# Running the DCDB Migration

The following approach can be used to migrate the DCDB to SOLA using a backup of the DCDB database and the open source ETL tool Pentaho Kettle.

## Preparation

Before performing the migration it is necessary to prepare the computer that will perform the migration with the necessary software, being MS SQL Server, Pentaho Kettle and PostgreSQL with the PostGIS extension.

### Download/Install MS SQL Server

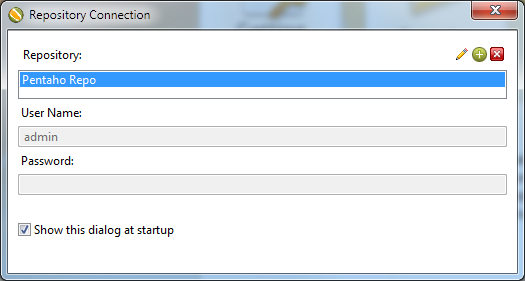
1. If the computer to use for the migration does not have MS SQL Server 2005 or later, you will need to download and install a SQL Server Express edition. The latest version available is MS SQL Server 2012 (<http://www.microsoft.com/sqlserver/en/us/editions/2012-editions/express.aspx>). You will need to download and install the “Express with Tools” package, but be aware this installation package is 600+ Mb.
2. During installation, choose the option for SQL Server Authentication and create a password for the sa account. Other than that you can customize the installation as you see fit.
3. Using SQL Server Management Studio, connect to the new Database Server and create a New database called **DCDB.**

### Download/Install PostgreSQL

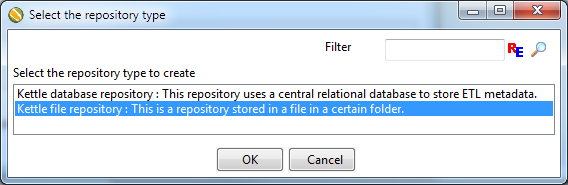
1. Instructions for installing PostgreSQL can be found in the SOLA Exercise Guide documentation. Follow those instructions to install a SOLA compatible PostgreSQL instance. It will also be necessary to create a base SOLA database (i.e. one that does not contain any test data) in preparation for the migration.

### Download/Install Pentaho Kettle

1. Download the latest stable release of Pentaho Kettle from <http://kettle.pentaho.com/> (approx 250Mb). For Windows download the .zip package. Note that Pentaho runs on Java, so you will need to have the Java 6 JRE or the Java 7 JRE installed on your computer as well.
2. Extract the zip to an appropriate location on your hard drive.
3. Start Pentaho by running the **.../Pentaho/data-integration/Spoon.bat** command file. The first time you run Pentaho you will need to set up a repository for it to use. The Pentaho UI is not overly intuitive. To create a repository, click the green **+** icon in the top right.



1. Select the **Kettle file repository** option and click OK and fill out the repo details accordingly.

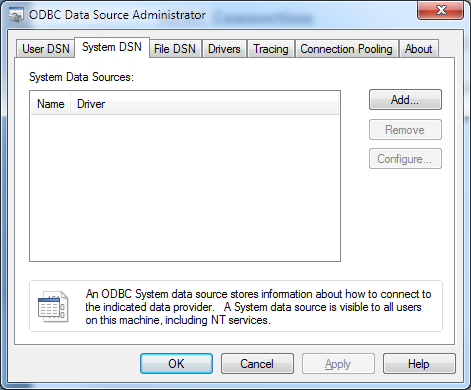


## Configuration

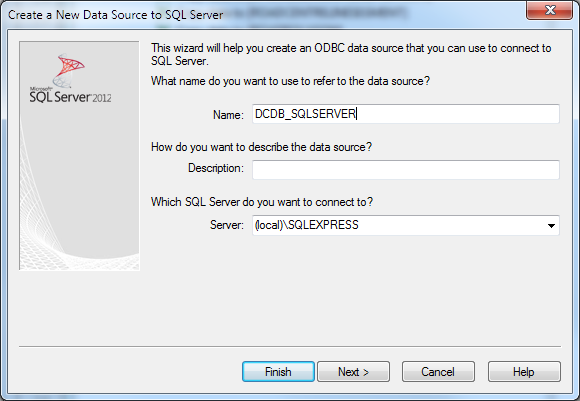
### ODBC Connection

Pentaho cannot connect natively to MS SQL Server so it is necessary to create an ODBC connection. Before performing this step, make sure you have a database called DCDB in your SQL Server instance.

