Tivoli software

IBM

Redpaper

Hélder Garcia

IBM Tivoli Storage Manager: How to Migrate the Library Manager Function

This IBM® Redpaper provides guidance and practical procedures for migrating the library manager role from one existing IBM Tivoli® Storage Manager server instance to another. Scripts that help automate certain details are also provided.

Basic information was extracted from *Get More Out of Your SAN with IBM Tivoli Storage Manager*, SG24-6687.

Overview

Figure 1 shows our scenario, which has Tivoli Storage Manager server instances using library sharing. In the current setup, the instance called NATAL plays the role of library manager and the instance called RECIFE is the library client.

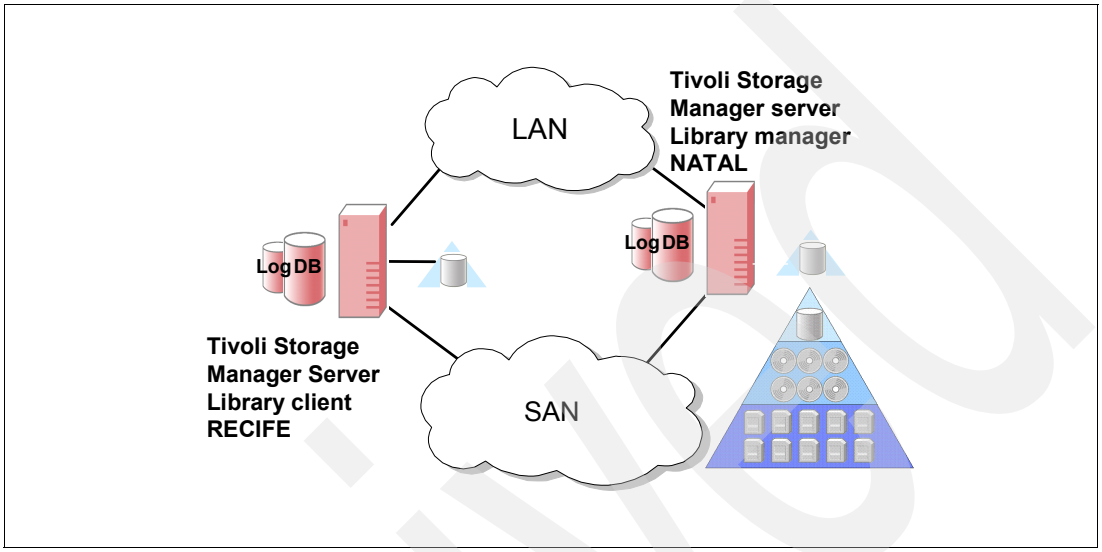

Figure 1 Library sharing configuration

The objective is to change the relationship between the two servers, so that RECIFE becomes the library manager and NATAL is demoted to a library client. This can be a complex procedure for an environment that is already in production, where hundreds or even thousands of tape volumes are currently tracked by the library sharing solution.

The library defined on server NATAL is a 3494 library (Example 1). From the library manager, NATAL, we ran the QUERY LIBRARY FORMAT=DETAIL command. We recommend saving this information for future reference.

Example 1 Library definition on the library manager

```
tsm: NATAL>query library f=d
Library Name: 3494
Library Type: 349X
ACS Id:
Private Category: 701
Scratch Category: 700
WORM Scratch Category:
External Manager:
Shared: Yes
LanFree:
ObeyMountRetention:
Primary Library Manager:
WWN:
Serial Number:
AutoLabel:
Reset Drives: Yes
```

Example 2 on page 3 shows the current list of library volumes on NATAL.

Example 2 List of volumes in the library manager inventory

```
tsm: NATAL>query libvolumes
Library Name        Volume Name        Status     Owner       Last Use
------------        -----------        ---------- --------    ----------
3494                J11771             Private    NATAL       DbBackup
3494                J11772             Scratch
3494                J11773             Private    RECIFE      Data
```

Example 3 shows the current volume history from NATAL.

Example 3 Volume history on current library manager

```
tsm: NATAL>query volhistory
Date/Time: 28.11.2007 17:54:54
Volume Type: BACKUPFULL
Backup Series: 4
Backup Operation: 0
Volume Seq: 1
Device Class: 3494_3592C
Volume Name: J11771
Volume Location:
Command:
Date/Time: 29.11.2007 17:47:24
Volume Type: REMOTE
Backup Series:
Backup Operation:
Volume Seq:
Device Class: 3494_3592C
Volume Name: J11773
Volume Location: RECIFE
Command:
```

Let's look at RECIFE. In a typical library client, the library type is defined as *shared* and the field *Primary Library Manager* specifies the name of the library manager. Example 4 shows the library definition on RECIFE.

Example 4 Library definition on the library client

```
tsm: RECIFE>query library f=d
Library Name: 3494
Library Type: SHARED
ACS Id:
Private Category:
Scratch Category:
WORM Scratch Category:
External Manager:
RSM Media Type:
Shared: No
LanFree:
ObeyMountRetention:
Primary Library Manager: NATAL
WWN:
Serial Number:
AutoLabel:
Reset Drives: No
```

Example 5 shows the list of volumes used on RECIFE.

Example 5 Volumes on library client

```
tsm: RECIFE>query volumes
Volume Name   Storage       Device          Estimated    Pct Volume
              Pool Name     Class Name      Capacity     Util         Status
-----------   -----------   ----------      ---------    -----------  -------
J11773        TAPE_3494     3494_3592C      900,000.0    0.4          Filling
```

Example 6 shows an entry in the volume history for RECIFE.

Example 6 Volume history on library client

```
tsm: RECIFE>query volhistory
Date/Time: 29.11.2007 17:50:24
Volume Type: STGNEW
Backup Series:
Backup Operation:
Volume Seq:
Device Class: 3494_3592C
Volume Name: J11773
Volume Location:
```

The information collected from Example 1 on page 2 to Example 6 illustrates three possible states of a volume in our scenario (Table 1).

