Tivoli software

IBM

Backup and Restore eDirectory

with Tivoli Storage Manager

By Nicholas Swenson, Jim Smith, & Mandeep Jandir

Version 1.0

Copyright Notice

Trademarks

IBM, the IBM logo, Tivoli, the Tivoli logo, AIX, Cross-Site, NetView, OS/2, Planet Tivoli, RS/6000, Tivoli Certified, Tivoli Enterprise, Tivoli Enterprise Console, Tivoli Ready, and TME are trademarks or registered trademarks of International Business Machines Corporation or Tivoli Systems Inc. in the United States, other countries, or both.

Lotus is a registered trademark of Lotus Development Corporation.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.
C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.
PC Direct is a trademark of Ziff Communications Company in the United States, other countries, or both and is used by IBM Corporation under license.
ActionMedia, LANDesk, MMX, Pentium, and ProShare are trademarks of Intel Corporation in the United States, other countries, or both. For a complete list of Intel trademarks, see http://www.intel.com/sites/corporate/trademarx.htm.
SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC. For further information, see http://www.setco.org/aboutmark.html.
Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

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

Notices

References in this publication to Tivoli Systems or IBM products, programs, or services do not imply that they will be available in all countries in which Tivoli Systems or IBM operates. Any reference to these products, programs, or services is not intended to imply that only Tivoli Systems or IBM products, programs, or services can be used. Subject to valid intellectual property or other legally protectable right of Tivoli Systems or IBM, any functionally equivalent product, program, or service can be used instead of the referenced product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by Tivoli Systems or IBM, are the responsibility of the user. Tivoli Systems or IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, New York 10504-1785,

U.S.A

1.0 Introduction

The intent of this document is to help in the process of building a backup solution by presenting several tools, techniques, and scenarios. This document is intended for professionals with a working knowledge of eDirectory and IBM Tivoli Storage Manager.

In the past Novell eDirectory was run on Novell NetWare as Novell Directory Services (NDS). Backup was fully handled by the Tivoli Storage Manager Backup-Archive client. It would run up and down the tree reading, and backing up, each object separately. Since then Novell has released backup tools which make the task far simpler. This document uses these tools in conjunction with basic Tivoli Storage Manager commands to fully backup the tree.

This document was tested with Novell eDirectory run on Novell OES 2 SP3, in conjunction with the Tivoli Storage Manager Backup-Archive client. The scenarios in this document were created on eDirectory version 8.8. The scenarios also used Tivoli Storage Manager Backup-Archive client version 6 release 2, though any version should work.

1.1 What is eDirectory?

eDirectory is a scalable, object-oriented, hierarchical database utilized to manage company resources (e.g. server, users, files, systems, printers) in a network. It is a global service that can be partitioned to reside on multiple servers within a network. This document is not intended to present a full description of eDirectory, though basic concepts will be presented in the context of backup and recovery.

For an excellent tutorial of basic eDirectory concepts please refer to:
http://developer.novell.com/education/tutorials/edirectory/edirec21.htm

A more in depth discussion can be found in the eDirectory online documentation:
http://www.novell.com/documentation/edir88/edir88/?page=/documentation/edir88/edir88/data/fbadjaeh.html

1.2 What is Tivoli Storage Manager?

Tivoli Storage Manager is a client/server program that provides centralized, automated data protection and storage management system. It is able to run in an environment that includes multiple vendors. Tivoli Storage Manager provides policy based backup, archive, and space-management facility for file servers, applications, and application server.

For more information please refer to Tivoli Storage Manager's online documentation:
http://publib.boulder.ibm.com/infocenter/tsminfo/v6r2/index.jsp

2.0 Backup Considerations

2.1.0 Backup Tools

This document uses several tools provided by Novell's eDirectory as well as Backup-Archive client. This section describes how to use them.

2.1.1 iManager

Novell's iManager is a web-based, management GUI provided in OES along with eDirectory. It provides easy wizard based services for one-time backup, repair, and restore functions. Functions are accessed in the roles/services tab, under the eDirectory maintenance section.

It is recommended to use iManager whenever possible. For more information, and detailed instructions on various tasks, please refer to iManager's online documentation.

After extending the schema, go to *New* under the show iManager 2.x Collection tab and click *Collection and Setup*.

