No	概要	対象	ユーザ	コマンド	確認（確認コマンド等）	備考
1	PostgreSQLの起動	既存PostgreSQLサーバ	postgres	$ pg_ctl start server starting.	$ pg_ctl status pg_ctl: server is running (PID: 23121) /usr/local/pgsql/bin/postgres	

2	PostgreSQLのバージョン確認	既存PostgreSQLサーバ	postgres	$ psql
				psql (9.2.4)
				Type "help" for help.
				
				postgres=# select version();
				version
				---------- 	
				PostgreSQL 9.2.4 on x86_64-unknown-linux-gnu, compiled by gcc (GCC) 4.4.7 20120313 (Red Hat 4.4.7-3), 64-bit
				(1 row)
					-	

3	JdbcRunnerのインストール	任意のサーバ	任意	JdbcRunnerのインストールは、「別紙_01_付帯ツールのインストール手順」を参照して下さい。	-	
4	検証用データベースの作成	既存PostgreSQLサーバ	postgres	$ createdb tpcc	$ psql -l List of databases Name | Owner | Encoding | Collate | Ctype | Access privileges ----------+----------+----------+-------------+-------------+----------------------- postgres | postgres | UTF8 | C | C | template0 | postgres | UTF8 | C | C | =c/postgres + | | | | | postgres=CTc/postgres template1 | postgres | UTF8 | C | C | =c/postgres + | | | | | postgres=CTc/postgres tpcc | postgres | UTF8 | C | C | (4 rows)	
5	検証用データの投入	任意のサーバ	任意	$ export CLASSPATH=$CLASSPATH:*JdbcRunnerを展開したディレクトリ*/jdbcrunner-1.2.jar $ java JR *JdbcRunnerを展開したディレクトリ*/scripts/tpcc_load.js -jdbcDriver org.postgresql.Driver -jdbcUrl jdbc:postgresql://*PostgreSQLサーバのIPアドレス*:5432/tpcc -jdbcPass postgres -jdbcUser postgres -logDir *任意のログ出力ディレクトリ*	$ psql tpcc tpcc=# ¥d List of relations Schema | Name | Type | Owner --------+-----------+-------+---------- public | customer | table | postgres public | district | table | postgres public | history | table | postgres public | item | table | postgres public | new_orders | table | postgres public | order_line | table | postgres public | orders | table | postgres public | stock | table | postgres public | warehouse | table | postgres (9 rows)	

6	検証用データの件数確認	既存PostgreSQLサーバ	postgres	検証用データの入力確認のため、以下のSQL文を実行します。
				
				$ psql tpcc
				
				tpcc=# select count(*) from customer;
				tpcc=# select count(*) from district;
				tpcc=# select count(*) from history;
				tpcc=# select count(*) from item;
				tpcc=# select count(*) from new_orders;
				tpcc=# select count(*) from order_line;
				tpcc=# select count(*) from orders;
				tpcc=# select count(*) from stock;
				Tpcc=# select count(*) from warehouse;
					count
					-------- 	
					480000
					(1 row)
					
					count
					------- 	
					160
					(1 row)
					
					count
					-------- 	
					480000
					(1 row)
					
					count
					-------- 	
					100000
					(1 row)
					
					count
					-------- 	
					144000
					(1 row)
					
					count
					--------- 	
					4798495
					(1 row)
					
					count
					-------- 	
					480000
					(1 row)
					
					count
					--------- 	
					1600000
					(1 row)
					
					count
					------- 	
					16
					(1 row)	

7	PostgreSQL 9.3.2のインストール	新規PostgreSQLサーバ	-	PostgreSQLのインストールは、「別紙_00_PostgreSQLインストール手順」を参照して下さい。	-	
8	新規PostgreSQLの起動	新規PostgreSQLサーバ	postgres	$ pg_ctl start server starting.	$ pg_ctl status pg_ctl: server is running (PID: 23121) /usr/local/pgsql/bin/postgres	

9	新規PostgreSQLのバージョン確認	新規PostgreSQLサーバ	postgres	$ psql
				psql (9.3.2)
				Type "help" for help.
				
				postgres=# select version();
				version
				---------- 	
				PostgreSQL 9.3.2 on x86_64-unknown-linux-gnu, compiled by gcc (GCC) 4.4.7 20120313 (Red Hat 4.4.7-3), 64-bit
				(1 row)
					-	

