Technical Report

Disaster Recovery Solution for Oracle Fusion Middleware 11*g* on Linux

Padmanabhan Sadagopan and Mike Doherty, NetApp
Pradeep Bhat, Oracle

June 2010 | TR-3855

TABLE OF CONTENTS

1 INTRODUCTION

This technical report describes the procedure to deploy Oracle® Fusion Middleware (OFM) 11*g* on Linux® using NetApp® SnapMirror®, Snapshot™, FlexVol®, and FlexClone® technology. It also describes the configuration and procedure to create a disaster recovery (DR) solution for Oracle Fusion Middleware 11*g* using a simple, fast, accurate, and cost-effective method.

NetApp disaster recovery (DR) solutions are simple to deploy and recover, and reduce downtime. They are flexible enough to address a broad range of recovery point objectives ranging from zero to one hour to one day. NetApp DR solutions can replicate over long distances, providing protection from both site and regional disasters. Customers have the flexibility to make a tradeoff between cost and data loss exposure.

For all prerequisites and configurations required to deploy this solution, see TR-3672: Oracle Fusion Middleware DR Solution Using NetApp Storage.

This report only covers the OFM 11*g* specific configuration steps.

1.1 ACRONYMS

Table 1) Acronyms.

Term	Definition
OFM	Oracle Fusion Middleware
IDM	Oracle Identity Management
SOA	Oracle Service-Oriented Architecture
OID	Oracle Internet Directory
EDG	Enterprise Deployment Guide
OEM	Oracle Enterprise Manager

2 DESIGN CONSIDERATIONS

The OFM Disaster Recovery solution relies on NetApp replication technology to replicate OFM's application and Web tier file system artifacts. At each site these artifacts are on a shared storage system configured for storage replication. For more information on the design considerations, see the OFM Disaster Recovery Guide: http://download.oracle.com/docs/cd/E15523_01/doc.1111/e15250/design_consid.htm#sthref426.

2.1 PROTOCOL CONFIGURATION

This DR solution uses the NFS protocol.

The following is an example of an NFS mount option configuration (sample fstab entry):

```
mount nasfiler:/vol/vol1/fmw11shared ORACLE_BASE/wls -t nfs -o
rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768
```

3 HARDWARE/SOFTWARE USED

This section describes the hardware and software environment used for this design validation. Actual customer deployments might vary.

3.1 SERVER HOST DEPLOYMENT DETAILS

The host system in our environment is configured as follows:

IBM® BladeCenter® H Series

- Processor: Dual-Core Intel® Xeon® 3.0 GHz
- Number of processors: 2
- Memory: 4GB per blade
- Network interface: Gigabyte connection
- Internal Hard Drive: 72GB

3.2 STORAGE SYSTEM DEPLOYMENT DETAILS

- Clustered FAS3050 storage controllers
- NetApp Data ONTAP® 7.3.2

3.3 OFM 11*G* SOFTWARE

- Oracle Fusion Middleware Enterprise deployments
- Oracle Fusion Middleware 11*g* R1 SOA suite
- Oracle Fusion Middleware 11*g* R1 Identity Management suite
- Oracle Access Manager 10*g*
-

OFM 11*g* components:

- Oracle WebLogic Server
- Oracle ADF
- Oracle WebCenter
- Oracle SOA Suite
- Oracle Identity Management
- Oracle portal, forms, reports, and Discoverer
- Oracle Web tier components

4 FILE SYSTEM LAYOUT

This section describes the file system layout for the OFM 11*g* components. For example, for this validation, the product suites used are the Oracle SOA and the Oracle Identity Management Product suite. The validation is done using the Oracle Enterprise Deployment Guide Topologies.

Similarly, this recommended structure can be used for other components.

4.1 DIRECTORY STRUCTURE FOR SOA COMPONENT

Figure 1 depicts the recommended directory structure for the OFM 11*g* – SOA component.

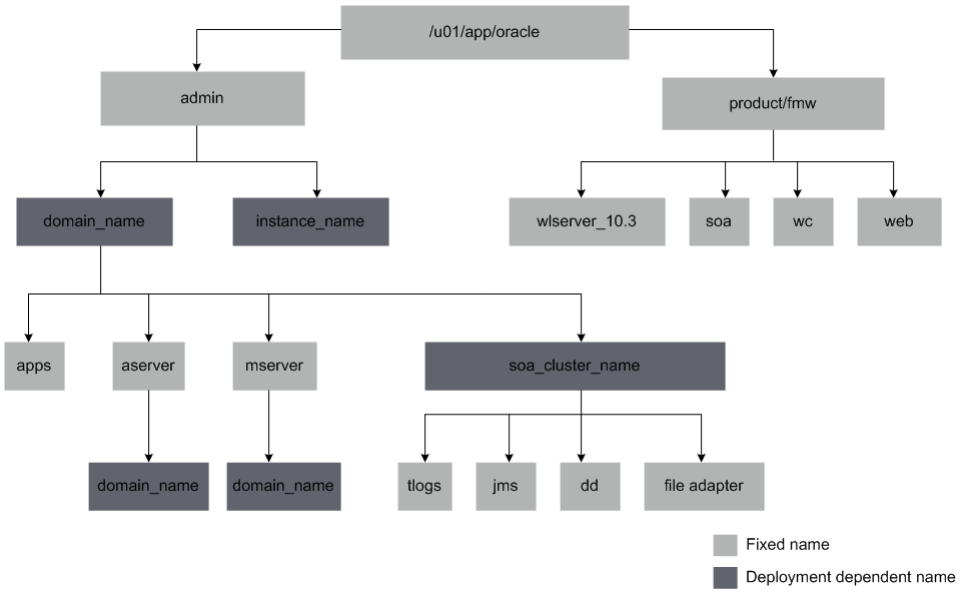


Figure 1) Directory structure for SOA.

In Figure 1, domain_name is a directory with a name that is deployment dependent. Everything in this directory applies to the "application tier" of the EDG. The Weblogic domain is also present under this directory.

Similarly, instance_name is a deployment-dependent directory, and everything in this directory applies to the "Web tier" of the EDG. The Oracle instance resides under this directory. In this case, the Oracle instance includes the Oracle HTTP server.

4.2 VOLUMES FOR SOA

Table 2) SOA volumes.

