**How to create a login in SQL Server SQL Server Management Studio or T-SQL**

A login is the identity of the person or process that is connecting to an instance of SQL Server. A SQL login is authenticated by a secure system

**All Users need a login to connect to SQL Server**

You can create a login based on a Windows principal (such as a domain user or a Windows domain group) or you can create a login that is not based on a Windows principal (such as an SQL Server login)

To use SQL Server Authentication, the Database Engine must use mixed mode authentication

The scope of a login is the whole Database Engine (mapped to the whole database)

To connect to a specific database on the instance of SQL Server, a login must be mapped to a database user

A database user is an already created LOGIN that is mapped to a specific database that can access the database itself, but ‘not the objects’ in the database

Permissions inside the database are granted and denied to the database user, not the login

**Permissions**

**To create a SQL Server login with SSMS or T-SQL**

**Server Roles**

The Server Roles page lists all possible roles that can be assigned to the new login.

bulkadmin

Members of the bulkadmin fixed server role can run the BULK INSERT statement.

dbcreator

Members of the dbcreator fixed server role can create, alter, drop, and restore any database.

diskadmin

Members of the diskadmin fixed server role can manage disk files.

processadmin

Members of the processadmin fixed server role can terminate processes running in an instance of the Database Engine.

public

All SQL Server users, groups, and roles belong to the public fixed server role by default.

securityadmin

Members of the securityadmin fixed server role manage logins and their properties. They can GRANT, DENY, and REVOKE server-level permissions. They can also GRANT, DENY, and REVOKE database-level permissions. Additionally, they can reset passwords for SQL Server logins.

serveradmin

Members of the serveradmin fixed server role can change server-wide configuration options and shut down the server.

setupadmin

Members of the setupadmin fixed server role can add and remove linked servers, and they can execute some system stored procedures.

sysadmin

Members of the sysadmin fixed server role can perform any activity in the Database Engine.

User Mapping

The User Mapping page lists all possible databases and the database role memberships on those databases that can be applied to the login. The databases selected determine the role memberships that are available for the login. The following options are available on this page:

Users mapped to this login

Select the databases that this login can access. When you select a database, its valid database roles are displayed in the Database role membership for: database\_name pane.

Map

Allow the login to access the databases listed below.

Database

Lists the databases available on the server.

User

Specify a database user to map to the login. By default, the database user has the same name as the login.

Default Schema

Specifies the default schema of the user. When a user is first created, its default schema is dbo. It is possible to specify a default schema that does not yet exist. You cannot specify a default schema for a user that is mapped to a Windows group, a certificate, or an asymmetric key.

Guest account enabled for: database\_name

Read-only attribute indicating whether the Guest account is enabled on the selected database. Use the Status page of the Login Properties dialog box of the Guest account to enable or disable the Guest account.

Database role membership for: database\_name

Select the roles for the user in the specified database. All users are members of the public role in every database and cannot be removed. For more information about database roles, see Database-Level Roles.

Securables

The Securables page lists all possible securables and the permissions on those securables that can be granted to the login. The following options are available on this page:

Upper Grid

Contains one or more items for which permissions can be set. The columns that are displayed in the upper grid vary depending on the principal or securable.

To add items to the upper grid:

Lists the possible permissions for the securable that are selected in the upper grid. Not all options are available for all explicit permissions.

Permissions

Grant

Select to grant this permission to the login. Clear to revoke this permission.

With Grant

Reflects the state of the WITH GRANT option for the listed permission. This box is read-only. To apply this permission, use the GRANT statement.

Deny

Select to deny this permission to the login. Clear to revoke this permission.

Status

The Status page lists some of the authentication and authorization options that can be configured on the selected SQL Server login.

The following options are available on this page:

Permission to connect to database engine

When you work with this setting, you should think of the selected login as a principal that can be granted or denied permission on a securable.

Select Grant to grant CONNECT SQL permission to the login. Select Deny to deny CONNECT SQL to the login.

Login

When you work with this setting, you should think of the selected login as a record in a table. Changes to the values listed here will be applied to the record.

A login that has been disabled continues to exist as a record. But if it tries to connect to SQL Server, the login will not be authenticated.

Select this option to enable or disable this login. This option uses the ALTER LOGIN statement with the either ENABLE or DISABLE option.

