

How to Set Up Slony-I Replication for Postgres Plus(R)

A Postgres Evaluation Quick Tutorial From EnterpriseDB

January 8, 2010

EnterpriseDB Corporation, 235 Littleton Road, Westford, MA 01866, USA
T +1 978 589 5700 **F** +1 978 589 5701 **E** info@enterprisedb.com **www**.enterprisedb.com

Introduction

Slony is a master to multiple slaves replication system with cascading and failover capabilities. When used with Postgres Plus, it becomes a powerful tool used for safeguarding data, failover/standby capabilities, optimizing online transaction processing, offloading reporting/Bi queries, backup and restore, and migrating systems. This tutorial shows how to efficiently setup a basic master-slave system that can be applied to a variety of applications.

This [EnterpriseDB Quick Tutorial](#) helps you get started with the [Postgres Plus Standard Server](#) or [Postgres Plus Advanced Server](#) database products in a Linux or Windows environment. It is assumed that you have already downloaded and installed Postgres Plus Standard Server or Postgres Plus Advanced Server on your desktop or laptop computer.

This Quick Tutorial is designed to help you expedite your Technical Evaluation of Postgres Plus Standard Server or Postgres Plus Advanced Server. For more informational assets on conducting your evaluation of Postgres Plus, visit the self-service web site, [Postgres Plus Open Source Adoption](#).

In this Quick Tutorial you will learn how to do the following:

- understand key Slony replication terms and concepts
- understand the basic components of a Slony Cluster
- prepare Slony replication nodes
- configure the replication cluster
- start replication

Feature Description

Slony-I is a pre-bundled enterprise module installed by default with Postgres Plus Standard Server and Postgres Plus Advanced Server.

Slony-I or Slony, as it will be referred to hereafter, provides the ability to replicate transactions from one master database to one or more slave databases. Slony is a trigger-based replication solution that asynchronously replicates every transaction that occurs on a source table or sequence object (sequence number generator created by the `CREATE SEQUENCE` statement) to the subscribed slave databases.

Slony replication is very versatile and can be used for a number of different applications:

- Safeguard data with failover/standby capabilities
- Optimize online transaction processing

- Offload reporting/Bi queries
- Backup and restore
- Migrate systems

The following is a summary of some of the basic characteristics of Slony:

- Replication is unidirectional from master to slave databases.
- Replication is from a single master.
- Replicated tables in slave databases cannot be modified by other applications.
- Replication is asynchronous. Transactions are committed independently on the master and each slave in near real time.
- The types of database objects that can be replicated are tables and sequence objects. Replicated tables must have a primary key or a unique, non-null index that can substitute for a primary key.
- Replication can be cascaded. Instead of receiving replicated data directly from the master, a slave may receive replicated data from another slave, instead.

The following is a brief description of some of the other features of Slony.

Controlled Switchover

At times, it may be necessary to bring the master node offline for a variety of reasons such as to perform system maintenance or a system upgrade. In that case, you would want one of the slave nodes to temporarily take over the role of the master node while the original master becomes a slave node and could hence, be taken offline.

This exchange of roles is called *controlled switchover*. When you perform a controlled switchover, the old master becomes a slave node and the slave node becomes the master node. All other slave nodes are notified of the change and become subscribers of the new master node.

Failover

In the event of a catastrophic failure of the master node, Slony supports the *failover* of the master node to a slave node. Failover is an irreversible action so it should only be done if the master node is not recoverable. Once the failover process is completed, the old master can be removed from the configuration. The new master takes over replication to the other slaves in the cluster.

