SPARC: Oracle® Solaris Cluster Data Replication Guide for MySQL

Part No: E69339
August 2018

Oracle Solaris Cluster Data Replication Guide for MySQL

Part No: E69339

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Référence: E69339

Accès aux services de support Oracle

Les clients Oracle qui ont souscrit un contrat de support ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` ou le site `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` si vous êtes malentendant.

Contents

Using This Documentation

- **Overview** – Describes how to administer MySQL data replication with the Oracle Solaris Cluster disaster recovery framework.
- **Audience** – Experienced system administrators with extensive knowledge of Oracle software and hardware.
- **Required knowledge** – Knowledge of the Oracle Solaris operating system, of Oracle Solaris Cluster software, and expertise with the volume manager software that is used with Oracle Solaris Cluster software.

This document is not to be used as a planning or presales guide.

Product Documentation Library

Documentation and resources for this product and related products are available at `http://docs.oracle.com/cd/E69294_01`.

Feedback

Provide feedback about this documentation at `http://www.oracle.com/goto/docfeedback`.

♦ ♦ ♦ CHAPTER 1

Replicating Data With MySQL

This chapter describes the Oracle MySQL (MySQL) data replication when used with the Oracle Solaris Cluster disaster recovery framework. It covers the following topics:

Some procedures and examples in this guide use the example cluster names `cluster-paris` and `cluster-newyork` to identify the primary and secondary clusters, respectively, in the partnership. Also used are the example cluster node names `phys-paris-`*N* and `phys-newyork-`*N*.

Overview of MySQL Replication Resource Groups

This section provides an overview of the MySQL replication resource groups. A protection group that secures MySQL databases with MySQL replication consists of the following two resource groups securing a third resource group that is not part of the protection group on each cluster:

- "MySQL Database Resource Groups" on page 10
- "MySQL Replication Resource Groups" on page 10
- "MySQL Application Resource Groups" on page 10

The failover MySQL database resource group holding a MySQL database is the foundation underneath the protection group. Therefore, strong positive affinities with failover delegation must exist from from the replication resource group to the failover MySQL database resource group.

The `mysql_geo_register` script creates the replication resource group and sets affinity from the replication resource group to the failover MySQL database resource group. You must add to this script an affinity from the application resource group to the failover MySQL database resource group. If affinity or dependency is needed from the application resource group that is added to the MySQL protection group, you must add this affinity or dependency.

MySQL Database Resource Groups

The MySQL database resource group must be configured as a failover resource group. It typically contains the following resources:

- Storage resource, which manages database storage
- Logical host resource, which provides the address to connect with the MySQL replication user
- MySQL database resource, which resides on top of the first two resources to make the database highly available locally

On single-node clusters, the storage resource can be omitted. The creation of the database resource group and its resources is the topic of *Oracle Solaris Cluster Data Service for MySQL Guide*.

The database failover resource group and its objects can have different names across the clusters.

MySQL Replication Resource Groups

The MySQL replication resource group contains the MySQL replication resource. This resource does not start or stop any process. Its only purpose is to monitor the status of the MySQL database replication.

MySQL Application Resource Groups

The MySQL application resource group must contain at least a logical host resource, which provides the address for all the clients to use for connections to the database.

Initial Configuration of MySQL Replication

This section describes the steps you need to perform to configure MySQL software on the primary and secondary clusters:

- "Installing MySQL and Configuring the MySQL Database Resource Group" on page 11
- "How to Configure the MySQL Replication" on page 11
- "Configuring the MySQL Application Resource Group" on page 15

Installing MySQL and Configuring the MySQL Database Resource Group

You can install MySQL and configure both clusters as described in *Oracle Solaris Cluster Data Service for MySQL Guide*.

The database resource group and its resources do not have to have the same name on both clusters.

Requirements for the disaster recovery framework controlling MySQL are as follows:

- Do not use the bind-address keyword in the MySQL configuration file `my.cnf`.
- Specify the nodes of all clusters in the `mysql_config` file for database preparation.

When preparing the MySQL database for cluster usage with the `mysql_register` script, you must provide all the physical node names of your clusters in the `MYSQL_NIC_HOSTNAME` variable. For example:

```
MYSQL_NIC_HOSTNAME="phys-paris-1,phys-paris-2,phys-newyork-1,phys-newyork-2"
```

When you configure the database resource, keep the following restrictions in mind:

- MySQL replication between the two clusters must be the only MySQL replication configured in the two databases.
- MySQL databases on both clusters must be configured to listen on the same port.

▼ How to Configure the MySQL Replication

Before You Begin Determine which cluster will contain the master database at the first start.

1. **Prevent the startup of the slave threads.**

On the primary cluster at the node where the MySQL database is active, add the `skip-slave-start` keyword to the `my.cnf` file. For example:

```
cl1-node1# echo skip-slave-start >> /mysql-data-directory/my.cnf
```

2. **(Optional) Prevent non-`root` modifications.**

   On the secondary cluster at the node where the MySQL database is active, add the `read-only=true` directive to the `my.cnf` file, and restart your database.

