CHU Security Applier

for SQL Server

**Révisions du document**

|  |  |  |  |
| --- | --- | --- | --- |
| Qui ? | Quand ? | Version | Commentaire |
| Jefferson Elias | 16/04/2014 | 0.0.1 | Création |
|  | 03/06/2014 | 0.0.2 | Quelques ajouts |
|  | 25/11/2014 | 0.0.3 | * Ajout d’une commande pour exécution à distance * Ajout de quelques exemples de commandes * Ajout d’un rôle au standard : prog\_executors * Ajout d’exemple de commandes pour créer le rôle prog\_executors et les privilèges associés + hiérarchie |
|  | 26/11/2014 | 0.0.4 | Ajout exemple de commande pour ajouter automatiquement tous les membres du rôle sai\_ra pour le MTECH pour les bases de données des serveurs MTECH où il est défini. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Versions de logiciel concernées**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Editeur | Nom | Version |
|  | RDBMS | Microsoft | SQL Server | 2005 |
|  | RDBMS | Microsoft | SQL Server | 2008 |
| X | RDBMS | Microsoft | SQL Server | 2008 R2 |
|  | RDBMS | Oracle | Oracle Database | 9.2 |
|  | RDBMS | Oracle | Oracle Database | 11.1 |
|  | RDBMS | Oracle | Oracle Database | 11.2 |
|  | RDBMS | Oracle | Oracle Database | 12.1 |
| X | Programme | CHULg/SAI/DBA | CHU Security Applier  For SQL Server | 0.0.1 |

**Table des matières**

[1 Introduction 3](#_Toc404770360)

[1.1 Contexte 3](#_Toc404770361)

[2 Documentation du schéma 4](#_Toc404770362)

[2.1 Tables 4](#_Toc404770363)

[2.2 Vues 5](#_Toc404770364)

[2.3 Fonctions scalaires 5](#_Toc404770365)

[3 Cas pratiques 6](#_Toc404770366)

[3.1 Ajout d’un nouveau contact 6](#_Toc404770367)

[3.2 Ajout d’un contact sur un serveur 6](#_Toc404770368)

[3.3 Ajout d’un contact sur une base de données 6](#_Toc404770369)

[3.4 Définition d’un rôle standard d’accès pour un contact donné 7](#_Toc404770370)

[3.5 Définition d’un rôle non standard d’accès pour une DB donnée 7](#_Toc404770371)

[3.6 Attribution d’un privilège sur un objet à un utilisateur ou à un rôle 7](#_Toc404770372)

[3.7 Ajout d’un membre à un rôle non standard 8](#_Toc404770373)

[3.8 Ajout d’un rôle standard et attribution des privilèges 8](#_Toc404770374)

[3.9 Créer le rôle sai\_ra pour les DBs référencées qui auraient été oubliées 9](#_Toc404770375)

[3.10 Ajout des membres du MTECH au rôle sai\_ra pour les DB des serveurs dont ils ont la charge. 9](#_Toc404770376)

[3.11 Ajouter le privilège SELECT sur dbo.SchemaChangeLog à tous les sai\_ra 11](#_Toc404770377)

[3.12 Génération du script d’application de la sécurité 12](#_Toc404770378)

# Introduction

## Contexte

Le présent document décrit la solution de mise en application du standard de sécurité développée au CHU de Liège pour les bases de données SQL Server.

Comme expliqué dans ce standard, le principe d’attribution des permissions tout juste nécessaires à la bonne exécution de l’application est prépondérant. Les permissions sont assignées soit sur des objets bien particuliers, soit sur des schémas de base de données. Toute autre permission qu’elle soit assignée au niveau d’une base de données ou au niveau d’une instance étant proscrite par le standard pour tout utilisateur non administrateur (DBA, gestionnaire de backup, etc.), leur application n’est donc pas actuellement implémentée dans la solution.