Tutorial Steps

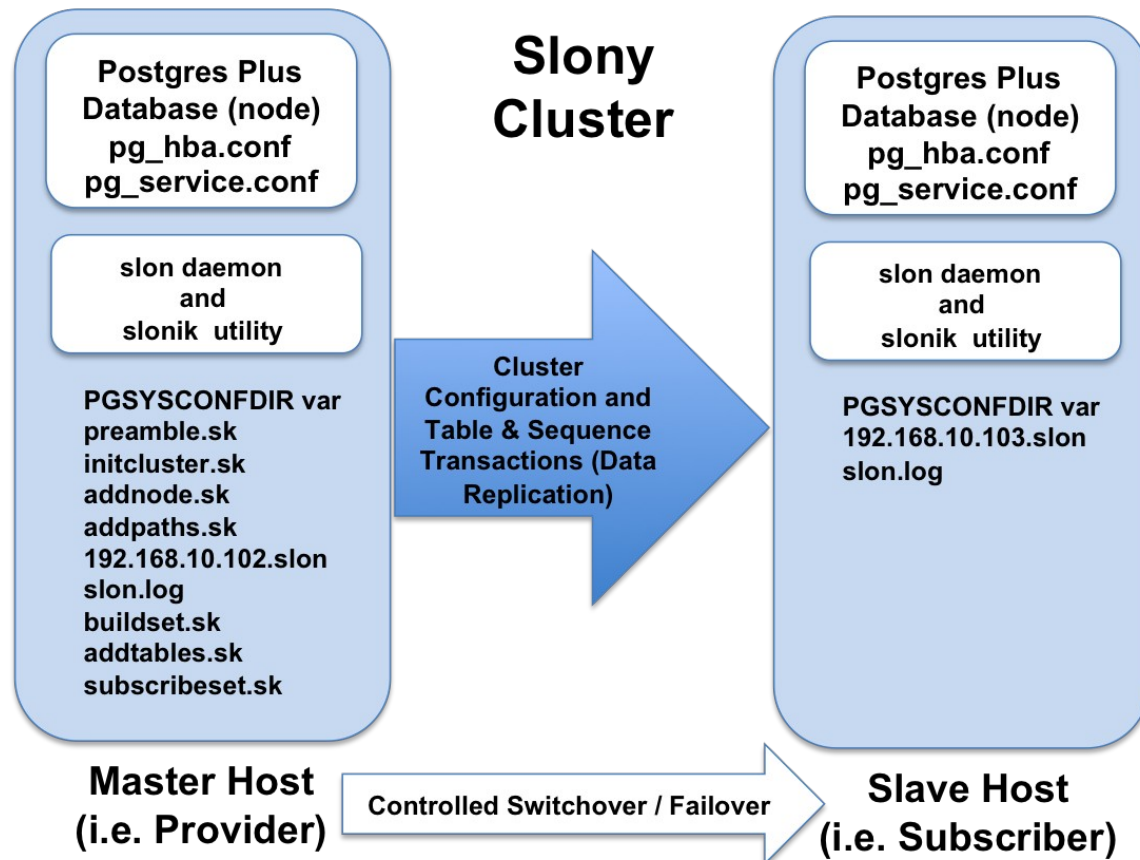
Slony Concepts

The following terms are used to describe the components of a Slony replication system:

- **Cluster.** A set of nodes (databases) participating in Slony replication.
- **Node.** A distinct database identified by a unique combination of IP address, port number, and database name.
- **Service.** A named, high level definition of a node.
- **Path.** Connection information between nodes in a cluster.
- **Replication Set.** A set of tables, and optionally, sequence objects, whose data is to be replicated.
- **Master.** Also referred to as the *origin*, the node that is the sole source of the table data for all other nodes subscribing to a replication set. There is exactly one master node for a replication set. The master node is the only node in which applications can modify table data belonging to the replication set.
- **Slave.** Also referred to as a *subscriber*, a node that receives replicated data from another node. Applications cannot modify table data on a slave node as the data will become out-of-sync with the master.
- **Provider.** A node that provides replicated data to another node. The master node is a provider. A slave node can also act as a provider in a cascaded subscription model.
- **Cascaded Subscription.** A replication model where at least one node acts as both a provider and a subscriber. That is, the node receives replicated data from a provider, and in turn, replicates it to one or more subscribers.
- **slon.** The Slony daemon that controls replication. For Linux each node has its own `slon` daemon. For Windows there is one `slon` service on the host. A Slony replication engine is registered with the `slon` service for each Windows node.
- **slonik.** The utility program that processes commands to create and update a Slony configuration.

Additional information about Slony and the Slony project can be found on the [Postgres Community Projects](#) page of the [EnterpriseDB](#) web site.

The remainder of this Quick Tutorial describes how to set up a basic Slony replication cluster (pictured below) with a master node and one slave node. The example is presented using Standard Server on Linux though the same steps apply to Advanced Server as well.



Differences in procedures for Microsoft Windows® systems are noted throughout the instructions.

Note: When a distinction must be made between a Postgres Plus database that participates in Slony replication and the operating environment in which the Postgres Plus database resides, the Postgres Plus database will be referred to as the *node* and the surrounding operating environment will be referred to as the *host*.

The environment is as follows:

Attribute	Master	Slave
Postgres Plus product	Postgres Plus Standard Server 8.4	Postgres Plus Standard Server 8.4
Postgres Plus home directory (Linux)	/opt/PostgresPlus/8.4SS	/opt/PostgresPlus/8.4SS
Postgres Plus home directory (Windows)	C:\Program Files\PostgresPlus\8.4SS	C:\Program Files\PostgresPlus\8.4SS
Host IP address	192.168.10.102	192.168.10.103
Port	5432	5432
Database name	reptest_node1	reptest_node2
Database username for cluster	slony	slony

Attribute	Master	Slave
configuration and replication processing		
Database superuser for all other miscellaneous operations	postgres	postgres
Replicated tables	sample.dept sample.emp	sample.dept sample.emp

Note: For Advanced Server, substitute `enterprisedb` for `postgres` as the database superuser.

Preparing the Nodes

Step 1: Verify that Slony is installed on the master host and on the slave host. The Slony files are located under the `bin` subdirectory (`dbserver/bin` for Advanced Server) of the Postgres Plus home directory. You should see files named `slon` and `slonik` in this subdirectory.