Tier	Volume Name	Mounted on Host	Mountpoint	Volume Used For
Web	VOLWEB1	WEBHOST1	/u01/app/oracle/product/fmw/web	Oracle HTTP Server installation
Web	VOLWEB2	WEBHOST2	/u01/app/oracle/product/fmw/web	Oracle HTTP Server installation
Web	VOLWEBINST1	WEBHOST1	/u01/app/oracle/admin/ohs_instance	Oracle HTTP Server instance
Web	VOLWEBINST2	WEBHOST2	/u01/app/oracle/admin/ohs_instance	Oracle HTTP Server instance
Web	VOLSTATIC1	WEBHOST1	/u01/app/oracle/admin/ohs_instance/config/static	Static HTML content
Web	VOLSTATIC2	WEBHOST2	/u01/app/oracle/admin/ohs_instance/config/static	Static HTML content
Application	VOLFMW1	SOAHOST1	/u01/app/oracle/product/fmw	WebLogic Server and Oracle SOA Suite binaries
Application	VOLFMW2	SOAHOST2	/u01/app/oracle/product/fmw	WebLogic Server and Oracle SOA Suite binaries.
Application	VOLADMIN	SOAHOST1	/u01/app/oracle/admin/soaDomain/admin	Administration Server domain directory
Application	VOLSOA1	SOAHOST1	/u01/app/oracle/admin/soaDomain/mng1	Managed Server domain directory
Application	VOLSOA2	SOAHOST2	/u01/app/oracle/admin/soaDomain/mng2	Managed Server domain directory
Application	VOLDATA	SOAHOST1, SOAHOST2	/u01/app/oracle/admin/soaDomain/soaCluster/jms /u01/app/oracle/admin/soaDomain/soaCluster/tlogs	Transaction logs and JMS data

The volumes for static HTML data are optional; Oracle Fusion Middleware can operate normally without it.

4.3 SOA EDG TOPOLOGY

Figure 2 depicts the SOA EDG topology for which the above directory structure has been used. The DR solution used in this validation deployed this topology at each site.

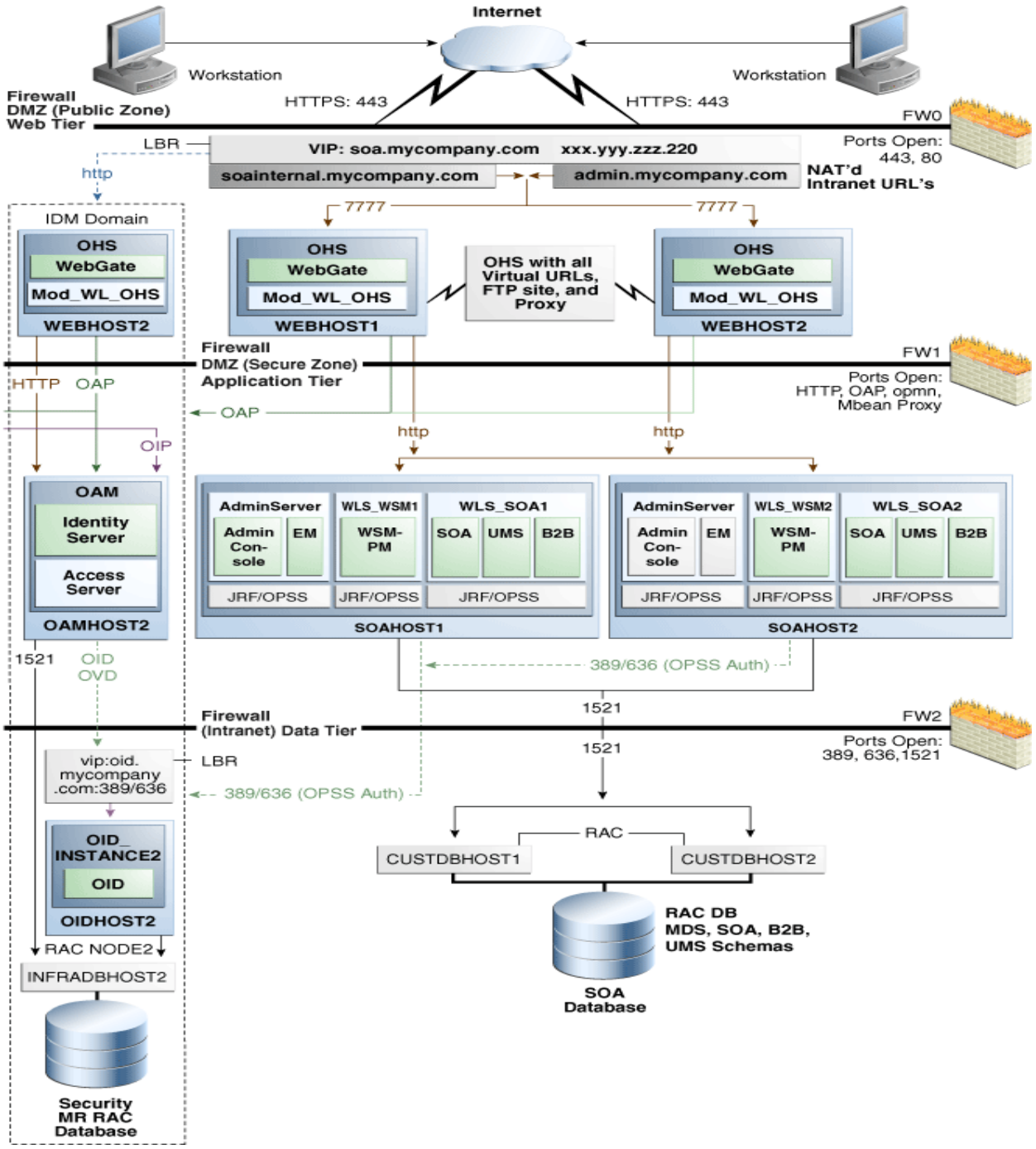


Figure 2) SOA EDG topology (graphics supplied by Oracle).

4.4 DIRECTORY STRUCTURE FOR IDM COMPONENT

Figure 3 depicts the directory structure used for the IDM Suite used in the DR validation.

