



Mammoth Replicator 1.7

Configuration Guide

© Command Prompt, Inc.



Table of Contents

Installation.....	4
The 12 steps to replication.....	4
Database requirements.....	5
A note about ADD/DROP column.....	5
init-mammoth-database.....	6
GUC variables for postgresql.conf.....	6
replication_enable.....	6
replication_mode.....	6
replication_slave_no.....	6
replication_database.....	6
replication_slave_batch_mode.....	6
replication_slave_batch_mode_timeout.....	6
replication_mcp_address.....	6
replication_mcp_port.....	6
replication_mcp_authkey.....	7
replication_mcp_require_ssl.....	7
replication_data_path.....	7
mcp_server syntax:.....	8
The mcp_server.conf parameter description:.....	8
mcp_listen_address	8
mcp_listen_port	8
mcp_authkey	8
mcp_max_slaves.....	8
mcp_echo_timeout.....	8
mcp_master_address.....	8
mcp_slave_addresses.....	9
mcp_queue_min_size.....	9
mcp_queue_max_size.....	9
mcp_txlog_path	9
mcp_require_ssl	9
mcp_log_level	9
mcp_promote_allow.....	9
User / Group (Role) replication.....	10
Grant / Revoke.....	10
Sync.....	11
MCP REFRESH.....	11
MCP BATCHUPDATE.....	11
Enabling.....	11
ALTER TABLE name ENABLE REPLICATION [ON SLAVE number].....	11
Large Object Replication.....	11
Promotion.....	12
PROMOTE.....	12
PROMOTE FORCE.....	12
PROMOTE BACK.....	12
Status.....	13
slave_stat [path to slave replication log].....	13
mcp_stat [path to mcp queue].....	13
DDL Changes.....	14
Useful Functions.....	15
is_replicated(REGCLASS).....	15
get_replication_status(REGCLASS).....	15
enable_replication(NAME, INTEGER[]).....	15



enable_replication_by_schema(NAME, INTEGER[])	15
enable_replication_global(INTEGER[])	15
disable_replication(REGCLASS)	15
disable_replication_by_schema(NAME)	15
disable_replication_global()	15
Getting support	16



Introduction

The following document describes, step by step how to configure Mammoth PostgreSQL Replicator 1.7. This document assumes **at least entry level administrator** knowledge of PostgreSQL and Linux/Solaris.

If you are not comfortable installing software, we are happy to install it for you. Installation support incidents are available for 350.00 USD. More information can be found at <http://www.commandprompt.com/support> .

Installation

Please follow the standard PostgreSQL installation instructions which can be found here:

<http://www.commandprompt.com/community/pgdocs8/installation>

If you are running a packaged version of Mammoth PostgreSQL Replicator please refer to your operating system instructions on installing software. Note that all versions of Mammoth Replicator install into /opt/mammoth and create a user mammoth. Further we do not conflict with installed copies of PostgreSQL.

Mammoth Replicator currently supports the 8.0.x, 8.1.x series of PostgreSQL releases and is 100% compliant with those releases. However, PostgreSQL/Replicator is not cross version compatible therefore if you are currently running 8.0.x you must run the 8.0.x version of Replicator.

The 12 steps to replication

1. If installing on an existing PostgreSQL database, **backup** before continuing to step 2.
2. Install Mammoth PostgreSQL Replicator on master and slave
3. Install mcp_server on additional machine or slave
4. Start Mammoth PostgreSQL Replicator without replication parameters on master and slave
5. Apply PGDATA/postgreql/replication_schema.sql to replicated database on master and slave[s]. (**8.0 only, if running 8.1 see section on: init-mammoth-database**)
6. Stop Mammoth PostgreSQL Replicator on master and slaves
7. Configure mcp_server
8. Set GUC variables on master and slave[s]
9. Start mcp
10. Start master
11. Start slaves
12. Connect to master and execute ALTER TABLE commands to enable replication on relations.



Database requirements

1. The table to be replicated must have a PRIMARY KEY.
2. You must be running X86/X86_64: Linux, FreeBSD, or MacOSX (X86 Only).
3. The database name on the Master & Slaves must be the same.
4. Replicator will not replicate the schema. You must restore your schema to the slaves from the master before you begin replication.
5. Replicator can only replicate one database. If you have multiple databases you can either initialize clusters for each database or move all databases into a single database using schemas/namespaces.

A note about ADD/DROP column

When initializing a slave it is common to use the output from a `pg_dump -s`. This will normally be just fine. However, if you have utilized ADD or DROP COLUMN functionality on any of your relations there is an extra step.

