matchIT SQL

Installation and Deployment Guide

www.helpIT.com

Contents

Introduction

matchIT SQL provides SQL CLR (.NET) stored procedures for SQL Server (version 2005 or newer). It also provides .NET tasks for use in SSIS (SQL Server Integration Services) packages.

The product can be deployed in a number of different environments. It can be installed and used, for example, on a computer with one SQL Server instance that contains the data being deduped; it can also be installed and used on a computer containing SQL Server accessing data located elsewhere on the network.

This guide is targeted both at new users of the product and at users wishing to use the product in a different deployment scenario. It is focussed specifically on installation of the product and deployment in common environments. Where possible, guidance and recommendations are provided to help you achieve maximum dedupe performance without compromising the security and stability of SQL Server.

Please refer to *Appendix A – checklists* to check what needs configuring in any particular environment. Further information can be found in the relevant sections of the document.

A working knowledge of both Windows operating systems and SQL Server is assumed. Some configuration tasks require sufficient security privileges, so it might be necessary to consult a network administrator or a SQL Server sysadmin for guidance. Full details of required security permissions are provided where necessary.

The accompanying document – the matchIT SQL Information Pack – includes a getting-started tutorial, information on how to configure and use the product's stored procedures, and details of the outputs produced by the stored procedures.

Installation

Installing matchIT SQL

When installing matchIT SQL, please ensure you have the activation code that helpIT have issued. If you wish to evaluate the product, please contact helpIT to request an activation code for the evaluation period. This activation code is applied during installation.

matchIT SQL must be installed on a computer containing SQL Server. Note that this doesn't necessarily need to contain the data – it can be located on a remote SQL Server instance – however please refer to the *Recommendations* below.

The matchIT SQL installer must be run by a member of the computer's Administrators group.

If a feature is disabled during installation (for example, automatic checking for updates, summary reporting, Integration Services components) then these can be enabled at any time by rerunning the installer. Note, however, that some features (such as summary reporting) also require that any existing stored procedure configuration files be modified accordingly. (Features can't currently be removed unless the product is uninstalled then reinstalled without the feature.)

Reactivating

Should an activation code expire, the product will cease to function. It will therefore be necessary to reactivate the product with a new code supplied by helpIT. There are two ways to do this, the first being the recommended action:

1. Run this program (as an administrator) from the Start Menu:

    All Programs -> matchIT SQL -> Utilities -> Reactivate

Simply paste the new activation code into the text box, then click OK. A confirmation message will be displayed on successful reactivation; if not, please contact helpIT for advice.

2. Re-run the matchIT SQL installer; in the Product Activation page, select the 'Enter a new activation code' option, paste the new activation code into the text box, then continue with reinstallation. Refer to the following *Upgrading* section for further details.

Upgrading

Both matchIT SQL – and the matchIT API core matching component – are under continual maintenance and development. There are many benefits to keeping the product up-to-date.

Upgrading matchIT SQL to a newer version is a simple process.

Before running the installer, please ensure that no stored procedures are running and that the matchIT API component is not in use.

All installed matchIT SQL files will be upgraded. *Any such files modified since the last installation will be replaced with the latest version.* However, modified .xml configuration files and .sql scripts (in the 'config' and 'scripts' subfolders) will be backed up (by default, to a dated folder within "C:\matchIT SQL\~backup"). NB: Any user-created files are completely safe during installation and will not be deleted or modified.

matchIT SQL has been designed to provide full compatibility with older configuration files. So, for example, a configuration that has been working with version 1.3.0 will continue to

work with all newer versions of the product. There will likely be additions to the product that the older configuration will not benefit from, but as these additions are not explicitly enabled by the configuration your process or workflow will therefore not be affected.

Whenever upgrading, we highly recommend that you check the release notes to see what additions and enhancements have been made to the product, and to perform a file comparison of the new Config.xml with your existing configuration files for keeping these up-to-date (for this we recommend WinMerge, freely available for download at http://www.winmerge.org).

Recommendations

1. For maximum performance and security, install matchIT SQL on the computer containing the data, rather than accessing data on remote SQL Server instances.