2.1.2 eMBox

Novell's eMBox utility is the main toolbox for backup and recovery operations. It combines backup, restore, and configuration tools into one source. The iManager interface invokes these backup and restore functions. It's also available for custom backup solutions as well; a key feature being that eMBox can be built into a script for an automated backup solution.

eMBox can be utilized with three major operations:

1. eMBox has a built-in interactive client that can be used to do single commands one a time.

   To access the client, input into the terminal:
   ```
   edirutil -i
   ```

   From the client, you must login to the respective server:
   ```
   login -s <server_ip> -p <port> -u <user.context> -w <password>
   ```

   Now you may take the required action, such as a full backup (designated by the –b):
   ```
   backup.backup -f /backup/bak.full -l /backup/bak.log -b
   ```

2. eMBox can be used to run the backup/restore commands directly from the terminal with `-t` switch:

```
java -cp <path>/eMBoxClient.jar embox -s <server_ip> -p <port>
-u <user.context> -w <pass> <options> -t <tool.function>
<tool switches>
```

Note: <path> is the directory containing eMBoxClient.jar. By default this is:
`/opt/novell/eDirectory/lib/nds-modules/embox`

3. eMBox can also process a batch command file with the `-b` switch:

```
java  -s <IP> -p <port> -u <user.context> -w <password> -l
<log> -b <mybatch.mbx>
```

An example batch file can be found in first link below.

More information and a list of available eMBox tools can be found:
http://www.novell.com/documentation/edir88/edir88/data/agaye59.html

A full list of eMBox backup/restore functions can be found in the online Novell documentation:
http://www.novell.com/documentation/edir88/edir88/data/bsl7jmp.html#agcgkmr
(Found in Section 17.6.6 Backup/Restore Command Line Options portion of the page)

NOTE: Before you can use eDirectory backup/restore functions through iManager or eMBox you may have to set up Role Based Services to gain access to those roles.

2.1.3 dsbk

The dsbk tool is another command line based backup/restore tool. It makes use of many of the same functions as eMBox. The key difference is that dsbk runs the commands locally, instead of remotely on eMBox. This circumvents the need to login, and can be useful if the database has been lost and you can't login with eMBox. Note that dsbk only runs backup/restore function, whereas eMBox runs others.

Before dsbk can be used, it must first be set up:
1) A temp file must be created, for example: `/tmp/dsbk.tmp`
2) A config file called `/etc/dsbk.conf` must be created containing one line with the full path of the temp file.

The output of dsbk is written to the ndsd.log file. The default log location is:
`/var/opt/novell/eDirectory/log/ndsd.log`

A full list of commands can be found using the link provided in the section above.

2.1.4 dsmc

The command line command for Backup-Archive client is dsmc. Commands can either be run inside dsmc (just run `dsmc`), or from the shell command line with the format:
`dsmc <command> -password=<password>`

The basic single file backup (called selective) is:
`backup <filename>`

To restore a file from a selective backup:
`restore <filename>`

Objects in Tivoli Storage Manager are saved in logical location very much like on a file system. Files are stored in filespaces, which act very much like directories. You can query these filespaces to see what they contain, allowing you to keep track of what is currently saved on the Tivoli Storage Manager Server.

The basic query format is:
`query backup <filespace>`

For example, say you have backed up files: `/backup/bak.full`, `/backup/bak.inc.1`, and `/backup/bak/inc.2`. If you wanted to see all the files in the `/backup` directory you would query with the command:
`query backup /backup/`

Objects in Tivoli Storage Manager can also be deleted from the server by using the following command:
`delete backup <filepath>`

The document also uses a tool very similar to dsmc: dsmadmc, which is the administrative client for Tivoli Storage Manager. Once logged in, this document uses the Backup-Archive client for one task, scheduling. The command is quite complex. A basic example that runs the script `/backup/auto_backup.sh` weekly:

```
define schedule employee_records weekly_backup action=command
objects='"/backup/auto_backup.sh"' startdate=06/07/1997
starttime=23:00 duration=4 durunits=hours perunits=weeks
dayofweek=saturday options=-quiet
```

For more information on defining schedules please read:
http://publib.boulder.ibm.com/infocenter/tsminfo/v6r2/index.jsp?topic=/com.ibm.itsm.client.doc/r_client_ref.html

This document doesn't involve more commands than those. A list for more options can be found:
http://publib.boulder.ibm.com/infocenter/tsminfo/v6r2/index.jsp?topic=/com.ibm.itsm.client.doc/r_client_ref.html

2.2.0 Types of Backup

Novell's eDirectory and the accompanying tools allow for three types of backup: full backup, incremental backup, and roll forward logs. It is important to understand the differences before designing a solution.