Table 1 Possible volume states

Volume	State
J11771	Volume allocated by NATAL for a Tivoli Storage Manager database backup. The current owner of this volume is NATAL. It appears in the library inventory of NATAL because it is the library manager. This information can be found in the LIBVOLUMES table.
J11772	Scratch volume, containing no valid data. No owner is associated with this kind of volume. It appears in the library inventory of NATAL because it is the library manager. This information can be found in the LIBVOLUMES table.
J11773	Volume allocated by RECIFE as a volume in a storage pool for backup or archive data. The current owner of this volume is RECIFE. It appears in the library inventory of NATAL because it is the library manager. This information can be found in the LIBVOLUMES table. In the VOLHISTORY table, in the library client (RECIFE), this volume type is STGNEW, showing that it was allocated to a locally defined storage pool. On the library manager (NATAL), this volume is identified as REMOTE, indicating that it does not belong to NATAL.

The LIBVOLUMES table keeps information about the library inventory, such as volume name, owner, and last used. This table tracks use of what we call *library volume* (or libvolume).

The VOLUMES and VOLHISTORY tables track volume information also, but from a different perspective. They track how the volume is used based on policies that are defined in the Tivoli Storage Manager environment, such as which storage pool it was allocated to, current occupation of the volume, and device class. Therefore, the LIBVOLUMES table on the library manager keeps the list of volumes in the library, because the library manager is responsible for managing media allocation and mount requests, for example. The LIBVOLUMES tables of the library clients are empty; however, each Tivoli Storage Manager instance that runs backup and archive operations makes use of its own VOLUMES and VOLHISTORY tables.

The rest of this document details the steps for moving the role of library manager from server NATAL to server RECIFE. In summary, these steps are:

1. Prepare target server.
2. Check communication.
3. Prevent client activities.
4. Back up Tivoli Storage Manager server databases.
5. Gather volume inventory from source server.
6. Check out volumes from source server.
7. Remove library definition from target server.
8. Define library on target server.
9. Check in volumes on target server.
10. Redefine library on source server.
11. Update source volume history.
12. Update additional library clients.

Preparing the target server

The steps for preparing the target server are:

1. Define server to server communication. In a production environment, it is likely that server to server communications are already defined and working properly. However, if you are creating a new instance to become the library manager, you can use the commands in Example 7 for RECIFE to define this communication.

Example 7 Defining server to server communication on a new Tivoli Storage Manager instance

```
tsm: RECIFE>set serverpassword secret
ANR2131I Server password set.
tsm: RECIFE>set servername RECIFE
ANR2094I Server name set to RECIFE.
tsm: RECIFE>set serverhladdress 9.11.213.9
ANR2132I Server hladdress set to 9.11.213.9.
tsm: RECIFE>set serverlladdress 1500
ANR2133I Server lladdress set to 1500.
tsm: RECIFE>define server natal serverpassword=secret HLADDRESS=9.11.213.110
lladdress=1500 crossdefine=yes
ANR1660I Server NATAL defined successfully.
```

2. Redefine storage agents. If storage agents are in your environment, you must redefine all of them as server entities in the new library manager. This can be time consuming if you do it manually. These scripts can help speed up this task:
   a. Collect all server definitions for the current library manager, NATAL, using a SELECT statement such as one that lists all definitions with the server name ending in suffix 'STA' (Example 8). If you have a similar naming convention in your environment for storage agents, you can specify this in the SELECT statement.

Example 8 SELECT statement for collecting storage agent server definitions

```
tsm: NATAL>select server_name,hl_address,ll_address from servers where server_name
like '%STA'
AGENT1_STA   192.168.4.15   1550
AGENT2_STA   192.168.4.16   1550
AGENT3_STA   192.168.4.17   1550
```

   b. Redirect the output to a text file (Example 9 on page 6).

Example 9 Redirecting SELECT storage agent server definitions to a text file

```
tsm: NATAL>select server_name,hl_address,ll_address from servers  where
server_name like '%STA' > servers.txt
Output of command redirected to file 'SERVERS.TXT'
```

c. Use the script *gen_servers.pl*, provided in "gen_servers.pl" on page 19 to generate a macro to define all servers included in the text file. Its syntax is:

perl gen_servers.pl <define|delete> <*inputfile*> <*outputfile*>

The *inputfile* parameter is a text file that contains one entry per line with three items: server name, IP address, and port number. No header line is expected, so you must edit the file to remove the header line if one is present before running the script.

The *outputfile* parameter is the file you want to write the macro commands. This macro is called later from the Tivoli Storage Manager Administrative CLI.

d. Generate a macro for the servers collected. In our scenario, we did this for the servers collected in the SERVERS.TXT file, using the *define* parameter (Example 10). The output is written to the define_servers.mac file.

Example 10 Running the gen_servers.pl script

```
# perl gen_servers.pl  define SERVERS.TXT define_servers.mac
```

Important: Each server definition has a password associated with its entry. The script generates commands to define the servers with passwords that are identical to the server names. You might have to edit the macro file to update the passwords accordingly before running the script.

e. Run the macro on RECIFE to define the servers on the future library manager (Example 11).

Example 11 Running the macro to create storage agent server definitions on RECIFE

```
tsm: RECIFE>macro define_servers.mac > define_servers_output.txt
Output of command redirected to file 'DEFINE_SERVERS_OUTPUT.TXT'
```

3. Recreate the path definitions to these servers by following this process:

a. Collect the definitions on NATAL (Example 12).

Example 12 Collecting path definitions for storage agents

```
tsm: NATAL>select source_name,destination_name,source_type,destination_type,
library_name,device,online from paths where source_name like '%STA'
AGENT1_STA   3494_03    SERVER    DRIVE3494    /dev/tsmscsi/mt9    YES
AGENT2_STA   3494_04    SERVER    DRIVE3494    /dev/tsmscsi/mt2    YES
AGENT3_STA   3494_05    SERVER    DRIVE3494    /dev/tsmscsi/mt3    YES
```

b. Redirect the output to a text file (Example 13).