2. Keep matchIT SQL up-to-date by installing new versions when they're made available for download. (A scheduled task can be created on installation of the product, to display a notification whenever a new release is available.)

3. Create and secure folders for .xml configuration files and .sql script files (e.g. "C:\matchIT SQL\config" and "C:\matchIT SQL\scripts" respectively). If using the web-based UI, change the path to the config folder by modifying the ConfigFilesDir setting in this file: "C:\Program Files\matchIT SQL\bin\UI\website\matchIT SQL\Web.config". Ensure that the hidden Template.xml file is copied into the new config folder. (Note that these steps are only required if the user has cancelled out of the matchIT SQL Configurator – this is executed at the end of the installation process.)

Deployment

matchIT SQL must be installed on the computer that will run the stored procedures. For best performance the data should be on the same computer; however, the data can also be located on any computer accessible via the network.

The stored procedures can be run using a variety of methods, including:

- an ADO.NET application;
- the SQL Server Management Studio;
- the SQL Server Agent;
- a SQL Server Integration Services (SSIS) package.

Each method uses a connection to the database containing the stored procedures; using this connection, a sequence of stored procedures (and T-SQL statements) can be executed. So, for example, an ADO.NET application can run from any computer on the network and connect to the SQL Server instance hosting the stored procedures; these stored procedures in turn connect to any SQL Server instance hosting the data.

Definitions

Local computer

This indicates the computer on which matchIT SQL has been installed; this computer requires a SQL Server instance to host the stored procedures.

Remote computer

This indicates the computer containing the SQL Server instance that hosts the data.

Scenario 1 – Local data

The most common deployment scenario that can provide maximum performance, in which both the stored procedures and the data are hosted within one SQL Server instance (or even separate instances on the same computer).

matchIT SQL must be installed on the computer on which SQL Server is already installed.

The stored procedures can be used by any authorised user from any networked computer (refer to *Configuring SQL Server Security* to configure execution privileges).

Ensure authorised SQL Server logins have been granted data access (refer to *Configuring SQL Server Security*).

Scenario 2 – Remote data

A common deployment scenario in which matchIT SQL cannot be installed on the computer containing the SQL Server instance hosting the data.

matchIT SQL will be installed on the local computer. The local computer, and all remote computers, must each contain a SQL Server instance.

The stored procedures can be used by any authorised user from any networked computer (refer to *Configuring SQL Server Security* to configure execution privileges).

Each remote SQL Server instance must be configured to allow access by authorised SQL Server logins (refer to *Configuring SQL Server for Remote Access*).

Ensure authorised SQL Server logins have been granted access to the remote database(s) (refer to *Configuring SQL Server Security*).

Scenario 3 – Local and remote data

A common deployment scenario in which matchIT SQL accesses data hosted on both local and remote SQL Server instances.

matchIT SQL will be installed on the local computer. The local computer, and all remote computers, must each contain a SQL Server instance.

The stored procedures can be used by any authorised user from any networked computer (refer to *Configuring SQL Server Security* to configure execution privileges).

Each remote SQL Server instance must be configured to allow access by authorised SQL Server logins (refer to *Configuring SQL Server for Remote Access*).

Ensure authorised SQL Server logins have been granted access to the local and remote database(s) (refer to *Configuring SQL Server Security*).

Recommendations

1. Wherever possible, for maximum performance and simpler configuration, install and use matchIT SQL on the same computer on which the data is stored.

2. Create a database on the local computer to be used only for hosting the stored procedures, containing no data, rather than having the stored procedures registered with multiple databases. This requires the stored procedures be registered once only, by an authorised login. When matchIT SQL is upgraded to a newer version, the stored procedures are simply dropped and then reregistered by ay authorised login. (Note that this is normally done automatically by the matchIT SQL Configurator during installation.)

   Alternatives involve registering the stored procedures on each relevant database, or registering the stored procedures only when they're needed then dropping them when they've completed (as per the matchIT SQL example .sql scripts). Both of these alternatives involve unnecessary repetition (in fact, the latter should not be done if multiple stored procedures will be run simultaneously!).

Configuring SQL Server for Remote Access

Since the 2005 version, SQL Server implements the *Secure by Default* principle, which means that upon installation SQL Server disables a number of features that are considered potentially insecure; this includes support for remote connections.