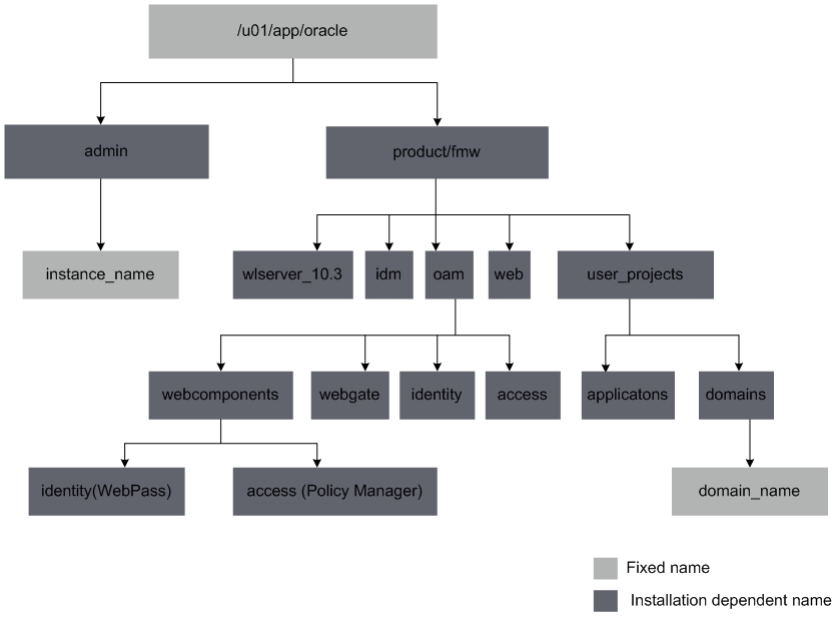


Figure 3) Directory structure for IDM.

In Figure 3, `domain_name` and applications are directories under `user_projects`. The `domain_name` has a name that is deployment dependent. Everything in this directory applies to the "application tier" of the IDM EDG. The Weblogic domain is present in this directory.

Similarly, `instance_name` is a deployment-dependent directory, and everything under it applies to the "Web tier" and the "data tier" of the IDM EDG. The Oracle Instance resides under this directory. It includes the Oracle HTTP server in the Web tier and the Oracle Internet Directory and Oracle Virtual Directory in the data tier.

4.5 VOLUMES FOR IDM

Table 3) IDM volumes.

Tier	Volume Names	Mounted on Nodes	Mountpoint	Volume Used For
Web	VOLWEB1	WEBHOST1	/u01/app/oracle/product/fmw/web	Volume for Oracle HTTP Server installations
Web	VOLWEB2	WEBHOST2	/u01/app/oracle/product/fmw/web	Oracle HTTP Server installations
Web	VOLWEBINST1	WEBHOST1	/u01/app/oracle/admin/ohs_instance	Oracle HTTP Server instances
Web	VOLWEBINST2	WEBHOST2	/u01/app/oracle/admin/ohs_instance	Oracle HTTP Server instances
Web	VOLSTATIC1[1]	WEBHOST1	/u01/app/oracle/admin/ohs_instance/config/static	static HTML content
Web	VOLSTATIC2[2]	WEBHOST2	/u01/app/oracle/admin/ohs_instance/config/static	static HTML content
Application	VOLIDM1	IDMHOST1	/u01/app/oracle/product/fmw	Identity Management Middleware homes
Application	VOLIDM2	IDMHOST2	/u01/app/oracle/product/fmw	Identity Management Middleware homes
Application	VOLIDMINST1	IDMHOST1	/u01/app/oracle/admin	Oracle instances
Application	VOLIDMINST2	IDMHOST2	/u01/app/oracle/admin	Oracle instances
Application	VOLOAM1	OAMHOST1	/u01/app/oracle/product/fmw/oam	Oracle Access Manager Identity Server and Access Server homes
Application	VOLOAM2	OAMHOST2	/u01/app/oracle/product/fmw/oam	Oracle Access Manager Identity Server and Access Server homes
Application	VOLOAMADMIN	OAMADMIN HOST	/u01/app/oracle	Oracle Access Manager administration components
Directory	VOLOID1	OIDHOST1	/u01/app/oracle/product/fmw/idm	Oracle Internet Directory Oracle homes
Directory	VOLOID2	OIDHOST2	/u01/app/oracle/product/fmw/idm	Oracle Internet Directory Oracle homes
Directory	VOLOIDINST1	OIDHOST1	/u01/app/oracle/admin	Oracle Internet Directory Oracle instances
Directory	VOLOIDINST2	OIDHOST2	/u01/app/oracle/admin	Oracle Internet Directory Oracle instances
Directory	VOLOVD1	OVDHOST1	/u01/app/oracle/product/fmw/idm	Oracle Virtual Directory Oracle homes
Directory	VOLOVD2	OVDHOST2	/u01/app/oracle/product/fmw/idm	Oracle Virtual Directory Oracle homes
Directory	VOLOVDINST1	OVDHOST1	/u01/app/oracle/admin	Oracle Virtual Directory Oracle instances
Directory	VOLOVDINST2	OVDHOST2	/u01/app/oracle/admin	Oracle Virtual Directory Oracle instances

[1] This volume for static HTML data is optional. Oracle Fusion Middleware will operate normally without it. See http://download.oracle.com/docs/cd/E15523_01/doc.1111/e15250/creating_sites.htm#sthref588.

[2] See http://download.oracle.com/docs/cd/E15523_01/doc.1111/e15250/creating_sites.htm#sthref589.

4.6 IDM EDG TOPOLOGY

Figure 4 represents the IDM EDG topology in which the above directory structure has been used.