   Note - If you perform this step, you must also set `READONLY=true` in the `mysql_geo_config` file that is used to create the MySQL protection group. That task is described in Step 2 of "How to Create and Configure a MySQL Protection Group" on page 17.

   For example:

```
cl2-node1# echo read-only=true >> /mysql-data-directory/my.cnf
cl2-node1# clresource disable newyork-mys-rs
cl2-node1# clresource enable newyork-mys-rs
```

3. **Create the replication user on both databases.**

   a. **On each cluster, pick the node where the MySQL database is active, and connect as an administrative user who can at least create users.**

   b. **Create the replication user, and stay connected.**

   Note - Be sure to create the replication with permissions to connect from any node.

   - **On the primary cluster, which listens to the socket `/tmp/paris.sock`, issue the following commands:**

```
cl1-node1:/ # /usr/local/mysql/bin/mysql -S /tmp/paris.sock -uroot -proot
mysql> use mysql
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'phys-paris-1' identified by
 'repl';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'phys-paris-2' identified by
 'repl';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'phys-newyork-1' identified by
 'repl';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'phys-newyork-2' identified by
 'repl';
```

   - **On the secondary cluster, which listens to the socket `/tmp/newyork.sock`:**

```
cl2-node3:/ # /usr/local/mysql/bin/mysql -S /tmp/newyork.sock -uroot -proot
mysql> use mysql
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'phys-paris-1' identified by
 'repl';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'phys-paris-2' identified by
 'repl';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'phys-newyork-1' identified by
 'repl';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'phys-newyork-2' identified by
 'repl';
```

4. **Establish the replication between the secondary and primary clusters.**

   a. **On the primary cluster, issue the following on the MySQL client:**

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> show master status;
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB  |

| bin-log.000002 |     1424 |              | sc3_test_database |

1 row in set (0.03 sec)

mysql> unlock tables;
```

Note the values for file and position. In the preceding example, they are `bin-log.000002` and `1424`, respectively.

   b. **On the c client on the secondary cluster, issue the following commands:**

```
mysql> change master to master_host='cluster-paris',

-> master_user='repl',

-> master_password='repl',

-> master_log_file='bin-log.000002',

-> master_log_pos=1424;

Query OK, 0 rows affected (0.04 sec)

mysql> start slave;

Query OK, 0 rows affected (0.03 sec)
```

c. **Check the slave status.**

```
mysql> show slave status\G
```

d. **Verify that the following status is reported in the output of the `show slave status` command.**

```
Slave_IO_State: Waiting for master to send event
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

Consult your MySQL documentation to troubleshoot any unexpected status.

5. **Configure the reverse replication to prepare the two clusters for a role swap.**

a. **On the secondary cluster, issue the following commands:**

```
mysql> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.01 sec)
mysql> show master status;

| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB  |

| bin-log.000020 |     1162 |              | sc3_test_database |

1 row in set (0.00 sec)

mysql> unlock tables;
```

Note the values for file and position. In the preceding example, they are `bin-log.000020` and `1162`, respectively.

b. **On the MySQL client on the primary cluster, issue the following commands:**

```
mysql> change master to master_host='cluster-newyork',

-> master_user='repl',

-> master_password='repl',

-> master_log_file='bin-log.000020',

-> master_log_pos=1162;

mysql> start slave;
```

```
Query OK, 0 rows affected (0.03 sec)
```

c. **Check the slave status.**

```
mysql> show slave status\G
```

d. **"Waiting for master to sent event" at `Slave_IO_State` check for the following message:**

```
Slave_IO_Running: Yes

Slave_SQL_Running: Yes
```

e. **Stop the slave, and exit the MySQL client.**

```
mysql> stop slave;
mysql> exit;
```

f. **On the MySQL client on the secondary cluster, start the slave, and exit the client.**

```
mysql> start slave;
mysql> exit;
```

Configuring the MySQL Application Resource Group

At a minimum, you must create a resource group that contains a logical host resource. You must leave the resource group in an unmanaged state.

For example, on the cluster-paris cluster, you would issue the following commands:

```
phys-paris-1# clresourcegroup create usa-rg
phys-paris-1# clresourcegroup set -p Auto_start_on_new_cluster=false usa-rg
phys-paris-1# clresourcegroup set -p RG_Affinities=+++paris-rg usa-rg
phys-paris-1# clreslogicalhostname create -g usa-rg cluster-usa
phys-paris-1# clresource enable cluster-usa
```

On the cluster-newyork cluster, you would issue the following commands:

```
phys-newyork-1# clresourcegroup create usa-rg
phys-newyork-1# clresourcegroup set -p Auto_start_on_new_cluster=false usa-rg
phys-newyork-1# clresourcegroup set -p RG_Affinities=+++newyork-rg usa-rg
phys-newyork-1# clreslogicalhostname create -g usa-rg cluster-usa
phys-newyork-1# clresource enable cluster-usa
```

Planning a MySQL Protection Group

A MySQL database resource group can belong to only one protection group.