10	Slony-Iのインストール	既存PostgreSQLサーバ 新規PostgreSQLサーバ	-	Slony-Iのインストールは、「別紙_01_付帯ツールのインストール手順」を参照してください。		
11	スキーマの移行	既存PostgreSQLサーバ	postgres	$ pg_dump -C -s tpcc | psql -h［新規PostgreSQLサーバのIPアドレス］ SET (中略) GRANT DDLコマンドによるデータ変更は、自動的にレプリケーションできません。 事前に新規PostgreSQLサーバにtpccデータベースのスキーマをコピーしておきます。	新規PostgreSQLでtpccデータベースのテーブルを確認します。 $ psql tpcc tpcc=# ¥d List of relations Schema | Name | Type | Owner --------+-----------+-------+---------- public | customer | table | postgres public | district | table | postgres public | history | table | postgres public | item | table | postgres public | new_orders | table | postgres public | order_line | table | postgres public | orders | table | postgres public | stock | table | postgres public | warehouse | table | postgres (9 rows)	

12	レプリケーション不可テーブルを確認	既存PostgreSQLサーバ	postgres	slony-Iは、PRIMARY KEYが付いていないテーブルをレプリケーションできません。 PRIMARY KEYが付いていないテーブルを確認します。 $ psql tpcc tpcc=# select relname as table_name from pg_stat_user_tables where relname not in (select distinct table_name from information_schema.table_constraints where constraint_type='PRIMARY KEY'); table_name ------------ history (1 row) historyテーブルはレプリケーションできないため, レプリケーション完了後, pg_dump/pg_restoreを用いてデータを移行します。	-	
13	Slony-Iの初期設定スクリプトの実行	新規PostgreSQLサーバ	postgres	シート「A_1_setup.sh」を参考に任意のディレクトリにsetup.shを作成します。 $ sh setup.sh (エラーが出力されないこと)	setup.shで指定した「CLUSTERNAME」に_(アンダーバー)が付与された slony-I用のスキーマが作成されていることを確認します。 $ psql tpcc pcc=# ¥dn List of schemas Name	Owner ---------+---------- _slony_cluster	postgres public	postgres (2 rows)	
14	Slony-I(マスタ)の起動	既存PostgreSQLサーバ	postgres	$ slon slony_cluster "dbname=tpcc user=postgres host=localhost" 2014-04-01 16:45:02 JST CONFIG main: slon version 2.2.2 starting up 2014-04-01 16:45:02 JST INFO slon: watchdog process started (省略) 2014-04-01 16:45:02 JST CONFIG storeListen: li_origin=2 li_receiver=1 li_provider=2 2014-04-01 16:45:02 JST CONFIG remoteWorkerThread_2: update provider configuration 標準出力にSlony-Iのログが出力されます。	$ ps aux	grep slon postgres 8514 0.0 0.0 8580 784 pts/3 S+ 16:45 0:00 slon slony_cluster dbname=tpcc user=postgres host=localhost postgres 8515 0.0 0.1 475608 3440 pts/3 Sl+ 16:45 0:00 slon slony_cluster dbname=tpcc user=postgres host=localhost root 8557 0.0 0.0 107468 936 pts/8 S+ 16:47 0:00 grep slon	
15	Slony-I(スレーブ)の起動	新規PostgreSQLサーバ	postgres	$ slon slony_cluster "dbname=tpcc user=postgres host=localhost" 2014-04-01 16:50:41 JST CONFIG main: slon version 2.2.2 starting up 2014-04-01 16:50:41 JST INFO slon: watchdog process starte (省略) 2014-04-01 16:51:55 JST INFO remoteWorkerThread_1: SYNC 5000000068 done in 0.005 seconds 2014-04-01 16:51:57 JST INFO remoteWorkerThread_1: SYNC 5000000069 done in 0.005 seconds 標準出力にSlony-Iのログが出力されます。	$ ps aux	grep slon postgres 1058 0.0 0.0 8584 788 pts/4 S+ 16:50 0:00 slon slony_cluster dbname=tpcc user=postgres host=localhost postgres 1059 0.0 0.1 475612 3492 pts/4 Sl+ 16:50 0:00 slon slony_cluster dbname=tpcc user=postgres host=localhost postgres 1135 0.0 0.0 107464 920 pts/1 S+ 16:52 0:00 grep slon	
16	レプリケーションの開始	既存PostgreSQLサーバ	postgres	シート「A_2_subscribe.sh」を参考に任意のディレクトリにsubscribe.shを作成します。 $ sh subscribe.sh (エラーが出力されないこと)	-	
17	レプリケーションの確認	新規PostgreSQLサーバ	postgres	Slony-I(スレーブ)のログに以下メッセージが出力されていることを確認します。 (Slony-I(スレーブ)のログ抜粋) 2014-04-01 16:59:49 JST CONFIG version for "dbname=tpcc host=10.36.239.24 user=postgres" is 90204 2014-04-01 16:59:49 JST INFO remoteWorkerThread_1: syncing set 1 with 8 table(s) from provider 1 2014-04-01 16:59:49 JST INFO remoteWorkerThread_1: SYNC 5000000104 done in 0.046 seconds 上記メッセージが出力されるまで、しばらく時間がかかります。※1	※1 レプリケーション開始の際に、スレーブはテーブルのデータを削除し、(TRUNCATE), マスタのデータをコピー(COPY)します。 (Slony-I(スレーブ)のログ抜粋) 2014-04-01 16:57:16 JST CONFIG remoteWorkerThread_1: Begin COPY of table "public"."customer" NOTICE: truncate of "public"."customer" failed - doing delete 2014-04-01 16:57:41 JST CONFIG remoteWorkerThread_1: 276918283 bytes copied for table "public"."customer" 2014-04-01 16:57:47 JST CONFIG remoteWorkerThread_1: 31.772 seconds to copy table "public"."customer"	
