/*

This script will script the role members for all roles on the database.

This is useful for scripting permissions in a development environment before refreshing
  development with a copy of production. This will allow us to easily ensure
  development permissions are not lost during a prod to dev restoration.

URL for this script: http://www.sqlservercentral.com/scripts/login/138379/
  Old URL http://www.sqlservercentral.com/scripts/Security/71562/ -- Periodically, updates are made to this script so check out the URL for updates.

Author: S. Kusen

Updates:
2017-07-10 v4.3:
I was unable to easily get this into a stored procedure / powershell script, so this update includes the changes/updates noted here:
	1. Incorporated Andrew G's updates from previous feedback (Much delayed to being updated on the main script page). Thanks Andrew!
	2. danmeskel2002 recommended a fix for the SID issue for "SQL User without login".
		Changed this line:
			SID = '' + CONVERT(varchar(1000), sid)
			to
			SID = '' + CONVERT(varchar(1000), sid, 1)

2016-10-31: AG 1. Added extended stored procedures and system object permissions for master database in OBJECT LEVEL PERMISSIONS area by removing join to sys.objects and using functions instead
	2. Added EXISTS check to all statements
	3. Added CREATE ROLE before adding principals to roles

2016-08-25: AG 1. Remove default database being specified for an AD group user as this option causes a failure on create

2015-08-21:
	1. Modified section 3.1 to load to a temp table and populate different users based on an error in 2005/2008 because of the update made for contained databases. Thanks to Andrew G for pointing that out.
	2. Altered section 4.1 to include COLLATE DATABASE_DEFAULT in the join statement. Thanks to Andrew G and PHXHoward for pointing that out.

2015-06-30:
	1. Re-numbered all sections based on additional updates being added inline.
	2. Added sections 8, 8.1; From Eddict, user defined types needed to be added.

3. Added sections 4, 4.1; From nhaberl, for orphaned users mapping (if logins don't exist, they will not be created by this script).
4. Updated section 3.1; From nhaberl, updated to include a default schema of dbo.

2014-07-25: Fix pointed out by virgo for where logins are mapped to users that are a different name. Changed ***+ ' FOR LOGIN ' + QUOTENAME([name]) +*** to ***+ ' FOR LOGIN ' + QUOTENAME(suser_sname([sid])) +***.

2014-01-24: Updated to account for 2012 contained db users

2012-05-14: Incorporated a fix pointed out by aruopna for Schema-level permissions.

2010-01-20: Turned statements into a cursor and then using print statements to make it easier to
copy/paste into a query window.
Added support for schema level permissions

Thanks to wsoranno@winona.edu and choffman for the recommendations.

