DB2BACKUP_LOGS is set to INCLUDE in the backup protocol file.
5. If so, remove these snapshot volumes by generating their names (line 229), and removing them (line 231) in the loop (lines 226 to 232).

Conclusion

This part of the series on the Scripted Interface for DB2 ACS demonstrated the use of LVM to create snapshot images of DB2 databases. It explained all actions of all operations in detail.