Cisco Prime Network Registrar IPAM 8.1.2 MySQL Database Replication Guide

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
        800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL- 30182-01

Contents

1 MySQL Database Replication Overview

The *MySQL Database Replication Guide* is designed to help you configure and maintain MySQL Replication on the Cisco Prime Network Registrar IPAM 8.1.2 Executive and Backup Executive. It is organized as follows:

MySQL Database Replication Overview	Describes the components in the replication system architecture and their respective roles.
Setup	Describes the steps necessary to configure the replication system.
Operations	Describes the procedures you use to work with the replication system:
	• Demote the master system
	• Promote the slave system
	• Resynchronize a slave system
Troubleshooting with the Replication Monitor	Describes the Replication Monitor and details the steps you take to resolve synchronization issues.

1.1 Database Replication Architecture

The Cisco Prime Network Registrar IPAM provides several scripts that allow Cisco Prime Network Registrar IPAM users who have a MySQL database server to take advantage of MySQL's built-in replication capability. Replication permits near real-time duplication of the primary Cisco Prime Network Registrar IPAM Executive database (the master) on a Backup Executive system (the slave).

MySQL accomplishes this by storing every data modification statement performed on the master in a binary log file. The slave receives a copy of the contents of this log file and executes the events that it contains on the local database. To ensure that the master and slave systems remain synchronized with one another, the slave keeps a record of the binary log coordinates – the file name and position in the file that it has read and processed from the master. The Cisco Prime Network Registrar IPAM has provided the scripts described in Table 1 to help create the log file and establish the coordinates, and then manage the replication process. These scripts are described in greater detail in Chapter 2, "Setup" on page 5.

Table 1 Cisco Prime Network Registrar IPAM Replication Scripts

Script	Description
prepmaster.sh	Defines the Active Master on the Production Executive
prepslave.sh	Defines the Passive Master on the Backup Executive
connect_to_passive.sh	Connects the Active Master to the Passive Master
demote_master.sh	Demotes the Active Master to become the Passive Master
promote_master.sh	Promotes the Passive Master to become the Active Master
resync_master.sh	Dumps the Active Master and calculates log coordinates
resync_slave.sh	Loads the Active Master data and sets the log coordinates on the Passive Master

1.1.1   Supported Operating Systems

MySQL Replication is supported on the following systems:

- Solaris 10

    Note: Refer to "Timing Issue" on page 21 for information on resolving an intermittent issue with scripts that involve shutting down the MySQL database.

- Red Hat Linux 5 and 6

1.1.2   Active-Passive Master Setup

The Cisco Prime Network Registrar IPAM has implemented a variant of the standard Master/Slave replication setup. The Cisco Prime Network Registrar IPAM Production Executive system serves as the replication master and is configured as an "Active" Master. The Backup Executive system serves as the replication slave and is configured as a read-only "Passive" Master. The Active-Passive Master setup means that both servers are configured as masters and slaves of each other. When the roles need to be reversed, it is a simple task of toggling the read-only flag on each server. Consequently, this configuration permits an easier transition of roles whenever that becomes necessary. The Active-Passive Master setup is illustrated in Figure 1.

Figure 1 Cisco Prime Network Registrar IPAM Implementation of MySQL Database Replication

1.1.3 Replication Monitor

The Cisco Prime Network Registrar IPAM provides the MySQL Replication Monitor daemon to help resolve a MySQL replication problem where the slave cannot always keep up with the pace of changes being performed on the master. As a result, the slave no longer represents a true replication of the master.

The Replication Monitor daemon makes periodic updates to a table on the master server, and then queries the slave to determine if the update was replicated. In this way, it can determine not only if the communication between the two servers is active but also how far the slave may be behind the master.

The Replication Monitor runs on the Cisco Prime Network Registrar IPAM Executive, or where the Active Master is running. The Replication Monitor is described in greater detail in Chapter 4, "Troubleshooting" on page 21.

1.1.4 Callout Manager

The Callout Manager is a facility within Cisco Prime Network Registrar IPAM that notifies other applications of alerts and programmatic events. The callouts for Replication Monitor are described in Table 3 on page 22.

