How to transfer the logins and the passwords between instances of SQL Server 2005

View products that this article applies to.

On This Page

Article ID : 918992
Last Review : July 11, 2006
Revision : 2.2

INTRODUCTION

This article describes how to transfer the logins and the passwords between instances of Microsoft SQL Server 2005 on different servers.

For more information about how to transfer the logins and the passwords between instances of other versions of SQL Server, click the following article number to view the article in the Microsoft Knowledge Base: 246133 (http://support.microsoft.com/kb/246133/) How to transfer logins and passwords between instances of SQL Server

MORE INFORMATION

In this article, server A and server B are different servers. Additionally, both server A and server B are running SQL Server 2005.

After you move a database from the instance of SQL Server on server A to the instance of SQL Server on server B, the users may not be able to log in to the database on server B. Additionally, the users may receive the following error message: **Login failed for user '*MyUser*'. (Microsoft SQL Server, Error: 18456)** This problem occurs because you did not transfer the logins and the passwords from the instance of SQL Server on server A to the instance of SQL Server on server B.

To transfer the logins and the passwords from the instance of SQL Server on server A to the instance of SQL Server on server B, follow these steps:

1. On server A, start SQL Server Management Studio, and then connect to the instance of SQL Server from which you moved the database.
2. Open a new Query Editor window, and then run the following script.

```
USE master GO IF OBJECT_ID ('sp_hexadecimal') IS NOT NULL DROP PROCEDURE sp_hexadecimal GO CREATE
PROCEDURE sp_hexadecimal @binvalue varbinary(256), @hexvalue varchar (514) OUTPUT AS DECLARE
@charvalue varchar (514) DECLARE @i int DECLARE @length int DECLARE @hexstring char(16) SELECT
@charvalue = '0x' SELECT @i = 1 SELECT @length = DATALENGTH (@binvalue) SELECT @hexstring =
'0123456789ABCDEF' WHILE (@i <= @length) BEGIN DECLARE @tempint int DECLARE @firstint int DECLARE
@secondint int SELECT @tempint = CONVERT(int, SUBSTRING(@binvalue,@i,1)) SELECT @firstint =
FLOOR(@tempint/16) SELECT @secondint = @tempint - (@firstint*16) SELECT @charvalue = @charvalue +
SUBSTRING(@hexstring, @firstint+1, 1) + SUBSTRING(@hexstring, @secondint+1, 1) SELECT @i = @i + 1
END SELECT @hexvalue = @charvalue GO IF OBJECT_ID ('sp_help_revlogin') IS NOT NULL DROP PROCEDURE
sp_help_revlogin GO CREATE PROCEDURE sp_help_revlogin @login_name sysname = NULL AS DECLARE @name
sysname DECLARE @type varchar (1) DECLARE @hasaccess int DECLARE @denylogin int DECLARE
@is_disabled int DECLARE @PWD_varbinary varbinary (256) DECLARE @PWD_string varchar (514) DECLARE
@SID_varbinary varbinary (85) DECLARE @SID_string varchar (514) DECLARE @tmpstr varchar (1024)
DECLARE @is_policy_checked varchar (3) DECLARE @is_expiration_checked varchar (3) DECLARE
@defaultdb sysname IF (@login_name IS NULL) DECLARE login_curs CURSOR FOR SELECT p.sid, p.name,
p.type, p.is_disabled, p.default_database_name, l.hasaccess, l.denylogin FROM
sys.server_principals p LEFT JOIN sys.syslogins l ON ( l.name = p.name ) WHERE p.type IN ( 'S',
