How To Use Data Abstraction Best Practices Privilege Scripts

An Open Source Asset for use with TIBCO® Data Virtualization

Project Name	AS Assets Data Abstraction Best Practices
Document Location	This document is only valid on the day it was printed. The source of the document will be found in the ASAssets_DataAbstractionBestPractices folder (https://github.com/TIBCOSoftware)
Purpose	Self-paced instructional

www.tibco.com

Global Headquarters
3303 Hillview Avenue
Palo Alto, CA 94304

Tel: +1 650-846-1000
+1 800-420-8450
Fax: +1 650-846-1005

Revision History

Version	Date	Author	Comments
8.1.8	05/24/2017	Mike Tinius	Updated for Best Practices v8.1.8 – added Privilege scripts.
8.1.9	12/06/2017	Mike Tinius	Transitioned to Tibco 8.1.9
2018Q1	03/20/2018	Mike Tinius	Release 2018Q1 – no changes.
2019Q1	01/25/2019	Mike Tinius	Release 2019Q1 - no changes.
2019Q200	06/13/2019	Mike Tinius	Release 2019Q200 – no changes.
2019.300	08/01/2019	Mike Tinius	Release 2019Q300 – no changes.

Related Documents

Name	Version
How To Use Utilities.pdf	2019Q301
How To Use Data Abstraction Best Practices View Generation.pdf	2019Q300
How To Test Data Abstraction Best Practices View Generation.pdf	2019Q300
How To Learn Data Abstraction Best Practices View Generation.pdf	2019Q300
How To Use Data Abstraction Best Practices Manage Annotations.pdf	2019Q300
How To Use Data Abstraction Best Practices Privilege Scripts.pdf	2019Q300
How To Use Data Abstraction Best Practices Dynamic File Framework.pdf	2019Q300

Supported Versions

Name	Version
TIBCO® Data Virtualization	7.0 or later
AS Assets Utilities open source	2019Q301 or later

Table of Contents

1 Introduction

Purpose

The purpose of Best Practices Privilege Scripts is to provide a framework for performing mass-updates on privileges based on rows maintained in a spreadsheet or a database.

The Privilege Scripts help the developer to maintain and execute privileges during deployment for the various layers. The three main areas for setting privileges are as follows:

/services/databases

/services/webservices

/shared

The strategy for setting privileges fits in generically with any folder structure within the Data Virtualization (DV) server but aligns nicely with the Data Abstraction Best Practices which uses the concept of an "organization" project folder in the three areas above. For example, it might look like this:

/services/databases/My_Organization

/services/webservices/My_Organization

/shared/My_Organization

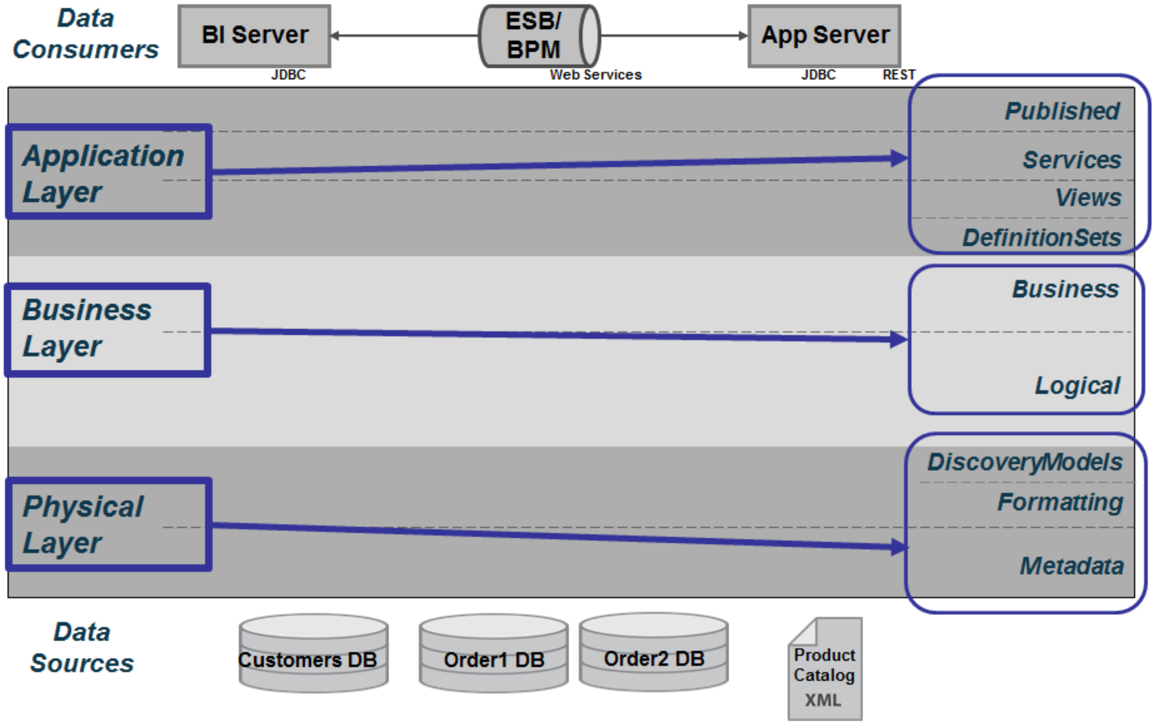


Figure one: Technical Data Abstraction Layers

Audience

This document is intended to provide guidance for the following users:

- Data Virtualization Administrators – provides a guide for installation.
- Architects – provides the data abstraction architecture.
- Data professionals – provides background on the published views and usage.
- Operations users – provides insight into triggers and procedures that are executed.
- Project Managers – provides general information on data abstraction best practices.

References

Product references are shown below. Any references to CIS or DV refer to the current TIBCO® Data Virtualization.

- TIBCO® Data Virtualization was formerly known as
  - Cisco Data Virtualization (DV)
  - Composite Information Server (CIS)

DV Folder Structure

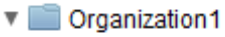
1) DV Folder Structure – establish the baseline concept with "Organization1" as an example.

   a. Databases – Publish views/procedures to a virtual database contained under: /services/databases/Organization1

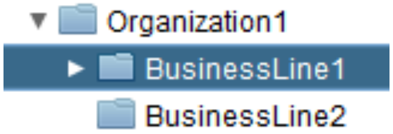
   b. Web Services – Publish views/procedures to a virtual web service contained under: /services/webservices/Organization1

   c. Shared – Developers for Servicing create resources under: /shared/Organization1

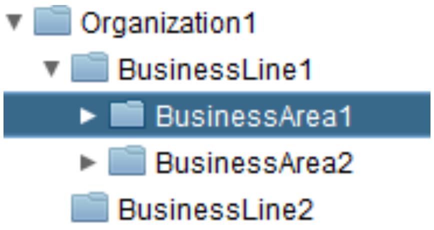
   d. Organization – /shared/Organization1

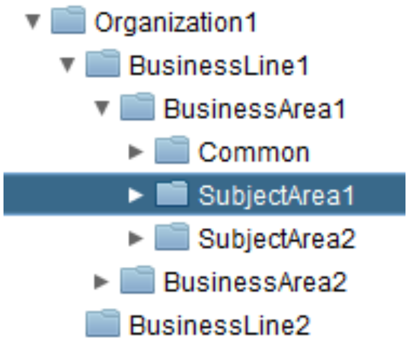
      i. Business Line – /shared/Organization1/BusinessLine1