Example 13 Redirecting storage agents path definitions to a text file

```
tsm: NATAL>select source_name,destination_name,source_type,destination_type,
library_name,device,online from paths where sourc_name like '%STA' > paths.txt
Output of command redirected to file 'PATHS.TXT'
```

c. Use the *gen_paths.pl* script that is provided in "gen_paths.pl" on page 20 to generate a macro to define all paths that are included in the text file. Its syntax is:

```
perl gen_paths.pl <define|delete> <inputfile> <outputfile>
```

The *inputfile* parameter is a text file that contains one entry per line with the following items: server name, drive name, source type, destination type, library name, device name, and online status. No header line is expected, so you must edit the file to remove the header line if present before running the script.

The *outputfile* parameter is the file you want to write the macro commands. This macro is called later from the Tivoli Storage Manager Administrative CLI.

d. Generate a macro for the paths collected in the PATHS.TXT file, using the *define* parameter (Example 14). The output is written to the define_paths.mac file.

Example 14 Running the gen_paths.pl script to create a macro to generate paths

```
perl gen_paths.pl  define PATHS.TXT define_paths.mac
```

You cannot run the *define_paths.mac* macro yet because a library must be defined and at this point, we have not defined the library on RECIFE.

e. Generate a macro to delete the storage agent paths that are no longer needed on NATAL using the *gen_paths.pl* script and specifying the *delete* parameter (Example 15).

Example 15 Running the gen_paths.pl script to create a macro to delete paths

```
perl gen_paths.pl  delete PATHS.TXT delete_paths.mac
```

f. You can run the generated macro on NATAL to delete the paths (Example 16).

Example 16 Deleting storage agent paths on the library manager

```
tsm: NATAL>macro delete_paths.mac > delete_paths_output.txt
Output of command redirected to file 'DELETE_PATHS_OUTPUT.TXT'
```

Checking communication

Example 17 shows how to verify that both servers can communicate with each other, using a **PING SERVER** command on each server.

Example 17 Testing server to server communication

```
tsm: RECIFE>ping server natal
ANR1706I Ping for server 'NATAL' was able to establish a connection.

tsm: NATAL>ping server recife
ANR1706I Ping for server 'RECIFE' was able to establish a connection.
```

This check is very important.

You can also verify the communication from the servers to the storage agents. Both the library manager and the library client must be able to communicate with the storage agents if the backup and archive node that uses the storage agent is registered at the library client instance.

Disabling client activities

From this point on, all client activities must be disabled. It is also a good idea to deactivate any administrative schedule that can make use of library volumes. To disable any new client connections, use the command in Example 18 on NATAL and RECIFE and wait for the current sessions to finish their jobs or cancel them. If you have other library clients, disable the client sessions there also.

Example 18 Disabling client sessions on all Tivoli Storage Manager instances

```
tsm: NATAL>disable session client
ANR2553I Server now disabled for Client access.
tsm: RECIFE>disable session client
ANR2553I Server now disabled for Client access.
```

Backing up Tivoli Storage Manager server databases

Back up all your Tivoli Storage Manager databases, volume history, and device configuration files. Do not proceed until all backups have completed successfully. Example 19 shows the back up commands for NATAL. Repeat them for all Tivoli Storage Manager instances in your environment, specifying a suitable device class.

Example 19 Backing up database, volume history, and device configuration files

```
tsm: NATAL>backup db devclass=3494_3592c type=full
ANR2280I Full database backup started as process 5.
ANS8003I Process number 5 started.
tsm: NATAL>backup volhistory filename=volhist
Do you wish to proceed? (Yes (Y)/No (N)) y
ANR2462I BACKUP VOLHISTORY: Server sequential volume history information was
written to volhist.
tsm: NATAL>backup devconfig filename=device
Do you wish to proceed? (Yes (Y)/No (N)) y
ANR2393I BACKUP DEVCONFIG: Server device configuration information was written to
device.
```

Gathering volume inventory from the source server

Follow these steps to gather the volume inventory from the source server:

1. Collect a list of all volumes from the inventory on the current library manager, which is NATAL, using a SELECT statement (Example 20).

Example 20 Collecting a list of volumes from the library manager inventory

```
tsm: NATAL>select LIBRARY_NAME,VOLUME_NAME,STATUS,OWNER,LAST_USE from libvolumes
LIBRARY_NAME   VOLUME_NAME   STATUS   OWNER      LAST_USE
-------------  ------------- ------   -----      ----------
3494           J11771        Private  NATAL      DbBackup
3494           J11772        Scratch
3494           J11773        Private  RECIFE     Data
3494           J11774        Private  NATAL      DbBackup
```

2. Separate this list by owner. The commands in Example 21 run several queries, redirecting the output to text files, to generate the list of scratch volumes, volumes owned by NATAL, and volumes owned by RECIFE. If you have more library clients in your environment, add the proper queries for these additional instances. The last command saves a list with all volumes to use in the check out process.

Example 21 Saving lists of volumes per owner

```
tsm: NATAL>select LIBRARY_NAME,VOLUME_NAME from libvolumes
where STATUS='Scratch' > scratch_libvolumes.txt
Output of command redirected to file 'SCRATCH_LIBVOLUMES.TXT'

tsm: NATAL>select LIBRARY_NAME,VOLUME_NAME from libvolumes
where OWNER='NATAL' > source_libvolumes.txt
Output of command redirected to file 'SOURCE_LIBVOLUMES.TXT'

tsm: NATAL>select LIBRARY_NAME,VOLUME_NAME from libvolumes
where OWNER='RECIFE' > target_libvolumes.txt
Output of command redirected to file 'TARGET_LIBVOLUMES.TXT'

tsm: NATAL>select LIBRARY_NAME,VOLUME_NAME from libvolumes
 > all_libvolumes.txt
Output of command redirected to file 'ALL_LIBVOLUMES.TXT'
```

Checking out volumes from the source server

To check out volumes from the source server, follow these steps:

1. Use the all_libvolumes.txt file (see Example 21) as input to the *gen_checkout.pl* script to create a macro (Example 22) with commands to check out all library volumes from NATAL.