2.2.1 Full Backup

Full backups are the most comprehensive backup function. It is exactly what the name states; a full backup of the entire directory service tree. It backs up every single object in the tree, and saves it to a single comprehensive backup file. Initially the file is not compressed, but it can normally be compressed down 80% using Tivoli Storage Manager compression.

There are a couple of options you must decide on while creating a full backup:

- Whether to backup stream files. Stream files contain additional information related to database such as login scripts. It is recommended to backup stream files.
- Whether to backup NICI files. If you are using NICI, it is heavily recommended to back them up.
- Whether you want to include other files. To do this, make an include file list, and specify it properly in the backup command.

2.2.2 Incremental Backup

An incremental backup backs everything since the last backup (full or incremental). It has the same options available as the full backup. The backup is not dependent on the previous backups being present; the position is stored internally in the database.

Example of an incremental backup (-i denotes the incremental):

```
dsbk backup –f <backup file> -l <log> -i
```

2.2.3 Roll Forward Logs

Roll forward logs are logs that keep track of all changes in real time. They can be used when restoring the server back to the exact point in time that it crashed.

Before eDirectory starts logging, roll forward logs must be turned on. This can be done in iManager, eMBox, or dsbk. Roll forward logging must also be re-enabled after a restore. After enabling the logging, it's recommended to perform a backup, to prevent the situation of having logs starting after the last backup. This log should not be stored on the same hard drive as eDirectory. Therefore, if the eDirectory drive is lost, the roll forward logs are still accessible.

It's important to monitor the roll forward log directory, as it can easily fill up the whole disk. If this happens eDirectory could become unresponsive. Every time a backup is performed a new

roll forward log is started, but the old logs are not deleted. These older logs must be manually deleted.

2.3.0 Best Practices in Designing a Backup Solution

2.3.1 Single Server

There is a simple, two step method when it comes to backing up a single server. First the database is backed up to a single flat file with one of the various eDirectory management tools. This file is then stored in a safe location.

2.3.2 Combining eDirectory tools with Tivoli Storage Manager

Tivoli Storage Manager can be used to store the eDirectory backup file. Once the backup file is created, a selective backup can be used to store the file on the server.

Once the file is backed up with Tivoli Storage Manager it can be deleted from the main server, saving space for other purposes.

Example:

1. Run a backup with dsbk:
   ```
   dsbk backup -f /backup/bak.full -l /backup/bak.log -b -t -w
   ```

2. Then run a selective backup on the file:
   ```
   dsmc backup /backup/bak.full
   ```

2.3.3 Automation

It is highly recommended that backups are done regularly. Incremental backups can be done frequently, for example daily. Full backups should be done less frequently, for instance weekly.

An easy method to ensure these backups are done regularly is to automate them. A script can be configured with the eMBox or dsbk, and the dsmc command line tools. It would create a backup file and store it on Tivoli Storage Manager. The script can then be automatically run with Tivoli Storage Manager schedule service.

To store all the backups, an automated naming convention must be used. For example, in the script at the end of this paper, `bak.full` was used for full backups, and `bak.inc.#` was used for the incremental backups. The script uses a small configuration file to store information about the last backup.

2.3.4 Multiple Backup Sets[1]

A useful feature you can easily add with automated scripting is saving multiple backup sets. A backup set would be a full backup and the associated incremental backups. This allows you to restore back to any point in time.

One possibility is to use the grouping capabilities for Tivoli Storage Manager backups. Essentially you create a file list of the files you want to backup, and submit to the Backup-Archive client, where these files are grouped together logically on the Tivoli Storage Manager server so that they can be easily queried or recovered as a group. The problem with this method is that all the files must be present locally to back them up to the group. If you delete the files after a backup, you would have to restore them before adding another to the group.

The get around this, it is suggested to just backup the files one by one. To keep them in an easy to remember order, just create a new directory for every backup set. For example, say we build a backup solution with daily incremental backups, and full weekly backups. For each full backup, a new directory would be created. The incremental backups would be grouped up in the same directory as the previous full backup. This makes it easier to manage large amounts of backups.