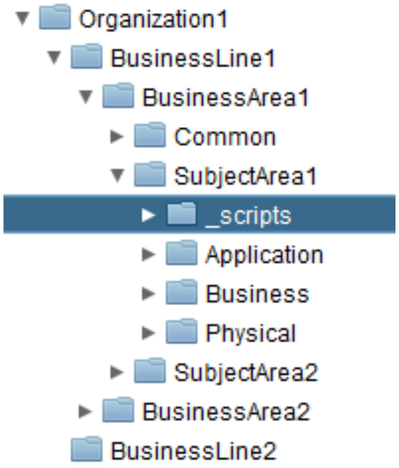
         1. Business Areas – /shared/Organization1/BusinessLine1\BusinessArea1



a. Subject Areas – shared/Organization1/BusinessLine1\BusinessArea1\SubjectArea1



i. Data Abstraction Folder Best Practices



e. Data Abstraction Best Practices Folders are contained under each subject area folder.

   i. /Application

   ii. /Business

   iii. /Physical

2) Privileges on Folders – LDAP groups are created down to the granularity of the subject areas.

   a. /BusinessLine1/BusinessArea1

      i. /Common

      ii. /SubjectArea1 – subject area 1 folder

      iii. /SubjectArea2 – subject area 2 folder

3) Roles and Rules – very much pattern based in terms of the conventions for creating LDAP names

   a. Admin - **DV_<BusLine>_Admin**

      i. Administrator for Servicing folders

   b. Architect - **DV_<BusLine>_<BusArea>_Arch**

      i. Architects are responsible for their own business area /Common and /Servicing/Common. They can promote resources to a /Common folder.

      ii. Architects have “G”rant capability.

      iii. Architects are responsible for published resources to virtual databases and web services.

   c. Developer - **DV_<BusLine>_<BusArea>_Dev**

      i. Developers are responsible for developing resources in their business area folders. They can create cache objects and execute.

   d. QA - **DV_<BusLine>_<BusArea>_QA**

      i. QA technicians are responsible for executing tests and can use Studio in the QA environment. They are not allowed to modify resources.

   e. Application Id - **DV_<BusLine>_<BusArea>_UserRO_AppId**

      i. This is a read-only user for applications such as Cognos to connect to Composite.

      ii. This user does not have Studio rights.

   f. Read-only user - **DV_<BusLine>_<BusArea>_<SubjArea>_UserRO**

      i. This is a read-only user for adhoc usage such as Toad users to connect to Composite.

ii. This user does not have Studio rights.

g. Read-Write user - **DV_<BusLine>_<BusArea>_<SubjArea>_UserRW**

i. This is a read-write user for adhoc usage such as Toad users to connect to Composite.

ii. This user does not have Studio rights.

Pre-Requisites

Follow the steps below to create a new project.

1. LDAP groups have been brought into the target environment.
2. /shared/ASAssets/Utilities have been installed and configured for 2018Q1.

2 Configuration

How to Configure

This section provides information on how to configure the Privilege Scripts.

Privilege Script Configuration Summary

1. Copy privilege spreadsheet to file system
2. Modify datasource connections
3. Create privilege database tables
4. Reintrospect datasources
5. Test datasources

Best Practices Installation

1. Install Best Practices Spreadsheets
   - 1.1. Follow the Data Abstraction Best Practices installation directions found in this document: "**How To Use AS Data Abstraction Best Practices.pdf**"
2. Copy privilege spreadsheet to file system
   - 2.1. This should have been done during installation of the Best Practices. Verify this.
3. Modify the default settings:
   - 3.1. Location: /shared/ASAssets/BestPractices_v81/_ProjectMaintenance/defaultValues
   - 3.2. datasource: determine whether to use EXCEL or DB_[LLE,PROD] (database) as the default source for the privilege rows. Value: EXCEL
   - 3.3. defaultDatabaseLLEMapping: provides a way to choose Oracle or SQL Server as the default LLE when using "DB_LLE" datasource. Value: DB_LLE_ORA
   - 3.4. defaultDatabasePRODMapping: provides a way to choose Oracle or SQL Server as the default LLE when using "DB_PROD" datasource. Value: DB_PROD_ORA
4. Modify the data source path or connection information:
   - 4.1. EXCEL: modify the root path if needed.
     - 4.1.1.DV Location: [EXCEL] /shared/ASAssets/BestPractices_v81/PrivilegeScripts/Metadata/Privileges_DS_EXCEL
     - 4.1.2.E.g. root path: C:/CIS7.0/BestPractices/Privileges
   - 4.2. DATABASE: modify connection information if needed for the lower-level environment