Example 22 Generating a macro to check out all library volumes from the library manager

```
perl gen_checkout.pl  ALL_LIBVOLUMES.TXT all_checkout.mac
```

Example 23 shows an example of the commands included in *all_checkout.mac*.

Example 23 Sample commands to check out volumes included in the macro

```
checkout libv 3494 J11771 remove=no checkl=no
checkout libv 3494 J11772 remove=no checkl=no
checkout libv 3494 J11773 remove=no checkl=no
checkout libv 3494 J11774 remove=no checkl=no
```

2. Run the macro on NATAL (Example 24) to clean the library inventory.

Example 24 Running the macro to check out volumes on the library manager

```
tsm: NATAL>macro all_checkout.mac
```

3. Monitor the activity log or use the administrative CLI with the *-consolemode* option to see the checkout processes issued and executed. These look similar to those in Example 25 on page 10.

Example 25 Console output while running checkout macro

```
ANR2017I Administrator ADMIN issued command: CHECKOUT libv 3494 J11771 remove=no
checkl=no
ANR0984I Process 6 for CHECKOUT LIBVOLUME started in the BACKGROUND at 15:10:25.
ANR8434I CHECKOUT LIBVOLUME: Operation for volume J11771 in library 3494 started
as process 6.
ANR2017I Administrator ADMIN issued command: CHECKOUT libv 3494 J11772 remove=no
checkl=no
ANR0984I Process 7 for CHECKOUT LIBVOLUME started in the BACKGROUND at 15:10:25.
ANR8434I CHECKOUT LIBVOLUME: Operation for volume J11772 in library 3494 started
as process 7.
ANR2017I Administrator ADMIN issued command: CHECKOUT libv 3494 J11773 remove=no
checkl=no
ANR0984I Process 8 for CHECKOUT LIBVOLUME started in the BACKGROUND at 15:10:25.
ANR8434I CHECKOUT LIBVOLUME: Operation for volume J11773 in library 3494 started
as process 8.
ANR2017I Administrator ADMIN issued command: CHECKOUT libv 3494 J11774 remove=no
checkl=no
ANR0984I Process 9 for CHECKOUT LIBVOLUME started in the BACKGROUND at 15:10:25.
ANR8434I CHECKOUT LIBVOLUME: Operation for volume J11774 in library 3494 started
as process 9.
ANR2017I Administrator ADMIN issued command: COMMIT
ANR2017I Administrator ADMIN issued command: COMMIT
ANR8438I CHECKOUT LIBVOLUME for volume J11771 in library 3494 completed
successfully.
ANR0985I Process 6 for CHECKOUT LIBVOLUME running in the BACKGROUND completed with
completion state SUCCESS at 15:10:26.
ANR8438I CHECKOUT LIBVOLUME for volume J11772 in library 3494 completed
successfully.
ANR0985I Process 7 for CHECKOUT LIBVOLUME running in the BACKGROUND completed with
completion state SUCCESS at 15:10:27.
ANR8438I CHECKOUT LIBVOLUME for volume J11773 in library 3494 completed
successfully.
ANR0985I Process 8 for CHECKOUT LIBVOLUME running in the BACKGROUND completed with
completion state SUCCESS at 15:10:28.
ANR8438I CHECKOUT LIBVOLUME for volume J11774 in library 3494 completed
successfully.
ANR0985I Process 9 for CHECKOUT LIBVOLUME running in the BACKGROUND completed with
completion state SUCCESS at 15:10:29.
```

4. Verify that there are no volumes in the library manager inventory (Example 26).

Example 26 Library inventory output on library manager

```
tsm: NATAL>q libv
ANR2034E QUERY LIBVOLUME: No match found using this criteria.
ANS8001I Return code 11.
```

Removing the library definition from the target server

On RECIFE, the current library definition, if any, must be removed (Example 27) so that you can create a new definition.

Example 27 Removing library definition on the library manager

```
tsm: RECIFE>delete library 3494
ANR8410I Library 3494 deleted.
```

Defining a library on the target server

Now you are ready to create the final definition of the library on the new library manager instance (RECIFE). Example 28 shows the commands for accomplishing this task. You must define the library first, then the drives and paths, and, finally, the device classes.

Example 28 Defining the library on the new library manager

```
tsm: RECIFE>define library 3494 LIBTYPE=349X SHARED=YES SCRATCHCATegory=700
PRIVATECATegory=701
ANR8400I Library 3494 defined.
tsm: RECIFE>define path recife 3494 srctype=server autodetect=yes desttype=library
device=3494c
ANR1720I A path from RECIFE to 3494 has been defined.
tsm: RECIFE>define drive 3494 3494_01
ANR8404I Drive 3494_01 defined in library 3494.
tsm: RECIFE>define drive 3494 3494_02
ANR8404I Drive 3494_02 defined in library 3494.
tsm: RECIFE>define path RECIFE 3494_01 srct=server autodetect=yes destt=drive
libr=3494 device=\\.\Tape2
ANR1720I A path from RECIFE to 3494 3494_01 has been defined.
tsm: RECIFE>define path RECIFE 3494_02 srct=server autodetect=yes destt=drive
libr=3494 device=\\.\Tape3
ANR1720I A path from RECIFE to 3494 3494_02 has been defined.
tsm: RECIFE>define path NATAL 3494_01 srct=server autodetect=yes destt=drive
libr=3494 device=/dev/rmt2
ANR1720I A path from NATAL to 3494 3494_01 has been defined.
tsm: RECIFE>define path NATAL 3494_02 srct=server autodetect=yes destt=drive
libr=3494 device=/dev/rmt3
ANR1720I A path from NATAL to 3494 3494_02 has been defined.
tsm: RECIFE>define devclass 3494_3592c library=3494 devtype=3592
ANR2203I Device class 3494_3592C defined.
```

If you generated a macro to define the paths to the storage agents, such as *define_paths.mac* in step d on page 7, execute the macro (Example 29).