Le standard définit un certain nombre de rôles pour l’octroi des permissions au niveau schéma, et ce afin de faciliter la gestion de ces permissions. Ainsi, dans le cas de privilèges au niveau schéma, le standard définit des rôles auxquels sont assignés les droits strictement nécessaires pour effectuer certaines actions. Ces rôles suivent une nomenclature stricte.

data\_reader

data\_modifier

struct\_viewer

struct\_modifier

enduser

manager

full\_access

responsible

prog\_executors

Ce diagramme est extrait de la définition du standard et peut encore changer. La solution devait donc être évolutive. La possibilité de créer des rôles autres que ceux du standard doit également rester possible.

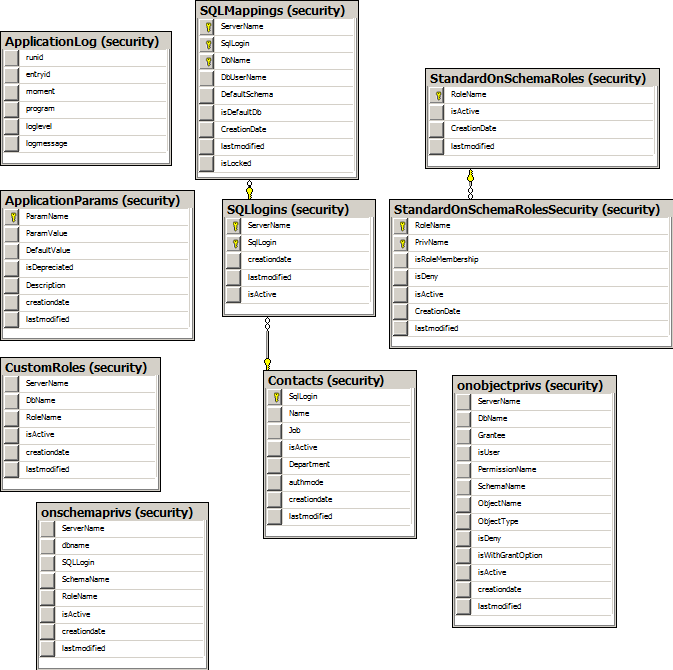
La solution se veut dynamique en ce sens que le programme tire ses paramètres dans des tables en base de données et que le résultat fourni est dépendant du paramétrage effectué dans celles-ci.

La solution n’applique pas directement les commandes générées sur le serveur demandé, mais délivre un script que son utilisateur peut parcourir et valider avant de l’appliquer sur ce serveur.

# Documentation du schéma

## Tables

Vous trouverez ci-dessous les tables du schéma de base de données correspondant à la solution.



Ainsi, nous trouvons une table « **Contacts** » dans laquelle se trouvera la liste des contacts qui peuvent se connecter à une instance, auquel cas ces contacts se voient associés une entrée dans la table « **SQLLogin** ». Vu que dans une instance SQL Server, il y a plusieurs bases de données, une liaison entre les logins SQL et les utilisateurs de base de données doit être prévue. Ces informations sont stockées dans la table « **SQLMappings** ».

Outre ces tables, une table nommée « **ApplicationParams** » constitue une table où seront stockés différents paramètres (la version de la solution, le niveau de journalisation, etc.). La table « **ApplicationLog** » constitue un journal général pour l’exécution d’applications.

La table « **CustomRoles** » permet de définir des rôles qui sortent du standard pour une base de données sur un serveur donné tandis que la table « **StandardOnSchemaRoles** » reprend les rôles définis dans le standard. Les permissions et l’adhésion d’un rôle standard à un autre rôle sont tous stockés dans la table « **StandardOnSchemaRolesSecurity** ». L’attribution d’un rôle à un contact pour un serveur et une base de données est quant à elle faite dans la table « **OnSchemaPrivs** ». Les contacts ne peuvent avoir 2 rôles standards pour une même base de données. Si cela devait être le cas, le standard prévoit des rôles composés.

Enfin, l’attribution des permissions sur des objets est définie dans la table « **OnObjectPrivs** ».

*Depuis la version 0.0.2*, une table « **CustomRoleMembers** » a été ajoutée afin d’ajouter des membres des rôles définis dans la table « CustomRoles ».