4.2.1. Note: This capability allows for a database in LLE for development and testing of privileges and a production database. In reality, a project really only needs a production database. All privileges are considered “production” no matter what DV environment they are being applied to. Therefore, while the scripts for LLE are provided it is not necessary to maintain an LLE privilege database.

4.2.2. Enable either Oracle or SQL Server LLE and/or PROD as needed.

4.2.3. Oracle database locations:

/shared/ASAssets/BestPractices_v81/PrivilegeScripts/Metadata/Privileges_DB_PROD_ORA

/shared/ASAssets/BestPractices_v81/PrivilegeScripts/Metadata/Privileges_DB_LLE_ORA

4.2.4. SQL Server database locations:

/shared/ASAssets/BestPractices_v81/PrivilegeScripts/Metadata/Privileges_DB_PROD_SS

/shared/ASAssets/BestPractices_v81/PrivilegeScripts/Metadata/Privileges_DB_LLE_SS

5. Create the tables

5.1. Oracle [LLE]: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/Metadata/DDL/Oracle_LLE/

1_pqCreate_DABP_privilege_tables (CMPPRV, CMPPRV_DATA)

2_pqCreate_DABP_privilege_triggers (CMPPRV)

3_pqCreate_DABP_privilege_sequences (CMPPRV)

5.2. Oracle [PROD]: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/Metadata/DDL/Oracle_PROD/

1_pqCreate_DABP_privilege_tables (CMPPRV, CMPPRV_DATA)

2_pqCreate_DABP_privilege_triggers (CMPPRV)

3_pqCreate_DABP_privilege_sequences (CMPPRV)

5.3. SQL Server [LLE]: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/Metadata/DDL/SQLServer_LLE/

1_pqCreate_DABP_privilege_tables (dbo)

2_pqCreate_DABP_privilege_triggers (dbo)

5.4. SQL Server [PROD]: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/Metadata/DDL/SQLServer_PROD/

1_pqCreate_DABP_privilege_tables (dbo)

2_pqCreate_DABP_privilege_triggers (dbo)

6. Reintrospect the data sources

7. Test the data sources

   7.1. Show Contents for one of the Excel data source worksheets to ensure that data is being retrieved.

   7.2. Show Contents for one of the tables in the chosen data source.

8. Construct the Privilege Spreadsheet to be loaded into the database

9. Load the database from the spreadsheet

   9.1. loadPrivileges_EXCEL - the first time, it will notify you of the missing support table values. This will load the privileges from the 3 privilege tabs in the spreadsheet.

   9.2. performSupportingTableMaintenance - run this to load the supporting tables.

   9.3. loadGroups_EXCEL - optional unless you want to set the Studio Rights for the DV groups in the Group_List tab of the spreadsheet.

   9.4. loadUsers_EXCEL – optional

10. Execute the update Groups, Privileges, and Users as needed.

    Go to the next section “Executing Resource Privileges” for more information.

3 Executing Resource Privileges

Introduction

This section provides guidance on how to make a request to the administrator to update the privileges using the manual execution of privileges.

Instructions

1. Run after hours or early in the morning so as not to impact Developers. Since these scripts are updating the Composite Metadata repository via the API, Developers will notice that their Studios will be unresponsive while these scripts are running.

2. Depending on activity in the network, activity on the Composite server and activity in the Composite repository database server, these scripts may take hours to run.