1. Open the ODBC Data Source Administrator (as Administrator) through the Control Panel or searching for odbc from the Start Menu



1. On the System DSN tab, choose **Add…**
2. Select **SQL Server Native Client** **<version>** as the Driver. Note that the version number will be dependent on the version of SQL Server you have installed. Click **Finish** to open the Create New Data Source dialog.
3. Enter **DCDB\_SQLSERVER** as the Name for the datasource. This is the datasource configured in the Pentaho DCDB Migration job. Also enter name of the server to connect to. Be aware the drop down list will not list the SQLEXPRESS instance. If the name of our SQL Server instance is SQLEXPRESS, then use **(local)\SQLEXPRESS** as your Server name as shown below. Click **Next >**



1. Choose **With SQL Server authentication…** and enter the sa login credentials. **Click Next >**
2. Set the default database to **DCDB** and click **Next >** then click **Finish**.
3. On the MSSQL Server Setup dialog, test the data source. You should get a TESTS COMPLETED SUCCESSFULLY message. If not, check the configuration and/or restart the ODBC Data Source Administrator and try again.

### Install the Base64 Encoding CLR Functions

Pentaho is not able to read the binary/blob data used by SQL Server. This is an issue as the geometry data is effectively binary/blob data. To get around this issue, the binary data can be base64 encoded as string data for the purpose of migration. This requires some custom CLR functions to be installed into SQL Server because SQL Server does not provide a native base64 encoding function.

1. Copy the **base64.cs** file from the …\database\migration\dcdb folder to the C:\temp folder. This file contains C# functions that can be loaded into SQL Server as CLR functions to base64 encode and decode binary data and strings.
2. Open a command window and execute the following command. This will compile the source code file into a dll using the .NET framework compiler (csc). This command assumes you have .NET 4.0 on our computer and that you are using SQL Server 2012. If do not have .NET 4.0, or you are using a different SQL Server version, you will need to use a different framework version. Check your …\Framework\ folder and execute the appropriate csc version. For SQL Server 2005, use .NET 2.0 and for SQL Server 2008, use .NET 3.0 or .NET 3.5.

|  |
| --- |
| **C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc /t:library /out:C:\temp\base64.dll C:\temp\base64.cs** |

1. Using SQL Server Management Studio, run the following SQL Script on the DCDB database

|  |
| --- |
| -- Configure the database to use CLR defined functions and procedures  sp\_configure 'clr enabled'**,** 1  reconfigure  **GO**  -- If using SQL Server 2005, uncomment the following line  -- EXEC sp\_dbcmptlevel 'DCDB', 90  -- If using SQL Server 2008, uncomment the following line  -- EXEC sp\_dbcmptlevel 'DCDB', 100  -- Drop the functions if they already exist  **IF** **OBJECT\_ID** **(**N'dbo.strToBase64'**,** N'FS'**)** **IS** **NOT** **NULL**  **DROP** **FUNCTION** dbo**.**strToBase64**;**  **GO**  **IF** **OBJECT\_ID** **(**N'dbo.strFromBase64'**,** N'FS'**)** **IS** **NOT** **NULL**  **DROP** **FUNCTION** dbo**.**strFromBase64**;**  **GO**  **IF** **OBJECT\_ID** **(**N'dbo.binToBase64'**,** N'FS'**)** **IS** **NOT** **NULL**  **DROP** **FUNCTION** dbo**.**binToBase64**;**  **GO**  **IF** **OBJECT\_ID** **(**N'dbo.binFromBase64'**,** N'FS'**)** **IS** **NOT** **NULL**  **BEGIN**  **DROP** **FUNCTION** dbo**.**binFromBase64  **DROP** ASSEMBLY Base64CLR**;**  **END**  **GO**  -- Create a reference to the assembly containing the base64 encode and  -- decode functions  **CREATE** ASSEMBLY Base64CLR **FROM** 'C:\temp\base64.dll' **WITH** PERMISSION\_SET **=** SAFE  **GO**  -- Create the CLR functions  **CREATE** **FUNCTION** strToBase64**(**@str NVARCHAR**(MAX))** **RETURNS** NVARCHAR**(MAX)**  **AS** **EXTERNAL** NAME base64CLR**.**base64Converter**.**strToBase64CLR  **GO**  **CREATE** **FUNCTION** binToBase64**(**@**bin** VARBINARY**(MAX))** **RETURNS** NVARCHAR**(MAX)**  **AS** **EXTERNAL** NAME base64CLR**.**base64Converter**.**binToBase64CLR  **GO**  **CREATE** **FUNCTION** strFromBase64**(**@str NVARCHAR**(MAX))** **RETURNS** NVARCHAR**(MAX)**  **AS** **EXTERNAL** NAME base64CLR**.**base64Converter**.**strFromBase64CLR  **GO**  **CREATE** **FUNCTION** binFromBase64**(**@str NVARCHAR**(MAX))** **RETURNS** VARBINARY**(MAX)**  **AS** **EXTERNAL** NAME base64CLR**.**base64Converter**.**binFromBase64CLR  **GO** |

## Running the Migration

### Obtain a DCDB Backup

The Migration can be run using a backup of the DCDB. Backups of DCDB are taken hourly and saved to the MNRE File Server. Prior to copying one of the backup files one of the MNRE IT Support team will need to run the **DCDBspatialBinaryExportver2.sql** on the DCDB. This script exports the spatial data to temporary tables in Well Known Binary (WKB) format which allows the spatial data to be imported into PostgreSQL.