2.4.0 Multi-Server Considerations

Backing up multiple eDirectory servers is a similar process as when backing up a single eDirectory server, except the backup process spans over multiple servers. It is not important to backup every server since restoring all eDirectory servers is not likely possible.

2.4.1 Partitions and Replicas

eDirectory trees are usually partitioned so that they can be more easily managed by multiple eDirectory servers. These partitions can then be replicated to spread out loads. Multiple replicas of the same partition interact in what is called a replica ring.

These replica rings present a problem that can be hard to work around. Each server keeps a list containing the timesync status of every server participating in the replica ring. These lists, called time vectors, need to stay properly synchronized. If the time vectors differ, the tree may, and most likely will, become unstable. The replicas might become unable to communicate, synchronization might not work, and fixing this problem might involve deleting the bad replica. To prevent this, restores have a time vector verification as part of the process. Unless specifically

---

1 The term ‘backup set’ in the context of this paper describes a full backup and associated incremental backups; it does not refer to the Tivoli Storage Manager concept of backup sets, i.e., a collection of a Tivoli Storage Manager Backup-Archive Client’s active backed up data, stored and managed as a single object, on specific media, in Tivoli Storage Manager server storage.

told not to verify, a restore will fail if the verification fails. *NOTE*: Only a server that has a set of unique, non-replicated partitions will be restored properly without verification.

Keeping replicas in proper timesync can be challenging during restore. The only way to guarantee synchronization after a restore is to use the roll forward logs to restore the server exactly to the point in time it crashed:

```
dsbk restore -f <backup file> -l <log> -d <roll-forward logs> -a
-o
```

The major problem with roll forward logs is that they may be destroyed. If this happens, restoring a server directly back into a replica ring is impossible. With this in mind, it isn't necessary to backup every server in a ring.

NOTE: There are several methods to view the current partitions and replicas. Either use iManager or run ndsrepair with the -P switch.

2.4.2 DSMaster servers

So which server should be backed up? Each replica ring has a master server. This master server holds partition information and controls the associated operations. This master server is what should be backed up.

Building a backup solution in a multi-server environment should be based around master servers. These master servers should each have a unique set of master replicas, so no two master servers have the same replica. The idea is that if a large portion of the tree is lost, the masters can be restored from the backup, and the rest of the tree can be restored using replication. If any of the master servers have conflicting replicas you will end up having to choose one over the other.

2.5 Sample Backup Solution

Jim is a computer administrator working for A Very Small Company, Inc (see Figure 1 below). His company recently installed Novell's eDirectory, and he needs to setup up a corresponding backup plan. Since his company is split between several offices, the tree has been partitioned up into three partitions: root, engineering, and marketing. Because the root partition is getting big, he decides that he will give it its own dsmaster server, *master server 1*. The other two partitions are smaller, so he decides to group them on a *master server 2*. Each partition has an associated replica ring of 4 servers, including the master. Although he knows that a full restore can be complicated, he decides to backup all the servers in the replica ring, so at the very least server identity can be restored.

Jim decides he is going to automate the backups to run daily. He will run an incremental backup every day and a full backup every Sunday. He is going to write a script that utilizes dsbk to create the flat backup file, and Tivoli Storage Manager Backup-Archive client to store the flat backup file on the Tivoli Storage Manager Server. The script will create multiple backup sets by

using an incrementing directory name to create new directory for each full backup. The first backup directory will be `/backup/bak0`, the next `/backup/bak1`, and so on. These backups will be scheduled using Tivoli Storage Manager automated scheduler.

Jim decided he will turn on the roll forward logs, just in case he might have them available during restore. He knows he will have to watch the space in the roll forward directory, so they won't end up filling up the disk. Jim also knows he will have to manage the space in the Tivoli Storage Manager storage pool. He will have to go through and manually delete backup sets he no longer wants.