SQL Server Authentication

The check box Login is locked out is only available if the selected login connects using SQL Server Authentication and the login has been locked out. This setting is read-only. To unlock a login that is locked out, execute ALTER LOGIN with the UNLOCK option.

Using Transact-SQL

To create a login using Windows Authentication

CREATE LOGIN [<domainName>\<loginName>] FROM WINDOWS;

GO

To create a login using SQL Server Authentication

CREATE LOGIN shcooper

WITH PASSWORD = 'Baz1nga' MUST\_CHANGE,

CREDENTIAL = RestrictedFaculty;

GO

For more information, see CREATE LOGIN (Transact-SQL).

Create a Database User

This topic describes how to create a database user mapped to a login in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The database user is the identity of the login when it is connected to a database. The database user can use the same name as the login, but that is not required. This topic assumes that a login already exists in SQL Server. For information about how to create a login, see Create a Login.

In This Topic

• Before you begin:

Background

Security

• To create a database user, using:

SQL Server Management Studio

Transact-SQL

Before You Begin

Background

A user is a database level security principal. Logins must be mapped to a database user to connect to a database. A login can be mapped to different databases as different users but can only be mapped as one user in each database. In a partially contained database, a user can be created that does not have a login. For more information about contained database users, see CREATE USER (Transact-SQL). If the guest user in a database is enabled, a login that is not mapped to a database user can enter the database as the guest user.

Security Note

The guest user is ordinarily disabled. Do not enable the guest user unless it is necessary.

As a security principal, permissions can be granted to users. The scope of a user is the database. To connect to a specific database on the instance of SQL Server, a login must be mapped to a database user. Permissions inside the database are granted and denied to the database user, not the login.

Security

Permissions

Requires ALTER ANY USER permission on the database.

Using SQL Server Management Studio

To create a database user

1. In Object Explorer, expand the Databases folder.

2. Expand the database in which to create the new database user.

3. Right-click the Security folder, point to New, and select User….

4. In the Database User – New dialog box, on the General page, select one of the following user types from the User type list: SQL user with login, SQL user without login, User mapped to a certificate, User mapped to an asymmetric key, or Windows user.

5. In the User name box, enter a name for the new user. If you have chosen Windows user from the User type list, you can also click the ellipsis (…) to open the Select User or Group dialog box.

6. In the Login name box, enter the login for the user. Alternately, click the ellipsis (…) to open the Select Login dialog box. Login name is available if you select either SQL user with login or Windows user from the User type list.

7. In the Default schema box, specifies the schema that will own objects created by this user. Alternately, click the ellipsis (…) to open the Select Schema dialog box. Default schema is available if you select either SQL user with login, SQL user without login, or Windows user from the User type list.

8. In the Certificate name box, enter the certificate to be used for the database user. Alternately, click the ellipsis (…) to open the Select Certificate dialog box. Certificate name is available if you select User mapped to a certificate from the User type list.

9. In the Asymmetric key name box, enter the key to be used for the database user. Alternately, click the ellipsis (…) to open the Select Asymmetric Key dialog box. Asymmetric key name is available if you select User mapped to an asymmetric key from the User type list.

10. Click OK.

Additional Options

The Database User – New dialog box also offers options on four additional pages: Owned Schemas, Membership, Securables, and Extended Properties.

• The Owned Schemas page lists all possible schemas that can be owned by the new database user. To add schemas to or remove them from a database user, under Schemas owned by this user, select or clear the check boxes next to the schemas.

• The Membership page lists all possible database membership roles that can be owned by the new database user. To add roles to or remove them from a database user, under Database role membership, select or clear the check boxes next to the roles.

• The Securables page lists all possible securables and the permissions on those securables that can be granted to the login.

• The Extended properties page allows you to add custom properties to database users. The following options are available on this page.

Database

Displays the name of the selected database. This field is read-only.

Collation

Displays the collation used for the selected database. This field is read-only.

Properties

View or specify the extended properties for the object. Each extended property consists of a name/value pair of metadata associated with the object.

Ellipsis (…)

Click the ellipsis (…) after Value to open the Value for Extended Property dialog box. Type or view the value of the extended property in this larger location. For more information, see Value for Extended Property Dialog Box.

Delete

Removes the selected extended property.

Using Transact-SQL

To create a database user

1. In Object Explorer, connect to an instance of Database Engine.

2. On the Standard bar, click New Query.

3. Copy and paste the following example into the query window and click Execute.

4. -- Creates the login AbolrousHazem with password '340$Uuxwp7Mcxo7Khy'.