### Restore the DCDB Backup

Use SQL Server Management Studio to restore the DCDB backup onto the migration computer. Once the restore is complete, you may also need to reapply the base64 encoding CLR functions. Refer to the **Install the Base64 Encoding CLR Functions** section for details on how to install the functions.

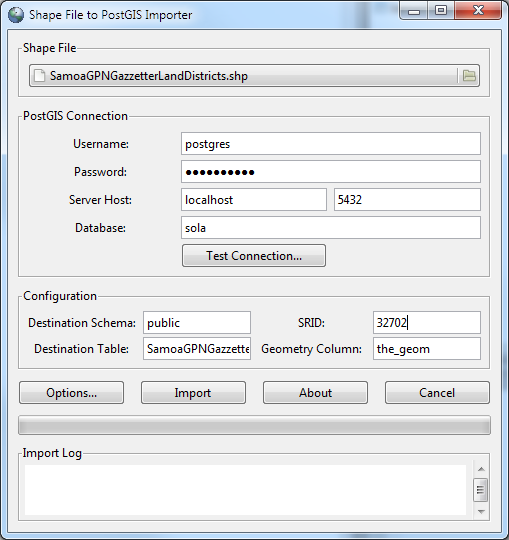
### Create the Target SOLA Database

Create the SOLA database that will be used as the target for the migration. The name of this database should be **sola**. Ideally use the .../database/create\_soladb\_samoa.bat script to create a SOLA database configured for the Samoan Pilot. If you need to use an existing SOLA database for the migration, run the …/database/extension/Samoa.sql script before performing the migration to install the Samoa configurations in the database.

### Load Static Shape Data

The Island, District and Village spatial data is loaded from static shape files. If the target SOLA database does not include the SamoaPlaceNames schema, load the static shape data as follows;

1. Unzip the …\database\migration\dcdb\static-shape-data.zip to the static-shape-data folder.
2. Using pgAdmin III, create a schema called **SamoaPlaceNames** in the target SOLA database.
3. Using pgAdmin III, launch the PostGIS Shape File Loader utility from the **Plugins > PostGIS Shapefile and DBF Loader** menu
4. In the Shape File section, browse to the location of the static-shape-file directory and choose one of the 3 SHP files.
5. Set the Database to the name of the target database (i.e. **sola**), the Destination Schema to **SamoaPlaceNames**, the SRID to **32702** and click the **Import** button.

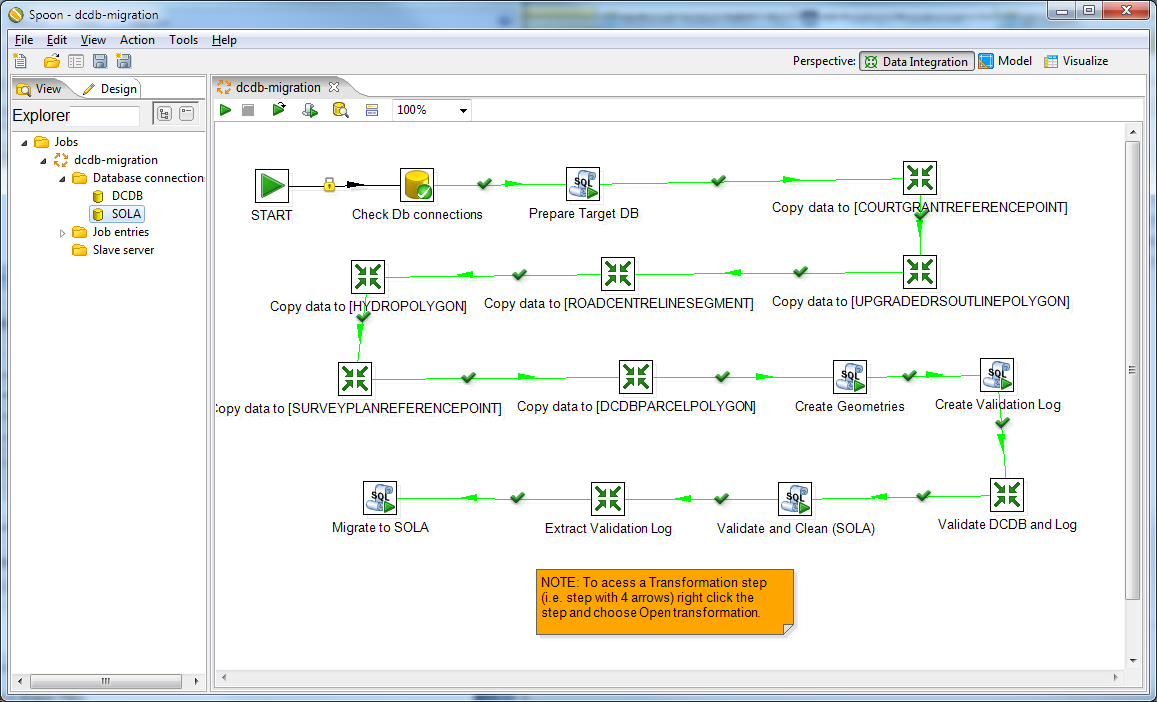


1. Repeat the Import for the remaining 2 shape files.

### Execute the DCDB Migration Job

Pentaho is used to transfer the data from the SQL Server database into the SOLA database. The transformation job also performs a number of validations that are exported to a file called dcdb\_migration\_log.txt in the C:\temp folder.

1. From Pentaho choose **File > Open…** and locate the …/database/migration/dcdb/pentaho-files/dcdb-migration.kjb file and Open the file. You should be represented with the dcdb-migration job as illustrated below.



1. The dcdb-migration job creates a staging schema in the target SOLA database called samoaview and moves each of the spatial tables from the DCDB into the target SOLA database. It then creates the PostGIS geometry for each spatial object before validating the migrated data and logging any issues as well as logging any records that did not get migrated from the DCDB. Once this validation is complete, the validation messages are written to the C:\temp\dcdb\_migration\_log.txt file before the data is moved from the staging samoaview schema into the cadastre schema tables.
2. Before running the migration you will probably need to reconfigure the database connections to match the passwords used on the migration computer. Right click the SOLA and DCDB nodes under the Database Connections node and Edit the details to match those on the migration computer.
3. Unfortunately Pentaho does not appear to share database connections with the transformation steps, so it is necessary to update the database details for each transformation step as well. Right click each transformation step (the steps with the 4 arrows icon) and choose **Open transformation**. Once in the tab for the transformation, update the database connection details under the Database connections node.
4. Now you should be ready to run the migration. Click the green arrow icon in the top left. The Job should take approx 1 minute.
5. If any errors occur, refer to the Logging tab and attempt to determine the cause of the error. If a transformation step fails, open the specific transformation step (i.e. Open transformation) and try running it independently. If it works, go back to the dcdb-migration job and rerun the job.

# To Refresh SamoaView Data in the SOLA Database (Original Approach)

1. Logon to a computer that is connected to MNRE04 server (host for SQL Server Database DCDB) that has SQL Server Management Studio software and some form of SQL Server to PostgreSQL ETL software (such as ESF Migration Toolkit) loaded
2. Run SQL Server Management Studio software and connect to SQL Server instance on MNRE04 using the SQL Server logon option with username/password of sa/Talofa01
3. Open SQL script DCDBspatialBinaryExportver2.sql to be found under Subversion/GitHub source control in the ..\trunk\database\Samoa\SQLserver folder and run it.
4. Using the ETL software migrate all the DCDB database tables to a staging schema in the SOLA database called SamoaView
5. In PostgreSQL pgAdmin check that there is a SamoaPlaceNames schema containing SamoaGPNGazzetterIslands, SamoaGPNGazzetterLandDistricts and SamoaVillageNamesPoint\_SMG. If not use the PostgreSQL shapefile loader (pgAdmin plugin )to load the shapefiles in ..\trunk\database\Samoa\SamoaStaticShapeFiles29July2010SRID32702 folder noting that the SRID value to be used for all the shapefiles is 32702.
6. In PostgreSQL pgAdmin run the SQL script sola\_populate\_samoa.sql in the ..\trunk\database\Samoa folder

**Future changes:**

Sola\_populate\_samoa.sql script should be extended to link the spatial point survey plan feature to the pdf image to be stored in the document table in the document schema

**Mapinfo Professional v11 reading of POSTGIS databases**

If you need to read (and potentially write) to POSTGIS databases using Mapinfo you need to add a mapinfo schema to the POSTGIS database and a mapinfo\_catalog table in that new schema. For each spatial table you need a row defining that table in the mapinfo\_catalog table. The cadastre\_object table can be viewed but so far no luck with the spatial\_unit. The spatial\_unit table is complicated in that there are two geometry columns and several features stored within that table.