The MySQL geographic replication was developed with the disaster recovery framework script-based plug-in module, so it must comply with all rules of the script-based plug-in. For each protection group, you must provide a script-based plug-in configuration file on each node. In addition, the MySQL geographic replication brings in its own configuration file, which is needed only at registration.

The MySQL geographic replication creation is an automated process that takes the MySQL geographic configuration file as input and performs the necessary actions. The essential content of this file consists of `key=`*value* pairs. The keys are:

`APPRG` Application resource group, which is unmanaged and contains at least the logical host for client access.

`CONFIGFILE` Absolute path for the script-based plug-in configuration file.

`DESC` Description for the protection group.

`LONGPING` Timeout for the extensive `ping` test. The default is 60 seconds if this variable is unset. This timeout is used at `check_takeover` where it must be verified as the remote site is unavailable.

`PG` Name of the protection group to create.

`PS` Name of the partnership.

`READONLY` Switch for setting the read-only variable on the MySQL slave. If the read-only variable should not be set, leave this value undefined. Any entry here triggers the read-only variable to be set.

`REALMYSRG` Resource group names that contain the MySQL database resource on the clusters. If the resource group names on the clusters differ, provide a comma-separated list.

Name of the partnership.

`REALMYSRS` Resource names configured as the master and slave MySQL database resources. If the resource names on the clusters differ, provide a comma-separated list.

`REPCOMP` Name of the replicated component to create in this protection group.

`REPRG` Name of the replication resource group.

`REPRS` Name of the replication resource.

`SHORTPING` Timeout for the short `ping` test. The default is 10 seconds if this variable is unset. The short `ping` timeout is used whenever a connection should succeed but is not required to succeed.

Creating and Validating a MySQL Protection Group

Protection group names are unique in the global disaster recovery framework namespace. You cannot use the same protection group name in two partnerships on the same system. In addition, you can replicate the existing configuration of a protection group from a remote cluster to a local cluster. For more information, see "Replicating a MySQL Protection Group Configuration to a Partner Cluster" on page 29.

This section contains the following information:

- "How to Create and Configure a MySQL Protection Group" on page 17
- "Validating a MySQL Protection Group" on page 19
- "Adding an Application Resource Group to a MySQL Protection Group" on page 21
- "Adding a Data-Replicated Component to a MySQL Protection Group" on page 23

▼ How to Create and Configure a MySQL Protection Group

Before You Begin Ensure that the following conditions are met:

- The local cluster is a member of a partnership.
- The protection group you are creating does not already exist, if you are going to create the MySQL configuration.
- The protection group exists if you want to do anything other than create the MySQL configuration.
- You must be assigned the Geo Management rights profile to complete this procedure. For more information, see "Securing Disaster Recovery Framework Software" in *Installing and Configuring the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

1. **Log in to a cluster node of the primary cluster.**

2. **Create the content for the `mysql_geo_config` file.**

Note - If you added readonly=true to the my.cnf file for the secondary cluster, you must also set READONLY=true in the mysql_geo_config file. Adding readonly=true to the my.cnf file is described in Step 2 of "How to Configure the MySQL Replication" on page 11.

3. **Copy the mysql_geo_config file to a different location.**

   For example:

```
phys-paris-1# cp /opt/SUNWscmys/geocontrol/util/mysql_geo_config /temp
```

4. **Specify the following variables in /temp/mysql_geo_config. This list uses sample values.**

```
PS=mysql-ps
PG=mysql-pg
REPCOMP=mysql.sbp
REPRS=mysql-rep-rs
REPRG=mysql-rep-rg
DESC="mysql replication pg"
CONFIGFILE=/geo-config/sbpconfig
REALMYSRG=paris-rg,newyork-rg
REALMYSRS=paris-mys-rs,newyork-mys-rs
READONLY=
APPRG=usa-rg
LONGPING=
SHORTPING=
```

5. **Create the script-based plug-in configuration file on all nodes of all clusters.**

   Assuming that the nodes of the primary cluster, cluster-paris, are phys-paris-1 and phys-paris-2, on each node of cluster-paris, you would issue the following commands:

```
phys-paris-1# mkdir /geo-config
phys-paris-1# echo "mysql.sbp|any|phys-paris-1,phys-paris-2" > /geo-config/sbpconfig
```

   Assuming that the nodes of the secondary cluster, cluster-newyork, are phys-newyork-1 and phys-newyork-2, on each node of cluster-newyork, you would issue the following commands:

```
phys-newyork-1# mkdir /geo-config
phys-newyork-1# echo "mysql.sbp|any|phys-newyork-1,phys-newyork-2" \
> /geo-config/sbpconfig
```

6. **Execute the mysql_geo_register script on the primary cluster.**

   For example:

```
phys-paris-1# ksh /opt/SUNWscmys/geocontrol/util/mysql_geo_register \
-f /temp/mysql_geo_config
```

7. **Replicate the protection group to the partner cluster.**

Note - You can also accomplish this step by using the Oracle Solaris Cluster Manager GUI. Click Partnerships, click the partnership name, highlight the protection group name, and click `Get Protection Groups`. For more information about Oracle Solaris Cluster Manager, see Chapter 12, "Using the Oracle Solaris Cluster Manager Browser Interface" in *Administering an Oracle Solaris Cluster 4.4 Configuration*.