Note: For Advanced Server on Windows, you should see files named `edb-replication` and `slonik`. The file `edb-replication` takes the place of `slon` as the Slony executable.

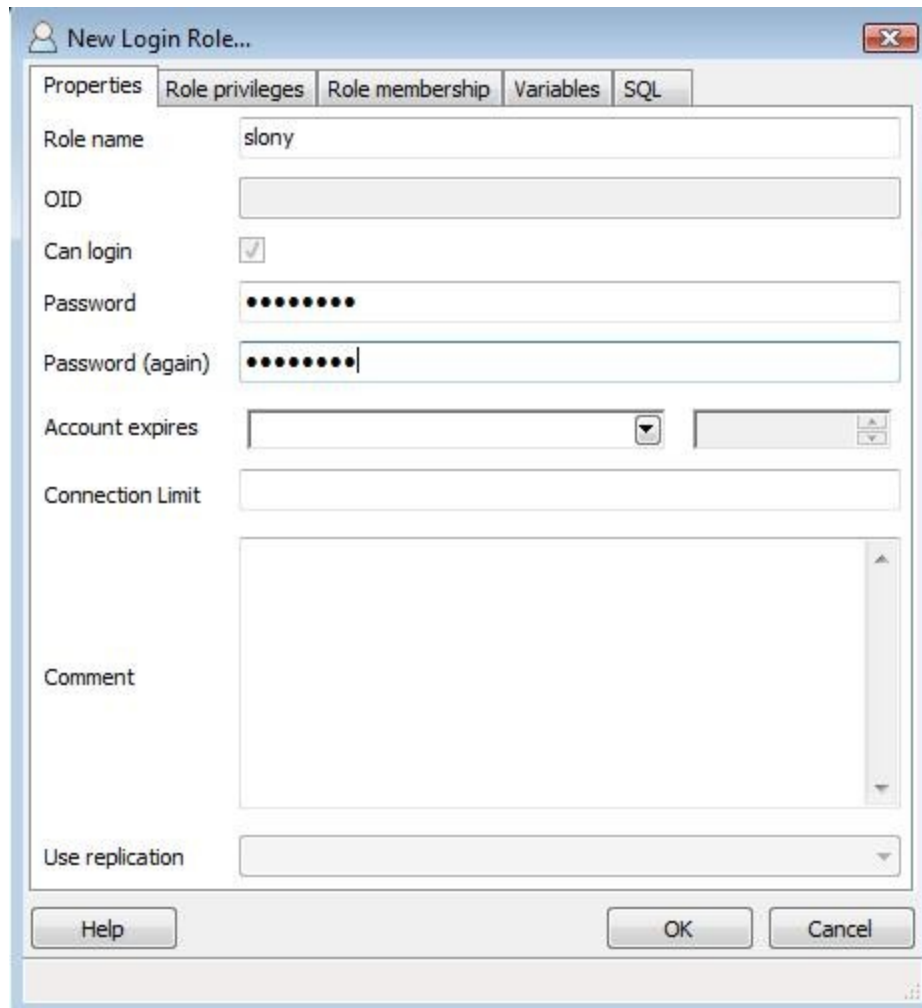
Note: When installing Standard Server, if you de-selected the Slony component, the `slon` and `slonik` programs are not installed. If you did not install Slony, you can use StackBuilder Plus to add Slony to your Standard Server configuration.

Step 2: Create a working directory on the master host and on the slave host where you will create and store the scripts to configure and run Slony.

For this example `/home/user/testcluster` is used as the working directory on both the master and on the slave (`C:\testcluster` for Windows hosts).

Step 3: Create a superuser with catalog modification privileges on the master node and on the slave node that will be used for Slony configuration and replication. In this example, the superuser is named `slony` on both the master and on the slave.