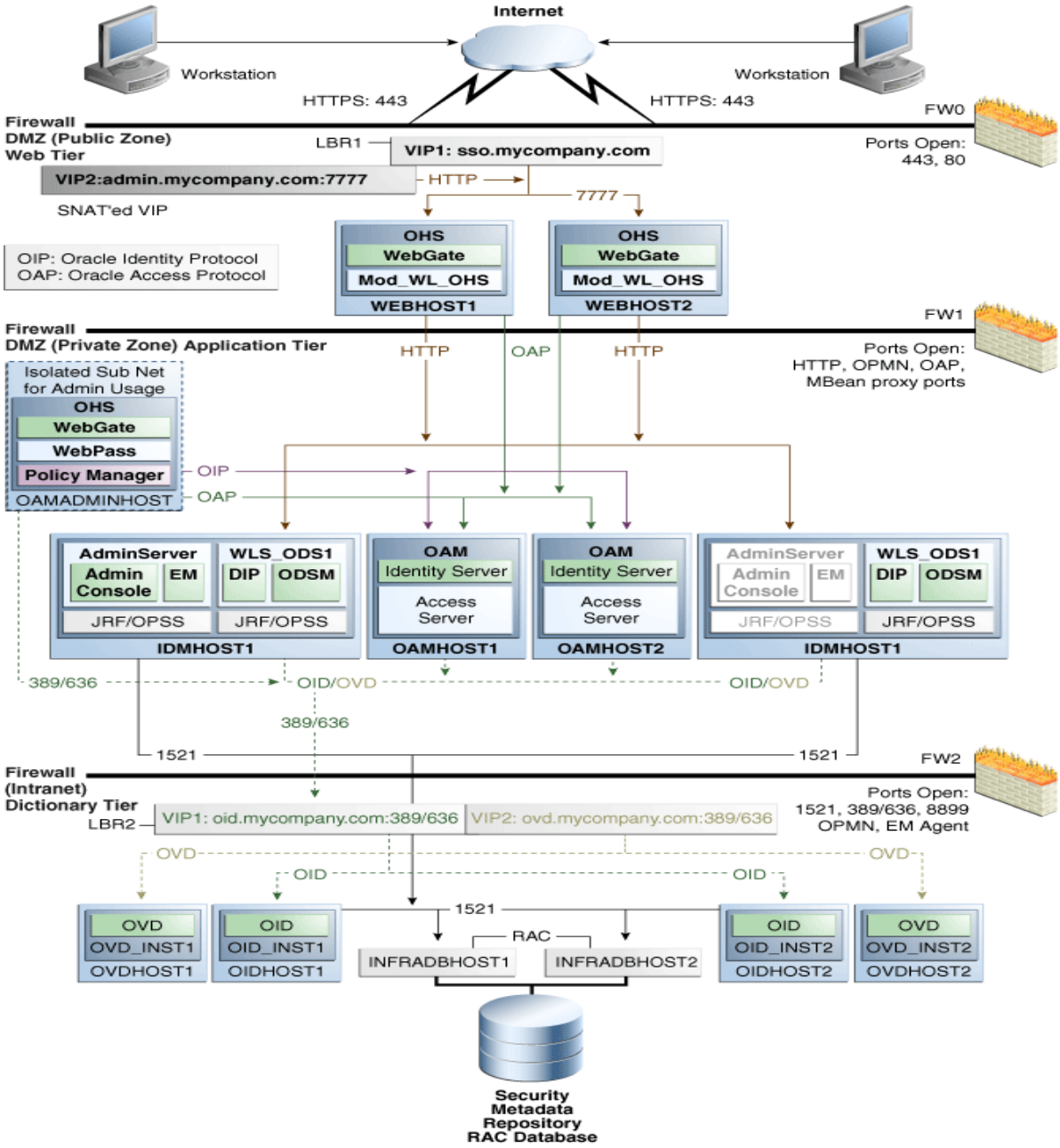


Figure 4) IDM EDG topology (graphics supplied by Oracle).

5 CONSISTENCY GROUPS

Consistency Group is a storage-level view of applications data stored in multiple volumes or controllers. Consistency groups are typically used in database applications where logs and database are part of the same consistency group. In this case, the database cannot get ahead of the logs as the writes are all ordered. Consistency groups are collections of objects that allow an administrator to take consistent point in time copies of, for example, volumes today and LUNs, files, block ranges in the future.

There are two levels of consistency:

- **Application Consistency**
  - Consistent copies are created after applications are gracefully shut down, quiesced, or put in hot backup mode
  - Provides application-defined benefits such as media recovery
- **Crash Consistency**
  - Creates point-in-time copy of storage that is usable with crash recovery applications
  - Creates crash consistent copies without coordinating with applications. However, write ordering is maintained for dependent writes in Snapshot copies across volumes.

Consistency Group can be enabled by running the following APIs from any servers where NetApp volumes are mounted:

- cg-start
  - Fences all writes for a volume per controller
  - Freezes volume contents during write fencing to prevents writes
  - Returns fencing success or failure
  - If fencing is successful, it continues with Snapshot copy creation based on frozen contents
- cg-commit
  - Unfences volumes after start of WAFL® Consistency Point (CP) to create a Snapshot copy
  - Returns success after creating a Snapshot copy
  - Snapshot copies created using the **`cg-start`** and **`cg-commit`** commands are replicated the same way as other Snapshot copies. No special handling exists for CG Snapshot copies.
  - Volume SnapMirror maintains a mirror of all Snapshot copies at the destination

Consistency Group (CG) is composed of three parts—Part 1 is the main library of the API, part 2 is the Perl script that calls the API libraries, and part 3 is the configuration file that lists the volumes and storage arrays in which these volumes reside.

It is important to note that part 2 and part 3 should be named to something meaningful and their names should correlate to each other.

Example:

Group 1 – Volumes VOLADMIN, VOLSOA1, VOLSOA2

Group 2 – Volumes VOLDATA

The following example shows the creation of four files, that is—two `create<>.pl` and two config files.

Group 1 would consist of `Cg_create_DOMAINGROUP.pl` and `DOMAINGROUP.cfg`

Group 2 would consist of `Cg_create_DATAGROUP.pl` and `DATAGROUP.cfg`

There is only one item to modify within `Cg_create_<>.pl`. If there are multiple files, create multiple consistency groups.

1. To create files, copy the original `Cg_create_<>.pl`:

   For example:

   `cp Cg_create_<>.pl  Cg_create_DOMAINGROUP.pl`

2. Modify `Cg_create_DOMAINGROUP.pl` as follows:

   i. Using VI open `Cg_create_DOMAINGROUP.pl`

   ii. Edit line #8 to change:

   ```
   open("CFG","cg.cfg" || die "Can't open config file: $!");
   ```

   to

   ```
   open("CFG"," DATAGROUP.cfg " || die "Can't open config file: $!");
   ```

3. Change `cg.cfg` to the configuration file name created:

   `cp cf.cfg DOMAINGROUP.cfg`

   `vi DOMAINGROUP.cfg`

4. Enter the storage and volume information:

   ```
   Config file ...
   FilerName        User      Password      Timeout    Volumes
   atlnetapp5       orabk      orabk        relaxed    VOLADMIN, VOLSOA1, VOLSOA2
   ```

5. If multiple storage arrays are used, add a second line:

   ```
   Config file ...
   FilerName        User    Password        Timeout    Volumes
   atlnetapp5       orabk      orabk        relaxed    VOLADMIN, VOLSOA1
   atlnetapp6       orabk      orabk        relaxed    VOLSOA2
   ```

6. To execute from GridControl, cron or any desired scheduler, use the following syntax :

   `perl Cg_create<>.pl <Snapshot copy_name>`

   ```
   [oracle@atl46004][asmdb4][~/cg]$ perl Cg_create_DOMAINGROUP.pl cgsnap_`date
   +%m%d%y%H%M`
   Input XML:
   <cg-start>
       <snapshot>snapname</snapshot>
       <timeout>relaxed</timeout>
       <volumes>
           <volume-name>VOLADMIN</volume-name>
           <volume-name>VOLSOA1</volume-name>
           <volume-name>VOLSOA2</volume-name>
       </volumes>
   </cg-start>

   Output XML1:
   <results status="passed">
       <cg-id>228</cg-id>
   </results>


   Commit XML2:
   <results status="passed"></results>
   ```

When you execute `Cg_create_DOMAINGROUP.pl` it parses the `DOMAINGROUP.cfg` configuration file and passes the information to NetApp APIs to create a single name Snapshot copy that spans multiple volumes/controllers.