Example 29 Running the macro to define storage agent paths on the new library manager

```
tsm: RECIFE>macro define_paths.mac > define_paths_output.txt
Output of command redirected to file 'DEFINE_PATHS_OUTPUT.TXT'
```

Checking in volumes on the target server

With the library, drives, and paths defined on the new library manager, it is time to check the volumes in the inventory of this instance. In “Gathering volume inventory from the source server” on page 8, you created a list of volumes for each owner and a list of scratch volumes in the form of text files. You use each of those files to generate macros to check the volumes back into the library of the new library manager.

1. Start with the scratch volumes. Use the file SCRATCH_LIBVOLUMES.TXT as input for the script *gen_checkin.pl* (Example 30) to create a macro with commands to check in these volumes. In “gen_checkin.pl” on page 21 and “gen_checkin_ACSLS.pl” on page 22, we provide two versions of this script: one generic version and another suitable for ACSLS libraries. You might have to adapt the code to your specific needs.

Example 30 Generating a macro to check in scratch volumes on the new library manager

```
perl gen_checkin.pl  SCRATCH_LIBVOLUMES.TXT  scratch_checkin.mac Scratch “”
```

Example 31 shows samples of the commands included in *scratch_checkin.mac*.

Example 31 Sample commands to check in volumes included in the macro

```
checkin libv 3494 vollist=J11772 status=Scratch owner="" checkl=barcode search=yes
checkin libv 3494 vollist=J11775 status=Scratch owner="" checkl=barcode search=yes
checkin libv 3494 vollist=J11776 status=Scratch owner="" checkl=barcode search=yes
```

2. Run the macro on RECIFE (Example 32).

Example 32 Running the macro to check in scratch volumes on the new library manager

```
tsm: RECIFE>macro scratch_checkin.mac
```

3. Check the activity log. If there are errors, reissue the command to the specific volumes.
4. Check in the volumes owned by the NATAL instance. Its volumes are listed in SOURCE_LIBVOLUMES.TXT. In Example 33, we show how to use this file as input for the *gen_checkin.pl* (or *gen_checkin_ACSLS.pl*) script to generate a macro. Note that the parameters have changed to indicate that the volumes are private and owned by NATAL.

Example 33 Generating a macro to check in NATAL private volumes on new library manager

```
perl gen_checkin.pl SOURCE_LIBVOLUMES.TXT  source_checkin.mac Private “NATAL”
```

Example 34 shows samples of the commands included in *source_checkin.mac*.

Example 34 Sample commands to check in volumes included in the macro

```
checkin libv 3494 vollist=J11773 status=Private owner=NATAL checkl=barcode
search=yes
checkin libv 3494 vollist=J11774 status=Private owner=NATAL checkl=barcode
search=yes
```

5. Run the macro on RECIFE (Example 35):

Example 35 Running the macro to check in NATAL private volumes on the new library manager

```
tsm: RECIFE>macro source_checkin.mac
```

6. Check the activity log. If you find errors, reissue the command to the specific volumes.

7. Repeat the steps to check in volumes owned by RECIFE. Its volumes are listed in TARGET_LIBVOLUMES.TXT. We used this file as input for the *gen_checkin.pl* (or *gen_checkin_ACSLS.pl*) script to generate a macro (Example 36). Note that the parameters have changed to indicate that the volumes are private and owned by RECIFE.

Example 36 Generating a macro to check in RECIFE private volumes on the new library manager

```
perl gen_checkin.pl TARGET_LIBVOLUMES.TXT target_checkin.mac Private "RECIFE"
```

Example 37 shows samples of the commands included in *target_checkin.mac*.

Example 37 Sample commands to check in volumes included in the macro

```
checkin libv 3494 vollist=J11770 status=Private owner=RECIFE checkl=barcode
search=yes
checkin libv 3494 vollist=J11771 status=Private owner=RECIFE checkl=barcode
search=yes
```

8. Run the macro on RECIFE (Example 38) to check in the private volumes for RECIFE.

Example 38 Running the macro to check in RECIFE private volumes

```
tsm: RECIFE>macro target_checkin.mac
```

9. Check the activity log. If you find errors, reissue the command to the specific volumes.
10. Repeat this process if you have other instances in your environment. Of course, in this case, you should have created a separate list of volumes for each of these instances.
11. Monitor the activity log or use the administrative CLI with the *-consolemode* option to see the check in processes issued and executed. These will look similar to Example 39.