For more detailed information on all aspects of the Callout Manager, refer to "Other Interfaces", in the *Cisco Prime Network Registrar IPAM 8.1.2 Command Line Interface (CLI) and Application Program Interface (API) Guide*.

2 Setup

2.1 Replication Setup

You set up MySQL replication on the Production and Backup Executives as follows.

1. Configure the Production Executive to begin logging any changes that are made to the database. This step is handled by the following script:

   INCHOME/etc/support/prepmaster.sh

   For further information on executing this script, refer to "Configure Production Executive as Active Master," on page 7.

2. Configure the Backup Executive to detect changes on the Production Executive and to become a read-only server. This step is handled by the following script:

   $INCHOME/etc/support/prepslave.sh

   For further information on executing this script, refer to "Configure Backup Executive as Passive Master," on page 8.

3. Complete the loop by setting up the Production Executive to listen for changes on the Backup Executive and begin monitoring the replication state. This step is handled by the following script:

   $INCHOME/etc/support/connect_to_passive.sh

   For further information on executing this script, refer to "Connect Active Master to Passive Master," on page 11.

2.2 Replication Software Prerequisites

Before you begin setting up MySQL replication, ensure you have completed the following prerequisites:

- The Cisco Prime Network Registrar IPAM Executive 8.1.2 or higher is installed on both the Production and Backup servers.

- MySQL 5.5 is installed on both the Production and Backup servers.

- A new license key has been obtained from Cisco Prime Network Registrar IPAM TAC to support both the Production and Backup Executives.

Enter the new license key under "System - System Policies/Options" before you begin replication installation and configuration. Ensure there are no spaces before or after the key when you enter it into the **License Key** field.

2.3 Configure Production Executive as Active Master

To configure the Production Executive as Active Master, execute the `prepmaster.sh` script on the Production Executive. When completed, the script displays instructions on the next steps you need to take to set up replication.

2.3.1   Executing the prepmaster.sh Script

To execute the `prepmaster.sh` script, follow these steps.

1   As **incadmin** user, change directory to /opt/incontrol/etc/support.

>**cd /opt/incontrol/etc/support**

2   Execute the script command.

>**./prepmaster.sh**

The script begins by granting access to the Backup Executive replication account and updating the configuration file, `my.cnf`, to support replication.

```
Creating replication monitor database with: CREATE DATABASE IF NOT EXISTS REPLMON
Configuring Active Master settings ...

Backed up /opt/incontrol/mysql/my.cnf to
/opt/incontrol/mysql/my.cnf.pre_replication
Configuration file updated: /opt/incontrol/mysql/my.cnf
```

After the appropriate changes have been made to the configuration file, the master is restarted.

```
Stopping MySQL Database Server...
MySQL Database Server stopped.
Waiting 5 seconds before starting...

Starting MySQL Database Server...
MySQL Database Server started.
Waiting 10 seconds for mysql to start ...
Stopping replication slave process with: STOP SLAVE
Disconnecting from old master with: CHANGE MASTER TO MASTER_HOST=''
Resetting slave with: RESET SLAVE
```

The script then uses `mysqldump` to create a dump file in the current directory. This defines the starting point for replication.

```
Dumping the master database and getting Log File Coordinates ...
Master Log File Coordinates:
MASTER_LOG_FILE='mysql-bin.000001'
MASTER_LOG_POS=2645

Dump File: dbforrepl.sql
```

Finally, the script displays the steps necessary to configure the slave (or Passive Master), including the correct parameters to pass to the `prepslave.sh` script on the Backup Executive.

```
Replication Setup Step 1 of 3 complete!

- Copy the Dumpfile: dbforrepl.sql to the slave server.
- Record the Active Master Log File Coordinates and log in to the passive master
system.
- Navigate to /opt/incontrol/etc/support.
- Once there, run the prepslave.sh script using the coordinates as follows:

./prepslave.sh -l mysql-bin.000001 -s 2645 -h 192.168.196.45 -d <location of
dbforrepl.sql>

Once the slave is configured, return to this system and run the connect_to_passive.sh
script to complete the setup.
```