## Vues

Quatre vues ont été définies pour une utilisation utilisateur ainsi que pour les procédures ou fonctions que nous verrons ci-après. Ces vues sont :

1. « **Logins** » : fusion de la table « **Contacts** » avec la table « **SQLLogin** »
2. « **DatabaseUsers** » : liste les utilisateurs de base de données associés à une occurrence de la table « **Contacts** ».
3. « **Login2DbUserMappings** » : information détaillée des « **SQLMappings** » sur base à la fois des « **Contacts** » et des « **SQLLogin** ».
4. « **ReferencedSchemas** » : liste tous les schémas référencés dans les différentes tables en remontant l’information de sa provenance.

## Fonctions scalaires

Il y a différentes fonctions scalaires. Chacune permet de générer une partie bien définie de la sécurité. Une fonction cependant permet de générer la totalité : « **getCmds4ServerSecurity** ».

Ainsi, toutes les fonctions commençant par « getCmds » génèrent des commandes utilisables et leur nom est suffisamment significatif pour ne rien dire de plus :

* getCmds4AllSchemasCreation
* getCmds4CustomRoles
* getCmds4DbUsersCreation
* getCmds4Login2DbUserMappings
* getCmds4LoginsCreation
* getCmds4ObjectPermissionAssignments
* getCmds4SchemaCreation
* getCmds4SchemaRoleAssignments
* getCmds4StandardRoles
* getCmds4StandardRolesOnDb

Outre ces fonctions, il y a aussi un certain nombre de fonctions « utilitaires » aux noms toujours aussi significatifs :

* getDbUserCreationStatement
* getLogin2DbUserMappingStatement
* getLoginCreationStatement

# Cas pratiques

## Ajout d’un nouveau contact

|  |
| --- |
| insert into [security].[Contacts]  (SqlLogin,Name,job,isActive,Department,authmode)  values (  'CHULG\c172610',  'Thomas Vanbrabant',  'BI Developer',  1,  'CHU Liege/Service T.B.E.',  'WINDOWS'  ); |

## Ajout d’un contact sur un serveur

*Attention : le contact doit être créé préalablement (cf. ).*

|  |
| --- |
| insert into security.SQLlogins (  ServerName,  SqlLogin,  isActive  )  values (  'SI-S-SERV53',  'CHULG\c172610',  1  ) |

## Ajout d’un contact sur une base de données

*Attention : le login doit être créé préalablement pour ce serveur (cf )*

|  |
| --- |
| insert into security.SQLMappings (  ServerName,  SqlLogin,  DbName,  DbUserName,  DefaultSchema,  isDefaultDb,  isLocked  )  values (  'SI-S-SERV53',  'CHULG\c172610',  'ASCLEPIOS\_DWH',  'CHULG\c172610',  'asclepios',  1,  0  ) |

## Définition d’un rôle standard d’accès pour un contact donné

*Attention : le login doit avoir accès à cette base de données sur ce serveur (Cf. )*

|  |
| --- |
| insert into security.onschemaprivs (  ServerName,  dbname,  SQLLogin,  SchemaName,  RoleName,  isActive  )  values (  'SI-S-SERV53',  'ASCLEPIOS\_DWH',  'CHULG\c172610',  'asclepios',  'full\_access',  1  ) |

## Définition d’un rôle non standard d’accès pour une DB donnée

|  |
| --- |
| insert into security.CustomRoles (  ServerName,  DbName,  RoleName,  isActive  )  values (  'SI-S-SERV53',  'DBA',  'sai\_ra',  1  ); |

## Attribution d’un privilège sur un objet à un utilisateur ou à un rôle

|  |
| --- |
| insert into security.onobjectprivs (  ServerName,  DbName,  Grantee,  isUser,  PermissionName,  SchemaName,  ObjectName,  ObjectType,  isDeny,  isWithGrantOption,  isActive  )  values (  'SI-S-SERV53',  'DBA',  'sai\_ra',  0,  'SELECT',  'dbo',  'SchemaChangeLog',  'TABLE',  0, 0, 1  ); |

## Ajout d’un membre à un rôle non standard

*Attention : le rôle non standard doit préalablement exister (Cf. )*

|  |
| --- |
| insert into security.CustomRoleMembers (  ServerName,  DbName,  RoleName,  MemberName,  MemberIsRole,  Reason,  isActive  )  values (  'SI-S-SERV308',  'msdb',  'SQLAgentUserRole',  'ppmgr',  'Permettre à ppmgr de créer des jobs SQLAgent',  0  ) |