Example 39 Console output when running check in macro

```
ANR2017I Administrator ADMIN issued command: CHECKIN libv 3494 vollist=J11771
status=Private
owner=POCO_1 checkl=barcode search=yes
ANR0984I Process 3 for CHECKIN LIBVOLUME started in the BACKGROUND at 16:10:49.
ANR8422I CHECKIN LIBVOLUME: Operation for library 3494 started as process 3.
ANR2017I Administrator ADMIN issued command: CHECKIN libv 3494 vollist=J11772
status=Scratch owner=
checkl=barcode search=yes
ANR0984I Process 4 for CHECKIN LIBVOLUME started in the BACKGROUND at 16:10:50.
ANR8422I CHECKIN LIBVOLUME: Operation for library 3494 started as process 4.
ANR2017I Administrator ADMIN issued command: CHECKIN libv 3494 vollist=J11773
status=Private
owner=SCAMP_1 checkl=barcode search=yes
ANR0984I Process 5 for CHECKIN LIBVOLUME started in the BACKGROUND at 16:10:50.
ANR8422I CHECKIN LIBVOLUME: Operation for library 3494 started as process 5.
ANR2017I Administrator ADMIN issued command: CHECKIN libv 3494 vollist=J11774
status=Private
owner=SCAMP_1 checkl=barcode search=yes
ANR0984I Process 6 for CHECKIN LIBVOLUME started in the BACKGROUND at 16:10:50.
ANR8422I CHECKIN LIBVOLUME: Operation for library 3494 started as process 6.
ANR2017I Administrator ADMIN issued command: COMMIT
ANR2017I Administrator ADMIN issued command: COMMIT
ANR8427I CHECKIN LIBVOLUME for volume J11771 in library 3494 completed
successfully.
ANR0985I Process 3 for CHECKIN LIBVOLUME running in the BACKGROUND completed with
```

```
completion state SUCCESS at 16:11:08.
ANR8427I CHECKIN LIBVOLUME for volume J11772 in library 3494 completed
successfully.
ANR0985I Process 4 for CHECKIN LIBVOLUME running in the BACKGROUND completed with
completion state SUCCESS at 16:11:11.
ANR8427I CHECKIN LIBVOLUME for volume J11773 in library 3494 completed
successfully.
ANR0985I Process 5 for CHECKIN LIBVOLUME running in the BACKGROUND completed with
completion state SUCCESS at 16:11:15.
ANR8427I CHECKIN LIBVOLUME for volume J11774 in library 3494 completed
successfully.
ANR0985I Process 6 for CHECKIN LIBVOLUME running in the BACKGROUND completed with
completion state SUCCESS at 16:11:18.
```

All volumes are now included in the RECIFE library inventory. Example 40 shows the **QUERY LIBVOLUMES** output. The output should look very similar to the original library manager (NATAL) except for the *Last Use* field, which should appear blank. This is not a problem because this field is updated when each volume is accessed.

Example 40 Query libvolumes command output on the new library manager

```
tsm: RECIFE>q libv
Library Name    Volume Name   Status     Owner   Last Use  Home     Device
                                                           Element  Type
------------    -----------  ---------- ------  --------  -------  -------
3494            J11771        Private    NATAL
3494            J11772        Scratch
3494            J11773        Private    RECIFE
3494            J117734       Private    RECIFE
```

Redefining the library on the source server

The former library manager must be demoted to a library client. To do this, you remove the current library definition and define a new shared library. Before deleting the library registry, you must delete all paths and drives related to the library (Example 41).

Example 41 Removing library definitions

```
tsm: NATAL>delete path natal 3494_01 srct=server destt=drive libr=3494
ANR1721I A path from NATAL to 3494 3494_01 has been deleted.
tsm: NATAL>delete path natal 3494_02 srct=server destt=drive libr=3494
ANR1721I A path from NATAL to 3494 3494_02 has been deleted.
tsm: NATAL>delete path natal 3494 srct=server destt=library
ANR1721I A path from NATAL to 3494 has been deleted.
tsm: NATAL>delete drive 3494 3494_01
ANR8412I Drive 3494_01 deleted from library 3494.
tsm: NATAL>delete drive 3494 3494_02
ANR8412I Drive 3494_02 deleted from library 3494.
tsm: NATAL>delete library 3494
ANR8410I Library 3494 deleted.
```

The access to the library manager is done by defining a shared library that points to the primary library manager server, RECIFE (Example 42 on page 15).

Example 42 New library entry on the new library client

```
tsm: NATAL>define library 3494 libtype=shared primarylibmanager=RECIFE
ANR8400I Library 3494 defined.
tsm: NATAL>q libr f=d
Library Name: 3494
Library Type: SHARED
ACS Id:
Private Category:
Scratch Category:
WORM Scratch Category:
External Manager:
Shared: No
LanFree:
ObeyMountRetention:
Primary Library Manager: RECIFE
```

Updating the source volume history

To update the source volume history, you must remove entries marked as *remote* on the volume history of NATAL as follows. Generate these entries using the command in Example 43.

Example 43 Volume history entries with type=remote

```
tsm: NATAL>query volhistory type=remote
Date/Time: 28.11.2007 18:00:31
Volume Type: REMOTE
Backup Series:
Backup Operation:
Volume Seq:
Device Class: 3494_3592C
Volume Name: J11774
Volume Location: RECIFE
Command:
Date/Time: 29.11.2007 17:47:24
Volume Type: REMOTE
Backup Series:
Backup Operation:
Volume Seq:
Device Class: 3494_3592C
Volume Name: J11773
Volume Location: RECIFE
Command:
```

Important: We strongly recommend the removal of these entries; otherwise, you can encounter the following situation:

1. Server NATAL requests a new media to library manager RECIFE.
2. The library manager selects scratch volume J11774 and inform s the library client.
3. The volume access is rejected with this message: *ANR8448E Scratch volume <VOLUME> from library <LIBRARY> rejected because the volume name is already in use*

To remove these entries:

1. Identify and redirect these entries to a text file using the SELECT statement (Example 44).

Example 44 Generating a list of volume history entries with type=remote

```
tsm: NATAL>set sqldisplaymode wide
tsm: NATAL>select VOLUME_NAME,substr(char(DATE_TIME), 0, 11) from volhistory where
type='REMOTE'
VOLUME_NAME   Unnamed[2]
-----------   ----------
J11774        2007-11-28
J11772        2007-11-29
J11773        2007-11-29
tsm: NATAL>select VOLUME_NAME,substr(char(DATE_TIME), 0, 11) from volhistory where
type='REMOTE' > source_volh_remote.txt
Output of command redirected to file 'SOURCE_VOLH_REMOTE.TXT'
```

2. Invoke *source_volh.pl* (Example 45) to create a macro that deletes these entries from the volume history.

Example 45 Using the source_volh.pl script to create a macro to remove type=remote entries

```
perl source_volh.pl  SOURCE_VOLH_REMOTE.TXT  del_remotes.mac
```

Example 46 shows an example of the commands included in *del_remotes.mac*.