3 Make a note of the following values from the screen output, which you need when executing the `prepslave.sh` script.

❑ The name of the MySQL log file, displayed in `MASTER_LOG_FILE=`

❑ The replication start coordinates of the log file, displayed in `MASTER_LOG_POS=`

❑ The name of the MySQL dump file

2.4 Configure Backup Executive as Passive Master

To configure the Backup Executive as Passive Master, copy the Production Executive dump file to the Backup Executive, and then execute the `prepslave.sh` script. When completed, the script displays instructions on the next steps you need to take to set up replication.

`prepslave.sh` accepts the following parameters:

Parameter	Required	Description
-d <dump_filename>	Yes	Defines the name of the MySQL database dump file.
-h <host_IP_address>	Yes	Defines the IP address of the current host or Master server.
-l <log_filename>	Yes	Defines the name of the MySQL database log file, as defined in the `MASTER_LOG_FILE=` statement.
-s <log_coordinate>	Yes	Defines the starting point coordinate in the log file where replication begins, as defined in the `MASTER_LOG_POS=` statement.

2.4.1 Copying the Production Executive Dump File

Before you execute the `prepslave.sh` script, copy the Production Executive dump file that was created when you executed the `prepmaster.sh` script to the Backup Executive server.

To copy the MySQL database dump file from the Production Executive, follow these steps.

1 Log into the Backup Executive server as **incadmin** user and change directory
to /opt/incontrol/etc/support.

>**cd /opt/incontrol/etc/support**

2 Execute the secure copy command, specifying the full pathname of the Production
Executive dump file.

> **scp incadmin@<host_IP_address>:/opt/incontrol/etc/support/<dump_filename> .**

For example:

```
>scp incadmin@192.168.196.45:/opt/incontrol/etc/support/dbforrepl.sql .
incadmin@192.168.196.45's password:
dbforrepl.sql
100%  390KB 390.1KB/s   00:00
```

2.4.2   Executing the prepslave.sh Script

To execute the prepslave.sh script, follow these steps.

1 As **incadmin** user, change directory to /opt/incontrol/etc/support.

>**cd /opt/incontrol/etc/support**

2 Execute the script command.

>**./prepslave.sh -l** <log_filename> **-s** <log_coordinate>
-h <host_IP_address> **-d** <dump_filename>**.sql**

For example:

```
>./prepslave.sh -l mysql-bin.000001 -s 2645 -h 192.168.196.45 -d dbforrepl.sql
```

The script begins by loading the dumpfile and updating the configuration file, my.cnf, to
support replication as the Passive Master.

```
Loading dumpfile: dbforrepl.sql ...


Creating replication user with: GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.*
TO repl@'192.168.196.45' IDENTIFIED BY 'fake_password'
Granting replication user rights to replication monitor db (any host) with: GRANT
SELECT ON REPLMON.* TO repl@'192.168.196.45' IDENTIFIED BY 'fake_password'


Configuring Slave as a Passive Master ...
Backed up /opt/incontrol/mysql/my.cnf to
/opt/incontrol/mysql/my.cnf.pre_replication
Configuration file updated: /opt/incontrol/mysql/my.cnf
```

After restarting the server, the script loads the starting point for the database from the
master.

```
Restarting MySQL daemon ...


Stopping MySQL Database Server...
MySQL Database Server stopped.
Waiting 5 seconds before starting...
```

```
Starting MySQL Database Server...
MySQL Database Server started.
Waiting 10 seconds for mysql to start ...

Connecting to master with: CHANGE MASTER TO
MASTER_HOST='192.168.196.45',MASTER_USER='repl',MASTER_PASSWORD='fake_password',
MASTER_LOG_FILE='mysql-bin.000001',MASTER_LOG_POS=2645

Starting slave process with: START SLAVE

Passive Master Log File Coordinates:
LOG_FILENAME='mysql-bin.000001'
LOG_START_POS=2645
```

Finally, the script displays the steps necessary to connect the Active Master to the Passive
Master, including the correct parameters to pass to the connect_to_passive.sh
script on the Production Executive.

```
Replication Setup Step 2 of 3 complete!

- Record the Passive Master Log File Coordinates and return to the master system.
- Navigate to /opt/incontrol/etc/support.
- Once there, run the connect_to_passive.sh script using the coordinates as
follows:

./connect_to_passive.sh -l mysql-bin.000001 -s 2645 -h 192.168.196.46
```

3 Make a note of the following values from the screen output, which you need when
executing the connect_to_passive.sh script.