18	ファイルオーバの実行	既存PostgreSQLサーバ	postgres	Slony-Iのスレーブとマスタを入れ替えます。(フェイルオーバ) フェイルオーバ後, 新規PostgreSQLで更新処理が実行可能になります。 シート「A_3_failover.sh」を参考に任意のディレクトリにfailover.shを作成します。 $ date ; sh failover.sh ;date 2014年 4月 1日 火曜日 17:34:21 JST executing preFailover(1,1) on 2 NOTICE: executing "_slony_cluster".failedNode2 on node 2 <stdin>:-4: NOTICE: calling restart node 1 NOTICE: executing "_slony_cluster".failedNode3 on node 2 debug: waiting for 1_5000000213 on 2 2014年 4月 1日 火曜日 17:34:43 JST ※フェイルオーバ以降は, 既存PostgreSQLでのデータ更新は, 新規PostgreSQLに反映されないため, 注意が必要です。	新規PostgreSQLで更新処理が実行可能なことを確認します。 $ psql tpcc tpcc=# insert into warehouse values (19); INSERT 0 1 tpcc=# delete from warehouse where w_id= 19; DELETE 1	
19	VACUUMの実施	新規PostgreSQLサーバ	postgres	$ psql tpcc =# ¥timing Timing is on. =# VACUUM VERBOSE; INFO: vacuuming "pg_catalog.pg_statistic" INFO: index "pg_statistic_relid_att_inh_index" now contains 422 row versions in 4 pages DETAIL: 0 index row versions were removed. 0 index pages have been deleted, 0 are currently reusable. CPU 0.00s/0.00u sec elapsed 0.00 sec. INFO: "pg_statistic": found 0 removable, 422 nonremovable row versions in 18 out of 18 pages (中略) VACUUM Time: 45253.710 ms	-	
20	ANALYZEの実施	新規PostgreSQLサーバ	postgres	$ psql tpcc =# ¥timing Timing is on. =# ANALYZE VERBOSE; INFO: analyzing "pg_catalog.pg_type" INFO: "pg_type": scanned 8 of 8 pages, containing 352 live rows and 0 dead rows; 352 rows in sample, 352 estimated total rows INFO: analyzing "public.history" INFO: "history": scanned 5180 of 5180 pages, containing 480000 live rows and 0 dead rows; 30000 rows in sample, 480000 estimated total rows INFO: analyzing "public.item" (中略) ANALYZE Time: 4976.217 ms	-	
21	Slony-I(新マスタ)の停止	新規PostgreSQLサーバ	任意	slony-Iを起動したコンソール上で(Ctrl+C)を押下 2014-04-01 17:36:26 JST INFO cleanupThread: 0.265 seconds for cleanupEvent() **(Ctrl+C)を押下** NOTICE: Slony-I: cleanupEvent(): Single node - deleting events < 5000000273 NOTICE: Slony-I: log switch to sl_log_2 complete - truncate sl_log_1 CONTEXT: PL/pgSQL function _slony_cluster.cleanupevent(interval) line 95 at assignment 2014-04-01 17:36:26 JST INFO cleanupThread: 0.265 seconds for cleanupEvent() ^C2014-04-01 17:40:33 JST CONFIG slon: child terminated signal: 9; pid: 1490, current worker pid: 1490 2014-04-01 17:40:33 JST INFO slon: done 2014-04-01 17:40:33 JST INFO slon: exit(0)	$ ps aux	grep slon postgres 1805 0.0 0.0 107464 924 pts/4 S+ 17:40 0:00 grep slon	
22	Slony-I(旧マスタ)の停止	既存PostgreSQLサーバ	任意	slony-Iを起動したコンソール上で(Ctrl+C)を押下 2014-04-01 17:43:36 JST FATAL main: Node is not initialized properly - sleep 10s **(Ctrl+C)を押下** 2014-04-01 17:43:36 JST ERROR cannot get sl_local_node_id - ERROR: schema "_slony_cluster" does not exist フェイルオーバ時に既存PostgreSQLからslony-I用のスキーマが削除されるため, 上記のメッセージが出力されます。	-	
23	pg_dumpの実行	新規PostgreSQLサーバ	postgres	slony-Iは、PRIMARY KEYが付いていないテーブルをレプリケーションできないため, historyテーブルのデータを既存PostgreSQLからバックアップします。 $ date; pg_dump -h [既存PostgreSQLのIPアドレス] -a -Fc --verbose -t history tpcc > /tmp/[バックアップファイル名] ; date	$ date; pg_dump -h 10.36.239.24 -a -Fc --verbose -t history tpcc > /tmp/history.sql ; date 2014年 4月 1日 火曜日 18:50:22 JST pg_dump: reading schema (省略) pg_dump: dumping contents of table history 2014年 4月 1日 火曜日 18:50:25 JST	