5. CREATE LOGIN AbolrousHazem

6. WITH PASSWORD = '340$Uuxwp7Mcxo7Khy';

7. GO

8.

9. -- Creates a database user for the login created above.

10. CREATE USER AbolrousHazem FOR LOGIN AbolrousHazem;

11. GO

For more information, see CREATE USER (Transact-SQL).

Create a Database Schema

SQL Server 2016

Other Versions

This topic describes how to create a schema in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL.

In This Topic

• Before you begin:

Limitations and Restrictions

Security

• To create a schema, using:

SQL Server Management Studio

Transact-SQL

Before You Begin

Limitations and Restrictions

• The new schema is owned by one of the following database-level principals: database user, database role, or application role. Objects created within a schema are owned by the owner of the schema, and have a NULL principal\_id in sys.objects. Ownership of schema-contained objects can be transferred to any database-level principal, but the schema owner always retains CONTROL permission on objects within the schema.

• When creating a database object, if you specify a valid domain principal (user or group) as the object owner, the domain principal will be added to the database as a schema. The new schema will be owned by that domain principal.

Security

Permissions

• Requires CREATE SCHEMA permission on the database.

• To specify another user as the owner of the schema being created, the caller must have IMPERSONATE permission on that user. If a database role is specified as the owner, the caller must have one of the following: membership in the role or ALTER permission on the role.

Using SQL Server Management Studio

To create a schema

1. In Object Explorer, expand the Databases folder.

2. Expand the database in which to create the new database schema.

3. Right-click the Security folder, point to New, and select Schema.

4. In the Schema - New dialog box, on the General page, enter a name for the new schema in the Schema name box.

5. In the Schema owner box, enter the name of a database user or role to own the schema. Alternately, click Search to open the Search Roles and Users dialog box.

6. Click OK.

Additional Options

The Schema– New dialog box also offers options on two additional pages: Permissions and Extended Properties.

• The Permissions page lists all possible securables and the permissions on those securables that can be granted to the login.

• The Extended properties page allows you to add custom properties to database users.

Using Transact-SQL

To create a schema

1. In Object Explorer, connect to an instance of Database Engine.

2. On the Standard bar, click New Query.

3. Copy and paste the following example into the query window and click Execute.

4. USE AdventureWorks2012;

5. GO

6. -- Creates the schema Sprockets owned by Annik that contains table NineProngs.

7. -- The statement grants SELECT to Mandar and denies SELECT to Prasanna.

8.

9. CREATE SCHEMA Sprockets AUTHORIZATION Annik

10. CREATE TABLE NineProngs (source int, cost int, partnumber int)

11. GRANT SELECT ON SCHEMA::Sprockets TO Mandar

12. DENY SELECT ON SCHEMA::Sprockets TO Prasanna;

13. GO

For more information, see CREATE SCHEMA (Transact-SQL).

Join a Role

SQL Server 2016

Other Versions

Topic Status: Some information in this topic is preview and subject to change in future releases. Preview information describes new features or changes to existing features in Microsoft SQL Server 2016 Community Technology Preview 2 (CTP2).

This topic describes how to assign roles to logins and database users in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. Use roles in SQL Server to efficiently manage permissions. Assign permissions to roles, and then add and remove users and logins to the roles. By using roles, permissions do not have to be individually maintained for each user.

SQL Server supports four types of roles.

• Fixed server roles

• User-defined server roles

• Fixed database roles

• User-defined database roles

The fixed roles are automatically available in SQL Server. Fixed roles have the necessary permissions to accomplish common tasks. For more information about fixed roles, see the following links. User-defined roles are created by you, and can be customized with the permissions that you select. For more information about user-defined roles, see the following links.

In This Topic

• Before you begin:

Limitations and Restrictions

Security

• To assign roles to logins and database users, using:

SQL Server Management Studio

Transact-SQL

Before You Begin

Limitations and Restrictions

• Changing the name of a database role does not change ID number, owner, or permissions of the role.

• Database roles are visible in the sys.database\_role\_members and sys.database\_principals catalog views.

[Top]

Security

Permissions

Requires ALTER ANY ROLE permission on the database, ALTER permission on the role, or membership in db\_securityadmin.

