# Creating Replication Nodes From Transactional Backups

## Introduction

This document describes a process for creating database replication nodes from a transactional backup of existing replicated databases. Let's assume that your database replication cluster is humming along nicely, but you find that you need to add a slave node to it in order to handle increased load on your application.

### Creating and Loading the Backup

Keeping the master and all of the existing slaves up and running during this process is important. Just as important is ensuring that you have a **transactional** backup of either the master database or one of the slaves. Check out the PostgreSQL documentation for On-line backup and point-in-time-recovery.

For the sake of argument, let's say that your full database backup is stored in the file `database-backup.sql`. After the backup file has been created, import it into the new slave database. For example.

```
psql new_slave_db -f database-backup.sql
```

This will create a replica of the database on your new slave.

### Provisioning the New Slave

Once you have the transactional backup restored on your new slave, you'll need to provision it in the Bruce configuration database. You'll use the admin tool for this. First, you'll need to create an XML file that contains metadata about your new slave. The format is very straightforward.

```
<!DOCTYPE dataset SYSTEM "dataset.dtd">
<dataset>

    <!-- Add the node to the bruce.yf_node
table -->
    <table name="bruce.yf_node">
        <column>id</column>
        <column>name</column>
        <column>uri</column>
        <column>includetable</column>
        <row>
            <value>10</value> <!-- ID for
the new node -->
            <value>Restored From
```

```
Backup</value> <!-- Name for new node -->

<value>jdbc:postgresql://localhost:5432/new_
slave_node?user=bruce</value> <!-- JDBC URL
-->

<value>replication_test\.replicate_.+</value
> <!-- Regex for tables to replicate -->
        </row>
    </table>

    <!-- Add the node to the cluster with ID
of 1000 -->
    <table name="bruce.node_cluster">
        <column>node_id</column>
        <column>cluster_id</column>
        <row>
            <value>10</value>
            <value>1000</value>
```

```
        </row>
    </table>
</dataset>
```

This XML document contains metadata for a single node. The format is self-explanatory. In this example, we are providing and ID, name, JDBC url, and a regular expression used to determine which tables to replicated. Then we add that new node to the cluster mapping table, `bruce.node_cluster`.

Now that you have the database configured and the node metadata in an XML file, the only other step is to use the admin tool to provision the new node.

```
# cd $BRUCE_HOME/bin
# ./admin.sh -data newnode.xml -initsnapshots NONE -operation INSERT -url
jdbc:postgresql://localhost:5432/bruce_config?user=bruce
```

Some notes on the command line options.

- `-data newnode.xml` - This is the XML file that contains the node metadata. It will be loaded into the bruce configuration database.
- `-initsnapshots NONE` - This tells the admin tool how the slave's snapshot status table should be initialized. **IMPORTANT:** If the transactional backup is from a slave node, you should specify `NONE` for this parameter. Since the original slave already has the most recent data in its snapshot status table. If the transactional backup is from a master node, however, use the `SLAVE` option. (Yes the `SLAVE` option). This causes the admin tool to assume the new node used to be a master, so the snapshot status is set based on the current state of the data in the new node. Confused yet? Good!
- `-operation INSERT` - Tells the admin tool to insert the new node data (as opposed to update, delete, or a clean insert where all other data is wiped out).
- `-url jdbc:postgresql://localhost:5432/bruce_config?user=bruce` - This URL should point to the configuration database that you used when you setup replication.

## Fire it Up

Now that you've backed up, restored and provisioned, the replication daemon needs to be restarted. Don't worry. It will pick up any changes that occurred in the master database while it was down, once it is back up again.

```
# cd $BRUCE_HOME/bin
# ./shutdown.sh
# ./startup.sh MyCluster
```

The `shutdown.sh` command can take a long time to complete. The prompt will return immediately, but you should wait until the log indicates that all threads have been stopped.