24	pg_restoreの実行	新規PostgreSQLサーバ	postgres	$ date ; pg_restore --verbose -d tpcc /tmp/*バックアップファイル名* ; date	$ date ; pg_restore --verbose -d tpcc /tmp/history.sql ;date 2014年 4月 1日 火曜日 18:51:11 JST pg_restore: connecting to database for restore pg_restore: implied data-only restore (省略) pg_restore: setting owner and privileges for TABLE DATA history 2014年 4月 1日 火曜日 18:51:41 JST	

25	検証用データの件数確認	新規PostgreSQLサーバ	postgres	各テーブルの件数を確認します。
				
				$ psql tpcc
				
				tpcc=# select count(*) from customer;
				tpcc=# select count(*) from district;
				tpcc=# select count(*) from history;
				tpcc=# select count(*) from item;
				tpcc=# select count(*) from new_orders;
				tpcc=# select count(*) from order_line;
				tpcc=# select count(*) from orders;
				tpcc=# select count(*) from stock;
				tpcc=# select count(*) from warehouse;
					count
					-------- 	
					480000
					(1 row)
					
					count
					------- 	
					160
					(1 row)
					
					count
					-------- 	
					480000
					(1 row)
					
					count
					-------- 	
					100000
					(1 row)
					
					count
					------- 	
					144000
					(1 row)
					
					count
					-------- 	
					4798495
					(1 row)
					
					count
					------- 	
					480000
					(1 row)
					
					count
					--------- 	
					1600000
					(1 row)
					
					count
					------- 	
					16
					(1 row)	