[Top]

Using SQL Server Management Studio

To add a member to a fixed server role

1. In Object Explorer, expand the server in which you want to edit a fixed server role.

2. Expand the Security folder.

3. Expand the Server Roles folder

4. Right-click the role you want to edit and select Properties.

5. In the Server Role Properties – server\_role\_name dialog box, on the Members page, click Add.

6. In the Select Server Login or Role dialog box, under Enter the object names to select (examples), enter the login or server role to add to this server role. Alternately, click Browse… and select any or all of the available objects in the Browse for Objects dialog box. Click OK to return to the Server Role Properties – server\_role\_name dialog box.

7. Click OK.

To add a member to a user-defined database role

1. In Object Explorer, expand the server in which you want to edit a user-defined database role.

2. Expand the Databases folder.

3. Expand the database in which you want to edit a user-defined database role.

4. Expand the Security folder.

5. Expand the Roles folder.

6. Expand the Server Roles folder.

7. Right-click the role you want to edit and select Properties.

8. In the Database Role Properties – database\_role\_name dialog box, in the General page, click Add.

9. In the Select Database User or Role dialog box, under Enter the object names to select (examples), enter the login or database role to add to this database role. Alternately, click Browse… and select any or all of the available objects in the Browse for Objects dialog box. Click OK to return to the Database Role Properties – database\_role\_name dialog box.

10. Click OK.

[Top]

Using Transact-SQL

To add a member to a fixed server role

1. In Object Explorer, connect to an instance of Database Engine.

2. On the Standard bar, click New Query.

3. Copy and paste the following example into the query window and click Execute.

4. ALTER SERVER ROLE diskadmin ADD [Domain\Juan] ;

5. GO

For more information, see ALTER ROLE (Transact-SQL).

To add a member to a user-defined database role

1. In Object Explorer, connect to an instance of Database Engine.

2. On the Standard bar, click New Query.

3. Copy and paste the following example into the query window and click Execute.

4. ALTER ROLE Marketing ADD MEMBER [Domain\Juan] ;

5. GO

For more information, see sp\_addrolemember (Transact-SQL).

[Top]

See Also

Reference

Server-Level Roles

Database-Level Roles

Concepts

Application Roles

Grant a Permission to a Principal

SQL Server 2016

Other Versions

This topic describes how to grant permission to a principal in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL.

In This Topic

• Before you begin:

Limitations and Restrictions

Security

• To grant permission to a principal, using:

SQL Server Management Studio

Transact-SQL

Before You Begin

Limitations and Restrictions

Consider the following best practices that can make managing permissions easier.

• Grant permission to roles, instead of individual logins or users. When one individual is replaced by another, remove the departing individual from the role and add the new individual to the role. The many permissions that might be associated with the role will automatically be available to the new individual. If several people in an organization require the same permissions, adding each of them to the role will grant them the same permissions.

• Configure similar securables (tables, views, and procedures) to be owned by a schema, then grant permissions to the schema. For example, the payroll schema might own several tables, views, and stored procedures. By granting access to the schema, all the necessary permissions to perform the payroll function can be granted at the same time. For more information about what securables can be granted permissions, see Securables.

Security

Permissions

The grantor (or the principal specified with the AS option) must have either the permission itself with GRANT OPTION or a higher permission that implies the permission being granted. Members of the sysadmin fixed server role can grant any permission.

Using SQL Server Management Studio

To grant permission to a principal

1. In Object Explorer, expand the database that contains the object to which you want to grant permissions.

Note

These steps deal specifically with granting permissions to a stored procedure, but you can use similar steps to add permissions to tables, views, functions, and assemblies, as well as other securables. For more information, see GRANT (Transact-SQL)

2. Expand the Programmability folder.

3. Expand the Stored Procedures folder.

4. Right-click a stored procedure and select Properties.

5. In the Stored Procedure Properties – stored\_procedure\_name dialog box, under select a page, select Permissions. Use this page to add users or roles to the stored procedure and specify the permissions those users or roles have.

6. When finished, click OK.

Using Transact-SQL

To grant permission to a principal

1. In Object Explorer, connect to an instance of Database Engine.

2. On the Standard bar, click New Query.

3. Copy and paste the following example into the query window and click Execute.

4. -- Grants EXECUTE permission on stored procedure HumanResources.uspUpdateEmployeeHireInfo to an application role called Recruiting11.