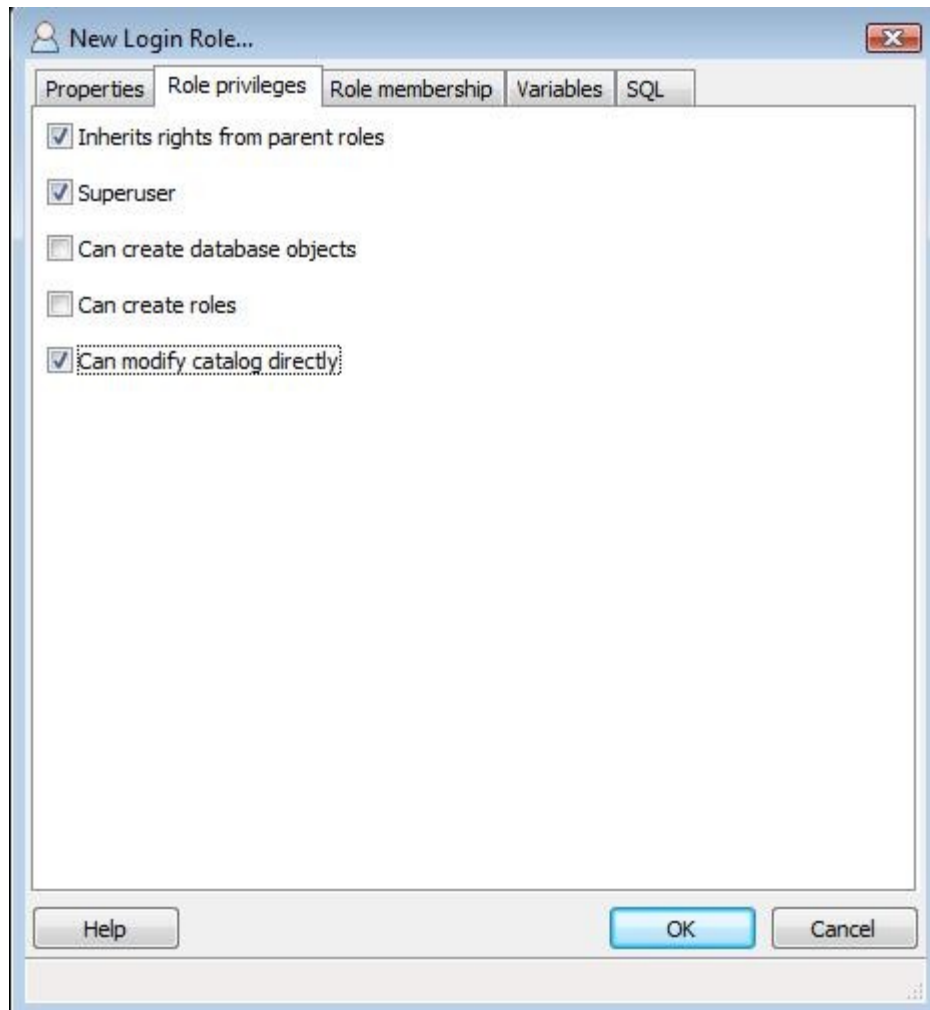
This can be done in pgAdmin in Standard Server (Postgres Studio in Advanced Server). Click the secondary mouse button on the Login Roles node in the Object Browser. Choose New Login Role and fill out the new Login Role dialog box.



The image shows a 'New Login Role...' dialog box with the following fields and controls:

- Properties** (selected tab), **Role privileges**, **Role membership**, **Variables**, **SQL**
- Role name**: slony
- OID**: (empty text box)
- Can login**: ☒
- Password**: (masked with dots)
- Password (again)**: (masked with dots)
- Account expires**: (calendar icon and date input)
- Connection Limit**: (empty text box)
- Comment**: (empty text area)
- Use replication**: (dropdown menu)
- Buttons**: Help, OK, Cancel

Click the Role Privileges tab and check the Superuser and Can Modify Catalog Directly check boxes. Click the OK button.



Step 4: Configure and reload the `pg_hba.conf` file on the master host and on the slave host.

The `pg_hba.conf` file is the host-based authentication configuration file. In a default Postgres Plus installation, this file is located in the `data` subdirectory of the Postgres Plus home directory.

You will need to make sure that it is configured properly on each host to allow connections from every other host in the Slony cluster.

On the master host in a 2-node replication system, the entry you add to permit connection to the master node from the slave host has the following form:

```
host  master_dbname  masterdb_user  slave_ipaddr/32  md5
```

On the slave host the entry you add to permit connection to the slave node from the

master host has the following form:

```
host  slave_dbname  slavedb_user  master_ipaddr/32  md5
```

For Linux only: Be sure there is an entry for the `local` node, which is used by the `slon` daemon to communicate with its own node.

The resulting `pg_hba.conf` file on the master host appears as follows:

```
# TYPE  DATABASE  USER  CIDR-ADDRESS  METHOD
# "local" is for Unix domain socket connections only
local   all             all                                     md5
# IPv4 local connections:
host    reptest_node1  slony  192.168.10.103/32  md5
host    all            all    127.0.0.1/32      md5
# IPv6 local connections:
host    all            all    ::1/128           md5
```

The resulting `pg_hba.conf` file on the slave host appears as follows:

```
# TYPE  DATABASE  USER  CIDR-ADDRESS  METHOD
# "local" is for Unix domain socket connections only
local   all             all                                     md5
# IPv4 local connections:
host    reptest_node2  slony  192.168.10.102/32  md5
host    all            all    127.0.0.1/32      md5
# IPv6 local connections:
host    all            all    ::1/128           md5
```

Be sure to reload the configuration file on each host after making the file modifications.

Choose Reload Configuration (Expert Configuration, then Restart Database on Advanced Server) from the Postgres Plus application menu. This will put the modified `pg_hba.conf` file into effect.

Step 5: Define a `pg_service.conf` file on the master host and on the slave host.