❑ The name of the MySQL log file, displayed in LOG_FILENAME=

❑ The replication start coordinates of the log file, displayed in LOG_START_POS=

2.5 Connect Active Master to Passive Master

To connect the Active Master to the Passive Master, execute
the `connect_to_passive.sh` script on the Production Executive. When completed, the
script defines the Active Master as a slave to the Passive Master. Additionally, it configures
the appropriate settings to run the MySQL Replication Monitor utility and starts this utility.

`connect_to_passive.sh` accepts the following parameters:

Parameter	Required	Description
-h `<passive_master_IP_address>`	Yes	Defines the IP address of the Passive Master server.
-l `<log_filename>`	Yes	Defines the name of the MySQL database log file, as defined in the `LOG_FILENAME=` statement.
-s `<log_coordinate>`	Yes	Defines the starting point coordinates in the log file where replication begins, as defined in the `LOG_START_POS=` statement.

2.5.1 Connecting the Active Master to the Passive Master

To execute the `connect_to_passive.sh` script, follow these steps.

1 As **incadmin** user, change directory to /opt/incontrol/etc/support.

>**cd /opt/incontrol/etc/support**

2 Execute the script command.

>**./connect_to_passive.sh -l** `<log_filename>` **-s** `<log_coordinate>`
-h `<passive_master_IP_address>`

For example:

```
>./connect_to_passive.sh -l mysql-bin.000001 -s 2645 -h 192.168.196.46
```

The script begins by connecting the Active Master to the Passive Master.

```
Creating replication user with: GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.*
TO repl@'192.168.196.46' IDENTIFIED BY 'fake_password'
Granting replication user rights to replication monitor db (passive master host)
with: GRANT SELECT ON replmon.* TO repl@'192.168.196.46' IDENTIFIED BY
'fake_password'
Connecting to passive master with: CHANGE MASTER TO
MASTER_HOST='192.168.196.46',MASTER_USER='repl',MASTER_PASSWORD='fake_password',
MASTER_LOG_FILE='mysql-bin.000001',MASTER_LOG_POS=2645

Starting slave process with: START SLAVE
```

After starting the slave process, the script modifies the Replication Monitor properties file
and starts the utility.

```
Modifying IP Addresses in /opt/incontrol/mysql_replication_monitor.properties
```

```
Turning on control of the replication monitor.
IPControl MySQL Replication Monitor install successful.
Starting the replication monitor ...
IPControl MySQL Replication Monitor is already stopped.
Waiting 5 seconds before starting...


Starting IPControl MySQL Replication Monitor...
IPControl MySQL Replication Monitor started.


Replication Setup Step 3 of 3 complete!
```

2.6 Replication Setup Post Configuration Steps

To set up the Production and Backup servers so that demoting and promoting of Master and Slave can occur, execute the following commands on both the Production and Backup servers.

```
# cd $INCHOME/etc
# cp default.incontrol default.incontrol.primary
# cp incstatus incstatus.primary
```

3 Operations

3.1 Operations Overview

This chapter describes the following procedures you use to work with the replication system:

- Demoting the master system

- Promoting the slave system

- Resynchronizing a slave system

3.2 Demoting a Master and Promoting a Slave

This section describes the steps you take when you need to bring down the Master MySQL server running on the Production Executive for some reason, and then make the Slave MySQL database assume the role of Active Master.

3.2.1 Demoting the Active Master

To demote the Active Master on the Production Executive to become a slave, you execute the `demote_master.sh` script.