*/
SET NOCOUNT ON

/*Prep statements*/
IF OBJECT_ID('tempdb..##tbl_db_principals_statements') IS NOT NULL DROP TABLE ##tbl_db_principals_statements
CREATE TABLE ##tbl_db_principals_statements (stmt varchar(max), result_order decimal(4,1))
IF ((SELECT SUBSTRING(convert(sysname, SERVERPROPERTY('productversion')), 1, charindex('.',convert(sysname, SERVERPROPERTY('productversion')))-1)) > 10)
EXEC ('
INSERT INTO ##tbl_db_principals_statements (stmt, result_order)
SELECT
CASE WHEN rm.authentication_type IN (2, 0) /* 2=contained database user with password, 0 =user without login; create users without logins*/ THEN (''IF NOT EXISTS (SELECT [name] FROM sys.database_principals WHERE [name] = '' + SPACE(1) + '''''' + [name] + '''''' + '') BEGIN CREATE USER '' + SPACE(1) + QUOTENAME([name]) + '' WITHOUT LOGIN WITH DEFAULT_SCHEMA = '' + QUOTENAME([default_schema_name]) + SPACE(1) + '', SID = '' + CONVERT(varchar(1000), sid, 1) + SPACE(1) + '' END; '')
ELSE (''IF NOT EXISTS (SELECT [name] FROM sys.database_principals WHERE [name] = '' + SPACE(1) + '''''' + [name] + '''''' + '') BEGIN CREATE USER '' + SPACE(1) + QUOTENAME([name]) + '' FOR LOGIN '' + QUOTENAME(suser_sname([sid])) + CASE WHEN [type] <>''G'' THEN '' WITH DEFAULT_SCHEMA = '' + QUOTENAME(ISNULL([default_schema_name], ''dbo'')) ELSE '''' END + SPACE(1) + ''END; '')

```
        END AS [-- SQL STATEMENTS --],
        3.1 AS [-- RESULT ORDER HOLDER --]
   FROM   sys.database_principals AS rm
   WHERE [type] IN ("U", "S", "G") /* windows users, sql users, windows groups */
     AND NAME <> "guest"')

ELSE IF ((SELECT SUBSTRING(convert(sysname, SERVERPROPERTY('productversion')),
1, charindex('.',convert(sysname, SERVERPROPERTY('productversion')))-1)) IN (9,10))
EXEC ('
INSERT INTO ##tbl_db_principals_statements (stmt, result_order)
   SELECT   ("IF NOT EXISTS (SELECT [name] FROM sys.database_principals WHERE
[name] = " + SPACE(1) + """" + [name] + """" + ") BEGIN CREATE USER " + SPACE(1) +
QUOTENAME([name]) + " FOR LOGIN " + QUOTENAME(suser_sname([sid])) + CASE
WHEN [type] <>"G" THEN " WITH DEFAULT_SCHEMA = " +
QUOTENAME(ISNULL([default_schema_name], "dbo")) ELSE "" END + SPACE(1) + "END;
") AS [-- SQL STATEMENTS --],
        3.1 AS [-- RESULT ORDER HOLDER --]
   FROM   sys.database_principals AS rm
   WHERE [type] IN ("U", "S", "G") /* windows users, sql users, windows groups */
   AND NAME <> "guest"')

--SELECT * FROM ##tbl_db_principals_statements




DECLARE
    @sql VARCHAR(2048)
    ,@sort INT

DECLARE tmp CURSOR FOR


/*********************************************/
/*********   DB CONTEXT STATEMENT    *********/
/*********************************************/
SELECT '-- [-- DB CONTEXT --] --' AS [-- SQL STATEMENTS --],
      1 AS [-- RESULT ORDER HOLDER --]
UNION
SELECT   'USE' + SPACE(1) + QUOTENAME(DB_NAME()) AS [-- SQL STATEMENTS --],
      1.1 AS [-- RESULT ORDER HOLDER --]

UNION

SELECT '' AS [-- SQL STATEMENTS --],
      2 AS [-- RESULT ORDER HOLDER --]
```

```
UNION

/*********************************************/
/*********     DB USER CREATION      *********/
/*********************************************/

  SELECT '-- [-- DB USERS --] --' AS [-- SQL STATEMENTS --],
      3 AS [-- RESULT ORDER HOLDER --]
  UNION

  SELECT
    [stmt],
      3.1 AS [-- RESULT ORDER HOLDER --]
  FROM  ##tbl_db_principals_statements
  --WHERE [type] IN ('U', 'S', 'G') -- windows users, sql users, windows groups
  WHERE [stmt] IS NOT NULL

UNION

/*********************************************/
/*********    MAP ORPHANED USERS     *********/
/*********************************************/

SELECT '-- [-- ORPHANED USERS --] --' AS [-- SQL STATEMENTS --],
    4 AS [-- RESULT ORDER HOLDER --]
UNION
SELECT   'ALTER USER [' + rm.name + '] WITH LOGIN = [' + rm.name + ']',
    4.1 AS [-- RESULT ORDER HOLDER --]
FROM   sys.database_principals AS rm
 Inner JOIN sys.server_principals as sp