The connection service file, `pg_service.conf`, contains connection information that can be referenced using a service name. Defining a service for each node in the Slony cluster greatly simplifies other scripts since you define this information once instead of in every script.

Note: A template named `pg_service.conf.sample` that can be copied and modified is located in the `share/postgresql` subdirectory (`dbserver/share` in Advanced Server) of the Postgres Plus home directory. For Standard Server on Windows hosts, this file is in the `share` subdirectory.

For this example, the service name assigned to the master node is `192.168.10.102-slonik`. The service name assigned to the slave node is `192.168.10.103-slonik`.

The service entries in the `pg_service.conf` file on the master host appear as follows:

```
[192.168.10.102-slonyk]
dbname=reptest_node1
user=slony
password=password

[192.168.10.103-slonyk]
dbname=reptest_node2
host=192.168.10.103
user=slony
password=password
```

The service entries in the `pg_service.conf` file on the slave host appear as follows:

```
[192.168.10.102-slonyk]
dbname=reptest_node1
host=192.168.10.102
user=slony
password=password

[192.168.10.103-slonyk]
dbname=reptest_node2
user=slony
password=password
```

Before running the `slon` or `slonik` programs, set the environment variable `PGSYSCONFDIR` to the directory containing the `pg_service.conf` file. (For Windows hosts, add a system environment variable named `PGSYSCONFDIR`.) This process is described in more detail in the next section.

For this example `pg_service.conf` is stored in directory `/home/user/testcluster` (`C:\testcluster` for Windows) on each host.

Step 6: Use the `pg_dump` utility program to create a backup file of the schema and table definitions of the master tables that you wish to replicate. **Do not include the table data in your backup file.**

The `pg_dump` command to use has the following form:

```
pg_dump -U user -n schema -s -f backup_file database_name
```

Program `pg_dump` is located in the `bin` subdirectory (`dbserver/bin` for Advanced Server) of the Postgres Plus home directory.

The following example creates a backup file named `sample.backup` containing the sample schema with the `dept` and `emp` tables from database `reptest_node1`:

```
cd /opt/PostgresPlus/8.4SS/bin
./pg_dump -U postgres -n sample -s -f /home/user/sample.backup reptest_node1
```

Step 7: Copy the backup file to the slave host. On the slave host, restore the backup file

to the database to which you want to replicate the master tables. Use the `psql` utility program to restore the backup file:

```
psql -U user -f backup_file database_name
```

Program `psql` is located in the `bin` subdirectory (`dbserver/bin` for Advanced Server) of the Postgres Plus home directory.

Note: For Advanced Server on Windows, use program `edb-psql`.

In the following example, the `createdb` program is used to create the `reptest_node2` database, and then the `psql` program is used to restore the `sample` schema from the `sample.backup` file into the `reptest_node2` database.

```
cd /opt/PostgresPlus/8.4SS/bin
./createdb -U postgres reptest_node2
./psql -U postgres -f /home/user/sample.backup reptest_node2
```

There are now table definitions, but no data, for `sample.dept` and `sample.emp` in the `reptest_node2` database:

```
./psql -U postgres reptest_node2
psql (8.4.1)
Type "help" for help.

reptest_node2=# SELECT * FROM sample.dept;
 deptno | dname | loc
-----+-----
(0 rows)

reptest_node2=# SELECT * FROM sample.emp;
 empno | ename | job | mgr | hiredate | sal | comm | deptno
-----+-----+-----+-----+-----+-----+-----+-----
(0 rows)
```

Configuring the Cluster and Starting Replication

Configuration of the Slony cluster is done by supplying commands to the `slonik` utility program. Separate script files are constructed for each step of the configuration process. This will help ensure that you successfully complete each step of the process before proceeding to the next one and will make troubleshooting much easier.

Step 1: Log on to the master host using any valid account on the computer. (For Windows, use a computer administrator account.)

For Linux only: Set the environment variable `PGSYSCONFDIR` to the directory containing the `pg_service.conf` file. Change to your working directory where you will create and run the `slonik` scripts.

```
export PGSYSCONFDIR=/home/user/testcluster
cd /home/user/testcluster
```

For Windows only: Add the environment variable `PGSYSCONFDIR` to the system. In the Control Panel, open System, select the Advanced tab, and click the Environment Variables button. Add `PGSYSCONFDIR` with value `C:\testcluster` to System Variables.