The following actions must be performed on the computer containing the SQL Server instance that will be remotely accessed (either by the stored procedures themselves, or by any application that will run the stored procedures):

SQL Server Configuration Manager

This can be found from the Start Menu:
Microsoft SQL Server 2005* -> Configuration Tools -> SQL Server Configuration Manager.

Select "SQL Server 2005* Network Configuration". (If the operating system is 64-bit and a 32-bit SQL Server instance is being configured, select "SQL Server 2005* Network Configuration (32-bit)" instead.)

Select "Protocols for *instance*". Note that *instance* refers to the name of the SQL Server instance; there will be at least one instance listed (the default instance is named MSSQLSERVER). Select the correct instance.

The right-hand pane lists the available remote connection protocols. By default, only the Shared Memory protocol is enabled. To enable remote connections, right-click the TCP/IP protocol and select Enable from the menu.

Acknowledge the confirmation box, then close the Configuration Manager.

SQL Server Browser

The SQL Server Browser service must be running on the remote computer.

Open up the Services console by using Run from the Start Menu, then typing:

```
services.msc
```

and clicking OK.

Within the console, locate the SQL Server Browser and open its properties dialog. If the service is stopped, click Start.

Additionally, the service's startup type can be set to Automatic so that the service is always available, even after the operating system is restarted.

Click OK to close the dialog, then close the Services window.

* or 2008 or 2008 R2 if either of these versions is installed

Firewall

Two exceptions must be added to any running firewall.

The exact steps to do this are dependent on which firewall is being used – or, in the case of Windows Firewall, the operating system itself. If unsure, please consult your network administrator.

Firstly, add the full pathname of the SQL Server instance to the firewall's exception list.
Instances can usually be found at "C:\Program Files\Microsoft SQL Server". For example, the default instance will likely be
"C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn\sqlservr.exe".
Alternatively, locate the service within the Services console (see above), bring up its properties, then look for the service's full pathname within the 'Path to executable' box (ensure that the arguments are ignored).

Secondly, add the full pathname of the SQL Server Browser to the firewall's exception list:
"C:\Program Files (x86)\Microsoft SQL Server\90\Shared\sqlbrowser.exe"
(for 32-bit operating systems use "Program Files" instead of "Program Files (x86)").

User Access

Refer to the following section, *Configuring SQL Server Security*.

Configuring SQL Server Security

Once a SQL Server instance has been configured to allow remote connections, it is also necessary to configure the instance's security to provide access for authorised users.

The highest level of access – providing full unlimited access to the server – is for a user to have the 'sysamin' server role. It is not recommended that this role is given to standard users except where strictly necessary. However, sysadmin privileges *are* required for certain one-off initialisation tasks, although these can be performed by an appropriate sysadmin-level user.

SQL Server security is provided by granting roles and permissions to logins and users.

Note that the T-SQL statements that follow should be executed within the SQL Server Management Studio by a login with the specified server role (alternatively, please refer to *Appendix B – Screenshots*).

SQL Server Logins

Each SQL Server instance maintains a list of *logins*. These are accounts through which a user connects to a SQL Server instance. A login can be a Windows domain user (domain\username), or it can be a SQL Server user (username and password, for example the 'sa' account).

As Microsoft recommends[1], "It is a best practice to use only Windows logins whenever possible" rather than a username and password combination. matchIT SQL configuration files specify connection strings to SQL Server instances, so using a password can be a potential security concern if configuration files are not protected correctly. (matchIT SQL does allow for encryption of connection strings, but please be aware that this provides a low level of security that isn't 100% secure.)

Logins can be assigned server roles (such as 'sysadmin' and 'bulkadmin') and permissions (such as 'Control server', 'Create any database', and 'Unsafe assembly').

To create a new login, use either of the following T-SQL statements:

```
CREATE LOGIN [domain\username] FROM WINDOWS

CREATE LOGIN [username] WITH PASSWORD='password'
```

Note that in all the following T-SQL statements, *domain|username* can be used in place of *username*.