`demote_master.sh` accepts the following parameters:

Parameter	Required	Description
-l <log_filename>	No	Defines the name of the MySQL database log file.

To execute the `demote_master.sh` script, follow these steps.

1 As **incadmin** user on the Production Executive, change directory to `/opt/incontrol/etc/support`.

>   **>cd /opt/incontrol/etc/support**

2 Execute the script command.

>   **>./demote_master.sh**

The script begins by stopping the Replication Monitor, changing the configuration file, and adding a flag that makes the MySQL server read-only.

```
Stopping the MySQL Replication Monitor

Stopping IPControl MySQL Replication Monitor...
```

```
IPControl MySQL Replication Monitor stopped.
IPControl MySQL Replication Monitor remove successful.
Adding the read_only option to configuration file ...
Turning on read_only flag with: set @@global.read_only := 1
```

The script continues by calculating the log file coordinates.

```
Master Log File Coordinates:
MASTER_LOG_FILE='mysql-bin.000001'
MASTER_LOG_POS=87592
```

Finally, the script displays the steps necessary to promote an existing slave (Passive Master), including the correct parameters to pass to the `promote_master.sh` script on the Backup Executive.

```
Promotion / Demotion Step 1 of 2 complete!
If you plan to promote an existing slave (Passive Master) follow these steps:

- Record the Active Master Log File Coordinates and log in to the existing Passive
Master system.
- Navigate to /opt/incontrol/etc/support.
- Once there, run the promote_master.sh script using the coordinates as follows:
- ./promote_master.sh -l mysql-bin.000001 -s 87592
```

3   Make a note of the following values from the screen output, which you need when executing the `promote_master.sh` script on the slave server.

❑ The name of the MySQL log file, displayed in `MASTER_LOG_FILE=`

❑ The replication start coordinates of the log file, displayed in `MASTER_LOG_POS=`

4   Execute the following commands.

```
# $INCHOME/etc/incontrol stop
# $INCHOME/etc/support/DemoteExec.sh
# $INCHOME/etc/incontrol start
```

3.2.2   Promoting the Slave

For the current slave (Passive Master) to assume the role of master after the current Active Master has been demoted, the `promote_master.sh` script must be run on the slave system, using the coordinates provided by the `demote_master.sh` script executed on the Production Executive (or Active Master).

`promote_master.sh` accepts the following parameters:

Parameter	Required	Description
-l <log_filename>	Yes	Defines the name of the MySQL database log file, as defined in the MASTER_LOG_FILE= statement.
-s <log_coordinate>	Yes	Defines the starting point coordinate in the log file where replication begins, as defined in the MASTER_LOG_POS= statement.

To execute the `promote_master.sh` script, follow these steps.

1 As **incadmin** user on the Backup Executive, change directory to `/opt/incontrol/etc/support`.

>**cd /opt/incontrol/etc/support**

2 Execute the script command.

>**./promote_master.sh -l** <log_filename> **-s** <log_coordinate>

For example:

```
>./promote_master.sh -l mysql-bin.000001 -s 87592
```

The script begins by allowing the Passive Master to mirror the content of the Active Master, as determined by the log coordinate parameter you just entered and then makes the database active by removing the read-only flag.

```
Waiting for Passive Master to catch up to Active Master using SELECT
MASTER_POS_WAIT('mysql-bin.000001', 87592) ...
+------------------------------------------+
| MASTER_POS_WAIT('mysql-bin.000001', 87592) |
+------------------------------------------+
|                                        0 |
+------------------------------------------+
Removing read_only option from configuration file ...
Turning off read_only flag with: set @@global.read_only := 0
IPControl MySQL Replication Monitor install successful.

Promotion / Demotion Step 2 of 2 complete!
```

3 Remove any existing files under `$INCHOME/etc/support/log`.

4 Execute the following commands.

 # $INCHOME/etc/incontrol stop

 # $INCHOME/etc/support/PromoteExec.sh

 # $INCHOME/etc/incontrol start

3.3 Resynchronizing a Slave

Even with the MySQL Production Executive and Backup Executive servers configured properly, the slave can get out of sync with the master. One method of resolving this is to use the resync_master.sh script to start over with a fresh database dump from the master. When completed, the script displays instructions on the next steps you need to take to resynchronize a slave.