PostgreSQL does not actually DROP a column, it marks it invisible. As replicator replicates tuples replicator will also replicate the invisible columns. If your slave is unaware of what to do with those invisible columns, the replication will fail.

There are three common ways to deal with this limitation:

1. Schedule an outage and use tar, rsync or other file system backup method to make a physical copy of your cluster. Then use that copy as the basis for your Slave.
2. Enable Point in Time Recovery and use a snapshot to create the basis for your slave.
3. Use the dump-table command. Dump table is a rudimentary perl script that will allow you generate a proper schema dump of the relation that has had columns added or dropped. The downside to dump-table is that it will not dump constraints or foreign keys. You will need to apply those afterward.



init-mammoth-database

The executable used to initialize a replicated database with replicator 8.1 or above. Execution assumes that the **postgres** binary is in your path, and that **PostgreSQL** is not currently running. Syntax as follows:

```
Usage: init-mammoth-database -D DATADIR DATABASE
```

```
DATADIR    is where the database files are stored
DATABASE  is the name of the database on which Replicator catalog's are
           to be installed
```

Where DATADIR is the data directory and DATABASE is the replicated database.

GUC variables for postgresql.conf

Mammoth PostgreSQL Replicator uses the same configuration file and syntax as the community PostgreSQL. However there are a few parameters that need to be added to the postgresql.conf file on the master and slaves. They are:

replication_enable

bool Whether Replicator is enabled

replication_mode

string Whether this server is a *master* or *slave*

replication_slave_no

integer On a slave, the ID of this server. This parameter is slave specific.

replication_database

string The database to be replicated

replication_slave_batch_mode

bool Enable "slave batch mode", on which a slave will connect only when the MCP BATCHUPDATE command is issued. This parameter is slave specific.

replication_slave_batch_mode_timeout

integer The amount of time in seconds a batch update will wait for new transactions before disconnecting. This parameter is slave specific.

replication_mcp_address

string The IP address of the MCP server

replication_mcp_port

integer The TCP port where the MCP server is listening

replication_mcp_authkey

string The auth token (*password*) of the MCP server



replication_mcp_require_ssl

bool Whether to require replicator to use SSL with communication to the MCP

replication_data_path

string The directory where Replicator data will be stored. The default is PGDATA/rlog



MCP_SERVER

The `mcp_server` is the mammoth control process (known affectionately as the Master Control Process, thanks Dave). The `mcp_server` is the server responsible for all communication of replicated data between the master and slaves.

The `mcp_server` binary is included with the Mammoth Replicator package but can run from any machine that is running a supported operating system. It is not suggested that the `mcp_server` be run from the master server.

mcp_server syntax:

```
mcp_server -c <configuration file> -l <log file> &
```

The `mcp_server` will create a pid file within the `mcp_txlog_path`. If you are running a supported Linux distribution you will have init scripts within `/etc/init.d` for the `mcp_server`.

The `mcp_server` should be run as the same user that initialized the PostgreSQL catalog. This user is typically called `postgres`.

The `mcp_server.conf` parameter description:

mcp_listen_address

IP address to listen on, use 0.0.0.0 for all interfaces

mcp_listen_port

IP Port to listen on

mcp_authkey

A token for authentication between the master/slave and `mcp`

mcp_max_slaves

The maximum number of slaves the `mcp` should accept connections from.

mcp_echo_timeout

Time, in seconds, that the MCP server will wait for the master before it will accept a PROMOTE FORCE command from a slave. Default is 15.

mcp_master_address

The IP address of the master



mcp_slave_addresses

The identifier (0:) and the IP address of the slave/s. Should be a comma separated list, e,g; "0:192.168.1.5,1:192.168.1.10" Where "0" is the first slave and 192.168.1.5 is the ip address of the first slave.

mcp_queue_min_size

The minimum size of the mcp_queue in bytes. In other words the minimum data set size that the mcp will store before it starts to recycle data. The larger this value is, the longer a slave can be offline without forcing MCP to request a full dump from master.

mcp_queue_max_size

The maximum size of the mcp_queue in bytes. Should be larger than the database. Similar to the mcp_queue_min_size parameter except that this is the largest the queue will be allowed to get before it starts to recycle data.

mcp_txlog_path

The path to the replicator transaction logs.