5. USE AdventureWorks2012;

6. GO

7. GRANT EXECUTE ON OBJECT::HumanResources.uspUpdateEmployeeHireInfo

8. TO Recruiting11;

9. GO

For more information, see GRANT (Transact-SQL) and GRANT Object Permissions (Transact-SQL).

Server-Level Roles

SQL Server 2016

Other Versions

SQL Server provides server-level roles to help you manage the permissions on a server. These roles are security principals that group other principals. Server-level roles are server-wide in their permissions scope. (Roles are like groups in the Windows operating system.)

Fixed server roles are provided for convenience and backward compatibility. Assign more specific permissions whenever possible.

SQL Server provides nine fixed server roles. The permissions that are granted to the fixed server roles cannot be changed. Beginning with SQL Server 2012, you can create user-defined server roles and add server-level permissions to the user-defined server roles.

You can add server-level principals (SQL Server logins, Windows accounts, and Windows groups) into server-level roles. Each member of a fixed server role can add other logins to that same role. Members of user-defined server roles cannot add other server principals to the role.

Fixed Server-Level Roles

The following table shows the fixed server-level roles and their capabilities.

Fixed server-level role Description

sysadmin Members of the sysadmin fixed server role can perform any activity in the server.

serveradmin Members of the serveradmin fixed server role can change server-wide configuration options and shut down the server.

securityadmin Members of the securityadmin fixed server role manage logins and their properties. They can GRANT, DENY, and REVOKE server-level permissions. They can also GRANT, DENY, and REVOKE database-level permissions if they have access to a database. Additionally, they can reset passwords for SQL Server logins.

Security Note

The ability to grant access to the Database Engine and to configure user permissions allows the security admin to assign most server permissions. The securityadmin role should be treated as equivalent to the sysadmin role.

processadmin Members of the processadmin fixed server role can end processes that are running in an instance of SQL Server.

setupadmin Members of the setupadmin fixed server role can add and remove linked servers by using Transact-SQL statements. (sysadmin membership is needed when using Management Studio.)

bulkadmin Members of the bulkadmin fixed server role can run the BULK INSERT statement.

diskadmin The diskadmin fixed server role is used for managing disk files.

dbcreator Members of the dbcreator fixed server role can create, alter, drop, and restore any database.

public Every SQL Server login belongs to the public server role. When a server principal has not been granted or denied specific permissions on a securable object, the user inherits the permissions granted to public on that object. Only assign public permissions on any object when you want the object to be available to all users. You cannot change membership in public.

Note

public is implemented differently than other roles. However, permissions can be granted, denied, or revoked from public.

Permissions of Fixed Server Roles

Each fixed server role has certain permissions assigned to it. For a chart of the permissions assigned to the server roles, see Database Engine Fixed Server and Fixed Database Roles.

Important

The CONTROL SERVER permission is similar but not identical to the sysadmin fixed server role. Permissions do not imply role memberships and role memberships do not grant permissions. (E.g. CONTROL SERVER does not imply membership in the sysadmin fixed server role.) However, it is sometimes possible to impersonate between roles and equivalent permissions. Most DBCC commands and many system procedures require membership in the sysadmin fixed server role. For a list of 171 system stored procedures that require sysadmin membership, see the following blog post by Andreas Wolter CONTROL SERVER vs. sysadmin/sa: permissions, system procedures, DBCC, automatic schema creation and privilege escalation - caveats.

Server-Level Permissions

Only server-level permissions can be added to user-defined server roles. To list the server-level permissions, execute the following statement. The server-level permissions are:

Transact-SQL

SELECT \* FROM sys.fn\_builtin\_permissions('SERVER') ORDER BY permission\_name;

For more information about permissions, see Permissions (Database Engine) and sys.fn\_builtin\_permissions (Transact-SQL).

Working with Server-Level Roles

The following table explains the commands, views, and functions that you can use to work with server-level roles.

Feature Type Description

sp\_helpsrvrole (Transact-SQL)

Metadata Returns a list of server-level roles.

sp\_helpsrvrolemember (Transact-SQL)

Metadata Returns information about the members of a server-level role.

sp\_srvrolepermission (Transact-SQL)

Metadata Displays the permissions of a server-level role.