To assign a server role to the login, use the following::

```
EXEC sp_addsrvrolemember 'username', 'role'
```

where *role* can be sysadmin, securityadmin, etc.

Note that a role can only be assigned if the login executing sp_addsrvrolemember is a sysadmin or has the server role being assigned to the target login.

[1] "Microsoft SQL Server 2005 Security Best Practices". Bob Beauchemin, SQLskills. 2007.

Database Users

Each SQL Server database maintains a list of *users*. These are SQL Server logins that have been given access to a particular database (a login can be given access to multiple databases).

Database users can be assigned database roles (such as 'db_owner') and permissions (such as 'Control', 'Insert', 'Delete', and 'Update').

To create a new user in a specific database, use the following T-SQL statements:

```
USE [database]
GO
CREATE USER [username] WITH DEFAULT_SCHEMA=dbo
```

To assign a database role to the user, use the following:

```
USE [database]
GO
EXEC sp_addrolemember 'role', 'username'
```

where *role* can be db_owner, db_ddladmin etc.

Data Access

When accessing a database that isn't hosting the stored procedures, a standard connection string is used that specifies the authentication method that will be used.

Local data

If Windows authentication is used (i.e. the connection string specifies "Integrated Security=SSPI"), then the user executing the stored procedure will be used to access the data.

If SQL Server authentication is used (i.e. the connection string specifies a user id and password), then that specific SQL Server login will be used to access the data. In this case, the stored procedures must be executed by a login that has execution privileges, but data will be accessed by a login that has the relevant data access privileges (the same login can be used for both execution and for data access).

Remote data

As per data access on a local machine, either Windows authentication or SQL Server authentication can be used.

To access remote data using Windows authentication, either of these requirements must be met (note that *instance* refers to the SQL Server instance on which the stored procedures are hosted):

1. the instance is running as a domain account, and the account has been given appropriate access to the database;

2. the instance is running as a service account (such as Network Service or Local System), and the *computer* is given appropriate access to the database;

3. the instance is running as a service account (such as Network Service or Local System), impersonation is enabled (see below), and the impersonated account has been given appropriate access to the database.

Impersonation

When running a matchIT SQL stored procedure from the SQL Server Management Studio, the stored procedure executes using the account that the SQL Server instance is running as.

For example, if the instance's service is running using the computer's Network Service account, then the stored procedure will run using that account; if a connection string (in a stored procedure configuration file) uses Windows authentication, then the *computer* will require access to the remote database.

matchIT SQL, therefore, allows the stored procedures to *impersonate* the actual user account that's executing the stored procedure. This account will be used only when connecting to a database.

To enable impersonation, simply set the impersonation node (in the generalSettings block of a stored procedure configuration file) to 'true':

```
<impersonation enabled="true" />
```

Note that impersonation isn't necessary when:
1. only local data access is performed; or
2. the connection string uses SQL Server authentication (username and password); or
3. the stored procedure is executed, for example, from a .NET application connecting to the database hosting the stored procedures; in such a case, the stored procedure will execute with the user account that's running the application.

Security Requirements for matchIT SQL

1. Allowing SQL CLR stored procedures on a SQL Server instance

To allow creation and execution of SQL CLR stored procedures on a particular SQL Server instance, a login must satisfy either of these requirements:

1. the login must have the 'sysadmin' server role; or

2. the login must have the 'Alter settings' server permission (however, the 'sysadmin' server role is still required for allowing 'external access' stored procedures, refer to the requirement that follows).

Execute the following T-SQL statements (this only needs to be done once on a SQL Server instance):

```
EXEC sp_configure 'clr enabled', 1
GO
RECONFIGURE
```

To grant the 'Alter settings' server permission, use the following T-SQL statements (this must be done by a login with the 'sysadmin' or 'securityadmin' server roles):

```
USE [master]
GO
GRANT ALTER SETTINGS TO [username]
```

Note that this only needs to be done on the SQL Server instance hosting the stored procedures. If accessing data on a remote instance, the stored procedures do not need to be registered on the remote instance.

2. Allowing 'external access' SQL CLR stored procedures on a SQL Server database

matchIT SQL provides stored procedures that require the 'External Access' permission. This means that no *unsafe* code is used (e.g. unmanaged compiled code), but that external resources including files and networks are accessed.