2.1. Update Groups does not take much time at all. This is a very low impact operation.

2.2. Update Resource Privileges will take the longest especially depending on the number of privileges to apply.

3. Update Group Studio Rights before updating privileges

3.1. Sheet [When Datasource=EXCEL]: Group_List

3.2. Table [When Datasource=DB_PROD or DB_LLE]: COMPOSITE_STUDIO_RIGHTS

3.3. Script: **updateGroupsDriver**

3.3.1. **Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/updateGroupsDriver**

3.3.2. Datasource: [DB_PROD|DB_LLE|EXCEL] – Indicates which datasource to use to execute from.

3.3.3. *Environment_Name*: [DEV|**UAT**|PROD**]**

3.3.4. *Organization:* ORG1

3.3.5. *Project*: Project1

3.3.6. *Subproject* *SB1*

3.3.7. All other parameters are left null

3.3.8. *inDebug1*: Y

3.3.9. *inDebug1Console*: Y

3.3.10. *inDebug1CISLog*: N

3.3.11. *inDebug1ReadOnly*: Y

3.3.12. **Repeat for these projects**:

Project:
Project2

4. Update Privileges

4.1. Sheet [When Datasource=EXCEL]: Privileges_shared + Privileges_databases + Privileges_webservices

4.2. Table [When Datasource=DB_PROD or DB_LLE]: COMPOSITE_PRIVILEGES

4.3. Script: **updatePrivilegesDriver**

4.3.1. **Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/updatePrivilegesDriver**

4.3.2. Datasource: [DB_PROD|DB_LLE|EXCEL] – Indicates which datasource to use to execute from.

4.3.3.

4.3.4. *Batch_Privileges*: 1

4.3.5. *Environment_Name*: [DEV|**UAT**|PROD**]**

4.3.6. *Organization:* ORG1

4.3.7. *Project*: **Project1**

4.3.8. *Subproject:* SB1

4.3.9. All other parameters are left null

4.3.10. *inDebug1*: Y

4.3.11. *inDebug1Console*: Y

4.3.12. *inDebug1CISLog*: N

4.3.13. *inDebug1ReadOnly*: Y

4.3.14. *inDebug1RevokeAll*: Y

4.3.15. *inDebug2*: N

4.3.16. *inBypassErrors:* Y

4.3.17. **Repeat for these projects**:

Project:	Subproject:
Project2	SB1|SB2|SB3

4 Privilege Spreadsheet and Database Table Mappings

Spreadsheet “sheet definition” and Associated Database Table

1) Composition of the spreadsheet / database table

    a. Privileges_shared – This sheet is used to establish privileges for a project and its /shared resources.

        i. Correlates to the database table COMPOSITE_PRIVILEGES where PRIVILEGE_TYPE = ‘SHARED’.

    b. Privileges_databases – This sheet is used to establish privileges for a project and its virtual database resource located at /services/databases.

        i. Correlates to the database table COMPOSITE_PRIVILEGES where PRIVILEGE_TYPE = ‘DATABASES’.

    c. Privileges_webservices – This sheet is used to establish privileges for a project and its virtual web service resources located at /services/webservices.

        i. Correlates to the database table COMPOSITE_PRIVILEGES where PRIVILEGE_TYPE = ‘WEBSERVICES’.

    d. Group_List – The group list sheet is used for applying studio access rights to a group.

        i. Correlates to the database table COMPOSITE_STUDIO_RIGHTS.

    e. User_List – The user list sheet is used for creating composite users that “mirror” the ldap groups for testing purposes.

        i. Correlates to the database table COMPOSITE_USER.

    f. Roles_and_Responsibilities – not used for any scripts (documentation only)

        i. Correlates to the database table ROLE_TYPE.

    g. Privilege_Template – not used for any scripts (documentation only)

        i. Correlates to the database table COMPOSITE_ROLE_TEMPLATE.