IS\_SRVROLEMEMBER (Transact-SQL)

Metadata Indicates whether a SQL Server login is a member of the specified server-level role.

sys.server\_role\_members (Transact-SQL)

Metadata Returns one row for each member of each server-level role.

sp\_addsrvrolemember (Transact-SQL)

Command Adds a login as a member of a server-level role. Deprecated. Use ALTER SERVER ROLE instead.

sp\_dropsrvrolemember (Transact-SQL)

Command Removes a SQL Server login or a Windows user or group from a server-level role. Deprecated. Use ALTER SERVER ROLE instead.

CREATE SERVER ROLE (Transact-SQL)

Command Creates a user-defined server role.

ALTER SERVER ROLE (Transact-SQL)

Command Changes the membership of a server role or changes name of a user-defined server role.

DROP SERVER ROLE (Transact-SQL)

Command Removes a user-defined server role.

IS\_SRVROLEMEMBER (Transact-SQL)

Function Determines membership of server role.

See Also

Database-Level Roles

Security Catalog Views (Transact-SQL)

Security Functions (Transact-SQL)

Securing SQL Server

GRANT Server Principal Permissions (Transact-SQL)

REVOKE Server Principal Permissions (Transact-SQL)

DENY Server Principal Permissions (Transact-SQL)

Create a Server Role

Database-Level Roles

SQL Server 2016

Other Versions

Updated: September 22, 2015

Applies To: Azure SQL Database, SQL Server 2016 Preview

To easily manage the permissions in your databases, SQL Server provides several roles which are security principals that group other principals. They are like groups in the Microsoft Windows operating system. Database-level roles are database-wide in their permissions scope.

There are two types of database-level roles in SQL Server: fixed database roles that are predefined in the database and flexible database roles that you can create.

Fixed database roles are defined at the database level and exist in each database. Members of the db\_owner database role can manage fixed database role membership. There are also some special-purpose fixed database roles in the msdb database.

You can add any database account and other SQL Server roles into database-level roles. Each member of a fixed database role can add other logins to that same role.

Important

Do not add flexible database roles as members of fixed roles. This could enable unintended privilege escalation.

The following table shows the fixed database-level roles and their capabilities. These roles exist in all databases.

Database-level role name Description

db\_owner Members of the db\_owner fixed database role can perform all configuration and maintenance activities on the database, and can also drop the database.

db\_securityadmin Members of the db\_securityadmin fixed database role can modify role membership and manage permissions. Adding principals to this role could enable unintended privilege escalation.

db\_accessadmin Members of the db\_accessadmin fixed database role can add or remove access to the database for Windows logins, Windows groups, and SQL Server logins.

db\_backupoperator Members of the db\_backupoperator fixed database role can back up the database.

db\_ddladmin Members of the db\_ddladmin fixed database role can run any Data Definition Language (DDL) command in a database.

db\_datawriter Members of the db\_datawriter fixed database role can add, delete, or change data in all user tables.

db\_datareader Members of the db\_datareader fixed database role can read all data from all user tables.

db\_denydatawriter Members of the db\_denydatawriter fixed database role cannot add, modify, or delete any data in the user tables within a database.

db\_denydatareader Members of the db\_denydatareader fixed database role cannot read any data in the user tables within a database.

msdb Roles

The msdb database contains the special-purpose roles that are shown in the following table.

msdb role name Description

db\_ssisadmin

db\_ssisoperator

db\_ssisltduser Members of these database roles can administer and use SSIS. Instances of SQL Server that are upgraded from an earlier version might contain an older version of the role that was named using Data Transformation Services (DTS) instead of SSIS. For more information, see Integration Services Roles (SSIS Service).

dc\_admin

dc\_operator

dc\_proxy Members of these database roles can administer and use the data collector. For more information, see Data Collection.

PolicyAdministratorRole Members of the db\_ PolicyAdministratorRole database role can perform all configuration and maintenance activities on Policy-Based Management policies and conditions. For more information, see Administer Servers by Using Policy-Based Management.

ServerGroupAdministratorRole

ServerGroupReaderRole Members of these database roles can administer and use registered server groups.

dbm\_monitor Created in the msdb database when the first database is registered in Database Mirroring Monitor. The dbm\_monitor role has no members until a system administrator assigns users to the role.