To be able to execute the following T-SQL statement, a login must have the 'sysadmin' server role:

```
ALTER DATABASE [database] SET TRUSTWORTHY ON
```

where *database* specifies the name of the database containing the stored procedures. As per the Deployment section above, the recommendation is to create one database that will host the stored procedures; this means that the TRUSTWORTHY option need be set on one database only.

3. To allow creation of SQL CLR stored procedures

In order to register SQL CLR stored procedures on a database, a login must satisfy any of these requirements:

1. the login must have the 'sysadmin' server role; or

2. the login must be a database user with the 'db_owner' or 'db_ddladmin' roles, must have the 'Control Server' server permission, and must have the 'Authenticate' database permission.

To prepare a login that satisfies the second requirement, use the following T-SQL statements (this can only be done by a login with the 'sysadmin' server role, or a login with the 'securityadmin' server role and the 'db_owner' database role):

```
USE [master]
GO
GRANT CONTROL SERVER TO [username]
GO
USE [database]
GO
GRANT AUTHENTICATE TO [username]
EXEC sp_addrolemember 'db_ddladmin', 'username'
```

The stored procedure assembly can then be registered using CREATE ASSEMBLY, and the stored procedures themselves using CREATE PROCEDURE. Please see one of the template .sql scripts – these can usually be found in C:\matchIT SQL\scripts.

Note that if the login doesn't have the sysadmin role, then impersonation of a sysadmin login will be required. In such a case, use `EXECUTE AS LOGIN = 'username'`
(where *username* specifies as sysadmin login) before creating the assembly, and `REVERT` after creating the stored procedures.

4. To allow execution of SQL CLR stored procedures

In order to execute SQL CLR stored procedures, a login must satisfy any these requirements:

1. the login must have the 'sysadmin' server role; or

2. the login must be a database user with the 'db_owner' role; or

3. the login must be a database user with the Execute database permission.

To allow a user to execute SQL CLR stored procedures:

```
USE [database]
GO
GRANT EXECUTE TO [username]
```

5. To allow the SQL CLR stored procedures to access data

To allow the SQL CLR stored procedures to access data within *any* database (either the database also hosting the stored procedures or any other database), the database user must have the Select, Alter, Insert, Update, and Delete permissions:

```
USE [database]
GO
GRANT SELECT, ALTER, INSERT, UPDATE, DELETE TO [username]
```

Alternatively, the user can have the 'db_datareader', 'db_datawriter' and 'db_ddladmin' database roles (or just 'db_owner'):

```
USE [database]
GO
EXEC sp_addrolemember 'db_datareader', 'username'
EXEC sp_addrolemember 'db_datawriter', 'username'
EXEC sp_addrolemember 'db_ddladmin', 'username'
```

Recommendations

1. Create a login with the 'sysadmin' role for performing these one-off or infrequent tasks: enabling SQL CLR on a SQL Server instance; enabling the TRUSTWORTHY option for the database that will host the stored procedures; registering the assembly and stored procedures in the database.

2. Create a login that has the necessary permissions for executing the stored procedures (see step 4 above) and for accessing each database that will be processed by the stored procedures (see step 5 above).

3. Use Windows authentication (domain accounts) for maximum security, rather than SQL Server authentication (username and password) – i.e. connection strings in stored procedure configuration files should specify "Integrated Security=SSPI", rather than a username and password.

Installing and Configuring Addressing

Installation

For matchIT SQL to be able to generate corrected addresses using the msp_GenerateCorrectAddresses stored procedure, a valid activation code must have been issue by helpIT systems. Use this activation code when installing matchIT SQL.

If matchIT SQL was previously installed without addressing support, then matchIT SQL will need to be reinstalled. Simply re-run the matchIT SQL installer, opting to apply a new activation code (provided by helpIT) in the Product Activation page.

During installation, an Address Correction page will be shown. Ensure the option "Install addressing components" is checked to add addressing support. For UK addressing, there will be an additional option, "Download data files", that, if checked, will allow automatic download of the addressing data. For US addressing, helpIT will provide installation media that will install the data separately.