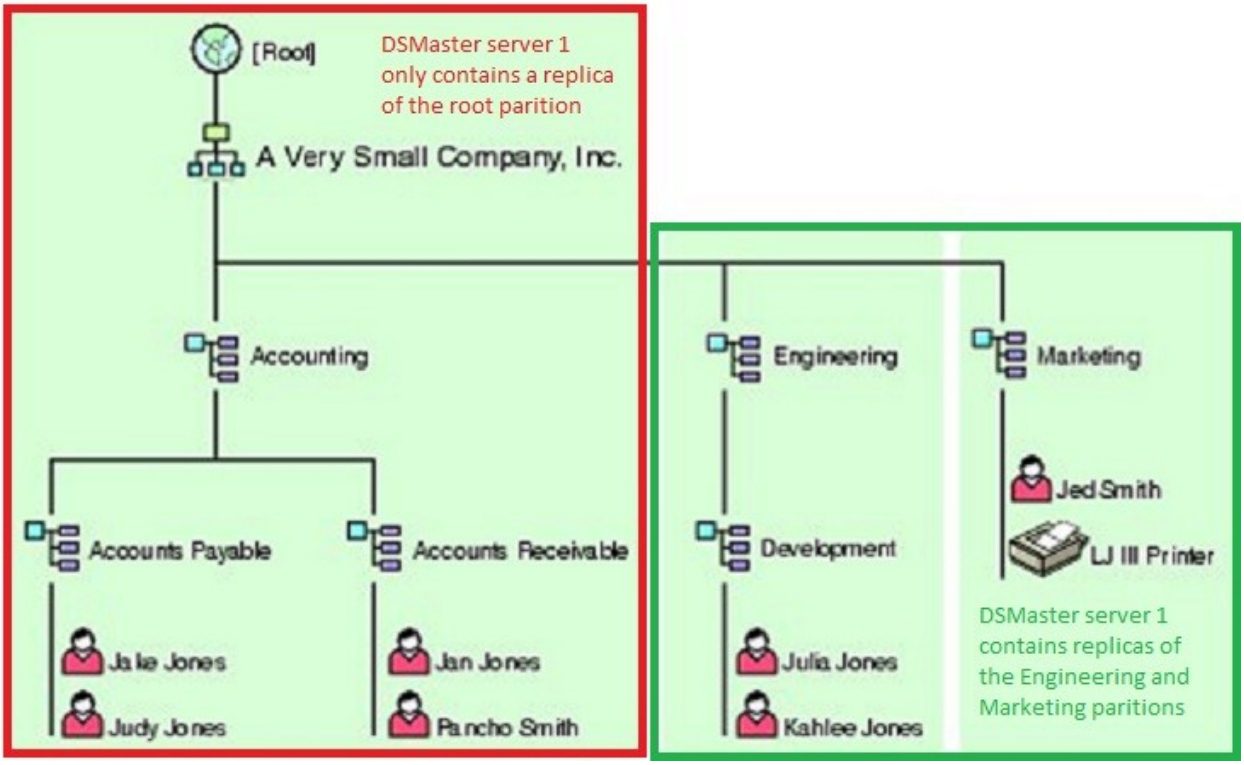

Figure 1: A Very Small Company, Inc. It has 3 partitions split among 2 master servers.

3.0 Restore Considerations

Restoring eDirectory to a working state is based around three different methods: repair, replicate, and restore.

3.1 Repair

Repair should be your first consideration with a problem server. If you have several small synchronization issues, or a single corrupt item, a full restore is not worth the trouble. A simple

repair can fix many of these smaller issues. Repairs can either be done in iManager under the repair section, or with the ndsrepair tool.

For more information on repairing a server:
http://www.novell.com/documentation/edir88/edir88/data/aese2hi.html

For more information about ndsrepair command line options:
http://www.novell.com/documentation/edir88/edir88/data/aflm3p7.html

3.2 Replicating to Restore

In a multi-server environment, replication should be the main method of restoration. The idea is to get the eDirectory server and environment to a clean state, and then just add the replicas to the eDirectory server.

To clean up the replica ring repeat these steps on every eDirectory server:

1. Run the ndsrepair partition advanced mode: `ndsconfig -P -Ad`
2. Select a replica the lost eDirectory server had (and repeat for all)
3. Select *View Replica Ring*
4. Select the lost eDirectory server
5. Select *Remove this eDirectory server from replica ring*

At this point there are a couple of options to bring back in the eDirectory server. You can either add the eDirectory server like it was new, or restore it and kill all replicas on it. Restoring and killing all replicas will allow you to keep the eDirectory server's rights, but it may end up causing odd synchronization issues.