mcp_require_ssl

Require SSL 1 (yes) or 0 (no)

mcp_log_level

Level of MCP output verbosity. Set it to debug5 if you need to send debugging output to CMD. This will produce a lot of output. If you are going to send the log to CMD please only include the last 1000 lines up to where an error occurs in the log file. Default is LOG. Possible values are PANIC, FATAL, ERROR, WARNING, NOTICE, INFO, COMERROR, LOG, DEBUG1, DEBUG2 ... DEBUG5 and their priorities are the same as in postgresql.conf (e.g. enabling NOTICE log level would show all PANIC, FATAL, ERROR, WARNING and NOTICE messages).

mcp_promote_allow

A comma separated list of slave IDs that are allowed to use the promote feature. The id is culled from the mcp_slave_addresses parameter. For example if you have "0:192.168.1.2" the id is "0".



User / Group (Role) replication

Mammoth Replicator automatically replicates all users and groups that are created on the master, to all the slaves. As soon as you create a user on the master, an identically-named user should appear on all the slaves. The only difference will be on the SYSIDs chosen; the slaves won't necessarily have the same SYSIDs as the master, not among themselves.

Note that you can create users or groups independently in the slaves, but as soon as you create a new user with the same name on the master, the user on the slave will be updated to match the characteristics of the user on the master. (The same thing applies if you rename a user on the master to a name that already exists on some slave -- the user on the slave will be updated to match the one on the master.) This may lead to surprising behavior. For example if you create user A on a slave, then create and drop user A on the master, user A on the slave will be dropped as well, even though it predated the creation on the master. To work around this behavior, one idea is to prefix slave-specific users with something that's not going to be used on the master; say, user foo on slave 1 would really be called "slavel_foo".

Mammoth Replicator only replicates users while it's active. If you disable replication on the master and then create a user, it will not be created on the slave, even after you turn replication back on. However, if you update the user on the master (say, rename it or change one of its attributes), it will be replicated to the slaves at that time.

Grant / Revoke

Mammoth Replicator also replicates GRANT and REVOKE statements, i.e., if you GRANT a privilege to a user on a table on the master, it will be granted on the slaves as well. If you grant a privilege to a user that doesn't exist on the slaves, it will be silently skipped (i.e., no error will occur, but there will be no additional privilege granted to this non-existent user.) But if said user is later created on the slave, the privilege will be updated to include that user as soon as some other privilege is changed on that table. (In other words, this doesn't happen right away, but as soon as the privileges for that table are processed again.)



Replication Commands

Sync

MCP REFRESH

A master only command used to resync all slaves from the master.

MCP BATCHUPDATE

A slave only command. Tells a batch replicated slave to request a batch. Use cron or other job scheduler to manage frequency

Enabling

```
ALTER TABLE name ENABLE REPLICATION [ON SLAVE number]
```

A master only command. Extends standard ALTER TABLE from PostgreSQL.

```
ALTER TABLE foo ENABLE REPLICATION ON SLAVE 0
```

Would configure replication on table foo for SLAVE 0.

```
ALTER TABLE foo ENABLE REPLICATION
```

Would enable replication on table foo for all configured slaves. Enabling replication on a table that contains data will cause a full sync. You can enable replication on empty tables without causing a full sync.

Large Object Replication

Replicator does not replicate the entire pg_largeobject table. In order to have large object replication work, you must have a referencing lookup table. This table must be replicated. Secondly you must tell Replicator which column is referencing the large objects. The command for enabling the large object relationship is below.

```
ALTER TABLE foo ALTER bar ENABLE LO
```

Where ALTER bar is the name of the oid column referencing large objects.

Disabling

```
ALTER TABLE foo DISABLE REPLICATION
```

Would disable replication on table foo. It will not erase the slave configuration, thus allowing you to make changes to the table on the slaves and then the master and ENABLE replication again without additional configuration.



Promotion

PROMOTE

A slave only command. Can only be executed if the slave number is allowed to promote per the `mcp_server.conf`. Below is the `PROMOTE` sequence:

1. Execute `PROMOTE` on slave. The slave will restore all transactions that the MCP has received to the point that `PROMOTION` was executed.
2. Master if still running converts to slave
3. Slave restores all current transactions from queue
4. Slave becomes master

PROMOTE FORCE

A slave only command. Can only be executed if the slave number is allowed to promote per the `mcp_server.conf`. Will not work if master is currently connected to MCP. Below is a `PROMOTE FORCE` sequence:

1. Execute `PROMOTE FORCE` on slave. The slave sends notification to MCP.
2. Slave becomes Master after restoring all completely received transactions from the queue.