Configuration

Additional configuration steps must be performed before the addressing stored procedure can be used (the following steps show how to configure addressing with the example database, matchIT_SQL_demo).

Firstly, the built-in Local Service account must be given write access to the database. This is because the address correction actually occurs outside the stored procedure; it is run under the context of the matchIT SQL Service which, by default, uses the Local Service account (a low privilege Windows service account).

Secondly, the account must be given the following roles for the matchIT_SQL_demo database: db_datareader, db_datawriter, and db_ddladmin. Alternatively, the account can be given the db_owner role instead of these three roles.

Here are the steps that should be run in the Management Studio:

```
USE [master]
GO
CREATE LOGIN [LocalService] FROM WINDOWS
GO
USE [matchIT_SQL_demo]
GO
CREATE USER [LocalService] WITH DEFAULT_SCHEMA=dbo
EXEC sp_addrolemember 'db_datareader', 'LocalService'
EXEC sp_addrolemember 'db_datawriter', 'LocalService'
EXEC sp_addrolemember 'db_ddladmin', 'LocalService'
```

If the matchIT SQL service isn't running as the Local Service account (as per the following recommendations), use the account name (i.e. domain\username) in place of LocalService.

Recommendations

Change the matchIT SQL Service to run as a domain user instead of the computer's Local Service account. Then create a login for this account within the SQL Server instance. Lastly, for each database within the SQL Server instance that the address correction will need access to, create a user for the login and give it the 'db_datareader', 'db_datawriter', and 'db_ddladmin' database roles.

The matchIT SQL Service can be configured from a command console as follows:

```
cd /d "C:\Program Files\matchIT SQL\bin"
matchITSQL /stop /uninstall
matchITSQL /install:domain\username:password /start
```

After reinstalling or upgrading matchIT SQL, the matchIT SQL service will be recreated using the default Local Service account. So you would need to rerun the above steps after each reinstallation or upgrade (a batch script, in a secure folder, could be created for this).

Appendix A – Checklists

Scenario 1 – Local data

Local computer:

1. Ensure a SQL Server instance has been installed;
2. Install matchIT SQL;
3. (Create the stored procedures, if necessary);
4. Allow authorised users to execute the stored procedures;
5. Allow authorised users to access data on the SQL Server database(s).

Scenario 2 – Remote data

Local computer:

1. Ensure a SQL Server instance has been installed;
2. Install matchIT SQL;
3. (Create the stored procedures, if necessary);
4. Allow authorised users to execute the stored procedures.

Remote computer(s):

5. Ensure a SQL Server instance has been installed;
6. Allow remote connections to the SQL Server instance;
7. Allow authorised users to access data on the SQL Server database(s).

Scenario 3 – Local and remote data

Local computer:

1. Ensure a SQL Server instance has been installed;
2. Install matchIT SQL;
3. (Create the stored procedures, if necessary);
4. Allow authorised users to execute the stored procedures;
5. Allow authorised users to access data on the SQL Server database(s).

Remote computer(s):

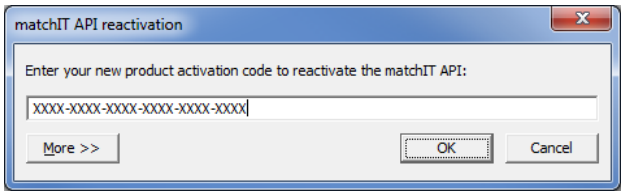
6. Ensure a SQL Server instance has been installed;
7. Allow remote connections to the SQL Server instance;
8. Allow authorised users to access data on the SQL Server database(s).