The final messages of the registration script outline the required `geopg get` command. You must log in to one node of the partner cluster and execute that exact command.

```
phys-newyork-1# geopg get --partnership mysql-ps mysql-pg
```

Validating a MySQL Protection Group

When the configuration status of a protection group is displayed as Error in the `geoadm` status output, you can validate the configuration by using the `geopg validate`command. This command checks the current status of the protection group and its entities.

If the protection group and its entities are valid, then the configuration status of the protection groups is set to OK. If the `geopg validate`command finds an error in the configuration files, then the command displays an error message, and the configuration remains in the Error state. In such a case, you can fix the error in the configuration, and rerun the `geopg validate`command.

This section provides the following information:

- "Data Replication Layer Process for Validating the Application Resource Groups and Data Replication Entities" on page 19
- "How to Validate a MySQL Protection Group" on page 20

Data Replication Layer Process for Validating the Application Resource Groups and Data Replication Entities

During protection group validation, the MySQL data replication layer validates the application resource groups and the data replication entities by verifying that an application resource group in the protection group has its `Auto_start_on_new_cluster` property set to `false`.

When you bring a protection group online on the primary cluster, bring the application resources groups participating in that protection group online only on the same primary cluster. Setting the `Auto_start_on_new_cluster` property to `false` prevents the Oracle Solaris Cluster resource group manager from automatically starting the application resource groups. In this case, the startup of resource groups is reserved for the disaster recovery framework.

Application resource groups should be online only on the primary cluster when the protection group is activated.

The MySQL `geocontrol` module supplies a script that is used by the script-based plug-in module. The script entry points require the same set of arguments. These arguments are validated for semantics and completeness. The following validation checks are performed:

- Are all of the mandatory arguments defined?
- Is the configured MySQL database resource defined?
- Is the specified replication resource configured with a correct start command, if the resource exists already?
- Are the long and short `ping` intervals numeric?

When the validation is complete, the disaster recovery framework creates and brings online the replication resource group and its resources if they don't already exist. If a resource group or resource of the same name already exists, the disaster recovery framework might modify its properties. The framework cannot create a new resource group or a resource of the same name if one already exists. After creating the necessary resources, the framework adds the application resource group to the protection group.

▼ How to Validate a MySQL Protection Group

The `geopg validate` command validates the configuration of the protection group on the local cluster only. To validate the protection group configuration on the partner cluster, rerun the command on the partner cluster.

Note - You can also accomplish this procedure by using the Oracle Solaris Cluster Manager GUI. Click Partnerships, click the partnership name, highlight the protection group name, and click `Validate`. For more information about Oracle Solaris Cluster Manager, see Chapter 12, "Using the Oracle Solaris Cluster Manager Browser Interface" in *Administering an Oracle Solaris Cluster 4.4 Configuration*.

Before You Begin Ensure that the following conditions are met:

- The protection group you want to validate exists locally.
- The common agent container is online on all nodes of both clusters in the partnership.

- **Validate the configuration of the protection group on the local cluster only.**

```
# geopg validate protection-group
```

`protection-group`

The unique name of a single protection group.

Example 1 Validating the Configuration of a Protection Group

In the following example, the configuration of a protection group is validated:

```
# geopg validate mysql-pg
```

Adding an Application Resource Group to a MySQL Protection Group

To make an application highly available, the application must be managed as a resource in an application resource group.

The initial registration of the protection group is performed with the `mysql_geo_register` script. This section explains how to manage the application resource groups on their own.

All of the entities that you configure for the application resource group on the primary cluster, such as application data resources, application configuration files, and resource groups, must be replicated manually on the secondary cluster. The resource group names must be identical on both clusters. Also, the data that the application resource uses must be replicated on the secondary cluster.

▼ How to Add an Application Resource Group to a MySQL Protection Group

You can add an existing application resource group to the list of application resource groups for a protection group.

Note - You can also accomplish this procedure by using the Oracle Solaris Cluster Manager GUI. Click Partnerships, click the partnership name, click the protection group name, and in the Resource Groups section click `Add Resource Groups`. For more information about Oracle Solaris Cluster Manager, see Chapter 12, “Using the Oracle Solaris Cluster Manager Browser Interface” in *Administering an Oracle Solaris Cluster 4.4 Configuration*.

Before You Begin Before you add an application resource group to a protection group, ensure that the following conditions are met:

- The protection group is defined.
- The resource group to be added already exists on both clusters and is in an appropriate state.
- The `Auto_start_on_new_cluster` property of the resource group is set to `false`. You can view this property by using the `clresourcegroup` show command. For example:

  ```
  # clresourcegroup show -p Auto_start_on_new_cluster apprg1
  ```

  You can set the `Auto_start_on_new_cluster` property to `false` as follows:

  ```
  # clresourcegroup set -p Auto_start_on_new_cluster=false apprg1
  ```

  Setting the `Auto_start_on_new_cluster` property to `false` prevents the Oracle Solaris Cluster resource group manager from automatically starting the resource groups in the protection group. Once the disaster recovery framework restarts and communicates with the remote cluster to ensure that the remote cluster is running and that the remote cluster is the secondary cluster for that resource group, the framework will not automatically start the resource group on the primary cluster.