The use of `PROMOTE FORCE` is for emergencies only. You can not `PROMOTE BACK` once a `PROMOTE FORCE` has been issued.

PROMOTE BACK

If executed on a slave, can only be executed if the slave number is allowed to promote per the `mcp_server.conf`. Below is a `PROMOTE BACK` sequence;

1. Execute `PROMOTE BACK` on the promoted slave or master.
2. Demoted Master receives all updates (possibly a full sync)
3. Demoted Master becomes Master
4. Promoted slave becomes slave



Status

slave_stat [path to slave replication log]

A shell command performed on the slave. Returns number of records received and number of records restored.

mcp_stat [path to mcp queue]

A shell command performed on the mcp machine. Returns a number of information as outlined below:

Global TransactionLog statistics:

```
sync: 1 (SYNC)
rec nums:      base    1 last  4455
              first   1 ack    0
txlog queue size: 106992
txlog data size: 190463
```

Slaves status:

max slaves: 5

```
Slave0 -- DUMP  first record: 1  ack record: 0  time: Tue Dec  5 09:03:09 2006
Slave1 -- DUMP  first record: 1  ack record: 0  time: Tue Dec  5 09:01:42 2006
Slave2 -- DUMP  first record: 1  ack record: 0  time: --
Slave3 -- DUMP  first record: 1  ack record: 0  time: --
Slave4 -- DUMP  first record: 1  ack record: 0  time: --
```

The important parameters of the above are:

sync: The parameter has two options SYNC and DESYNC to reflect the status of replication in general.

txlog queue size: The size of the transaction queue. This can be used to help determine the mcp_min/max_queue sizes.

Slaves status: A numerical listing of the slaves in production and there current state. Typically state is one of either SYNC, DESYNC or DUMP. SYNC means that the slave is replicating normally. DESYNC means the slave is out of sync. DUMP means that the slave is disconnection, has just connected (and a full sync is occurring) or has been out of sync for too long and a full dump of the master has been requested.

first record: The next record in the transaction list.

ack record: The last acknowledged record sent to the slaves.

There will always been a gap of at least 1 between first and ack record.



DDL Changes

It is possible to add and drop columns to replicated tables within Replicator. This type of change to your table structure will require a full sync and therefore is best done in batch or after hours. Below is an example schema change:

On Master:

```
BEGIN;  
ALTER TABLE foo DISABLE REPLICATION;  
ALTER TABLE bar DISABLE REPLICATION;  
  
ALTER TABLE foo ADD COLUMN last_name text;  
ALTER TABLE bar DROP COLUMN ssn;  
COMMIT;
```

On Slave:

```
BEGIN;  
ALTER TABLE foo ADD COLUMN last_name text;  
ALTER TABLE bar DROP COLUMN ssn;  
COMMIT;
```

On Master:

```
BEGIN;  
ALTER TABLE foo ENABLE REPLICATION;  
ALTER TABLE bar ENABLE REPLICATION;  
COMMIT;
```

Remember, this will cause a full sync to occur. This should be done during a scheduled maintenance period.



Useful Functions

The following functions can be optionally enabled by installing plpgsql and applying the replicator-functions.sql. The functions must be executed by a PostgreSQL superuser. Depending on install the file will be located either in \$PGDATA/share/postgresql or /usr/share/postgresql .

is_replicated(REGCLASS)

Takes name of relation, returns boolean if relation is replicated

get_replication_status(REGCLASS)

Takes name of relation, returns which slaves, if any the relation is replicated to.

enable_replication(NAME, INTEGER[])

Takes name of relation and array of slaves that the relation should be replicated to.

enable_replication_by_schema(NAME, INTEGER[])

Takes name of schema and array of slaves that the relation should be replicated to.

enable_replication_global(INTEGER[])

Takes array of slave numbers to enable replication to.

disable_replication(REGCLASS)

Takes name of relation and disables replication.

disable_replication_by_schema(NAME)

Takes name of schema and disables replication for all relations within the schema.

disable_replication_global()

Disables all replication.



Getting support

Support is available 8am – 5pm, Monday – Friday, excluding federal holidays. You can contact support in the following manner:

By Phone: 503-667-4564

By Fax: 503-210-0334

By Email: support@commandprompt.com

By Web: <http://www.commandprompt.com>

24x7 support is available on a contract basis for as little as 2995.00 per year.

Before support can be delivered a signed services agreement must be on file with Command Prompt. The services agreement can be found here:

<http://www.commandprompt.com/support>

Just sign and fax back to us at 503-210-0334.