26	JdbcRunnerの実行	任意のサーバ	任意	$ export CLASSPATH=$CLASSPATH:*JdbcRunnerを展開したディレクトリ*/jdbcrunner-1.2.jar
				
				$ java JR *JdbcRunnerを展開したディレクトリ*/scripts/tpcc.js -jdbcDriver org.postgresql.Driver -jdbcUrl jdbc:postgresql://*PostgreSQLサーバのIPアドレス*:5432/tpcc -jdbcPass postgres -jdbcUser postgres -logDir *任意のログ出力ディレクトリ*
					# java JR /usr/local/src/jdbcrunner-1.2/scripts/tpcc.js -jdbcDriver org.postgresql.Driver -jdbcUrl jdbc:postgresql://localhost:5432/tpcc -jdbcPass postgres -jdbcUser postgres -logDir /tmp
					13:48:29 [INFO] > JdbcRunner 1.2
					13:48:29 [INFO] [Config]
					Program start time : 20140220-134828
					Script filename : /usr/local/src/jdbcrunner-1.2/scripts/tpcc.js
					JDBC driver : org.postgresql.Driver
					JDBC URL : jdbc:postgresql://localhost:5432/tpcc
					JDBC user : postgres
					Warmup time : 0 sec
					Measurement time : 60 sec
					Number of tx types : 5
					Number of agents : 16
					Connection pool size : 16
					Statement cache size : 40
					Auto commit : false
					Sleep time : 0,0,0,0,0 msec
					Throttle : - tps (total)
					Debug mode : false
					Trace mode : false
					Log directory : /tmp
					Parameter 0 : 0
					Parameter 1 : 0
					Parameter 2 : 0
					Parameter 3 : 0
					Parameter 4 : 0
					Parameter 5 : 0
					Parameter 6 : 0
					Parameter 7 : 0
					Parameter 8 : 0
					Parameter 9 : 0
					13:48:30 [INFO] Tiny TPC-C 1.1.25,60,62,62 tx
					13:48:52 [INFO] [Progress] 21 sec, 68,61,9,6,4 tps, 677,686,69,68,66 tx
					(中略)
					13:49:32 [INFO] [Response time (99%tile)] 1122,1045,285,1168,609 msec
					13:49:32 [INFO] [Response time (maximum)] 1476,1548,303,1245,1088 msec
					13:49:32 [INFO] < JdbcRunner SUCCESS
						デフォルトの900秒では長いため、60秒で処理が終わるように設定を以下のように変更する。
						# vi /usr/local/jdbcrunner-1.2/scripts/tpcc.js
						var warmupTime = 0;
						var measurementTime = 60;

27	既存PostgreSQLの停止	既存PostgreSQLサーバ	postgres	$ pg_ctl stop waiting for server to shut down....LOG: received smart shutdown request LOG: autovacuum launcher shutting down LOG: shutting down LOG: database system is shut down done server stopped	$ pg_ctl status pg_ctl: no server running	

28	slony-I用のスキーマを削除	新規PostgreSQLサーバ	postgres	$ psql tpcc
				
				tpcc=# DROP SCHEMA _slony_cluster CASCADE;
				(省略)
				DROP SCHEMA
					tpcc=# ¥dn
					List of schemas
					Name
						Owner
						------+----------
						public
							postgres
							(1 row)	

29	既存PostgreSQLのアンインストール	既存PostgreSQLサーバ	root	# cd /usr/local/src/postgresql-9.2.4 # make uninstall	# su - postgres $ which psql /usr/bin/which: no psql in (/usr/local/pgsql/bin:/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/zookeeper-3.4.5/bin:/usr/local/storm/bin:/home/postgres/bin)	