To repair the eDirectory server:

1. Perform a standard restore that will fail:
   `dsbk restore -f <backupfile> -l <log> -a -o`
2. Override the restore: `dsbk restadv -v -l <log>`
3. Change all references to external references: `ndsrepair -R -Ad -xk2`
4. Remove lockout and open database: `dskb restadv -o -k -l <log>`
5. Use iManager to re-add the replicas

To add the eDirectory server like it was new:

1. Make sure the eDirectory server is removed from the tree
   a. If available, try a remove on the broken eDirectory server: `ndsconfig rm`
      NOTE: this needs to be done if eDirectory isn't going to be reinstalled fresh
   b. Or just delete all the eDirectory server-info objects from the tree.
2. Using Yast, install/reconfigure the tree as if you installing a new eDirectory server
3. Once installed you can add rights and replicas.

After either method, use iManager to replicate the replicas back onto the eDirectory server.

3.3 Restoration from a Backup

In a single server environment, restoration from a Tivoli Storage Manager backup is you're only choice. In a multi-server environment, restoring from a Tivoli Storage Manager backup should be your last resort, when rebuilding the tree is impossible from the live servers. Even then only dsmaster servers should be restored. The rest of the tree should be rebuilt using replication. A restore from file can be done from iManager or one of the management tools.

To understand restore, it's important to understand the steps the restore process takes:

1. Directory Service Agent is closed.
2. The active Data Information Base (DIB) set is switched from the .NDS set to the .RST set.
3. The restore is performed, saving the restored DIB as .RST.
4. The DIB set is disabled.
5. Roll forwards logs are reset to OFF (In a command line restore, this can be prevented with the –s switch)
6. Verification of the .RST set is performed. The time vectors of other servers must match the time of the restored dataset. For more information read section 2.4.1 (*Replicas and Paritions*)
7. If the verification passes, the .RST is renamed to .NDS. If the verification fails, the active DIB value is set back to the current .NDS. The .RST set stays on the server.

Before restoring from a file, the files must be restored from Tivoli Storage Manager.

A standard restore from dsbk or eMBox (-a activates DIB after verify, -o opens):

```
restore –f <full_backup> -l <log> -a –o -r
```

where the following files, `<full_backup> -l <log>`, came from the TSM server restore operation.

With roll forward logs:

```
restore –f <full_backup> -l <log> -d <roll_forward_log> -a –o
```

With incrementals:

```
restore –f <full_backup> -l <log> -a –o –i
<inc.1>,<inc.2>,<inc.3>
```

A full list of eMBox backup/restore functions can be found in the online Novell documentation: http://www.novell.com/documentation/edir88/edir88/data/bsl7jmp.html#agcgkmr

3.4 Sample Restore from Tivoli Storage Manager

Mandeep is a computer administrator working for A Very Small Company, Inc (see Figure 1). An electrical fire happened, and now she has to restore the tree back to a working order. Jim had recently left the company, and she is on her own. But Jim had taught her how the backup solution was set up, so she knows enough to restore the tree.

In the fire, the company lost all the servers containing the root partition (including the master server). Mandeep decides she will first fully restore *master server 1*, then restore the identity of the others in that replica ring, and finally replicate the root partition out to the ring. First she sets up Tivoli Storage Manager, with each server registered with the proper node, and eDirectory, configuring each server to have a temporary tree. She then finds the backup of the master server in Tivoli Storage Manager, and restores it as the master server. Then Mandeep does a full restore of *master server 1*. This doesn't conflict with any of the other servers, or the other master, because none of them contain the root partition.

After restoring the master, she has to restore each of servers in the replica ring. She attempts a restore on each one, but it fails since the time syncs do not match the master. After each server has a failed attempt, she has the clean up the replica ring (see Section 3.2) by removing them from the replica ring known by the master server. With the replica ring clean, she can now, on each server with a failed restore, force a restore, and change the databases to external references (see Section 3.2). Mandeep can finally unlock the databases on the server, and using iManager, re-add the replicas to the servers. Now the whole tree should be complete again.

