Date: 18/02/2013
Procedure: Mapping SQL Server Logins to Database Users (SQL Server)
Source: LINK
Permalink: LINK
Created by: HeelpBook Staff
Document Version: 1.0

MAPPING SQL SERVER LOGINS TO DATABASE USERS (SQL SERVER)

If you develop applications for **SQL Server**, you've probably had to move a database from one computer to another. For example, you may grab a snapshot of the production or stage database before starting a new round of development.

The problem with moving from one machine to another is that **the login accounts don't go with the database**. Login accounts, as opposed to database user accounts, are defined at the server level. In fact, you usually don't want the login accounts associated with the dev, test, and production environments to be the same anyway.

One way to solve the problem is to create a new login account for the database, but then you run into "mapping" issues. If the database already has a user account that matches the login account name, you'll get a duplicate name error. If the database does not already have the user account, you'll be stuck re-assigning any special permissions the new account needs (*such as stored procedure permissions*).

Sure, you can set the new database user account to be a database owner, but that just masks privilege issues that will come up during deployment later on.

Fortunately, SQL Server provides a simple solution for this situation. The approach you take depends upon which version of **SQL Server** you are running. The "**old way**" (prior to **SQL Server 2008**) is to use "sp_change_users_login" system stored procedure. The "**new way**" (starting with SQL Server 2008) is to use the "**ALTER USER**" statement. I'll show you both ways.
For the examples, assume you just restored the Enterprise database to your development **SQL Server**. The database user name is "Captain." In production, the login is "**JamesTKirk**". On your development machine, you want to map the "ChristopherPike" login to the "**Captain**" database user.

USING SP_CHANGE_USERS_LOGIN

I'm not sure how far "back" the sp_change_users_login stored procedure goes, but you can continue to use it through the next release of **SQL Server** (the one after 2008). After that, **Microsoft** claims it will be dropped in favor of **ALTER USER**.

Note that this procedure only works with SQL Server logins, not with logins mapped to a **Windows** user account.

```
USE Enterprise

GO

sp_change_users_login

@Action='update_one',

@UserNamePattern='Captain',

@LoginName='ChristopherPike'

GO
```

These Transact-SQL statements map the "**ChristopherPike**" login to the "Captain" user of the **Enterprise** database. You can even use **sp_change_users_login** to create the new login account for you, if you include the **@Password** parameter. For more info about how to do that, see SQL Server Books Online.

Using ALTER USER

Since sp_change_users_login is deprecated as of **SQL Server 2008**, you should start using the **ALTER USER** approach instead. **ALTER USER** can even be used with Windows-based logins, as long as the database user was originally mapped to a Windows-based login. In other words, you can't change from a Windows-based login to a SQL Server login or vice-versa.

The same example implemented with **ALTER USER** looks like this:

```
USE Enterprise

GO

ALTER USER Captain WITH LOGIN = ChristopherPike

GO
```

ALTER USER has nice, clean syntax, but it didn't support the WITH LOGIN clause until SQL Server 2008.

Keep Those Permissions

Using **sp_change_users_login** or **ALTER USER** to map a login to a database user can save you a bunch of time and prevent permission configuration errors when you move a SQL Server database from one computer to another. The database user keeps all of its permissions, and you can use whatever name you want for the login account.