**How to Avoid Orphan Users in SQL Server ALWAYSON, Create Logins Correctly**

SQL Server [logins](https://msdn.microsoft.com/en-us/library/aa337562.aspx) are stored in the [master] database. System databases (master, model, msdb and tempdb) cannot be included in an availability group so a login created on the PRIMARY replica will not show up on the SECONDARY replica automatically. It has to be created manually on the SECONDARY replica.

Logins have an [SID](https://msdn.microsoft.com/en-us/library/ms403629.aspx) associated with them. When a Windows authentication login is created in SQL Server, it uses the SID from the Windows Active Directory. So the SID for such a login will be the same on every SQL Server in the network domain.

When a SQL authentication login is created, it gets a new auto-generated SID assigned by the SQL Server. This auto-generated SID will be different in each SQL Server even if the login name and the password combination are the same.

The database [users](https://msdn.microsoft.com/en-us/library/aa337545.aspx) are mapped to the logins internally using the SID, not the login name/user name. There are some situations where the SID may have a mismatch, e.g. when a database is restored to a different server where the supposedly matching login already exists, or a login is dropped and recreated without consideration to the mapped users, or a login is recreated between a database detach and reattach etc. Such users that do not have a login with a matching SID are known as orphan users. This SID mismatch means that although an application or a user can connect to the server using the login, but it can not access the database.

The following options with varying degree of effectiveness can be used to fix the SID mapping between a login and an orphan user –

* Drop and recreate the user in the restored database. Of course the user permissions will get deleted too and have to be granted again.
* Drop and recreate the login with same SID as the restored database. This is a definite no-no if there are other databases on the server linked to that login. It will only complicate matters.
* Run the system stored procedure [sp\_change\_users\_login](https://msdn.microsoft.com/en-us/library/ms174378.aspx" \t "_blank). It has parameters to just report, fix one or fix all orphans. But the stored procedure is now marked as deprecated so there are no guarantees of future availability.
* [ALTER](https://msdn.microsoft.com/en-us/library/ms176060.aspx) the user (there are some restrictions, like there cannot be a one-to many mapping etc.) –

|  |  |
| --- | --- |
| 1  2  3 | USE [MyUserDBName]  GO  ALTER USER someuser WITH LOGIN = somelogin |

* As the user databases on the SECONDARY replica are read-only, the role of the SECONDARY server has to be changed to PRIMARY by doing a failover before the above ALTER command can be executed.

All of this can be avoided if the login is created correctly on the SECONDARY replica. We just have to make sure that the SID for the login on the SECONDARY replica matches the PRIMARY replica.

Let us begin with creating a new login on the PRIMARY replica of an existing Availability Group –

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | /\* On the PRIMARY replica \*/    -- Create the Login  USE [master]  GO  CREATE LOGIN [TestLogin]  WITH PASSWORD=N'abc123#'  , DEFAULT\_DATABASE=[master]  GO |

Grant privileges to this login if necessary. They will NOT automatically replicate to the SECONDARY replica. Now find the SID of this new login.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | -- Get the SID for the new Login  SELECT name, sid  FROM sys.server\_principals  WHERE name = 'TestLogin'    /\* Results:  name          sid  TestLogin     0x8EA0E033BD83524180CF813A20C5265B  \*/ |

On the SECONDARY replica, create the login with the same SID. The GUI wizard to create logins does not have this feature to specify the SID, so the login has to be created using TSQL with an additional parameter.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | /\* On the SECONDARY replica \*/  -- Create the Login with the same SID as  -- the PRIMARY replica  CREATE LOGIN [TestLogin]  WITH PASSWORD=N'abc123#'  -- use the SID retrieved above  , SID = 0x8EA0E033BD83524180CF813A20C5265B  , DEFAULT\_DATABASE=[master]  GO |

Grant the same privileges to this login as done on the PRIMARY replica.

Now go back to the PRIMARY replica and create database user mapped to the login and grant required permissions at the database level. This new database user will be automatically replicated on the SECONDARY replica with its permissions and correctly map to the login. No action on SECONDARY required because the user database is in an Availability Group that is synced across replicas.

**Further reading:**  
For the sake of completeness, I must mention the widely cited [KB 918992 article (How to transfer logins and passwords between instances of SQL Server)](http://support.microsoft.com/kb/918992) which provides a stored procedure [sp\_rev\_login] to move the logins from one server to another. This stored procedure generates the CREATE LOGIN script with the password hash and the SID. You would need that stored procedure only if you don’t have access to the clear text passwords or, if you want to include that script as a scheduled job but not hardcode the password in the job. If you do not have those constraints then you can simply use the steps described in my blog post here.