2) Spreadsheet Concepts

    a. Applying privileges

i. Apply privileges to lowest level folders first. For example, given /shared/folder1 and /shared/folder1/folder2, apply privileges to /shared/folder1 before /folder2.

ii. READ only privileges – The update privileges have the ability to automatically set READ on parent folders. The benefit of this is that these rows do not have to be specified in the spreadsheet any longer like they did with version 1.

iii. Apply a concept of initializing folders to “NONE” recursively first to create a blank pallet on which to paint privileges.

iv. If a folder shares privileges with another sibling folder then don’t apply privileges recursively. Simply set privileges on that folder for your groups. Example:

   1. /shared/folder1 (your folder)

      a. Set recursive mode=”N” for “not” recursive.

   2. /shared/folder2 (another groups folder)

v. Spreadsheet column: Recurse Child

   Recurse Child Resources and Folders:
   N or blank = do not recurse. Set specified privileges for the resource only.
   YM = Only apply modification recursively (always recommended when using the spreadsheet)
   YC = Make child resources look like this resource recursively. This feature is **not recommended** when using the spreadsheet and making multiple modifications because the entire context is not taken into consideration like it does in studio.

   Concept 1: Only select YM to recurse child resources when the resource being set is at the lowest level at which it is safe to recursively set privileges for a given USER or GROUP. If the resource is an intermediate level folder with other branches offshoot from it then only apply the privilege modification to the resource folder using "N" unless all branches are required to have the same privilege settings. "YC" should only be used when you want to reset all privileges for other USERS or GROUPS to "NONE" except for the current USER or GROUP being set.

   Concept 2: If the higher-level resource folder is set with "YC" then it is not necessary to set this USER or GROUP for branch-level resources.

Caveat: It has been determined through testing because the Composite API is being used to set privileges, only "YM" should be used since each transaction on the spreadsheet is executed individually. This differs from the concept in Studio, where Studio has a holistic view of all privileges for all groups for a given resource. Therefore, in Studio, you can set the equivalent of "YC" which is recurse all children and make them look exactly like the parent. Unfortunately, that concept is not the same when executed from this spreadsheet and API.

b. Batching privileges

   i. Privileges are batched and executed as a single unit request to DV. The batches are based on finding a set of rows in the spreadsheet with the same path, type and recurse child settings. If any of those change, then that delineates the batch request to DV.

c. Spreadsheet Line

   i. Each spreadsheet line (like a database row) needs to include all of the information to be able to set a row independently if requested. However, if batching privileges is requested then the first row of a batch is used to set the resource path, type, recurse child.

   ii. If the resource path or type are missing, that line in the spreadsheet will *NOT* be applied.

   iii. It is important to note that the first row of a batch is used exclusively to set the owner and owner domain since this can only be set at the resource path level. By definition of a batch [resource path, type, recurse child] will start a new batch.

5 Privilege Scripts Method Definition

Detailed Definitions

Detailed documentation on the inputs and outputs can be found in the header and annotation section of each procedure.

1) Composite groups and users – If you are adding groups and users, you must first execute the "updateGroupsDriver" followed by the "udpateUsersDriver". Groups must be present before users are created because users are assigned to groups upon creation of the user.

2) Validation – you can validate the groups and users were created by using "validateGroupsDriver", "validateUsersDriver" and "validatePrivilegesDriver".

3) Delete groups and users – If you want to clean up the composite created users and groups perform these tasks in reverse order of creation. First execute "deleteUsersDriver" to remove the users followed by "deleteGroupsDriver" to remove the groups.

Privilege Maintenance: Get Privileges

1. **getPrivileges –** Construct a SQL statement based on filters passed in and retrieve the set of privileges that match the criteria.

   Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/getPrivileges

Direction	Parameter Name	Parameter Type
IN	**Datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL - Excel spreadsheet which gets uploaded to each DV server. • DB_LLE - SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD - SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**Environment_Name** – REQUIRED FILTER: The Composite server environment in which to get/update privileges for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**Organization** – RECOMMENDED FILTER: The name of the organization such as ORG1 or ORG2.	VARCHAR(255)

Direction	Parameter Name	Parameter Type
IN	**Project** – RECOMMENDED FILTER: The name of the project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given project. If left blank, all projects will be updated.	VARCHAR(255)
IN	**SubProject** – OPTIONAL FILTER*:* The name of the sub-project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given sub-project. If left blank, all rows for the project will be updated.	VARCHAR(255)
IN	**Sheet_Name** – OPTIONAL FILTER: The name of the Excel spreadsheet in which to get/update privileges from: [shared, databases, webservices]	VARCHAR(255)
IN	**Resource_Path** – OPTIONAL FILTER: The resource path in which to get/update privileges.	LONGVARCHAR
IN	**Resource_Type** – OPTIONAL FILTER: The resource type in which to get/update privileges. This will only be used when no "Resource_Path" or a single "Resource_Path" is provided. It is not used when a list of "Resource_Path" entries are provided. Example: • DATA_SOURCE - a published datasource or physical metadata datasource. • CONTAINER - a folder path, a catalog or schema path. • COLUMN - a column from a table • LINK - a published table or procedure. If it resides in the path /services and points to a TABLE or PROCEDURE then it is a LINK. • TABLE - a view in the /shared path. • PROCEDURE a procedure in the /shared path.	LONGVARCHAR
IN	**Name** – OPTIONAL FILTER: The user/group name in which to get/update privileges.	VARCHAR(255)
IN	**Name_Type** – OPTIONAL FILTER: Valid values are USER or GROUP.	VARCHAR(255)
IN	**Domain_Name** – OPTIONAL FILTER: The domain name in which to get/update privileges.	VARCHAR(255)
IN	**debug** – OPTIONAL: Y=debug on, N=debug off	CHAR(1)
OUT	DECLARE PUBLIC TYPE privilegeRowType ROW (rownum INTEGER, sheetRownum INTEGER, datasource VARCHAR(255), sheetName VARCHAR(255), privilegeType VARCHAR(255), Organization VARCHAR(255), Project VARCHAR(255), SubProject VARCHAR(255), "Resource Path" VARCHAR(1024),	privilegeRowType

Direction	Parameter Name	Parameter Type
	"Resource Type" VARCHAR(255), "Recurse Dependencies" VARCHAR(1), "Recurse Dependents" VARCHAR(1), "Recurse Child" VARCHAR(3), "Revoke All" VARCHAR(1), Name VARCHAR(255), "Name Type" VARCHAR(255), "Domain" VARCHAR(255), "Env Type" VARCHAR(255), OrderPrecedence CHAR(1), R VARCHAR(1), W VARCHAR(1), E VARCHAR(1), S VARCHAR(1), U VARCHAR(1), I VARCHAR(1), D VARCHAR(1), G VARCHAR(1), Owner VARCHAR(255), "Owner Domain" VARCHAR(255), Comments VARCHAR(1024), Initialize VARCHAR(255), isActive CHAR(1))	

Privilege Maintenance: Update Privileges

2. **updatePrivilegesDriver –** This script is used to execute the update privileges. This script will access the spreadsheet or database table, batch rows together into a request and submit to DV via the DV admin API.   The script runs within the Composite Server. This procedure is used to update privileges by reading the privileges from an excel spreadsheet or database table.

   Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/updatePrivilegesDriver

   2.1. The excel sheets include:

      2.1.1.   Privileges_shared - Update privileges for /shared

      2.1.2.   Privileges_databases - Update privileges for /services/databases

      2.1.3.   Privileges_webservices - Update privileges for /services/webservices

      2.1.4.   COMPOSITE_PRIVILEGES is a table that contains a type field that indicates which area of composite the row is specifying [shared, databases, webservices].