3.3.1 Dumping the Master Database

To execute the resync_master.sh script, follow these steps.

1 As **incadmin** user on the Production Executive, change directory to /opt/incontrol/etc/support.

> **>cd /opt/incontrol/etc/support**

2 Execute the script command.

> **>./resync_master.sh**

The script begins by dumping the master database and determining the log file coordinates.

```
Stopping replication slave process with: STOP SLAVE

Dumping the master database and getting Log File Coordinates ...

Master Log File Coordinates:

MASTER_LOG_FILE='mysql-bin.000001'

MASTER_LOG_POS=14158333


Dump File: dbforrepl.sql
```

Finally, the script displays the steps necessary to resynchronize the slave, including the correct parameters to pass to the resync_slave.sh script on the Backup Executive.

```
Re-synchronization Setup Step 1 of 3 complete!


- Copy the Dumpfile: dbforrepl.sql to the slave server.

- Record the Active Master Log File Coordinates and log in to the passive master
system.

- Navigate to /opt/incontrol/etc/support.

- Once there, run the resync_slave.sh script using the coordinates as follows:


./resync_slave.sh -l mysql-bin.000001 -s 14158333 -h 192.168.196.45 -d <location of
dbforrepl.sql>
```

3 Make a note of the following values from the screen output, which you need when executing the resync_slave.sh script.

> ❑ The name of the MySQL log file, displayed in MASTER_LOG_FILE=

❑ The replication start coordinates of the log file, displayed in
MASTER_LOG_POS=

3.3.2 Copying the Production Executive Dump File to the Backup Executive

Before you execute the resync_slave.sh script, copy the Production Executive dump file that
was created when you executed the resync_master.sh script to the Backup Executive server.

To copy the MySQL database dump file from the Production Executive, follow these steps.

1 Log into the Backup Executive server as incadmin user and change directory to
/opt/incontrol/etc/support.

> **>cd /opt/incontrol/etc/support**

2 Execute the secure copy command, specifying the full pathname of the Production
Executive dump file.

> **> scp incadmin@<host_IP_address>:/opt/incontrol/etc/support/<dump_filename>.**

For example:

```
>scp incadmin@192.168.196.45:/opt/incontrol/etc/support/dbforrepl.sql .
incadmin@192.168.196.45's password:
dbforrepl.sql
100%  391KB 391.2KB/s   00:00
```

3.3.3 Reloading the Dump File and Setting Coordinates

To reload the Production Executive dump file on the Backup Executive and set the log file
coordinates, execute the resync_slave.sh script. When completed, the script displays
instructions on the next steps you need to take to complete the resynchronization process.

resync_slave.sh accepts the following parameters:

Parameter	Required	Description
-d <dump_filename>	Yes	Defines the name of the MySQL database dump file.
-h <host_IP_address>	Yes	Defines the IP address of the Active Master server.
-l <log_filename>	Yes	Defines the name of the MySQL database log file, as defined in the MASTER_LOG_FILE= statement.
-s <log_coordinate>	Yes	Defines the starting point coordinate in the log file where replication begins, as defined in the MASTER_LOG_POS= statement.

To execute the `resync_slave.sh` script, follow these steps.

1   As **incadmin** user, change directory to `/opt/incontrol/etc/support`.

>**cd /opt/incontrol/etc/support**

2   Execute the script command.

>**./resync_slave.sh -l** \<log_filename> **-s** \<log_coordinate>
>**-h** \<host_IP_address> **-d** \<dump_filename>**.sql**

For example:

>./resync_slave.sh -l mysql-bin.000001 -s 14158333 -h 192.168.196.45 -d
>dbforrepl.sql