  Application resource groups should be online only on the primary cluster when the protection group is activated.
- The `Nodelist` property of the failover application resource group that has affinities with a replicated component defined by the resource must contain the same entries in identical order as the `Nodelist` property of the protection group.
- The application resource group must not have dependencies on resource groups and resources outside of this protection group. To add several application resource groups that share dependencies, you must add all the application resource groups that share dependencies to the protection group in a single operation. If you add the application resource groups separately, the operation will fail.

  The protection group can be activated or deactivated, and the resource group can be either online or unmanaged. If the resource group is unmanaged and the protection group is activated after the configuration of the protection group has changed, then the local state of the protection group becomes Error. If the resource group to add is online, and the protection group is deactivated, the request is rejected. You must activate the protection group before adding an online resource group.
- You must be assigned the Geo Management rights profile to complete this procedure. For more information, see "Securing Disaster Recovery Framework Software" in *Installing and Configuring the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

1. **Log in to a cluster node.**

2. **Add an application resource group to the protection group.**

```
# geopg add-resource-group resource-group-list protection-group
```

In this syntax, *resource-group-list* specifies the name of the application resource group. You can specify more than one resource group in a comma-separated list. Also, *protection-group* specifies the name of the protection group.

This command adds an application resource group to a protection group on the local cluster. Then, the command propagates the new configuration information to the partner cluster if the partner cluster contains a protection group with the same name.

For information about the names and values that are supported by the disaster recovery framework, see Appendix B, "Legal Names and Values of Disaster Recovery Framework Entities," in *Administering the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

If the add operation is unsuccessful on the local cluster, the configuration of the protection group is not modified. Otherwise, the configuration status is set to OK on the local cluster. If the configuration status is OK on the local cluster, but the add operation is unsuccessful on the partner cluster, the configuration status is set to Error on the partner cluster.

After the application resource group is added to the protection group, the application resource group is managed as an entity of the protection group. Then, the application resource group is affected by protection group operations such as start, stop, switchover, and takeover.

Example 2 Adding an Application Resource Group to a MySQL Protection Group

In the following example, the application resource groups `apprg1` and `apprg2`, are added to the `mysql-pg` protection group.

```
# geopg add-resource-group apprg1,apprg2 mysql-pg
```

Adding a Data-Replicated Component to a MySQL Protection Group

A protection group is the container for the application resource groups, which contain data for services that are protected from disaster. The disaster recovery framework protects the data by replicating it from the primary cluster to the secondary cluster. By adding a data-replicated component to a protection group, the framework monitors the replication status of a MySQL database. The framework also controls the role and state of the database during protection group operations such as start, stop, switchover, and takeover.

This section provides the following information:

- "How to Add a Data-Replicated Component to a MySQL Protection Group" on page 24

- "Data Replication Subsystem Process for Verifying the Replicated Component" on page 25

How to Add a Data-Replicated Component to a MySQL Protection Group

---

Note - You can also accomplish this procedure by using the Oracle Solaris Cluster Manager GUI. Click Partnerships, click the partnership name, click the protection group name, and in the Data Replication Components section click `Add`. For more information about Oracle Solaris Cluster Manager, see Chapter 12, "Using the Oracle Solaris Cluster Manager Browser Interface" in *Administering an Oracle Solaris Cluster 4.4 Configuration*.

Before You Begin Ensure that the following conditions are met:

- The protection group is defined on the local cluster.
- The protection group is offline on the local cluster and the partner cluster, if the partner cluster can be reached.
- The underlying MySQL database resources exist on both the local cluster and the partner cluster.
- You must be assigned the Geo Management rights profile to complete this procedure. For more information, see "Securing Disaster Recovery Framework Software" in *Installing and Configuring the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

1. **Log in to a cluster node.**

2. **Add a data-replicated component to the protection group.**

   The following command adds a replication component to a protection group on the local cluster and propagates the new configuration to the partner cluster if the partner cluster contains a protection group with the same name.

   ```
   # geopg add-replication-component -p property [-p…] mysql-replicated-component protection-group
   ```

   In this syntax, `-p` *property* specifies the properties of the data-replicated component group. You can specify the following script-based plug-in properties:

   - `switchover_args` – Specifies the command-line arguments for the switchover script
   - `takeover_args` – Specifies the command-line arguments for the takeover script
   - `start_replication_args` – Specifies the command-line arguments for the `start_replication` script

- `remove_config_args` – Specifies the command-line arguments for the `remove_configuration` script
- `create_config_args` – Specifies the command-line arguments for the `create_configuration` script
- `stop_replication_args` – Specifies the command-line arguments for the `stop_replication` script

Caution - Make sure that the command-line arguments are the same for all scripts.