5.1 CONSISTENCY GROUPS FOR SOA EDG

The volumes created earlier are grouped together into consistency groups as shown in Table 4.

Table 4) SOA EDG consistency groups.

Tier	Group Name	Members	Consistency Group Elements
Application	DOMAINGROUP	VOLADMIN VOLSOA1 VOLSOA2	Consistency group for the Administration Server, Managed Server domain directory
Application	DATAGROUP	VOLDATA	Consistency group for the JMS file store and transaction log data
Application	FMWHOMEGROUP	VOLFMW1 VOLFMW2	Consistency group for the Middleware homes
Web	WEBHOMEGROUP	VOLWEB1 VOLWEB2	Consistency group for the Oracle HTTP Server Oracle homes
Web	WEBINSTANCEGROUP	VOLWEBINST1 VOLWEBINST2 VOLSTATIC1 VOLSTATIC2	Consistency group for the Oracle HTTP Server Oracle instances

5.2 CONSISTENCY GROUPS FOR IDM EDG

The volumes created earlier are grouped together into consistency groups as shown in Table .

Table 5) IDM EDG consistency groups.

Tier	Group Name	Members	Consistency Group Elements
Directory	OIDHOMEGROUP	VOLOID1 VOLOID2	Oracle Internet Directory Oracle homes
Directory	OIDINSTGROUP	VOLOIDINST1 VOLOIDINST2	Oracle Internet Directory Oracle instances
Directory	OVDHOMEGROUP	VOLOVD1 VOLOVD2	Oracle Virtual Directory Oracle homes
Directory	OVDINSTGROUP	VOLOVDINST1 VOLOVDINST2	Oracle Virtual Directory Oracle instances
Application	IDMMWGROUP	VOLIDM1 VOLIDM2	Middleware homes
Application	IDMINSTGROUP	VOLIDMINST1 VOLIDMINST2	Identity Management instances
Application	OAMGROUP	VOLOAM1 VOLOAM2	Oracle Access Manager Identity Server and Access Server homes
Application	OAMADMINGROUP	VOLOAMADMIN	Oracle Access Manager administration host components

Tier	Group Name	Members	Consistency Group Elements
Web	WEBHOMEGROUP	VOLWEB1 VOLWEB2	Oracle HTTP Server Oracle homes
Web	WEBINSTGROUP	VOLWEBINST1 VOLWEBINST2 VOLSTATIC1 VOLSTATIC2	Oracle HTTP Server Oracle instances

6 PLANNED AND UNPLANNED DOWNTIME

6.1 PERFORMING A PLANNED SWITCHOVER

When you plan to take down the production site (for example, to perform maintenance) and make the current standby site the new production site, you must perform a switchover operation so that the standby site takes over the production role.

Follow these steps to perform a switchover operation:

1. Shut down any processes that are still running on the production site. This includes the database instances in the data tier, Oracle Fusion Middleware instances, and any other related processes in the application tier and Web tier.
2. Stop the replication (SnapMirror relationship) between the production site NetApp storage system and the standby site.
3. Use Oracle Data Guard to switch over the database(s).
4. On the standby site hosts, manually start all processes. This includes the database instances in the data tier, Oracle Fusion Middleware instances and any other processes in the application and Web tier.
5. Make sure that all user requests are routed to the standby site by performing a global DNS push or something similar, such as updating the global load balancer.
6. Use a browser client to perform postswitchover application testing to confirm that requests are being resolved and redirected to the standby site.

   At this point, the former standby site is the new production site, and the former production site is the new standby site.
7. Reestablish the replication between the two sites, but configure the replication so that the Snapshot copies go in the opposite direction (from the current production site to the current standby site). Refer to the documentation for your shared storage to learn how to configure the replication so that Snapshot copies are transferred in the opposite direction.

After these steps have been performed, the former standby site is the new production site and the former production site is the new standby. At this point, you can perform maintenance (if any) at the new standby site.

6.2 PERFORMING A SWITCHBACK

After a switchover operation has been performed, a switchback operation can be performed to revert the current production site and the current standby site to the roles they had prior to the switchover operation.

Follow these steps to perform a switchback operation:

1. Shut down any processes running on the current production site. This includes the database instances in the data tier, Oracle Fusion Middleware instances, and any other processes in the application and Web tier.
2. Stop the replication (SnapMirror relationship) between the production site NetApp storage system and the standby site.

3. Use Oracle Data Guard to switch back the databases.
4. On the new production site hosts, manually start all processes. This includes the database instances in the data tier, Oracle Fusion Middleware instances, and any other processes in the application tier and Web tier.
5. Make sure that all user requests are routed to the new production site by performing a global DNS push or something similar, such as updating the global load balancer.
6. Use a browser client to perform postswitchback testing to confirm that requests are being resolved and redirected to the new production site.
7. At this point, the former standby site is the new production site and the former production site is the new standby site.
8. Reestablish the replication between the two sites, but configure the replication so that the Snapshot copies go in the opposite direction (from the new production site to the new standby site). Refer to the documentation for your shared storage to learn how to configure the replication so that Snapshot copies are transferred in the opposite direction.

6.3 PERFORMING AN UNPLANNED FAILOVER

When the production site becomes unavailable unexpectedly, you must perform a failover operation so that the standby site takes over the production role.

Follow these steps to perform a failover operation:

1. Stop the replication (SnapMirror relationship) between the production site NetApp storage system and the standby site. From the standby site, use Oracle Data Guard to fail over the databases.
2. On the standby site hosts, manually start all processes. This includes the database instances in the data tier, Oracle Fusion Middleware instances, and any other processes in the application and Web tier.
3. Make sure that all user requests are routed to the standby site by performing a global DNS push or something similar, such as updating the global load balancer.
4. Use a browser client to perform postfailover testing to confirm that requests are being resolved and redirected to the production site.
5. At this point, the standby site is the new production site. You can examine the issues that caused the former production site to become unavailable.
6. To use the original production site as the current standby site, you must reestablish the replication between the two sites, but configure the replication so that the Snapshot copies go in the opposite direction (from the current production site to the current standby site). Refer to the documentation for your shared storage system to learn how to configure the replication so that Snapshot copies are transferred in the opposite direction.