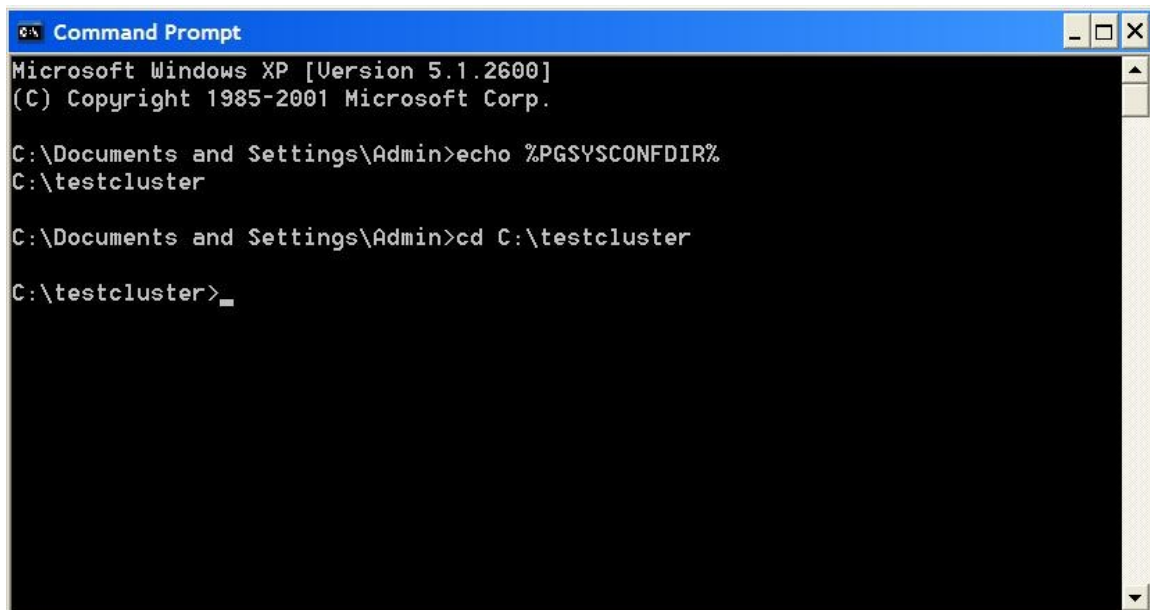
Note: Other applications using `libpq` may also be affected by the use of this environment variable.

For Windows XP only: Restart your computer at this point. (The Slony service that you will register in Step 8 will not have access to the new `PGSYSCONFDIR` system environment variable unless you restart the computer.)

Open a Command Prompt window.

For Windows Vista only: If you are using User Account Control, run the Command Prompt window as an administrator. (Click the secondary mouse button on Command Prompt, and then in the Command Prompt submenu, click the primary mouse button on Run as Administrator.)

In the Command Prompt window, verify `PGSYSCONFDIR` is set to the working directory. Change to your working directory where you will create and run the `slonik` scripts.



```
C:\> Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Admin>echo %PGSYSCONFDIR%
C:\testcluster

C:\Documents and Settings\Admin>cd C:\testcluster

C:\testcluster>_
```

Step 2: Repeat Step 1 on the slave host. You should now have two terminal sessions running – one on the master host and one on the slave host.

Step 3: Continue with the following steps on the master host. Create a preamble file to provide connection information for each node in the cluster. This connection information is used by the `slonik` program to set up and control administration of the cluster.

The following is preamble file `preamble.sk`:

```
# file preamble.sk

cluster name = testcluster;
node 1 admin conninfo = 'service=192.168.10.102-slonik';
node 2 admin conninfo = 'service=192.168.10.103-slonik';
```

The name assigned to the cluster is `testcluster`. The `CONNINFO` parameters reference the service names defined in the `pg_service.conf` file.

Use the `INCLUDE` statement in other `slonik` scripts to reference the preamble file:

```
include <preamble.sk>;
```

Step 4: Create a script to define the Slony replication cluster. The replication cluster is defined using the `INIT CLUSTER` command.

The following is script `initcluster.sk`:

```
#!/opt/PostgresPlus/8.4SS/bin/slonik

#file initcluster.sk

include <preamble.sk>;

init cluster (id=1, comment='hostname=192.168.10.102');
```

In the `INIT CLUSTER` command, the master node is assigned a numeric identifier (typically, 1) using the `ID` parameter.

Run the script you just created.

For Linux only: Be sure you add execute permission to this script (and all other scripts created in subsequent steps) before running it.

```
$ chmod ugo+x initcluster.sk
$ ./initcluster.sk
./initcluster.sk:7: Possible unsupported PostgreSQL version (80401) 8.4,
defaulting to 8.3 support
```

Note: The warning message displayed in the output can be ignored.

For Windows only: Create a batch file to run script `initcluster.sk` and all subsequent `slonik` scripts created in these instructions.

The following is Windows batch file `runslonik.bat`:

```
@ECHO OFF
REM file runslonik.bat
REM
REM Batch file to run slonik

C:\Program Files\PostgresPlus\8.4SS\bin\slonik < %1
```

Run the batch file with the `initcluster.sk` script as a parameter as follows:

```
C:\testcluster>runslonik initcluster.sk
<stdin>:7: Possible unsupported PostgreSQL version (80401) 8.4, defaulting to
8.3 support
```

Step 5: Create a script to add the slave node to the replication cluster. The node is added using the `STORE NODE` command.