'G', 'U' ) AND p.name <> 'sa' ELSE DECLARE login_curs CURSOR FOR SELECT p.sid, p.name, p.type,
p.is_disabled, p.default_database_name, l.hasaccess, l.denylogin FROM sys.server_principals p LEFT
JOIN sys.syslogins l ON ( l.name = p.name ) WHERE p.type IN ( 'S', 'G', 'U' ) AND p.name =
@login_name OPEN login_curs FETCH NEXT FROM login_curs INTO @SID_varbinary, @name, @type,
@is_disabled, @defaultdb, @hasaccess, @denylogin IF (@@fetch_status = -1) BEGIN PRINT 'No login(s)
found.' CLOSE login_curs DEALLOCATE login_curs RETURN -1 END SET @tmpstr = '/* sp_help_revlogin
script ' PRINT @tmpstr SET @tmpstr = '** Generated ' + CONVERT (varchar, GETDATE()) + ' on ' +
@@SERVERNAME + ' */' PRINT @tmpstr PRINT '' WHILE (@@fetch_status <> -1) BEGIN IF (@@fetch_status
<> -2) BEGIN PRINT '' SET @tmpstr = '-- Login: ' + @name PRINT @tmpstr IF (@type IN ( 'G', 'U'))
BEGIN -- NT authenticated account/group SET @tmpstr = 'CREATE LOGIN ' + QUOTENAME( @name ) + '
FROM WINDOWS WITH DEFAULT_DATABASE = [' + @defaultdb + ']' END ELSE BEGIN -- SQL Server
authentication -- obtain password and sid SET @PWD_varbinary = CAST( LOGINPROPERTY( @name,
'PasswordHash' ) AS varbinary (256) ) EXEC sp_hexadecimal @PWD_varbinary, @PWD_string OUT EXEC
sp_hexadecimal @SID_varbinary,@SID_string OUT -- obtain password policy state SELECT
@is_policy_checked = CASE is_policy_checked WHEN 1 THEN 'ON' WHEN 0 THEN 'OFF' ELSE NULL END FROM
```

```
sys.sql_logins WHERE name = @name SELECT @is_expiration_checked = CASE is_expiration_checked WHEN
1 THEN 'ON' WHEN 0 THEN 'OFF' ELSE NULL END FROM sys.sql_logins WHERE name = @name SET @tmpstr =
'CREATE LOGIN ' + QUOTENAME( @name ) + ' WITH PASSWORD = ' + @PWD_string + ' HASHED, SID = ' +
@SID_string + ', DEFAULT_DATABASE = [' + @defaultdb + ']' IF ( @is_policy_checked IS NOT NULL )
BEGIN SET @tmpstr = @tmpstr + ', CHECK_POLICY = ' + @is_policy_checked END IF (
@is_expiration_checked IS NOT NULL ) BEGIN SET @tmpstr = @tmpstr + ', CHECK_EXPIRATION = ' +
@is_expiration_checked END END IF (@denylogin = 1) BEGIN -- login is denied access SET @tmpstr =
@tmpstr + '; DENY CONNECT SQL TO ' + QUOTENAME( @name ) END ELSE IF (@hasaccess = 0) BEGIN --
login exists but does not have access SET @tmpstr = @tmpstr + '; REVOKE CONNECT SQL TO ' +
QUOTENAME( @name ) END IF (@is_disabled = 1) BEGIN -- login is disabled SET @tmpstr = @tmpstr + ';
ALTER LOGIN ' + QUOTENAME( @name ) + ' DISABLE' END PRINT @tmpstr END FETCH NEXT FROM login_curs
INTO @SID_varbinary, @name, @type, @is_disabled, @defaultdb, @hasaccess, @denylogin END CLOSE
login_curs DEALLOCATE login_curs RETURN 0 GO
```

Note This script creates two stored procedures in the **master** database. The two stored procedures are named the **sp_hexadecimal** stored procedure and the **sp_help_revlogin** stored procedure.

3. Run the following statement.

   ```
   EXEC sp_help_revlogin
   ```

   The output script that is generated by the **sp_help_revlogin** stored procedure is the login script. This login script creates the logins that have the original Security Identifier (SID) and the original password.

4. On server B, start SQL Server Management Studio, and then connect to the instance of SQL Server to which you moved the database.

   Important Before you go to step 5, review the information in the "Remarks" section.