7 SCREEN CAPTURES

Figures 5 through 8 depict the switchover and failover process.

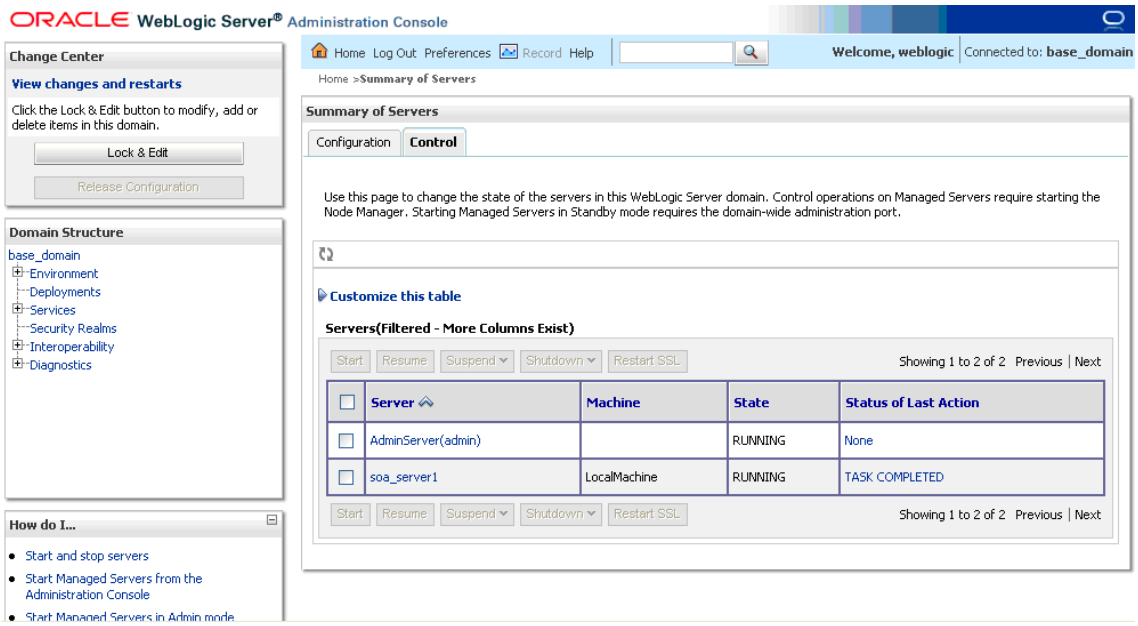


Figure 5) SOA server is installed and running on the primary site.

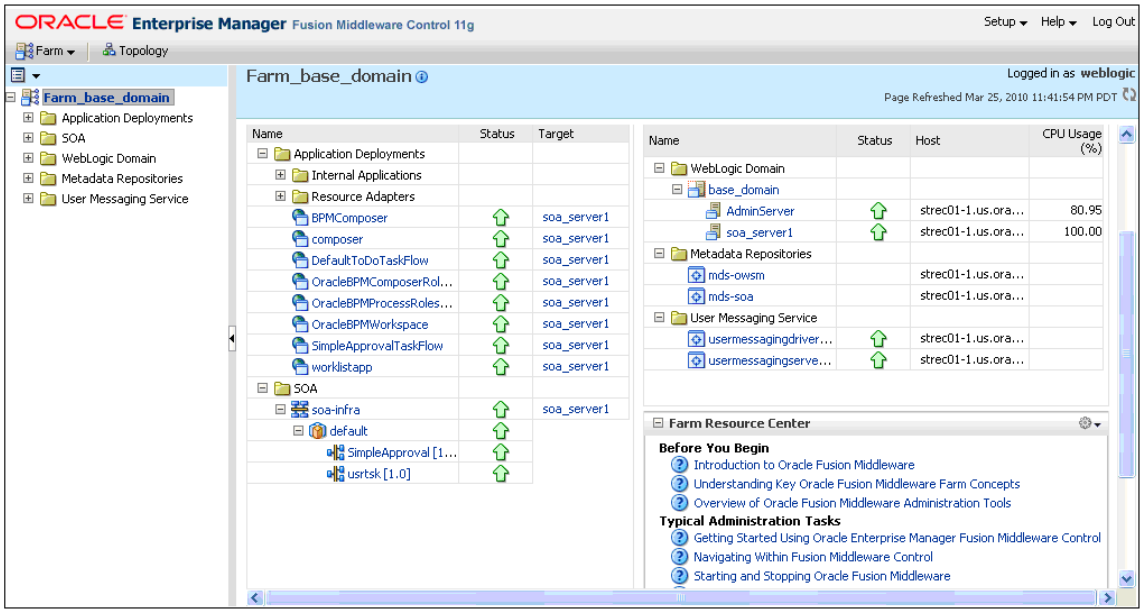


Figure 6) All processes up and running in the primary site.

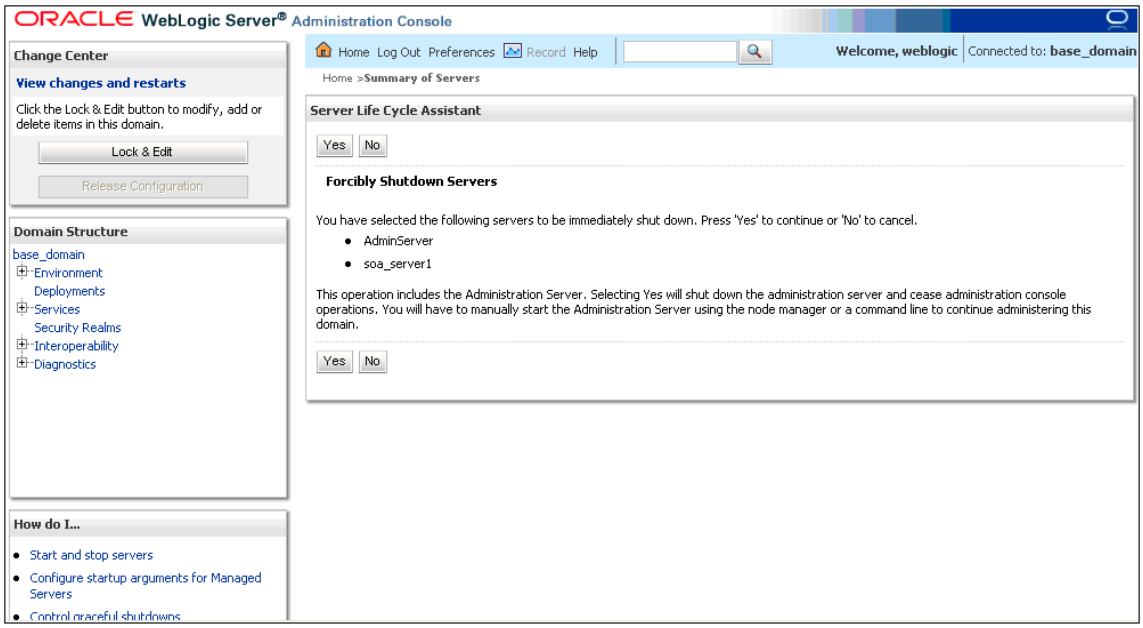
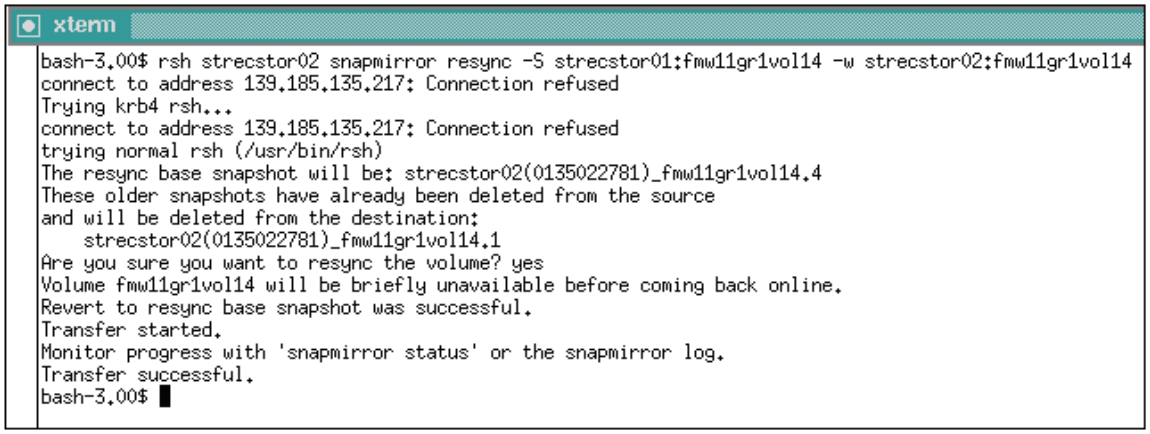


Figure 7) Shutting down all processes running on the primary site.

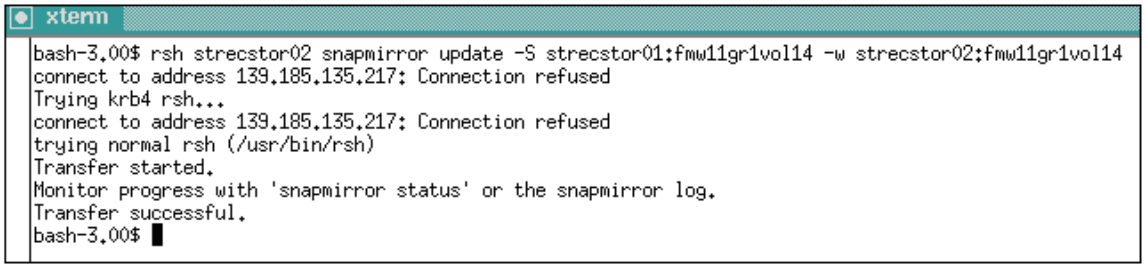
To Check the Mirror Status

```
xterm
bash-3.00$ rsh strecstor02 snapmirror status  strecstor02:fmw11gr1vol01
connect to address 139.185.135.217: Connection refused
Trying krb4 rsh...
connect to address 139.185.135.217: Connection refused
trying normal rsh (/usr/bin/rsh)
Snapmirror is on.
Source                      Destination                 State          Lag        Status
strecstor01:fmw11gr1vol01   strecstor02:fmw11gr1vol01   Broken-off     362:20:49  Pending
strecstor02:fmw11gr1vol01   strecstor01:fmw11gr1vol01   Source         171:28:54  Idle
bash-3.00$
```

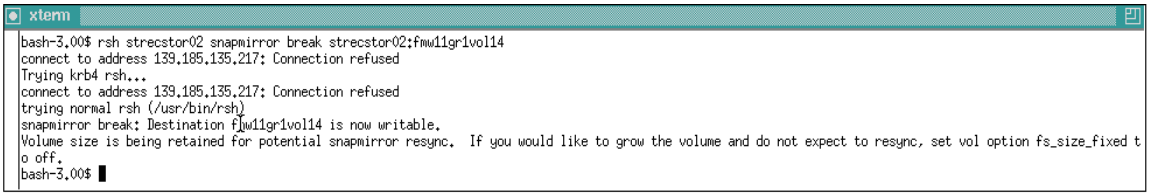
To Resync the Mirror



To Update the Mirror



To Break the Mirror



Start all the processes after the switchover and verify the EM console. The EM console is the Fusion Middleware console that manages Fusion Middleware domains.

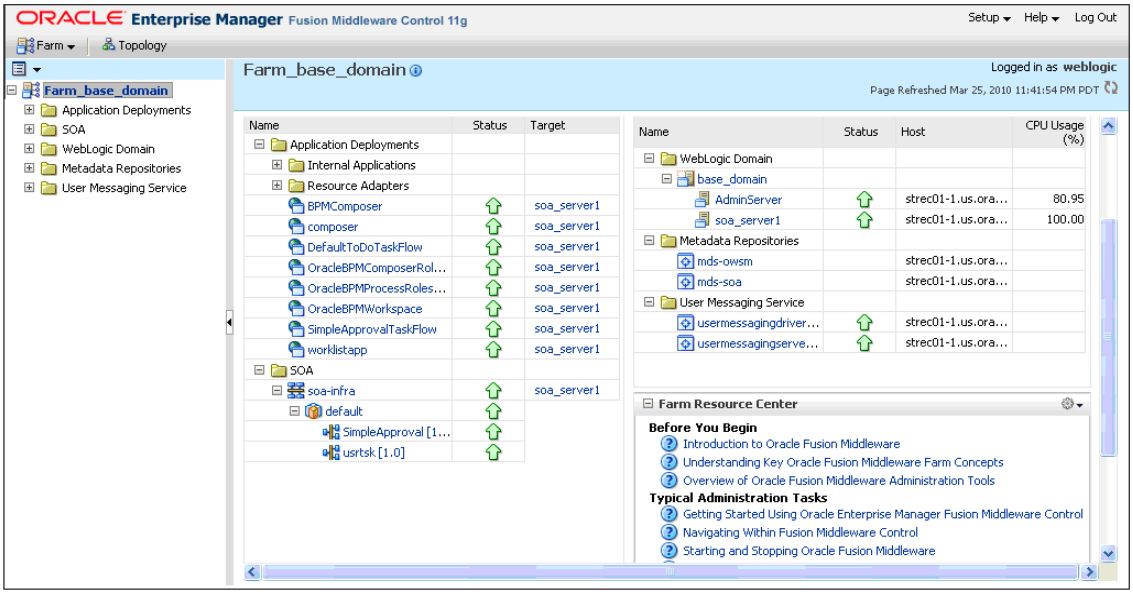


Figure 8) EM console.

8 CONCLUSION

NetApp SnapMirror simplifies the Oracle Fusion Middleware replication process; the use of storage-level mirroring allows the copies to be created quickly, efficiently, and independently of the server. This maximizes the resources on the source server available for production/online use. The mirroring can also be started in advance so that only the last incremental changes need to be transferred during cloning, thus shortening the whole process.

This DR solution provides an optimal process for Oracle Fusion Middleware replication. This in turn enables flexibility in setting the frequency of cloning to satisfy the cloning requirements of the enterprise, be it for development, testing, reporting, or whatever the case may be. SnapMirror is easy to set up, configure, maintain and, most important, is cost-effective as a mirroring solution.

Using NetApp storage systems and SnapMirror in conjunction with Oracle DataGuard greatly simplifies and speeds up the Oracle Fusion Middleware replication process. This provides users with the maximum benefit out of their investment in the overall system.

ACKNOWLEDGEMENTS

Shilpa Shree and Shailesh Dwivedi, Oracle

Neto, Bill Heffelfinger, Lynne Thieme, Steve Schuettinger, Generosa Litton, and Esther Smitha, NetApp

APPENDIXES

APPENDIX A: REFERENCES

High Availability Guide

Disaster Recovery Guide

Enterprise Deployment Guide for Oracle WebCenter

Enterprise Deployment Guide for Oracle SOA Suite

Enterprise Deployment Guide for Oracle Identity Management

http://download.oracle.com/docs/cd/E15523_01/doc.1111/e15250/creating_sites.htm#BABGJFDC

APPENDIX B: SCRIPTS

CREATE_CG_SNAPS

This script creates a consistency group Snapshot copy.

```
#!/opt/local/bin/perl
use lib "NetApp";
use NaServer;
use NaElement;
sub open_cfg {
        open("CFG","cg.cfg" || die "Can't open config file: $!");
        while(<CFG>) {
                chomp;
                if(/^\#/ || /^FilerName/) {
                        next;
                }
                @cfgline = split /\s+/;
                push(@FILERLIST,[@cfgline]);
        }

    close(CFG);
}

&open_cfg;
$snapname = shift;
&loop_cgstart;
&loop_cgcommit;

sub loop_cgstart()
{
        for $i ( 0 .. $#FILERLIST ) {
                &cg_start(${FILERLIST[$i]});
        }
}

sub loop_cgcommit()
{
        for $i ( 0 .. $#FILERLIST ) {
                &cg_commit(${FILERLIST[$i]});
        }
}
sub cg_start()
{
```

```
        $filername = $FILERLIST[$i][0];
        $username = $FILERLIST[$i][1];
        $password = $FILERLIST[$i][2];
        $timeout = $FILERLIST[$i][3];
        @volumes = split(",",$FILERLIST[$i][4]);

        chomp ($filername);
        chomp ($username);
        chomp ($password);
        chomp ($timeout);
        chomp ($snapname);
        chomp (@volumes);
        my $zapicon = NaServer->new($filername, 1, 3);

        $zapicon->set_style(LOGIN_PASSWORD);
        $zapicon->set_admin_user($username, $password);

                if (!defined($zapicon))
        {
                print "Connection to $filername failed.\n";
                exit 2;
        }

        $zapicon->set_transport_type(NA_SERVER_TRANSPORT_HTTP);
        if (!defined($zapicon))

        {
                print "Unable to set HTTP transport.\n";
                exit 2;
        }

        my $zapicmd = NaElement->new("cg-start");
        $zapicmd->child_add_string("snapshot",$snapname);
        $zapicmd->child_add_string("timeout",$timeout);

        my $volumecount = @volumes;
        chomp (@volumes);
        my $zapivols = NaElement->new("volumes");
        while ($volumecount > 0) {
                $zapivols->child_add_string("volume-name",shift(@volumes));
                $volumecount--;
        }
        $zapicmd->child_add($zapivols);

        my $zapiin=$zapicmd->sprintf();
        print "Input XML:\n$zapiin \n";

        my $zapiout = $zapicon->invoke_elem($zapicmd);
        my $zapiout=$zapiout->sprintf();
        print "Output XML1:\n$zapiout \n";
        @precgid = split(/<cg-id>/,$zapiout);
        @cgid = split(/<\/cg-id>/,$precgid[1]);
        $cgid = @cgid[0];
        $filenames{$i} = $cgid;
#       &loop_cgcommit;
}
sub cg_commit()
{
```

```
        $filername = $FILERLIST[$i][0];
        $username = $FILERLIST[$i][1];
        $password = $FILERLIST[$i][2];
        $timeout = $FILERLIST[$i][3];
        @volumes = split(",",$FILERLIST[$i][4]);

        chomp ($filername);
        chomp ($username);
        chomp ($password);
        chomp ($timeout);
        chomp ($snapname);
        chomp (@volumes);

        my $zapicon = NaServer->new($filername, 1, 3);

        $zapicon->set_style(LOGIN_PASSWORD);
        $zapicon->set_admin_user($username, $password);

                if (!defined($zapicon))
        {
                print "Connection to $filername failed.\n";
                exit 2;
        }

        $zapicon->set_transport_type(NA_SERVER_TRANSPORT_HTTP);
        if (!defined($zapicon))

        {
                print "Unable to set HTTP transport.\n";
                exit 2;
        }

        $cgid = $filenames{$i};
        print "\n\n";
        my $zapiout = $zapicon->invoke("cg-commit","cg-id",$cgid);
        my $zapiout=$zapiout->sprintf();
        print "Commit XML2:\n$zapiout \n";
}

sub syntax_printer()
{
        print "USAGE: !$ <snapshot name> <config file>\n";
        exit 0;
}
```

www.netapp.com

TR-3855