The script begins by stopping the replication slave process and loading the dumpfile.

```
Stopping replication slave process with: STOP SLAVE
Disconnecting from old master with: CHANGE MASTER TO MASTER_HOST=''
Resetting slave with: RESET SLAVE


Loading dumpfile: dbforrepl.sql ...
```

The script ends by resetting the log file coordinates and restarting the replication slave process.

```
Connecting to master with: CHANGE MASTER TO
MASTER_HOST='192.168.196.45',MASTER_USER='repl',MASTER_PASSWORD='replpass1',
MASTER_LOG_FILE='mysql-bin.000001',MASTER_LOG_POS=14158333

Starting replication slave process with: START SLAVE


Re-synchronization Step 2 of 3 complete!

- To complete the synchronization, log in to the Active Master system.

- Navigate to $INCHOME/etc/support.

- Once there, run the resync_MasterSlaveProcess.sh script
```

3.3.4 Restarting the Slave Process on the Active Master

The resync_master.sh script stops the slave process on the Active Master to prevent statements run on the Passive Master during the re-synchronization process from executing on the Active Master. Once the Passive Master has been synchronized, the slave process needs to be restarted on the Active Master.

To restart the slave process on the Active Master, run the resync_MasterSlaveProcess.sh script on the Active Master.

To execute the **resync_MasterSlaveProcess.sh** script, follow these steps.

1. As i**ncadmin** user, change directory to /opt/incontrol/etc/support.

>cd /opt/incontrol/etc/support

2. Execute the script command.

    ```
    >./resync_MasterSlaveProcess.sh
    ```

    ```
    For example:
    ```

    ```
    >./resync_MasterSlaveProcess.sh
    ```

    The script simply starts the replication slave process.

    ```
    Starting replication slave process with: START SLAVE


    Re-synchronization step 3 of 3 complete!
    ```

4 Troubleshooting

4.1 Troubleshooting MySQL

4.1.1   Timing Issue

On Solaris systems only, an intermittent problem (Reference: IN-6134) occurs with any script that needs to shut down the MySQL database, such as `prepslave.sh`. If you see the following message when any replication script is executed, use the appropriate workarounds described in Table 2.

```
"Stopping MySQL Database Server...
MySQL Database Server could not be stopped: pid=<pid#> user=incadmin"
```

Table 2 MySQL Timing Issue

Problem	Workaround
MySQL stopped but the script reported it could not stop the server.	No action is needed and you can ignore message.
MySQL could not stop and the script also reported it could not stop the server and failed.	Follow these steps: 1. Execute commands to stop and start the MySQL database. 2. Retry the script that just failed.

4.2 Troubleshooting with the Replication Monitor

The Replication Monitor can report information in two ways. The first is a simple HTML page and the second is via scripts launched by the Cisco Prime Network Registrar IPAM Callout Manager.

4.2.1   Monitoring with ReplmonStats.html

Cisco Prime Network Registrar IPAM provides an HTML page called `ReplmonStats.html`, which is served by the Tomcat server installed on the Production Executive. This page (`http://localhost:8080/incontrol/ReplmonStats.html`) shows the sequence number value on both the master and the slave, and calculates the lag between the two values. The user can see the recent history of the sequence numbers for both servers. A sample page showing a lag between the sequence numbers of the master and slave servers is shown in Figure 2.
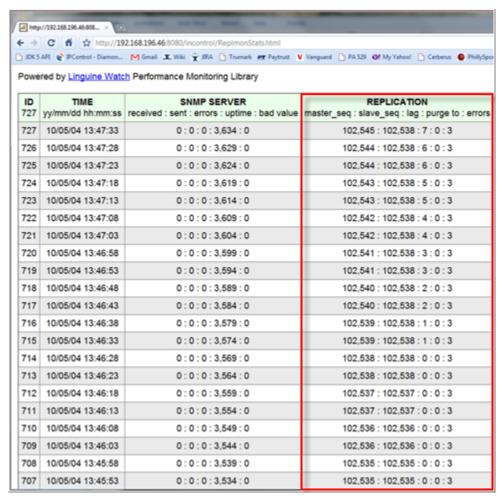
Figure 2 ReplMonStats.html showing sequence number lag

4.2.2   Monitoring with Callout Manager Scripts

The other means of obtaining replication monitor information is by using scripts that can be executed by the Cisco Prime Network Registrar IPAM Callout Manager when predefined monitoring events occur. The Callout Manager can issue callouts for the replication monitor events described in Table 3.