## Ajout d’un rôle standard et attribution des privilèges

*Attention : Tout rôle standard ou non doit préalablement exister (Cf. )*

|  |
| --- |
| insert into security.StandardOnSchemaRoles (  RoleName,  isActive  )  values (  'prog\_executors',  1  )  insert into security.StandardOnSchemaRolesSecurity (  RoleName,  PrivName,  isRoleMembership,  isDeny,  isActive  )  values (  'prog\_executors',  'EXECUTE',  0,  0,  1  )  insert into security.StandardOnSchemaRolesSecurity (  RoleName,  PrivName,  isRoleMembership,  isDeny,  isActive  )  values (  'endusers',  'prog\_executors',  1,  0,  1  ) |

## Créer le rôle sai\_ra pour les DBs référencées qui auraient été oubliées

|  |
| --- |
| BEGIN TRAN  insert into [security].[CustomRoles] (  ServerName ,  DbName,  RoleName,  isActive,  Reason  )  select s.ServerName,s.DbName, 'sai\_ra', 1,'Actions permises aux responsables applicatifs sur la DB mentionnée.'  from (  select ServerName,DbName  from security.SQLMappings  EXCEPT (  select ServerName,DbName  from [security].[CustomRoles]  where rolename = 'sai\_ra'  )  ) s  COMMIT; |

## Ajout des membres du MTECH au rôle sai\_ra pour les DB des serveurs dont ils ont la charge.

*Attention :*

* *Le rôle sai\_ra doit avoir été défini.*
* *Cette requête sera revue si la solution vient à être utilisée dans le cadre d’un inventaire.*

|  |
| --- |
| merge  security.[CustomRoleMembers] c  using (  select  users.ServerName,  DbName,  RoleName,  DbUserName as RoleMemberName,  isActive,  'Responsable applicatif' as Reason  from (  select distinct  ServerName,  DbUserName  from  security.sqlmappings  where SQLLogin in (  select  SqlLogin  from  security.Contacts  where  Department = 'CHU Liege/SAI/MTECH'  and Job = 'Responsable applicatif'  )  ) users  inner join (  select  ServerName,  DbName,  RoleName,  isActive  from  security.CustomRoles  where  RoleName = 'sai\_ra'  and ServerName not in ('SI-S-SERV24','SI-S-SERV53')  )dbs  on  dbs.ServerName = users.ServerName  ) n  on c.ServerName = n.ServerName  and c.DbName = n.DbName  and c.RoleName = n.RoleName  and c.MemberName = n.RoleMemberName  WHEN MATCHED THEN  UPDATE SET  [Reason] = n.[Reason],  [isActive] = n.[isActive]  WHEN NOT MATCHED BY TARGET THEN  INSERT (ServerName,DbName,RoleName,MemberName,isActive,Reason,MemberIsRole)  VALUES (n.ServerName,n.DbName,n.RoleName,n.RoleMemberName,n.[isActive],n.Reason,0)  /\*WHEN NOT MATCHED BY SOURCE THEN  DELETE \*/  ; |