 ON rm.name = sp.name COLLATE DATABASE_DEFAULT and rm.sid <> sp.sid
WHERE rm.[type] IN ('U', 'S', 'G') -- windows users, sql users, windows groups
 AND rm.name NOT IN ('dbo', 'guest', 'INFORMATION_SCHEMA', 'sys',
'MS_DataCollectorInternalUser')

UNION

/*********************************************/
/*********    DB ROLE PERMISSIONS    *********/
/*********************************************/
SELECT '-- [-- DB ROLES --] --' AS [-- SQL STATEMENTS --],
    5 AS [-- RESULT ORDER HOLDER --]
UNION

SELECT   'IF DATABASE_PRINCIPAL_ID(' + QUOTENAME([name],'') COLLATE
```

```
database_default + ') IS NULL' + SPACE(1) + 'CREATE ROLE'
  + SPACE(1) + QUOTENAME([name]),
    5.1 AS [-- RESULT ORDER HOLDER --]
FROM sys.database_principals
WHERE [type] ='R' -- R = Role
  AND [is_fixed_role] = 0
--ORDER BY [name] ASC
UNION

SELECT   'IF DATABASE_PRINCIPAL_ID(' +
QUOTENAME(USER_NAME(rm.member_principal_id),'') COLLATE database_default + ')
IS NOT NULL' + SPACE(1) + 'EXEC sp_addrolemember @rolename ='
  + SPACE(1) + QUOTENAME(USER_NAME(rm.role_principal_id), '') COLLATE
database_default + ', @membername =' + SPACE(1) +
QUOTENAME(USER_NAME(rm.member_principal_id), '') COLLATE database_default AS
[-- SQL STATEMENTS --],
    5.2 AS [-- RESULT ORDER HOLDER --]
FROM   sys.database_role_members AS rm
WHERE   USER_NAME(rm.member_principal_id) IN (
                        --get user names on the database
                        SELECT [name]
                        FROM sys.database_principals
                        WHERE [principal_id] > 4 -- 0 to 4 are system users/schemas
                        and [type] IN ('G', 'S', 'U') -- S = SQL user, U = Windows user, G =
Windows group
                       )
--ORDER BY rm.role_principal_id ASC


UNION

SELECT '' AS [-- SQL STATEMENTS --],
    7 AS [-- RESULT ORDER HOLDER --]

UNION

/*********************************************/
/*********  OBJECT LEVEL PERMISSIONS *********/
/*********************************************/
SELECT '-- [-- OBJECT LEVEL PERMISSIONS --] --' AS [-- SQL STATEMENTS --],
    7.1 AS [-- RESULT ORDER HOLDER --]
UNION
SELECT   'IF DATABASE_PRINCIPAL_ID(' +
QUOTENAME(USER_NAME(usr.principal_id),'') COLLATE database_default + ') IS NOT
NULL' + SPACE(1) +
    CASE
```

```
        WHEN perm.state <> 'W' THEN perm.state_desc
        ELSE 'GRANT'
      END
     + SPACE(1) + perm.permission_name + SPACE(1) + 'ON ' +
QUOTENAME(OBJECT_SCHEMA_NAME(perm.major_id)) + '.' +
QUOTENAME(OBJECT_NAME(perm.major_id)) --select, execute, etc on specific objects
     + CASE
          WHEN cl.column_id IS NULL THEN SPACE(0)
          ELSE '(' + QUOTENAME(cl.name) + ')'
       END
     + SPACE(1) + 'TO' + SPACE(1) + QUOTENAME(USER_NAME(usr.principal_id))
COLLATE database_default
     + CASE
          WHEN perm.state <> 'W' THEN SPACE(0)
          ELSE SPACE(1) + 'WITH GRANT OPTION'
       END
        AS [-- SQL STATEMENTS --],
     7.2 AS [-- RESULT ORDER HOLDER --]
FROM
  sys.database_permissions AS perm

  /* No join to sys.objects as it excludes system objects such as extended stored procedures
*/
  /*   INNER JOIN
  sys.objects AS obj
       ON perm.major_id = obj.[object_id]
  */
     INNER JOIN
  sys.database_principals AS usr
       ON perm.grantee_principal_id = usr.principal_id
     LEFT JOIN
  sys.columns AS cl
       ON cl.column_id = perm.minor_id AND cl.[object_id] = perm.major_id
 WHERE /* Include System objects when scripting permissions for master, exclude
elsewhere */
     (    DB_NAME() <> 'master' AND perm.major_id IN (SELECT [object_id] FROM
sys.objects WHERE type NOT IN ('S'))
       OR DB_NAME() =  'master'
       )



--WHERE   usr.name = @OldUser
--ORDER BY perm.permission_name ASC, perm.state_desc ASC
```

```
UNION

/*****************************************/
/*********  TYPE LEVEL PERMISSIONS *********/
/*****************************************/