Upgrading matchIT SQL

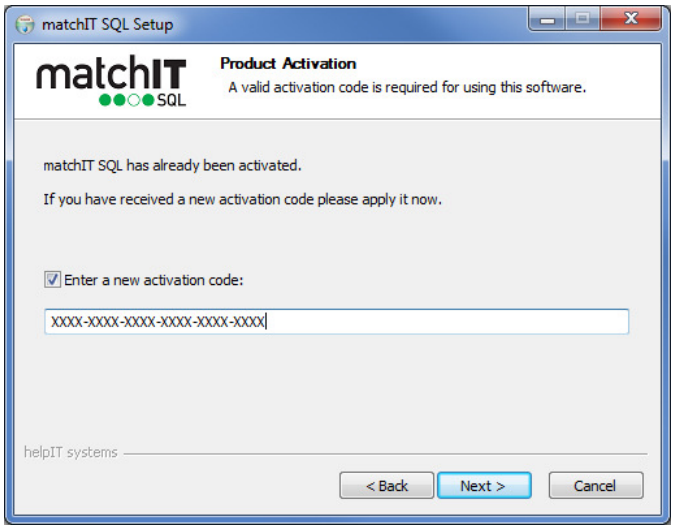
1. Ensure that no stored procedures are running;
2. Ensure that the matchIT API component is not in use;
3. (Drop the existing stored procedures, if necessary);
4. Install the new matchIT SQL;
5. (Create the new stored procedures, if necessary);
6. Check the release notes for the new version;
7. Upgrade configuration files by comparing with the new Config.xml.

Appendix B – Screenshots

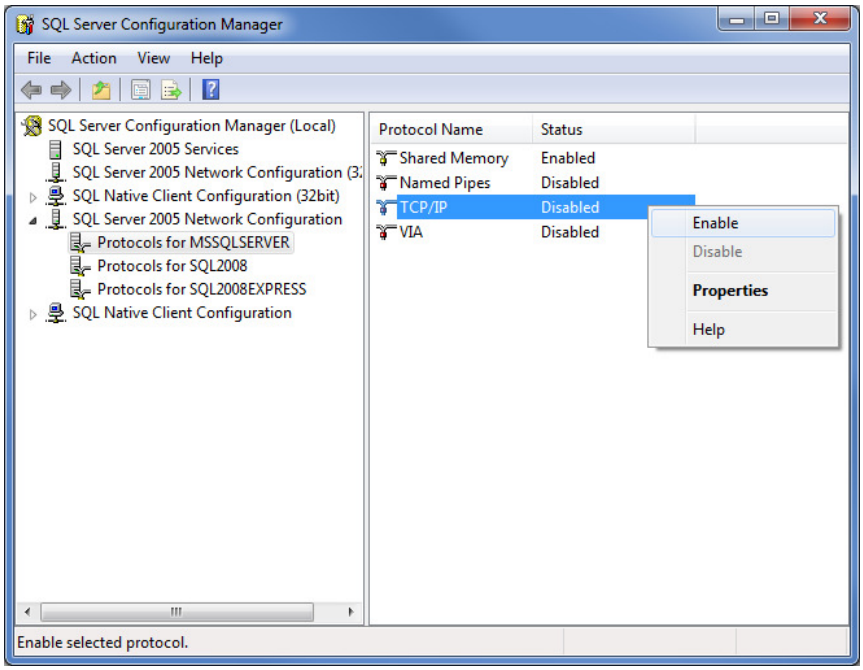
1. Reactivating matchIT SQL with a new activation code, without reinstalling the product:



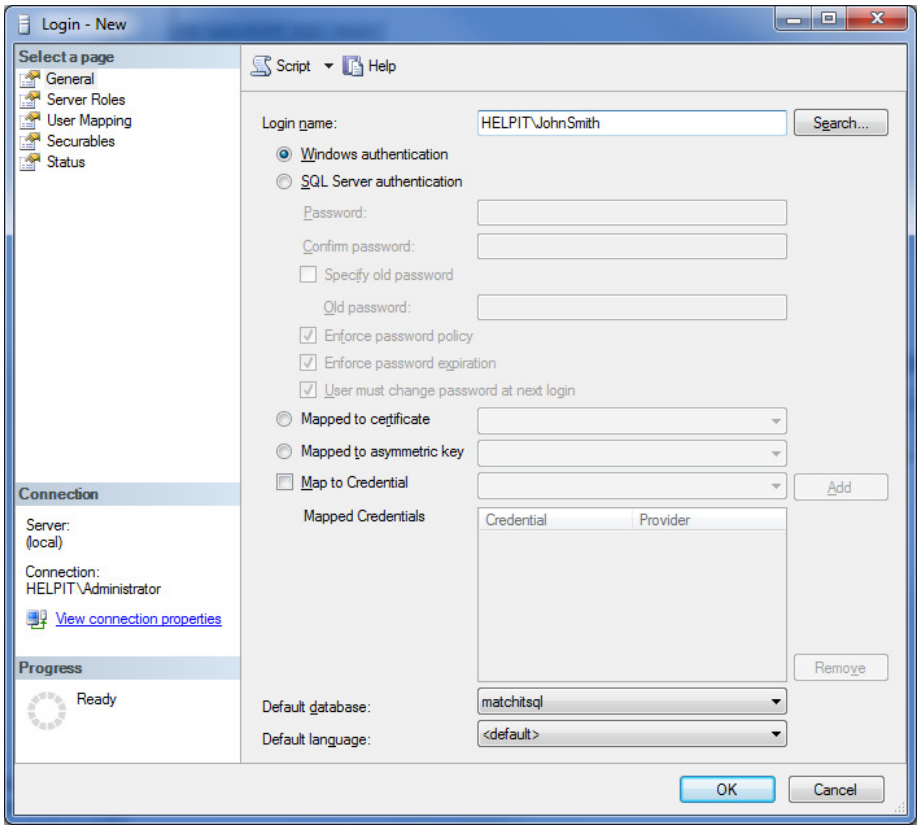
2. Reactivating matchIT SQL with a new activation code when reinstalling the product:



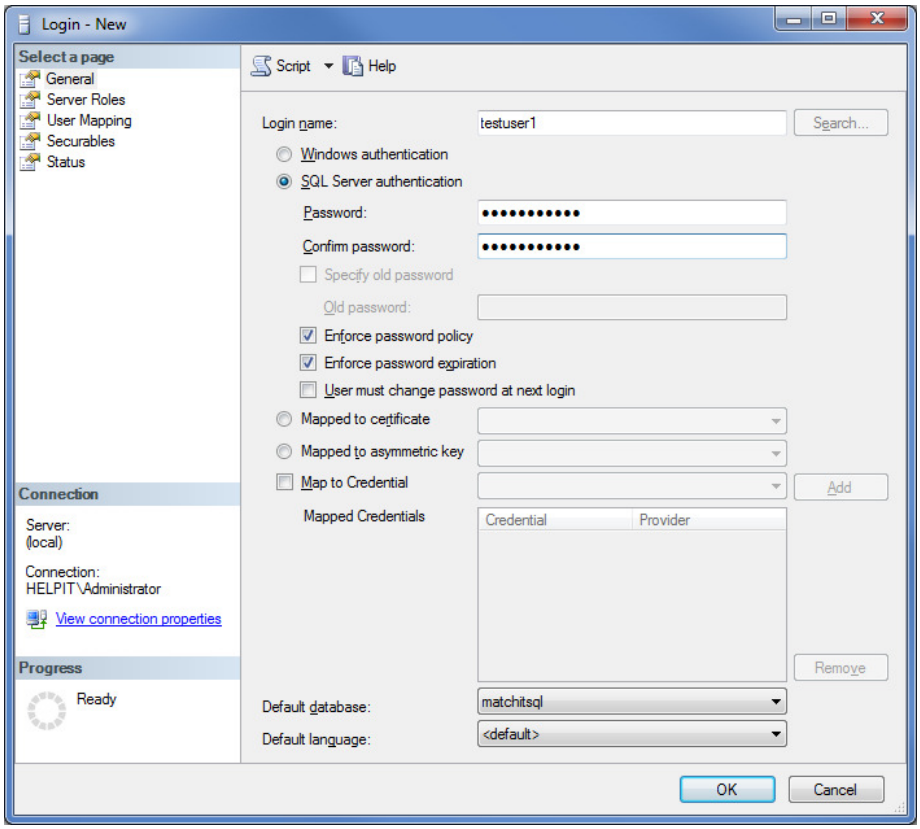
3. Allowing a SQL Server instance to be accessed from computers on the network:



4. Creating a new login that uses Windows authentication:



5. Creating a new login that uses SQL Server authentication:



6. Providing the login with read-write access to a database: