# Settings

File WEB.Config, section applicationSettings/CfCServiceTester.Properties.Settings

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Value** | **Action** |
| LocalServersOnly | Boolean | true | Enumerate SQL servers installed on the same computer as WCF service |
|  |  | false | Enumerate all available SQL servers |
| AccessibleOnly | Boolean | true | Enumerate accessible databases only |
|  |  | false | Enumerate all databases on the SQL server |
| BackupDirectory | String |  | Directory where backup files are written. **This directory must be available for SQL server and service.** Map directory into SQL server’s file system and service’s file system under the same device letter if backup device, SQL server and CfC service are on separate computers. |

# Testing the service

Deployment package is prepared in file

~\CfCServiceTester\CfCServiceTester\obj\Debug\Package

Follow article

<http://go.microsoft.com/fwlink/?LinkId=124618>

for installing this package on your IIS. Solution is built for framework 3.5, it should work on .NET v4.x too. Launching it in .NET 2.0 will require Linq to be installed.

Archive contains ready for launching solution. You can open it with Visual Studio 2010. Test projects weren’t be used so Express Edition will work too.

Database maintenance software is located in directory WEBservice and consists of AJAX enabled WEB service. All other content is debugger and demo application only. Recompile WEB service as standalone application if you are going to call it from environments different from ASP.NET.

## Preparing database

Prepare database user with server roles:

1. Public,
2. Dbcreator,

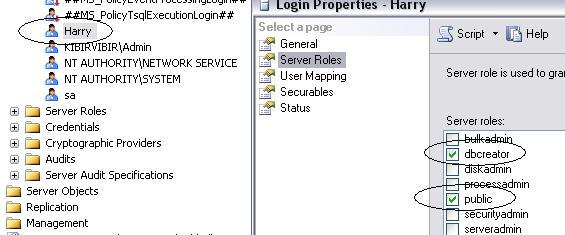


Fig.1. Database login that will be used on service must have dbcreator and public roles.

On target database this user must have listed below roles:

1. Db\_backupoperator,
2. Db\_datareader,
3. Db\_datawriter,
4. Db\_owner.

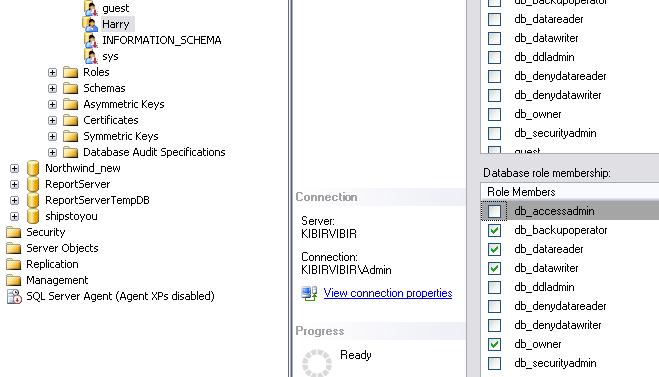


Fig. 2. User’s roles on target database.

## Launching testing tool

Written ASP.NET application is testing tool only. Launch it for verifying that service is working as you wish. The service itself may be called from any application that is able to call JSON enabled services.

Launching application from Visual Studio (select any of listed below ways):

1. Click “Start Debugging” speed button,
2. Click “F5” key or “Start Debugging” menu item in Debug menu,
3. Click “Ctrl + F5” key or “Start Without Debugging” menu item in Debug menu,
4. Right click on “Default.aspx” in solution explorer and select “View in Browser” from the popup menu.

Type application’s URL in browser if you installed it using supplied deployment package.

In either way you’ll see login screen:

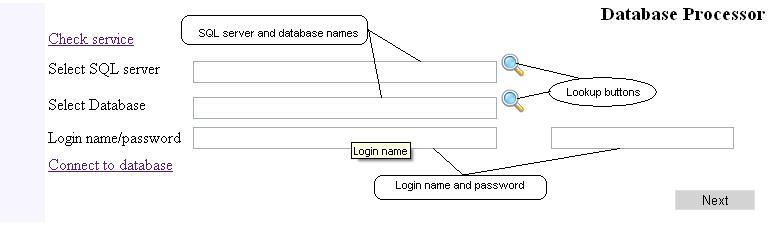


Fig. 3. Login screen.

Fill up all text fields before clicking “Connect to database”. Don’t worry about security; the service does not require SSL (HTTPS). It generates new pair of RSA public/private keys for every new session and the page sends name/password encrypted with public key.

SQL server and database may be selected from drop down menu. Click on lookup button and select data from drop down box. It is possible to enter some text in the text box before clicking lookup button. The service will use this phrase for filtering items. Output will consist of items with defined phrase inside (case insensitive search).

The service will build connection string and store it in the session container after clicking on “Connect to database”. Connection string uses SQL server authentication, ensure that defined here account has relevant rights on target database. For example: restoring database will fail if the user is not granted with “Create database” right.

You will see roles of the selected user if connection goes well:

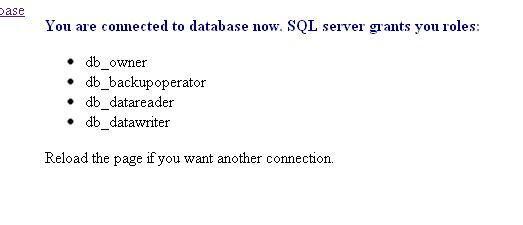


Fig.4. Message about successful connection.

Click on Next button or equivalent link on left for entering to Backup/Restore page.

## Backup database

Backup/Restore page consists of 3 parts separated by lines:

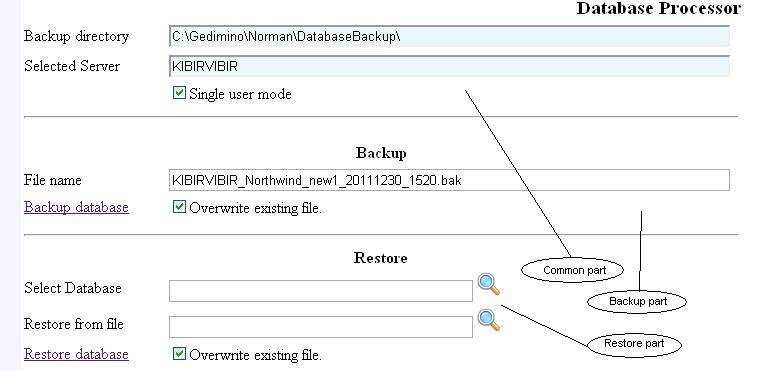


Fig. 5. Backup/Restore screen.

Backup directory is defined in the WEB.CONFIG file; see “BackupDirectory” key in “appSettings” section. The service and SQL must have access to this directory. Share this directory and map it to the same device letter on both computers if service and SQL server are located on different computers.

Test application builds file name following this template:

{SQL server’s name}\_{Database name}\_{Sortable date}\_{24 hours time: hours+minutes}

File name may be changed, but don’t change the extension: lookup button in restore part will look for \*.bak files only.

Checkbox “Single user mode” switches database into single mode before backup process only. During restoring process this checkbox acts existing database only (new database is created in single user’s mode ignoring value of this checkbox). The service switches restored database into multiuser mode after restoring it.

If you leave “Overwrite existing file.” Checkbox in checked state, the service will delete file with the same name from backup directory. Old file will be renamed appending “.bak” if you uncheck this box.

Click on “Backup database” link for creating backup file.

## Restore database

“Restore” process shares the same screen with “Backup” but is independent of “Backup” process: fill up “Select database”, “Restore from file” and click on “Restore database” if your backup file was created earlier. You can select both values from drop down clicking on lookup button but watch: you’ll get an exception after unchecking “Overwrite existing file” if your database server contains defined database. I’d recommend you select database and modify the name appending something like “\_new”.

## Renaming tables

“Modify table” tab allows you renaming every table in the database.

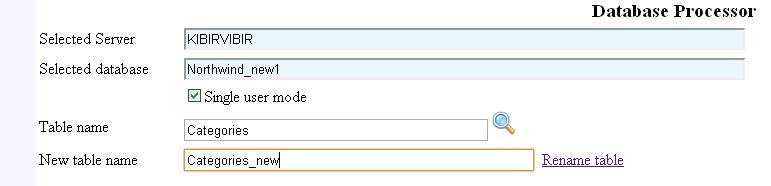


Fig. 7. Renaming tables.

Process of renaming tables has no impact on relations: they remains active, foreign keys are pointing to new table. Data in table are preserved too.

The service modifies bodies of stored procedures, views, user defined functions and triggers replacing old table name with the new one. Response contains list of modified objects, see property “AlteredDependencies” in the “RenameTableStatus”. Encrypted and CLR objects aren’t modified.

## About Service

The service was started as WCF Java script service but I had to rewrite it as WEB service because of slow work and faults that came from WCF environment.

Web methods (you can get more details generating documentation from CfcWebService.asmx.cs file):

|  |  |
| --- | --- |
| **Name** | **Description** |
| HelloWorld | Ping method for testing service itself |
| EnumerateSqlServers | Get list of available SQL servers |
| EnumerateDatabases | Get list of available databases on selected server |
| EnumerateBackupFiles | Get list with file names from the backup directory |
| CreateDbConnection | Verifies credentials and generates connection string; connection string is stored in the session container. |
| BackupDatabase | Creates backup file |
| RestoreDatabase | Restores database from the backup file |

# Known issues

Launching tester from Visual Studio debugging environment may result to “No SQL servers were found.” during first loading of the page. This comes not from service but from long preparing of debugging environment in Visual Studio (operation was canceled because of timeout). Click on “Magnifier” icon again or reload the page in this case.

Windows XP SP2 and later systems have “Messenger” service disabled. Change status of the service to “Automatic” or “Manual” and start it for receiving messages about switching SQL server to single user mode and restoring normal multiuser mode.