Appendix A: Backup/Restore Script Example

These scripts are run with a configuration file /backup/backup.conf:

```
NUM_INC=0
SUB_DIR=-1
SERVER_IP=<server_ip>
SERVER_PORT=<server_port>
LOGIN_ID=<user.context>
LOGIN_PASS=<pass>
```

auto_backup.sh:

```
#!/bin/bash

#This script does a full, hot, continuous backup and backs
#the backup in TSM. Uses embox tools with a login.

ROOT_UID=0
E_NONROOT=87
EMBOX_PATH=/opt/novell/eDirectory/lib/nds-modules/embox
BACKUP_DIR=/backup
CONF_FILE=$BACKUP_DIR/backup.conf
SUBDIR=/bak
LOG=/backup/scripts/auto_backup.log
DET_LOG=/backup/scripts/auto_backup_det.log

#Needs to run a root
if [ "$UID" -ne "$ROOT_UID" ]
then
    echo "Must be root to run this script." #>> $LOG
    exit $E_NOTROOT
fi

echo -e "\n\nFull Backup started" #>> $LOG
date #>> $LOG

#function to process variable input
process_var()
{
    case $1 in
        NUM_INC)
            NUM_INC=$2 ;;
        SERVER_IP)
            SERVER_IP=$2 ;;
        SERVER_PORT)
            SERVER_PORT=$2 ;;
        LOGIN_ID)
            LOGIN_ID=$2 ;;
        LOGIN_PASS)
            LOGIN_PASS=$2 ;;
        SUBDIR_NUM)
            SUBDIR_NUM=$(($2+1)) ;;
        *)
            echo "Unrecognized option: $1" #>> $LOG
    esac
}

#checks if a process has failed or not
check_no_fail()
{
if [ "$1" -ne "0" ]
then
    echo $2 #>> $LOG
    date #>> $LOG
    exit
```

```
fi

}

#check if config file exists:
if [ -e "$CONF_FILE" ]
    then echo "Config file found." #>> $LOG
    else echo "Configuration file not found. Please create one." #>> $LOG
        date #>> $LOG
        exit 1
fi

#read in config file
OLDIFS=$IFS
IFS='='
while read line
do
    read opt val <<< "$line"
    process_var $opt $val
done < $CONF_FILE
IFS=$OLDIFS

BAK_FILE_PATH=$BACKUP_DIR$SUBDIR$SUBDIR_NUM/bak.full
BAK_LOG_PATH=$BACKUP_DIR/bak.log

#back up old config:
if [ "$SUBDIR_NUM" -gt "0" ]
then echo "Backup up old config ..."
    OLD_CONF=$BACKUP_DIR$SUBDIR$(($SUBDIR_NUM-1))/backup.conf
    cp $CONF_FILE $OLD_CONF
    dsmc selective $OLD_CONF -password=server1 #>> $DET_LOG
    check_no_fail $? "Old config backup failed."
fi

#make subdirectory:
mkdir $BACKUP_DIR$SUBDIR$SUBDIR_NUM

#run backup:
echo "Backing up eDirectory ... " #>> $LOG
java -cp $EMBOX_PATH/eMBoxClient.jar embox \
    -s $SERVER_IP -p $SERVER_PORT -u $LOGIN_ID -w $LOGIN_PASS \
    -t backup.backup -f $BAK_FILE_PATH -l $BAK_LOG_PATH -b \
    -w -t #>> $DET_LOG

#back up eDirectory backup
echo "Backing up eDirectory file in TSM ... " #>> $LOG
dsmc backup $BAK_FILE_PATH -password=server1 #>> $DET_LOG
check_no_fail $? "eDirectory TSM backup failed"

#Remove File
echo "Removing file from local system" #>> $LOG
rm -f $BAK_FILE_PATH

#edit options file to save number of incs
sed -e "s/NUM_INC=$NUM_INC/NUM_INC=0/g" -i $CONF_FILE
sed -e "s/SUBDIR_NUM=$(($SUBDIR_NUM-1))/SUBDIR_NUM=$SUBDIR_NUM/g" -i $CONF_FILE

#back up new config
echo "Backing up config file ... " #>> $LOG
dsmc backup $CONF_FILE -password=server1 #>> $DET_LOG
check_no_fail $? "Config file backup failed"

echo "Done."
echo "Backup Completed: " #>> $LOG
date #>> $LOG
```