5. Open a new Query Editor window, and then run the output script that is generated in step 3.

Remarks

Review the following information before you run the output script on the instance on server B:

- Review the output script carefully. If server A and server B are in different domains, you have to modify the output script. Then, you have to replace the original domain name with the new domain name in the CREATE LOGIN statements. The integrated logins that are granted access in the new domain do not have the same SID as the logins in the original domain. Therefore, the users are orphaned from these logins. For more information about how to resolve these orphaned users, click the following article number to view the article in the Microsoft Knowledge Base: 240872 (http://support.microsoft.com/kb/240872/) How to resolve permission issues when you move a database between servers that are running SQL Server If server A and server B are in the same domain, the same SID is used. Therefore, the users are not likely to be orphaned.
- In the output script, the logins are created by using the encrypted password. This is because of the HASHED argument in the CREATE LOGIN statement. This argument specifies that the password that is entered after the PASSWORD argument is already hashed.
- By default, only a member of the **sysadmin** fixed server role can run a SELECT statement from the **sys.server_principals** view. Unless a member of the **sysadmin** fixed server role grants the necessary permissions to the users, the users cannot create or run the output script.
- The steps in this article do not transfer the default database information for a particular login. This is because the default database may not always exist on server B. To define the default database for a login, use the ALTER LOGIN statement by passing in the login name and the default database as arguments.
- The sort order of server A may be case insensitive, and the sort order of server B may be case sensitive. In this case, the users must type all the letters in the passwords as uppercase letters after you transfer the logins and the passwords to the instance on server B.

  Alternatively, the sort order of server A may be case sensitive, and the sort order of server B may be case insensitive. In this case, the users cannot log in by using the logins and the passwords that you transfer to the instance on server B unless one of the following conditions is true:
  - The original passwords contain no letters.
  - All the letters in the original passwords are uppercase letters.

  The sort order of both server A and server B may be case sensitive, or the sort order of both server A and server B may be case insensitive. In these cases, the users do not experience a problem.
- A login that already is in the instance on server B may have a name that is the same as a name in the output script. In this case, you receive the following error message when you run the output script on the instance on server B: **Msg 15025, Level 16, State 1, Line 1**
  The server principal '*MyLogin*' already exists. Similarly, a login that already is in the instance on server B may have a SID that is the same as a SID in the output script. In this case, you receive the following error message when you run the output script on the instance on server B: **Msg 15433, Level 16, State 1, Line 1**

Supplied parameter sid is in use. Therefore, you must do the following:

1. Review the output script carefully.
2. Examine the contents of the **sys.server_principals** view in the instance on server B.
3. Address these error messages accordingly.

- In SQL Server 2005, the SID for a login is used as the basis for implementing database-level access. A login may have two different SIDs in two different databases on a server. In this case, the login can only access the database that has the SID that matches the SID in the **sys.server_principals** view. This problem may occur if the two databases are consolidated from two different servers. To resolve this problem, manually remove the login from the database that has a SID mismatch by using the DROP USER statement. Then, add the login again by using the CREATE USER statement.

REFERENCES

For more information about how to troubleshoot orphaned users, visit the following Microsoft Developer Network (MSDN) Web site: http://msdn2.microsoft.com/en-us/library/ms175475.aspx (http://msdn2.microsoft.com/en-us/library/ms175475.aspx) For more information about the CREATE LOGIN statement, visit the following MSDN Web site: http://msdn2.microsoft.com/en-us/library/ms189751.aspx (http://msdn2.microsoft.com/en-us/library/ms189751.aspx) For more information about the ALTER LOGIN statement, visit the following MSDN Web site: http://msdn2.microsoft.com/en-us/library/ms189828.aspx (http://msdn2.microsoft.com/en-us/library/ms189828.aspx)

APPLIES TO

- Microsoft SQL Server 2005 Standard Edition
- Microsoft SQL Server 2005 Workgroup Edition
- Microsoft SQL Server 2005 Developer Edition
- Microsoft SQL Server 2005 Enterprise Edition

Keywords: kbexpertiseadvanced kbhowto kbinfo KB918992