For more information about the properties you can set, see "Property Descriptions for Script-Based Plug-Ins" in *Administering the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

Also in this syntax, *mysql-replicated-component* specifies the name of the new data-replicated component, and *protection-group* specifies the name of the protection group that will contain the new data-replicated component.

For information about the names and values that are supported, see Appendix B, "Legal Names and Values of Disaster Recovery Framework Entities," in *Administering the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

For more information about the `geopg` command, refer to the `geopg`(8) man page.

Note - Because the add operation for the replication component is performed during the scripted registration, an example is not provided here.

Data Replication Subsystem Process for Verifying the Replicated Component

During protection group validation, the MySQL data replication layer validates the application resource groups and the data replication entities by verifying that an application resource group in the protection group has its `Auto_start_on_new_cluster` property set to `false`.

When you bring a protection group online on the primary cluster, bring the application resources groups participating in that protection group online only on the same primary cluster. Setting the `Auto_start_on_new_cluster` property to `false` prevents the Oracle Solaris Cluster resource group manager from automatically starting the application resource groups. In this case, the startup of resource groups is reserved for the disaster recovery framework.

Application resource groups should be online only on the primary cluster when the protection group is activated.

The `Mysql geocontrol` module supplies a script that is used by the script-based plug-in module. The script entry points require the same set of arguments. These arguments are validated for semantics and completeness. The following validation checks are performed:

- Are all of the mandatory arguments defined?
- Is the configured MySQL database resource defined?
- Is the specified replication resource configured with a correct start command, if the resource exists already?
- Are the long and short `ping` intervals numeric?

When the validation is complete, the disaster recovery framework adds the application resource group to the protection group.

Note - Every entry point of the underlying script-based plug-in has a validation method. In the case of the MySQL replication, all the validation methods are the same.

Administering MySQL Application Resource Groups

This section describes how to delete an application resource group from a MySQL protection group.

▼ How to Delete an Application Resource Group From a MySQL Protection Group

You can remove an application resource group from a protection group without altering the state or contents of the application resource group.

Note - You can also accomplish this procedure by using the Oracle Solaris Cluster Manager GUI. Click Partnerships, click the partnership name, click the protection group name, highlight the resource group name, and click `Remove`. For more information about Oracle Solaris Cluster Manager, see Chapter 12, "Using the Oracle Solaris Cluster Manager Browser Interface" in *Administering an Oracle Solaris Cluster 4.4 Configuration*.

Before You Begin Ensure that the following conditions are met:

- The protection group is defined on the local cluster.
- The resource group to be removed is part of the application resource groups of the protection group.

- You must be assigned the Geo Management rights profile to complete this procedure. For more information, see "Securing Disaster Recovery Framework Software" in *Installing and Configuring the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

1. **Log in to a cluster node.**

2. **Remove the application resource group from the protection group.**

   The following command removes an application resource group from a protection group on the local cluster. If the partner cluster contains a protection group with the same name, the application resource group is also removed from the protection group on the partner cluster.

   ```
   # geopg remove-resource-group resource-group-list protection-group
   ```

   In this syntax, *resource-group-list* specifies the name of the application resource group. You can specify more than one resource group in a comma-separated list. In addition, *protection-group* specifies the name of the protection group.

   If the resource group that is being removed shares dependencies with other resource groups in the protection group, then you must also remove all other resource groups that share dependencies with the resource group that is being removed.

   If the remove operation fails on the local cluster, the configuration of the protection group is not modified. Otherwise, the configuration status is set to OK on the local cluster. If the configuration status is OK on the local cluster but the remove operation is unsuccessful on the partner cluster, the configuration status is set to Error on the partner cluster.

Example 3 Deleting an Application Resource Group From a Protection Group

In the following example, two application resource groups, `apprg1` and `apprg2`, are removed from the `mysql-pg` protection group.

```
# geopg remove-resource-group apprg1,apprg2 mysql-pg
```

Administering MySQL Data-Replicated Components

This section describes the following tasks for administering data-replicated components in a MySQL protection group:

- "How to Modify a MySQL Data-Replicated Component" on page 28
- "How to Delete a Data-Replicated Component From a MySQL Protection Group" on page 28

For details about configuring a MySQL protection group, see "How to Create and Configure a MySQL Protection Group" on page 17.

▼ How to Modify a MySQL Data-Replicated Component

Note - You can also accomplish this procedure by using the Oracle Solaris Cluster Manager GUI. Click Partnerships, click the partnership name, click the protection group name, click the data replication component name, and click `Edit`. For more information about Oracle Solaris Cluster Manager, see Chapter 12, "Using the Oracle Solaris Cluster Manager Browser Interface" in *Administering an Oracle Solaris Cluster 4.4 Configuration*.

Before You Begin You must be assigned the Geo Management rights profile to complete this procedure. For more information, see "Securing Disaster Recovery Framework Software" in *Installing and Configuring the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

1. **Log in to a cluster node.**

2. **Modify the replication component.**

   The following command modifies the properties of a device group in a protection group on the local cluster. Then, the command propagates the new configuration to the partner cluster if the partner cluster contains a protection group with the same name.

   ```
   # geopg modify-replication-component -p property [-p…] mysql-replicated-component protection-group
   ```

   In this syntax, the `-p` *property* option specifies the properties of the data-replicated component.

   For more information about the properties you can set, see "Property Descriptions for Script-Based Plug-Ins" in *Administering the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

   Also in this syntax, *mysql-replicated-component* specifies the name of the data-replicated component, and *protection-group* specifies the name of the protection group that will contain the new data-replicated component.

▼ How to Delete a Data-Replicated Component From a MySQL Protection Group

You might need to delete a data-replicated component from a protection group if you previously added a data-replicated component to that protection group. Normally, after an application is configured to connect to the database, you would not change the database.

Note - You can also accomplish this procedure by using the Oracle Solaris Cluster Manager GUI. Click Partnerships, click the partnership name, click the protection group name, highlight the data replication component name, and click `Remove`. For more information about Oracle Solaris Cluster Manager, see Chapter 12, "Using the Oracle Solaris Cluster Manager Browser Interface" in *Administering an Oracle Solaris Cluster 4.4 Configuration*.

Before You Begin Before you delete a data-replicated component, ensure that the following conditions are met:

- The protection group is defined on the local cluster.
- The protection group is offline on the local cluster and the partner cluster, if the partner cluster can be reached.
- The device group is managed by the protection group.
- You must be assigned the Geo Management rights profile to complete this procedure. For more information, see "Securing Disaster Recovery Framework Software" in *Installing and Configuring the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

For information about deleting protection groups, refer to "Deleting Protection Groups and Data Replication Components" in *Administering the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

1. **Log in to a cluster node.**

2. **Remove the replicated component.**

```
# geopg remove-replication-component mysql-replicated-component protection-group
```

In this syntax, *mysql-replicated-component* specifies the name of the data-replicated component, and *protection-group* specifies the name of the protection group.

Example 4 Deleting a Replicated Component From a MySQL Protection Group

In the following example, the `mysql-dg` data-replicated device group is deleted from the `mysql-pg` protection group.

```
# geopg remove-replication-component mysql.sbp mysql-pg
```

Replicating a MySQL Protection Group Configuration to a Partner Cluster

This section contains information about replicating the MySQL protection group.

How to Replicate a MySQL Protection Group to a Partner Cluster

Before You Begin Ensure that the following conditions are met:

- The protection group is defined on the remote cluster, not on the local cluster.
- The MySQL database resources in the protection group on the remote cluster exist on the local cluster.
- The application resource groups in the protection group on the remote cluster exist on the local cluster.
- Application resource groups are online only on the primary cluster when the protection group is activated.
- The `Auto_start_on_new_cluster` property of the resource groups is set to `false`.
  - To view the `Auto_start_on_new_cluster` property, use following command:

    ```
    # clresourcegroup show -p auto_start_on_new_cluster apprg1
    ```
  - If you need to set the `Auto_start_on_new_cluster` property to `false`, use the following command:

    ```
    # clresourcegroup set -p Auto_start_on_new_cluster=false apprg1
    ```

  Setting this property to `false` prevents the Oracle Solaris Cluster Resource Group Manager from automatically starting the resource groups in the protection group. Once the disaster recovery framework restarts and communicates with the remote cluster to ensure that the remote cluster is running and that the remote cluster is the secondary cluster for the resource group, the framework does not automatically start the resource group on the primary cluster.

- **Replicate the protection group configuration to the partner cluster.**

```
# geopg get -s partnership-name mysql-pg
```

The command retrieves the configuration information of the protection group from the remote cluster and creates the protection group on the local cluster.

In this syntax, the `-s` *partnership-name* option specifies the name of the partnership from which the protection group configuration information is retrieved. In addition, *mysql-pg* specifies the name of the protection group.

Note - Resynchronizing a protection group updates only entities that are related to the the disaster recovery framework. To update Oracle Solaris Cluster resource groups, resource types, and resources, use the `cluster export -t rg,rt,rs` command to generate an XML cluster configuration file, modify the XML file for the expected configuration on the secondary cluster, and run the `clresource create` command with the `-a` option to apply the configuration updates.

For more information, see "How to Configure Oracle Solaris Cluster Software on All Nodes (XML)" in *Installing and Configuring an Oracle Solaris Cluster 4.4 Environment* and the `cluster`(8CL)and `clresource`(8CL) man pages.

Example 5 Replicating a MySQL Protection Group to a Partner Cluster

In the following example, `paris-newyork-ps` is a partnership between the `cluster-paris`and `cluster-newyork` clusters. The `mysql-pg` protection group is replicated to the `cluster-newyork` cluster. The configuration of the protection group is retrieved from the remote cluster `cluster-paris` and then validated by the data replication subsystem on the local cluster `cluster-newyork`.

```
#phys-newyork-1# geopg get -s paris-newyork-ps mysql-pg
```

If the validation is successful, the configuration status is set to OK, and the protection group is created on the local cluster. This protection group contains a replicated component and an application group that are configured almost identically to the replicated component and application group on the remote cluster. If the validation fails, the protection group is not created on the local cluster. Fix the cause of the error, and replicate it again.

See Also To perform switchover and takeover of protection groups, see Chapter 11, "Migrating Services" in *Administering the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

To perform recovery after takeover of a MySQL protection group, see "Recovery Strategy After a Takeover of a MySQL Protection Group" on page 32.

Debugging the MySQL Replication Configuration

This section describes how to set different debugging methods for the MySQL replication configuration.

- Debugging the replication status resource by obtaining extensive logging and a Korn shell trace.

In the file /opt/SUNWscmys/geocontrol/etc/config, set the variable SET_DEBUG to ALL or set it to the replication component that you are debugging on the node that you want to debug. On this node, call the probe script that you want to debug.

```
ksh -x /opt/SUNWscmys/geocontrol/bin/mysql_geo_replication
-R rep-rs -G rep-rg -r mysql-rs -g mysql-rg probe
```

- Debugging the script-based plug-ins control scripts by obtaining extensive logging and a Korn shell trace for a script.
  - In the file /opt/SUNWscmys/geocontrol/etc/config, set the variable SET_DEBUG to ALL or set it to the replication component that you are debugging on the node that you want to debug. On this node, call the SBP control script that you want to debug.

    ```
    ksh -x /opt/SUNWscmys/geocontrol/bin/mysql_geo_control
    -O mysql-comp -r rep-rs -g rep-rg -R mysql-rs -G mysql-rg -t 10 -T 60
    -o paris:mysqlex1,paris:mysqlre2,newyork:mysqlex2,newyork:mysqlre1
    function=create_configuration pg=mysql-pg validate_parameters=true
    currentRole=PRIMARY is Modify=false
    ```

  - To obtain all the SBP controls and its parameters, scan the common agent container log.

    ```
    /var/cacao/instances/default/logs
    ```

Recovery Strategy After a Takeover of a MySQL Protection Group

When an old primary cluster is restarting for the first time after a successful takeover, the MySQL database does not detect that the cluster should no longer act as a master and the disaster recovery framework still keeps the primary role, but leaves it deactivated. The goal for the recovery is to configure the old master to run as a slave and to update the disaster recovery framework configuration to reflect this role change.

You can check for the status with the following command:

```
# geoadm status
```

The recovery strategy after a takeover involves the following actions:

1. Configuring the old master to run as a slave
2. Manually starting the slave threads on the old master
3. Resynchronizing the protection group to switch the role

▼ How to Recover After a Takeover

Note - You can also accomplish Step 8 and Step 9 by using the Oracle Solaris Cluster Manager browser interface. Click Partnerships, click the partnership name, and highlight the protection group name. Click Update Protection Group and when the update is completed click Start Protection Group. For more information about Oracle Solaris Cluster Manager, see Chapter 12, "Using the Oracle Solaris Cluster Manager Browser Interface" in *Administering an Oracle Solaris Cluster 4.4 Configuration*.

1. **Log in to a node of the old primary cluster.**

   You must be assigned the Geo Management rights profile to complete this procedure. For more information, see "Securing Disaster Recovery Framework Software" in *Installing and Configuring the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

2. **Allow the MySQL slave threads to be started if the database resource performs a restart or similar action.**

   a. **Remove the `skip-slave-start` keyword from the appropriate `my.cnf` file.**

   b. **If, when the protection group was created, the `mysql_geo_config` registration file contained `READONLY=true`, remove the `read-only=true` entry from the appropriate `my.cnf` file.**

3. **Log in to MySQL as the `root` role, then start the slave.**

```
mysql> start slave;
```

4. **Verify that the slave is running, and wait until it is synchronized with the master.**

```
mysql> show slave status\G
```

   If the slave status shows that at least one slave thread is not running, fix the root cause, and retry the operation. As a last resort, you could take a backup from the current master and perform a fresh slave setup.

5. **From a node of the old primary cluster, update the protection group to change the role from a deactivated primary cluster to a secondary cluster.**

```
# geopg update protection-group
```

6. **Prepare the protection group on the original primary cluster for recovery.**

   a. **View the status of the protection group.**

```
phys-paris-1# geopg status protection-group
```

b. **If the protection group is still active, stop it.**

```
phys-paris-1# geopg stop -e local protection-group
```

c. **If the protection group in in an Error state, validate it.**

```
phys-paris-1# geopg validate protection-group
```

7. **Ensure that the protection group is valid.**

8. **Resynchronize the protection group.**

```
# geopg update protection-group
```

9. **Start the protection group locally.**

```
# geopg start -e local protection-group
```

For more information, see "Resynchronizing a Protection Group" in *Administering the Disaster Recovery Framework for Oracle Solaris Cluster 4.4* and "Activating and Deactivating a Protection Group" in *Administering the Disaster Recovery Framework for Oracle Solaris Cluster 4.4*.

Index