A.1 setup.sh

参考URL:http://www.slony.info/documentation/2.2/tutorial.html#FIRSTDB
(*赤字下線*部分は、上記URLのサンプルからの変更点)

```
#!/bin/sh

CLUSTERNAME=slony_cluster
MASTERDBNAME=tpcc
SLAVEDBNAME=tpcc
MASTERHOST=[親側PostgreSQLサーバのIPアドレス]
SLAVEHOST=[子側PostgreSQLサーバのIPアドレス]
REPLICATIONUSER=postgres
    #--
/usr/local/slony1/bin/slonik <<_EOF_
    #--
  # define the namespace the replication system
  # uses in our example it is slony_example
  #--
  cluster name = $CLUSTERNAME;
  # admin conninfo's are used by slonik to connect to
  #--
  # admin conninfo's are used by slonik to connect to
  # the nodes one for eachnode on each side of the cluster,
  # the syntax is that of PQconnectdb in
  # the C-API
  # --
  node 1 admin conninfo = 'dbname=$MASTERDBNAME ¥
        host=$MASTERHOST user=$REPLICATIONUSER';
  node 2 admin conninfo = 'dbname=$SLAVEDBNAME ¥
        host=$SLAVEHOST user=$REPLICATIONUSER';

  #--
  # init the first node.  Its id MUST be 1.  This creates
  # the schema _$CLUSTERNAME containing all replication
  # system specific database objects.
  #--
  init cluster ( id=1, comment = 'Master Node');

  #--
  # Slony-I organizes tables into sets.  The smallest unit
  # a node can subscribe is a set. The master or origin of
  # the set is node 1.
  #--
  create set (id=1, origin=1, comment='All tpcc tables');
  set add table (set id=1, origin=1, id=1,
                fully qualified name = 'public.customer',
                comment='customer table');
  set add table (set id=1, origin = 1, id = 2,
                fully qualified name = 'public.district',
                comment = 'district table');
  set add table (set id=1, origin = 1, id = 4,
                fully qualified name = 'public.item',
                comment = 'item table');
  set add table (set id=1, origin = 1, id = 5,
                fully qualified name = 'public.new_orders',
                comment = 'new_orderw table');
  set add table (set id=1, origin = 1, id = 6,
                fully qualified name = 'public.order_line',
                comment = 'order_line table');
  set add table (set id=1, origin = 1, id = 7,
                fully qualified name = 'public.orders',
                comment = 'orders table');
  set add table (set id=1, origin = 1, id = 8,
                fully qualified name = 'public.stock',
                comment = 'stock orders');
  set add table (set id=1, origin = 1, id = 9,
                fully qualified name = 'public.warehouse',
                comment = 'warehouse table');

  #--
  # Create the second node (the slave) tell the 2 nodes how
  # to connect to each other and how they should listen for events.
  #--

  store node (id=2, comment = 'Slave Node', event node=1);
  store path (server = 1, client = 2, conninfo='dbname=$MASTERDBNAME ¥
        host=$MASTERHOST user=$REPLICATIONUSER');
  store path (server = 2, client = 1, conninfo='dbname=$SLAVEDBNAME ¥
        host=$SLAVEHOST user=$REPLICATIONUSER');
_EOF_
```

A_2_subscribe.sh

参考URL:http://www.slony.info/documentation/2.2/tutorial.html#FIRSTDB
(*赤字下線*部分は、上記URLのサンプルからの変更点)

```
#!/bin/sh

CLUSTERNAME=slony_cluster
MASTERDBNAME=tpcc
SLAVEDBNAME=tpcc
MASTERHOST=[親側PostgreSQLサーバのIPアドレス]
SLAVEHOST=[子側PostgreSQLサーバのIPアドレス]
REPLICATIONUSER=postgres

/usr/local/slony1/bin/slonik <<_EOF_
    # ----
    # This defines which namespace the replication system uses
    # ----
    cluster name = $CLUSTERNAME;

    # ----
    # Admin conninfo's are used by the slonik program to connect
    # to the node databases.  So these are the PQconnectdb arguments
    # that connect from the administrators workstation (where
    # slonik is executed).
    # ----
    node 1 admin conninfo = 'dbname=$MASTERDBNAME host=$MASTERHOST ¥
                             user=$REPLICATIONUSER';
    node 2 admin conninfo = 'dbname=$SLAVEDBNAME host=$SLAVEHOST ¥
                             user=$REPLICATIONUSER';

    # ----
    # Node 2 subscribes set 1
    # ----
    subscribe set ( id = 1, provider = 1, receiver = 2, forward = yes);
_EOF_
```

A_3_failover.sh

参考URL:http://www.slony.info/documentation/2.2/tutorial.html#FIRSTDB
(*赤字下線*部分は、上記URLのサンプルからの変更点)

```
#!/bin/sh

CLUSTERNAME=slony_cluster
MASTERDBNAME=tpcc
SLAVEDBNAME=tpcc
MASTERHOST=[現行PostgreSQLサーバのIPアドレス]
SLAVEHOST=[新規PostgreSQLサーバのIPアドレス]
REPLICATIONUSER=postgres

# execute slonik command
/usr/local/slony1/bin/slonik <<_EOF_
cluster name = $CLUSTERNAME;
node 1 admin conninfo = 'dbname=$MASTERDBNAME host=$MASTERHOST ¥
                        user=$REPLICATIONUSER;
node 2 admin conninfo = 'dbname=$SLAVEDBNAME host=$SLAVEHOST ¥
                        user=$REPLICATIONUSER;
failover (id = 1, backup node = 2);
drop node (id = 1, event node = 2);
_EOF_

exit 0;
```