Example 46 Sample commands to delete remote entries included in the macro

```
DELETE VOLHISTORY t=remote vol=J11774 todate=11/28/2007 force=yes
DELETE VOLHISTORY t=remote vol=J11772 todate=11/29/2007 force=yes
DELETE VOLHISTORY t=remote vol=J11773 todate=11/29/2007 force=yes
```

3. Run the macro on NATAL (Example 47):

Example 47 Running the macro to remove volume history remote entries

```
tsm: NATAL>macro del_remotes.mac
```

4. Monitor the activity log or use the administrative CLI with the *-consolemode* option to see the delete processes that were issued and executed. These will look similar to those in Example 48.

Example 48 Console output while running del_remotes macro

```
ANR2017I Administrator ADMIN issued command: DELETE VOLHISTORY t=remote vol=J11774
todate=11/28/2007 force=yes
ANR2467I DELETE VOLHISTORY: 1 sequential volume history entries were successfully
deleted.
ANR2017I Administrator ADMIN issued command: DELETE VOLHISTORY t=remote vol=J11772
todate=11/29/2007 force=yes
ANR2467I DELETE VOLHISTORY: 1 sequential volume history entries were successfully
deleted.
ANR2017I Administrator ADMIN issued command: DELETE VOLHISTORY t=remote vol=J11773
todate=11/29/2007 force=yes
ANR2467I DELETE VOLHISTORY: 1 sequential volume history entries were successfully
deleted.
ANR2017I Administrator ADMIN issued command: COMMIT
ANR2017I Administrator ADMIN issued command: COMMIT
```

Updating additional library clients

The final task is to update any other library clients. The shared library definition for those library clients currently points to NATAL as the primary library manager. Example 49 shows how to query one, update it so that it points to RECIFE, the new library manager, and check to make sure that it now points to RECIFE.

Example 49 Updating other library clients

```
tsm: SCAMP_1>query library 3494 f=d
Library Name: 3494
Library Type: SHARED
ACS Id:
Private Category:
Scratch Category:
WORM Scratch Category:
External Manager:
RSM Media Type:
Shared: No
LanFree:
ObeyMountRetention:
Primary Library Manager: NATAL
WWN:
Serial Number:
AutoLabel:
Reset Drives: No
Last Update by (administrator): ADMIN
Last Update Date/Time: 24.11.2007 12:05:57

tsm: SCAMP_1>update library 3494 primarylibmanager=RECIFE
ANR8465I Library 3494 updated.

tsm: SCAMP_1>query library 3494 f=d
Library Name: 3494
Library Type: SHARED
ACS Id:
Private Category:
Scratch Category:
WORM Scratch Category:
External Manager:
RSM Media Type:
Shared: No
LanFree:
ObeyMountRetention:
Primary Library Manager: RECIFE
WWN:
Serial Number:
AutoLabel:
Reset Drives: No
Last Update by (administrator): ADMIN
Last Update Date/Time: 30.11.2007 16:33:06
```

Completion

It is a good idea to run **AUDIT LIBRARY** for all Tivoli Storage Manager instances just to be sure everything is in order.

Finally, reenable client sessions on your servers (Example 50). You have successfully completed the task.

Example 50 Enabling client sessions on all Tivoli Storage Manager instances

```
tsm: NATAL>enable session client
ANR2553I Server now enabled for Client access.

tsm: RECIFE>enable session client
ANR2553I Server now enabled for Client access.
```

Appendix: Auxiliary perl scripts

This section shows the perl scripts that are used in this paper. The scripts are available for download from:

ftp://www.redbooks.ibm.com/redbooks/REDP0140

Note: You need a perl interpreter to run these scripts.

gen_servers.pl

This macro (Example 51) is used for defining servers on the library manager.

Example 51 Generating gen_servers.pl

```
#!/usr/bin/perl
use Switch;
#
# Generate a macro to define servers on the library manager
# The servers shall be listed on the input text file
# gen_servers.pl
# Parameters
# $1 = action: define or delete
# $2 = name of the input text file. This text file can be created with the
#          following SELECT statement:
#          select server_name,hl_address,ll_address from servers
# $3 = name of the output text file. This file will be the macro containing
#          the macro commands
$mode   = shift;
$infile    = shift;
$outfile = shift;
open(INPUTFILE, "<$infile") or die "Can't open $infile for input: $!";
open(OUTPUTFILE, ">$outfile") or die "Can't open $outfile for input: $!";
while ( my $line = <INPUTFILE> ) {
   switch ($mode) {
   case "define" {
      $line =~
s/^(\w+|\w+-\w+|\w+-\w+-\w+|\w+-\w+-\w+-\w+)\s+(\d+\.\d+\.\d+\.\d+)\s+(\d+)/define
server $1 serverpassword=$1 hladdress=$2 lladdress=$3/; }
   case "delete" {
      $line =~
s/^(\w+|\w+-\w+|\w+-\w+-\w+|\w+-\w+-\w+-\w+)\s+(\d+\.\d+\.\d+\.\d+)\s+(\d+)/delete
server $1/; }
   }
   print OUTPUTFILE lc($line);
}
```

gen_paths.pl

This macro (Example 52) is used for defining paths on the library manager.