SELECT '-- [-- TYPE LEVEL PERMISSIONS --] --' AS [-- SQL STATEMENTS --],
       8 AS [-- RESULT ORDER HOLDER --]
UNION
SELECT  'IF DATABASE_PRINCIPAL_ID(' +
QUOTENAME(USER_NAME(usr.principal_id),'') COLLATE database_default + ') IS NOT
NULL' + SPACE(1) +
     CASE
          WHEN perm.state <> 'W' THEN perm.state_desc
          ELSE 'GRANT'
       END
       + SPACE(1) + perm.permission_name + SPACE(1) + 'ON ' +
QUOTENAME(SCHEMA_NAME(tp.schema_id)) + '.' + QUOTENAME(tp.name) --select,
execute, etc on specific objects
       + SPACE(1) + 'TO' + SPACE(1) + QUOTENAME(USER_NAME(usr.principal_id))
COLLATE database_default
       + CASE
            WHEN perm.state <> 'W' THEN SPACE(0)
            ELSE SPACE(1) + 'WITH GRANT OPTION'
         END
          AS [-- SQL STATEMENTS --],
       8.1 AS [-- RESULT ORDER HOLDER --]
FROM
   sys.database_permissions AS perm
       INNER JOIN
   sys.types AS tp
          ON perm.major_id = tp.user_type_id
       INNER JOIN
   sys.database_principals AS usr
          ON perm.grantee_principal_id = usr.principal_id


UNION

SELECT '' AS [-- SQL STATEMENTS --],
  9 AS [-- RESULT ORDER HOLDER --]

UNION

/*****************************************/
/*********    DB LEVEL PERMISSIONS   *********/
/*****************************************/
```

```
SELECT '-- [--DB LEVEL PERMISSIONS --] --' AS [-- SQL STATEMENTS --],
     10 AS [-- RESULT ORDER HOLDER --]
UNION
SELECT   'IF DATABASE_PRINCIPAL_ID(' +
QUOTENAME(USER_NAME(usr.principal_id),"") COLLATE database_default + ') IS NOT
NULL' + SPACE(1) +
     CASE
       WHEN perm.state <> 'W' THEN perm.state_desc --W=Grant With Grant Option
       ELSE 'GRANT'
     END
  + SPACE(1) + perm.permission_name --CONNECT, etc
  + SPACE(1) + 'TO' + SPACE(1) + '[' + USER_NAME(usr.principal_id) + ']' COLLATE
database_default --TO <user name>
  + CASE
       WHEN perm.state <> 'W' THEN SPACE(0)
       ELSE SPACE(1) + 'WITH GRANT OPTION'
    END
     AS [-- SQL STATEMENTS --],
     10.1 AS [-- RESULT ORDER HOLDER --]
FROM   sys.database_permissions AS perm
  INNER JOIN
  sys.database_principals AS usr
  ON perm.grantee_principal_id = usr.principal_id
--WHERE   usr.name = @OldUser

WHERE   [perm].[major_id] = 0
  AND [usr].[principal_id] > 4 -- 0 to 4 are system users/schemas
  AND [usr].[type] IN ('G', 'S', 'U') -- S = SQL user, U = Windows user, G = Windows group

UNION

SELECT '' AS [-- SQL STATEMENTS --],
     11 AS [-- RESULT ORDER HOLDER --]

UNION

SELECT '-- [--DB LEVEL SCHEMA PERMISSIONS --] --' AS [-- SQL STATEMENTS --],
     12 AS [-- RESULT ORDER HOLDER --]
UNION
SELECT   'IF DATABASE_PRINCIPAL_ID(' +
QUOTENAME(USER_NAME(grantee_principal_id),"") COLLATE database_default + ') IS
NOT NULL' + SPACE(1) +
       CASE
       WHEN perm.state <> 'W' THEN perm.state_desc --W=Grant With Grant Option
       ELSE 'GRANT'
       END
```

```
        + SPACE(1) + perm.permission_name --CONNECT, etc
        + SPACE(1) + 'ON' + SPACE(1) + class_desc + '::' COLLATE database_default --TO
<user name>
        + QUOTENAME(SCHEMA_NAME(major_id))
        + SPACE(1) + 'TO' + SPACE(1) +
QUOTENAME(USER_NAME(grantee_principal_id)) COLLATE database_default
        + CASE
          WHEN perm.state <> 'W' THEN SPACE(0)
          ELSE SPACE(1) + 'WITH GRANT OPTION'
          END
       AS [-- SQL STATEMENTS --],
    12.1 AS [-- RESULT ORDER HOLDER --]
from sys.database_permissions AS perm
  inner join sys.schemas s
    on perm.major_id = s.schema_id
  inner join sys.database_principals dbprin
    on perm.grantee_principal_id = dbprin.principal_id
WHERE class = 3 --class 3 = schema

ORDER BY [-- RESULT ORDER HOLDER --]


OPEN tmp
FETCH NEXT FROM tmp INTO @sql, @sort
WHILE @@FETCH_STATUS = 0
BEGIN
      PRINT @sql
      FETCH NEXT FROM tmp INTO @sql, @sort
END

CLOSE tmp
DEALLOCATE tmp

IF OBJECT_ID('tempdb..##tbl_db_principals_statements') IS NOT NULL DROP TABLE
##tbl_db_principals_statements
```