The following is script `addnode.sk`:

```
#!/opt/PostgresPlus/8.4SS/bin/slonyk

#file addnode.sk

include <preamble.sk>;

store node (id=2, comment='hostname=192.168.10.103', event node=1);
```

The slave node is assigned an identifier of 2 using the `ID` parameter.

Note: Specification of the `EVENT NODE` parameter is required in the `STORE NODE` command for Slony version 2.x. Prior 1.x versions of Slony allowed omission of this parameter for a default value.

Run the script you just created.

```
$ ./addnode.sk
./addnode.sk:7: Possible unsupported PostgreSQL version (80401) 8.4,
defaulting to 8.3 support
```

Note: The warning message displayed in the output can be ignored.

Step 6: Create a script to add communication paths to the replication cluster. Communication paths are added using the `STORE PATH` command. The communication paths are used by Slony daemons to connect to remote nodes in the cluster.

The following is script `addpaths.sk`:

```
#!/opt/PostgresPlus/8.4SS/bin/slonyk

#file addpaths.sk

include <preamble.sk>;

store path (server=1, client=2, conninfo='service=192.168.10.102-slonyk');
```

```
store_path (server=2, client=1, conninfo='service=192.168.10.103-slunik');
```

There should be a `STORE PATH` command from each node to every other node. (Connections are not established unless they are actually used.)

For this 2-node example, the first `STORE PATH` command provides the communication path for the Slony daemon running on the slave host to connect to the master node. The second `STORE PATH` command provides the communication path for the Slony daemon running on the master host to connect to the slave node.

Run the script you just created.

```
$ ./addpaths.sk
```

Step 7: Create a Slony daemon configuration file on the master host and on the slave host for the parameters needed to start the Slony daemon for each respective node. The parameters define how the daemon connects to the named cluster.

The following is configuration file `192.168.10.102.slon` on the master host:

```
#file 192.168.10.102.slon

cluster_name='testcluster'
conn_info='service=192.168.10.102-slunik'
```

The following is configuration file `192.168.10.103.slon` on the slave host:

```
#file 192.168.10.103.slon

cluster_name='testcluster'
conn_info='service=192.168.10.103-slunik'
```

Step 8: Start Slony on the master host.

For Linux only: Start the Slony daemon on the master host. The Slony daemon executable named `slon`, is found in the `bin` subdirectory (`dbserver/bin` for Advanced Server) of the Postgres Plus home directory.

The command to start the Slony daemon using the configuration file created in the prior step is the following:

```
slon -f configfile > logfile 2>&1 &
```

On the master host, start the Slony daemon as follows:

```
$ /opt/PostgresPlus/8.4SS/bin/slons -f 192.168.10.102.slons > slons.log 2>&1 &
[1] 5877
```

You can verify that the Slony daemon is running by using the following command:

```
ps aux | grep slon
```

This is shown by the following:

```
$ ps aux | grep slon
user      5877  0.0  0.1  4756 1204 pts/2    S   13:03   0:00
/opt/PostgresPlus/8.4SS/bin/slony -f 192.168.10.102.slony
user      5879  0.0  0.1  54064 1116 pts/2    Sl  13:03   0:00
/opt/PostgresPlus/8.4SS/bin/slony -f 192.168.10.102.slony
postgres  5883  0.0  0.4  48752 4964 ?        Ss   13:03   0:00 postgres:
slony retest_node1 [local] idle
postgres  5885  0.0  0.5  49088 5688 ?        Ss   13:03   0:00 postgres:
slony retest_node1 [local] idle
postgres  5888  0.0  0.3  48216 3152 ?        Ss   13:03   0:00 postgres:
slony retest_node1 [local] idle
postgres  5890  0.0  0.5  48784 5204 ?        Ss   13:03   0:00 postgres:
slony retest_node1 [local] idle
user      5896  0.0  0.0   3064   732 pts/2    S+  13:04   0:00 grep slon
```

Check the log file, `slon.log`, to verify that there are no error messages.

For Windows only: Register the Slony service, and then register a Slony replication engine with the Slony service.

A Slony service is registered with the following command:

```
slon -regservice service_name
```

A Slony replication engine is registered to the service with the following command:

```
slon -addengine service_name configfile
```

Note: For Advanced Server, use program `edb-replication` in place of `slon`.

The following shows the creation of a service named Slony along with a replication engine:

```
C:\testcluster>set PATH=C:\Program Files\PostgresPlus\8.4SS\bin;%PATH%

C:\testcluster>slon -regservice Slony
Service registered.
Before you can run Slony, you must also register an engine!

WARNING! Service is registered to run as Local System. You are
encouraged to change this to a low privilege account to increase
system security.

C:\testcluster>slon -addengine Slony C:\testcluster\192.168.10.102.slony
Engine added.
NOTE! You need to restart the Slony service before this takes effect.
```

Start the Slony service by opening Control Panel, Administrative Tools, and then Services. Select the Slony service and click the Start link.