Example 52 Generating gen_paths.pl

```
#!/usr/bin/perl
use Switch;
#
# Generate a macro to define paths on the library manager
# The paths shall be listed on the input text file
# gen_paths.pl
# Parameters
# $1 = action: define or delete
# $2 = name of the input text file. This text file can be created with the
#          following SELECT statement:
#          select source_name, destination_name, source_type, destination_type,
#          library_name, device, online from paths
# $3 = name of the output text file. This file will be the macro containing
#          the macro commands
$mode   = shift;
$infile    = shift;
$outfile = shift;
open(INPUTFILE, "<$infile") or die "Can't open $infile for input: $!";
open(OUTPUTFILE, ">$outfile") or die "Can't open $outfile for input: $!";
while ( my $line = <INPUTFILE> ) {
   switch ($mode) {
   case "define" {
      $line =~
s/^(\w+|\w+-\w+|\w+-\w+-\w+|\w+-\w+-\w+-\w+)\s+(\w+)\s+(\w+)\s+(\w+)\s+(\w+)\s+(\/
\w+\/\w+|\/\w+\/\w+\/\w+|\w+\.\d\.\d\.\d)\s+(\w+)/define path $1 $2 srctype=$3
desttype=$4 library=$5 device=$6 online=$7/; }
   case "delete" {
      $line =~
s/^(\w+|\w+-\w+|\w+-\w+-\w+|\w+-\w+-\w+-\w+)\s+(\w+)\s+(\w+)\s+(\w+)\s+(\w+)\s+(\/
\w+\/\w+|\/\w+\/\w+\/\w+|\w+\.\d\.\d\.\d)\s+(\w+)/delete path $1 $2 srctype=$3
desttype=$4 library=$5 \ncommit/; }
   }
   print OUTPUTFILE lc($line);
}
```

gen_checkout.pl

This macro (Example 53) is used for checking out volumes.

Example 53 Generating gen_checkout.pl

```
#!/usr/bin/perl
#
# Generate a macro to check out volumes
# The volumes shall be listed on the input text file
# Parameters
# gen_checkout.pl
# $1 = name of the input text file
# $2 = name of the output text file. This file will be the macro containing
#          the macro commands
$infile = shift;
$outfile = shift;
open(INPUTFILE, "<$infile") or die "Can't open $infile for input: $!";
open(OUTPUTFILE, ">$outfile") or die "Can't open $outfile for input: $!";
while ( my $line = <INPUTFILE> ) {
   $line =~ s/^(\w+)\s+(\w+)/checkout libv $1 $2 remove=no checkl=no/;
   print OUTPUTFILE $line;
}
```

gen_checkin.pl

This macro (Example 54) is used for checking in volumes.

Example 54 Generating gen_checkin.pl

```
#!/usr/bin/perl
#
# Generate a macro to check in volumes
# The volumes shall be listed on the input text file
# gen_checkin.pl
# Parameters
# $1 = name of the input text file
# $2 = name of the output text file. This file will be the macro containing
#          the macro commands
# $3 = status in which the volumes will be checked in: Scratch or Private
# $4 = onwer to which the volumes will be associated. Can be the name
#          of a Tivoli Storage Manager server or “” for scratch volumes
# You can change checkl=barcode to checkl=no below if you have too many volumes
$infile = shift;
$outfile = shift;
$status = shift;
$owner = shift;
open(INPUTFILE, "<$infile") or die "Can't open $infile for input: $!";
open(OUTPUTFILE, ">$outfile") or die "Can't open $outfile for input: $!";
while ( my $line = <INPUTFILE> ) {
   $line =~ s/^(\w+)\s+(\w+)/checkin libv $1 vollist=$2 status=$status
owner=$owner checkl=barcode search=yes/;
   print OUTPUTFILE $line;
}
```

gen_checkin_ACSLS.pl

This macro (Example 55) is used for checking in volumes for ACSLS libraries.

Example 55 Generating gen_checkin_ACSLS.pl

```
#!/usr/bin/perl
#
# This version is for ACSLS libraries
# Generate a macro to check in volumes
# The volumes shall be listed on the input text file
# gen_checkin_ACSLS.pl
# Parameters
# $1 = name of the input text file
# $2 = name of the output text file. This file will be the macro containing
#          the macro commands
# $3 = status in which the volumes will be checked in: Scratch or Private
# $4 = onwer to which the volumes will be associated. Can be the name
#          of a Tivoli Storage Manager server or “” for scratch volumes
$infile = shift;
$outfile = shift;
$status = shift;
$owner = shift;
open(INPUTFILE, "<$infile") or die "Can't open $infile for input: $!";
open(OUTPUTFILE, ">$outfile") or die "Can't open $outfile for input: $!";
while ( my $line = <INPUTFILE> ) {
   $line =~ s/^(\w+)\s+(\w+)/checkin libv $1 $2 search=no status=$status checkl=no
owner=$owner/;
print OUTPUTFILE $line;
}
```

source_volh.pl

This macro (Example 56) is used for removing remote entries from volume history.

Example 56 Generating source_volh.pl

```
#!/usr/bin/perl
#
# Generate a macro to remove remote entries from volume history
# The volumes shall be listed on the input text file
# source_volh.pl
# Parameters
# $1 = name of the input text file
# $2 = name of the output text file. This file will be the macro containing
#          the macro commands
$infile = shift;
$outfile = shift;
open(INPUTFILE, "<$infile") or die "Can't open $infile for input: $!";
open(OUTPUTFILE, ">$outfile") or die "Can't open $outfile for input: $!";
while ( my $line = <INPUTFILE> ) {
   $line =~ s/^(\w+)\s+(\d\d\d\d)-(\d\d)-(\d\d)/delete volhistory t\=remote
vol\=$1 todate\=$3\/$4\/$2 force\=yes/;
   print OUTPUTFILE $line;
}
```

The team that wrote this Redpaper

Hélder Garcia is an IT Specialist for IBM Brazil. He has 6 years of experience with Tivoli Storage Manager and16 years in the IT industry, with a background in network protocols and management. Before joining IBM in 2005, he worked for an IBM Business Partner in Brazil. His areas of expertise include consulting, planning, and implementing Tivoli Storage Manager backup solutions and storage management. He has also taught Tivoli Storage Manager classes for clients and IBM Business Partners. He is an IBM Certified Deployment Professional for Tivoli Storage Manager and was on the development team for the V5.4 certification exam. He has the ITIL Foundation Certificate and is an author of *IBM Tivoli Storage Manager: Building a Secure Environment*, SG24-7505.

Thanks to the following people for their contributions to this project:

Forsyth Alexander
Charlotte Brooks
International Technical Support Organization

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Send us your comments in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ® IBM® Tivoli®

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

Other company, product, or service names may be trademarks or service marks of others.