## Ajouter le privilège SELECT sur dbo.SchemaChangeLog à tous les sai\_ra

|  |
| --- |
| BEGIN TRAN  merge security.onobjectprivs o  using (  SELECT [ServerName]  ,[DbName]  ,[RoleName]  ,0 as isUser  ,'SELECT' as PermissionName  ,'dbo' as SchemaName  ,'SchemaChangeLog' as ObjectName  ,'TABLE' as ObjectType  ,0 as isDeny  ,0 as isWithGrantOption  ,1 as isActive  ,'Permettre l''accès en lecture aux Responsables applicatifs à l''historique des changements de structure' as Reason  FROM [security].[CustomRoles]  where RoleName = 'sai\_ra'  )n  on  o.ServerName = n.ServerName  and o.DbName = n.DbName  and o.Grantee = n.RoleName  and o.PermissionName = n.PermissionName  and o.SchemaName = n.SchemaName  and o.ObjectName = n.ObjectName  WHEN MATCHED THEN  update SET  isUser = n.isUser,  ObjectType = n.ObjectType,  isDeny = n.isDeny,  isWithGrantOption = n.isWithGrantOption,  isActive = n.isActive  WHEN NOT MATCHED BY TARGET THEN  insert (  ServerName,  DbName,  Grantee,  isUser,  PermissionName,  SchemaName,  ObjectName,  ObjectType,  isDeny,  isWithGrantOption,  isActive  )  values (  n.ServerName,  n.DbName,  n.[RoleName],  n.isUser,  n.PermissionName,  n.SchemaName,  n.ObjectName,  n.ObjectType,  n.isDeny,  n.isWithGrantOption,  n.isActive  )  /\*WHEN NOT MATCHED BY SOURCE THEN  DELETE \*/  ;  commit |

## Modification du rôle sur les accès à un schéma

Tableau

|  |
| --- |
| BEGIN TRAN  update security.onschemaprivs  set  RoleName = 'endusers'  /\*  select \*  from  security.onschemaprivs  \*/  where  ServerName = 'SI-S-SERV308'  and (  (  dbname = 'ppdb\_sys'  and SQLLogin = 'ppmgr'  )  or (  dbname = 'ppdb\_mgr'  and SQLLogin = 'ppsys'  )  )  /\*  Check it's ok  select \* from security.onschemaprivs  where ServerName = 'SI-S-SERV308'  and RoleName = 'endusers'  \*/  commit |

## Génération du script d’application de la sécurité

Se connecter sur le serveur central de gestion et exécuter la commande suivante :

Exemple de commande

|  |
| --- |
| SQLCMD.EXE -E -d CHU\_Security\_JEL -Q "SET NOCOUNT ON; SELECT [security].[getCmds4ServerSecurity]('SI-S-SERV53')" -b -o INDIC\_TEST\_ApplySecurity.sql -y 0  OU (en powershell)  invoke-command -computer "si-s-serv236" -ScriptBlock {  SQLCMD.EXE -E -d CHU\_Security\_JEL -Q "SET NOCOUNT ON; SELECT [security].[getCmds4ServerSecurity]('SI-S-SERV308')" -b -o \\SI-S-SERV236\sources\SWISSLOG\_EVAL\_ApplySecurity\_20141128.sql -y 0  } |

Retour attendu dans le fichier de sortie

|  |
| --- |
| /\*\*  \* Security Settings Applier version 0.1.3.  \* ServerName : SI-S-SERV53  \*/  /\*\*  \* Server logins Creation version 0.1.2.  \* ServerName : SI-S-SERV53  \*/  PRINT '. Commands for "Thomas Vanbrabant" from department "CHU Liege/Service T.B.E."'  /\*\*  \* SQL Login Creation (both authentication) version 0.1.3.  \* LoginName : CHULG\c172610  \* AuthMode : WINDOWS  \* Passwd : 123456a.  \* DefaultDatabase : ASCLEPIOS\_DWH  \*/  -- 1.1 Check that the default database actually exists  if (NOT exists (select \* from sys.databases where name = 'ASCLEPIOS\_DWH'))  BEGIN  RAISERROR ( 'The given default database (ASCLEPIOS\_DWH) does not exist',0,1 ) WITH NOWAIT  return  END  Use [ASCLEPIOS\_DWH]  IF NOT EXISTS (SELECT loginname FROM master.dbo.syslogins WHERE name = 'CHULG\c172610')  BEGIN  -- create it !  EXEC ('USE [master]; CREATE LOGIN [CHULG\c172610] FROM WINDOWS , DEFAULT\_DATABASE=[ASCLEPIOS\_DWH]')  END  -- give the login the permission to connect the database engine  EXEC ('USE [master] ; GRANT CONNECT SQL TO [CHULG\c172610]');  -- set the default database for this login  exec sp\_defaultdb @loginame = 'CHULG\c172610' , @DefDb = 'ASCLEPIOS\_DWH'  -- by default : no password policy is defined  GO  … ETC … |