Important

Members of the db\_ssisadmin role and the dc\_admin role may be able to elevate their privileges to sysadmin. This elevation of privilege can occur because these roles can modify Integration Services packages and Integration Services packages can be executed by SQL Server using the sysadmin security context of SQL Server Agent. To guard against this elevation of privilege when running maintenance plans, data collection sets, and other Integration Services packages, configure SQL Server Agent jobs that run packages to use a proxy account with limited privileges or only add sysadmin members to the db\_ssisadmin and dc\_admin roles.

Working with Database-Level Roles

The following table explains the commands, views and functions for working with database-level roles.

Feature Type Description

sp\_helpdbfixedrole (Transact-SQL)

Metadata Returns a list of the fixed database roles.

sp\_dbfixedrolepermission (Transact-SQL)

Metadata Displays the permissions of a fixed database role.

sp\_helprole (Transact-SQL)

Metadata Returns information about the roles in the current database.

sp\_helprolemember (Transact-SQL)

Metadata Returns information about the members of a role in the current database.

sys.database\_role\_members (Transact-SQL)

Metadata Returns one row for each member of each database role.

IS\_MEMBER (Transact-SQL)

Metadata Indicates whether the current user is a member of the specified Microsoft Windows group or Microsoft SQL Server database role.

CREATE ROLE (Transact-SQL)

Command Creates a new database role in the current database.

ALTER ROLE (Transact-SQL)

Command Changes the name of a database role.

DROP ROLE (Transact-SQL)

Command Removes a role from the database.

sp\_addrole (Transact-SQL)

Command Creates a new database role in the current database.

sp\_droprole (Transact-SQL)

Command Removes a database role from the current database.

sp\_addrolemember (Transact-SQL)

Command Adds a database user, database role, Windows login, or Windows group to a database role in the current database.

sp\_droprolemember (Transact-SQL)

Command Removes a security account from a SQL Server role in the current database.

public Database Role

Every database user belongs to the public database role. When a user has not been granted or denied specific permissions on a securable object, the user inherits the permissions granted to public on that object.

Related Content

Security Catalog Views (Transact-SQL)

Security Stored Procedures (Transact-SQL)

Security Functions (Transact-SQL)

Securing SQL Server

sp\_helprotect (Transact-SQL)

Application Roles

SQL Server 2016

Other Versions

An application role is a database principal that enables an application to run with its own, user-like permissions. You can use application roles to enable access to specific data to only those users who connect through a particular application. Unlike database roles, application roles contain no members and are inactive by default. Application roles work with both authentication modes. Application roles are enabled by using sp\_setapprole, which requires a password. Because application roles are a database-level principal, they can access other databases only through permissions granted in those databases to guest. Therefore, any database in which guest has been disabled will be inaccessible to application roles in other databases.

In SQL Server, application roles cannot access server-level metadata because they are not associated with a server-level principal. To disable this restriction and thereby allow application roles to access server-level metadata, set the global flag 4616. For more information, see Trace Flags (Transact-SQL) and DBCC TRACEON (Transact-SQL).

Connecting with an Application Role

The following steps make up the process by which an application role switches security contexts:

1. A user executes a client application.

2. The client application connects to an instance of SQL Server as the user.

3. The application then executes the sp\_setapprole stored procedure with a password known only to the application.

4. If the application role name and password are valid, the application role is enabled.

5. At this point the connection loses the permissions of the user and assumes the permissions of the application role.

The permissions acquired through the application role remain in effect for the duration of the connection.

In earlier versions of SQL Server, the only way for a user to reacquire its original security context after starting an application role is to disconnect and reconnect to SQL Server. Beginning with SQL Server 2005, sp\_setapprole has an option that creates a cookie. The cookie contains context information before the application role is enabled. The cookie can be used by sp\_unsetapprole to revert the session to its original context. For information about this new option and an example, see sp\_setapprole (Transact-SQL).

Security Note

The ODBC encrypt option is not supported by SqlClient. When you are transmitting confidential information over a network, use Secure Sockets Layer (SSL) or IPsec to encrypt the channel. If you must persist credentials in the client application, encrypt the credentials by using the crypto API functions. In SQL Server 2005 and later versions, the parameter password is stored as a one-way hash.

Related Tasks

Create an application role. Create an Application Role and CREATE APPLICATION ROLE (Transact-SQL)

Alter an application role. ALTER APPLICATION ROLE (Transact-SQL)

Delete an application role. DROP APPLICATION ROLE (Transact-SQL)

Using an application role. sp\_setapprole (Transact-SQL)

See Also

Securing SQL Server