Use the Windows Event Viewer for applications to check for any Slony errors. In the same Administrative Tools window of the Control Panel that you used to start the Slony service, open Event Viewer, (then open Windows Logs if you are using Windows Vista), then open Application.

Step 9: Start Slony on the slave host.

For Linux only: On the slave host repeat Step 8 using the Slony daemon configuration file you created in Step 7 for the slave node:

```
$ /opt/PostgresPlus/8.4SS/bin/slon -f 192.168.10.103.slony > slon.log 2>&1 &
[1] 9522
```

For Windows only: On the slave host repeat Step 8 using the Slony daemon configuration file you created in Step 7 for the slave node.

```
C:\testcluster>set PATH=C:\Program Files\PostgresPlus\8.4SS\bin;%PATH%

C:\testcluster>slon -regservice Slony
Service registered.
Before you can run Slony, you must also register an engine!

WARNING! Service is registered to run as Local System. You are
encouraged to change this to a low privilege account to increase
system security.

C:\testcluster>slon -addengine Slony C:\testcluster\192.168.10.103.slony
Engine added.
NOTE! You need to restart the Slony service before this takes effect.
```

Step 10: Continue with this step on the master host. Create a script to add a replication set to the replication cluster. A replication set is added using the `CREATE SET` command.

A replication set contains the database objects that you wish to replicate from the master node to the slave node.

The following is script `buildset.sk`:

```
#!/opt/PostgresPlus/8.4SS/bin/slonyk

#file buildset.sk

include <preamble.sk>;

create set (id=1, origin=1, comment='source tables for repset #1');
```

This replication set is assigned an identifier of 1 using the `ID` parameter.

Run the script you just created.

```
$ ./buildset.sk
```

Step 11: Create a script to add tables to the replication set. Tables are added using the `SET ADD TABLE` command.

Slony requires a primary key or a unique, non-null index on each replicated table otherwise the `SET ADD TABLE` command will fail, and an error message will be displayed.

For each table provide values for the following parameters:

- **SET ID.** Replication set identifier
- **ORIGIN.** Origin node identifier
- **ID.** Identifier to be assigned to the table that must be unique amongst all table identifiers in the cluster
- **FULLY QUALIFIED NAME.** Fully qualified table name
(*schema_name.table_name*)
- **COMMENT.** Description of the table
- **KEY.** Primary key name or unique, non-null index name (not qualified by the schema name)

Note: Tables that are dependent upon each other by foreign key constraints must be added to the same replication set.

The following is script `addtables.sk`:

```
#!/opt/PostgresPlus/8.4SS/bin/slonik

#file addtables.sk

include <preamble.sk>;

set add table(set id=1, origin=1, id=1, fully qualified name='sample.dept',
comment='source table #1', key='dept_pk');
set add table(set id=1, origin=1, id=2, fully qualified name='sample.emp',
comment='source table #2', key='emp_pk');
```

Run the script you just created.

```
$ ./addtables.sk
```

Step 12: Create a script to subscribe the slave node to the replication set. Nodes are subscribed to a replication set using the `SUBSCRIBE SET` command.

For each slave joining the replication set, provide the following parameters:

- **ID.** Replication set identifier
- **PROVIDER.** Node identifier of the data provider for the slave
- **RECEIVER.** Node identifier of the slave subscribing to the set
- **FORWARD.** Set to `YES` to allow this slave to become a provider at some point in the future

The following is script `subscribeset.sk`:

```
#!/opt/PostgresPlus/8.4SS/bin/slonyk  
  
#file subscribeset.sk for adding subscribers to replication set  
  
include <preamble.sk>;  
  
subscribe set (id=1, provider=1, receiver=2, forward=yes);  
sync(id=1);  
wait for event (origin=1, confirmed=2, wait on=1);
```

Run the script you just created.

```
$ ./subscribeset.sk
```

The Slony daemons synchronize with each other and initialization of the slave database begins. The tables in the slave are truncated and the data in the master node is copied to the slave node. If data fails to replicate to the slave node, look for error messages in the `slon.log` file (for Windows, use Event Viewer for applications) on the master host and on the slave host.

You now have a complete Slony replication environment.

Conclusion

In this Quick Tutorial you learned how to set up Slony-I replication on a Postgres Plus database.

EnterpriseDB has the expertise and services to assist you in setting up a Slony cluster, fully testing it, and providing management scripts. The [Postgres Plus Replication Setup Service](#) is described on the Packaged Services page of the [EnterpriseDB](#) web site.

You should now be able to proceed confidently with a Technical Evaluation of Postgres Plus.

The following resources should help you move on with this step:

- [Postgres Plus Technical Evaluation Guide](#)
- [Postgres Plus Getting Started resources](#)
- [Postgres Plus Quick Tutorials](#)
- [Postgres Plus User Forums](#)
- [Postgres Plus Documentation](#)
- [Postgres Plus Webinars](#)