auto_restore.sh:

```
#This script is for doing a full restore.
#Uses embox tools with a login.

ROOT_UID=0
E_NONROOT=87
EMBOX_PATH=/opt/novell/eDirectory/lib/nds-modules/embox
BACKUP_DIR=/backup
CONF_FILE=$BACKUP_DIR/backup.conf
SUBDIR=/bak
LOG=/backup/scripts/auto_backup.log
DET_LOG=/backup/scripts/auto_backup_det.log
OLDIFS=$IFS

#Needs to run a root
if [ "$UID" -ne "$ROOT_UID" ]
then echo "Must be root to run this script."
    exit $E_NOTROOT
fi

echo -e "Restore started."
echo -e "\n\nRestore started." >> $DET_LOG
date >> $DET_LOG

#Check for config file
if [ -e "$CONF_FILE" ]
then echo "Config file found."
else echo "Config file missing. Attmepting to restore ..."
        dsmc restore "$CONF_FILE" -password=server11 >> $DET_LOG
        if [ $? -ne "0" ]
        then echo "Config restoration failed. Either create a new " \
                  "one for check the detailed log for more info"
            exit 1
        else echo "Config file restored."
        fi
fi

#function to process variable input
process_var()
{
    case $1 in
        NUM_INC)
            NUM_INC=$2 ;;
        SERVER_IP)
            SERVER_IP=$2 ;;
        SERVER_PORT)
            SERVER_PORT=$2 ;;
        LOGIN_ID)
            LOGIN_ID=$2 ;;
        LOGIN_PASS)
            LOGIN_PASS=$2 ;;

        SUBDIR_NUM)
            SUBDIR_NUM=$2 ;;
    esac
}

read_conf()
{
    IFS='='
    while read line
    do
        read opt val <<< "$line"
        process_var $opt $val
    done < $1
    IFS=$OLDIFS
}

#read in config
read_conf $CONF_FILE
```

```
#prompt for which backup set
echo "You currently have $(($SUBDIR_NUM+1)) backup sets."
echo "Which would you like to restore? (0 - $SUBDIR_NUM): "
read NUM_SUB_RESTORE

if [ "$NUM_SUB_RESTORE" -gt "$SUBDIR_NUM" -o "$NUM_SUB_RESTORE" -lt "0" ]
then echo "Number must be >= 1 and <= $SUBDIR_NUM"
    exit 1
fi

#restore and read in previous config
if [ "$NUM_SUB_RESTORE" -ne "$SUBDIR_NUM" ]
then echo "Using previous config file."
    CONF_FILE=$BACKUP_DIR$SUBDIR$NUM_SUB_RESTORE/backup.conf
    if [ -e "$CONF_FILE" ]
    then echo "Previous config file found."
    else echo "Previous config file missing. Attmepting to restore ..."
            dsmc restore "$CONF_FILE" -password=server11 >> $DET_LOG
            if [ $? -ne "0" ]
            then echo "Restoration failed. Either create a new " \
                    "one for check the detailed log for more info"
                exit 1
            else echo "Previous config file restored."
            fi
    fi
    read_conf $CONF_FILE
fi

#Prompt for incremental
echo "Currently you have a full restore and $NUM_INC incrementals."
echo "Which would you like to restore to? (0 for full, or # of inc): "
read NUM_INC_RESTORE

if [ "$NUM_INC_RESTORE" -gt "$NUM_INC" -o "$NUM_INC_RESTORE" -lt "-1" ]
then echo "Number must be <= $NUM_INC and > -1"
    exit 1
fi

FULL_BAK=$BACKUP_DIR$SUBDIR$NUM_SUB_RESTORE/bak.full
INC_BAK=$BACKUP_DIR$SUBDIR$NUM_SUB_RESTORE/bak.inc

#run restore:
echo "Restoring full backup file ... "
dsmc restore "$FULL_BAK" -password=server11 -replace=true

if [ "$?" -ne "0" ]
    then
    echo "Full file restoration failed, exiting"
    exit
fi

if [ "$NUM_INC_RESTORE" -gt "0" ]
then echo "" > inc_filelist

    INC=1
    while [ "$INC" -le "$NUM_INC_RESTORE" ]
    do
        echo "$INC_BAK.$INC" >> inc_filelist
        INC=$(($INC + 1))
    done

    echo "Restoring incremental files ... "
    dsmc restore -filelist=inc_filelist -password=server11 -replace=true

    if [ "$?" -ne "0" ]
        then
        echo "Incremental file restorations failed, exiting"
        exit
    fi
fi
```

```
#echo "Restoring database"
java -cp $EMBOX_PATH/eMBoxClient.jar embox \
    -s $SERVER_IP -p $SERVER_PORT -u $LOGIN_ID -w $LOGIN_PASS \
    -t backup.restore -f $FULL_BAK -l $BAK_LOG_PATH -i \
    -r -a -o -s
echo "Done."

exit 0
```