Table 3 Replication Monitor Events

Replication Monitor Event	Callout Manager Script
The Monitor was started	`replmon_started_callout.sh`
The Monitor was stopped	`replmon_stopped_callout.sh`
The monitor could not connect to the Active Master	`replmon_master_connect_error_callout.sh`
The monitor reconnected to the Active Master after a prior disconnect	`replmon_master_reconnect_callout.sh`

Replication Monitor Event	Callout Manager Script
The monitor could not connect to the Passive Master (slave)	`replmon_slave_connect_error_callout.sh`
The monitor reconnected to the Passive Master (or slave) server after a prior disconnect	`replmon_slave_reconnect_callout.sh`
The monitor detected that the slave has begun to lag behind the master beyond the lag NotificationThreshold as defined in the replication monitor's configuration file (mysql_replication_monitor.properties), located under `$INCHOME`.	`replmon_slave_lag_start_callout.sh`
The monitor detected that the slave has completely resynchronized with the master after a prior notified lag	`replmon_slave_resynched_callout.sh`

To configure the Callout Manager to fire on any of the events listed above, you need to create the shell script to be executed for each event, and then define these scripts in the Callout Manager properties file for the appropriate events.

The section of the `callout_manager.properties` file that pertains to the Replication Monitor is shown below.

```
#
# Callouts defined for the MySQL Replication Monitor
#   Each of the callouts defined below will be passed the
#   latest replication status values.
#   These include:
#   MasterSeq - The sequence number found on the Active Master
#   SlaveSeq - The sequence number found on the Passive Master (or slave)
#   PurgeTo - The last binary log file number that was purged.
#   Errors - The number of errors encountered since the monitor was started.
#            This includes connecting to the master or slave and any purge errors.
#   Lag - The current lag between Active and Passive Masters.  (Lag = MasterSeq - SlaveSeq).
#
# The monitor was started
#replmon.started = replmon_started_callout.sh

# The monitor was stopped
#replmon.stopped = replmon_stopped_callout.sh

# The monitor could not connect to the Active Master server
#replmon.master.connect.error = replmon_master_connect_error_callout.sh

# The monitor reconnected to the Active Master server after a prior disconnect
#replmon.master.reconnected = replmon_master_reconnect_callout.sh

# The monitor could not connect to the Passive Master (or slave) server
#replmon.slave.connect.error = replmon_slave_connect_error_callout.sh
```

```
# The monitor reconnected to the Passive Master (or slave) server after a prior disconnect
#replmon.slave.reconnected = replmon_slave_reconnect_callout.sh

# The monitor detected that the Passive Master has begun to lag behind the
# Active Master beyond the lagNotificationThreshold defined in the
# replication monitor's configuration file.
#replmon.slave.lag.start = replmon_slave_lag_start_callout.sh

# The monitor detected that the Passive Master has completely resynchronized with the master
# after a prior notified lag.
#replmon.slave.resynched = replmon_slave_resynched_callout.sh
```

4.2.3   MySQL Binary Log Purge

You can set up the Replication Monitor to purge binary logs that have already been read by the slave server, as well as control the interval at which it checks to see whether there are logs to purge. Set up the following parameters in the `mysql_replication_monitor.properties` file.

MySqlReplManager.Context.canPurgeBinLogs

When set to True, instructs the Active Master to purge any binary logs that have already been read by the slave. It purges logs up to the penultimate log used by the slave.

For example, if the slave is current to `mysql-bin.00020`, the purge log removes all files except for `mysql-bin.00020` and `mysql-bin.00019`.

MySqlReplManager.Context.purgeDelayMillis

By default, instructs the Replication Monitor to check if it can purge logs every 300000 milliseconds, which is the equivalent of 300 seconds (or five minutes).