   2.2. Pre-requisites to executing the script.

2.2.1. LDAP groups must have been brought into Composite.

2.2.2. The spreadsheet needs to be uploaded to the Composite Server machine.

2.2.3. The spreadsheet data source is updated with the spreadsheet path.

2.2.4. All of the resources being referenced must be present in the Composite server or an exception will be thrown.

2.2.5. The user executing the script must have administrative privileges.

Direction	Parameter Name	Parameter Type
IN	**Datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL - Excel spreadsheet which gets uploaded to each DV server. • DB_LLE - SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD - SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**Batch_Privileges** – REQUIRED_FILTER: When the resource path in the spreadsheet changes, a batch is executed. When the resource type in the spreadsheet changes, a batch is executed. When the recursion identifier in the spreadsheet changes, a batch is executed. A group of rows with like recursion may only be batched together otherwise the meaning of the privilege setting is not the same. • 1=Batch all privileges for the same path, type and recursion setting. • 0=Process each spreadsheet line separately (no batching).	INTEGER
IN	**Environment_Name** – REQUIRED FILTER: The Composite server environment in which to get/update privileges for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**Organization** – RECOMMENDED FILTER: The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**Project** – RECOMMENDED FILTER: The name of the project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given project. If left blank, all projects will be updated.	VARCHAR(255)

Direction	Parameter Name	Parameter Type
IN	**SubProject** – OPTIONAL FILTER*:* The name of the sub-project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given sub-project. If left blank, all rows for the project will be updated.	VARCHAR(255)
IN	**Sheet_Name** – OPTIONAL FILTER: The name of the Excel spreadsheet in which to get/update privileges from: [shared, databases, webservices]	VARCHAR(255)
IN	**Resource_Path** – OPTIONAL FILTER: The resource path in which to get/update privileges.	LONGVARCHAR
IN	**Resource_Type** – OPTIONAL FILTER: The resource type in which to get/update privileges. This will only be used when no "Resource_Path" or a single "Resource_Path" is provided. It is not used when a list of "Resource_Path" entries are provided. Example: • DATA_SOURCE - a published datasource or physical metadata datasource. • CONTAINER - a folder path, a catalog or schema path. • COLUMN - a column from a table • LINK - a published table or procedure. If it resides in the path /services and points to a TABLE or PROCEDURE then it is a LINK. • TABLE - a view in the /shared path. • PROCEDURE a procedure in the /shared path.	LONGVARCHAR
IN	**Name** – OPTIONAL FILTER: The user/group name in which to get/update privileges.	VARCHAR(255)
IN	**Name_Type** – OPTIONAL FILTER: Valid values are USER or GROUP.	VARCHAR(255)
IN	**Domain_Name** – OPTIONAL FILTER: The domain name in which to get/update privileges.	VARCHAR(255)
IN	**preview** – N or null(default)=Do not preview. Execute the privileges. Y=preview what will get set but don't actually execute the operation.	CHAR(1)
IN	**forceDomainAsComposite** – OPTIONAL PARAM: Y=true, N=false. For example, the LDAP group DV_Demo_Dev is automatically converted to the composite domain group dv_demo_dev. • Provides a way to force a conversion of the group from an LDAP domain name like "ldap" to the "composite" domain. • This is useful when setting up the privileges for testing with composite groups that mirror the LDAP groups. This way the spreadsheet does not have to be modified.	CHAR(1)
IN	**inDebug1** – N/Null=No debug, Y (default)=Debug for this	CHAR(1)

Direction	Parameter Name	Parameter Type
	procedure - basic messages.	
IN	**inDebug1Console** – N/Null (default)=No debug, Y=Debug console. Determines whether to print to the Studio console window or not.	CHAR(1)
IN	**inDebug1CISLog** – N/Null (default)=No debug, Y=Debug log. Determines whether to print to the DV log or not.	CHAR(1)
IN	**inDebug1ReadOnly** – N/Null (default)=No debug, Y=Debug for READ only messages.	CHAR(1)
IN	**inDebug1RevokeAll** – N/Null (default)=No debug, Y=Debug for Revoke All messages.	CHAR(1)
IN	**inDebug2** – N/Null (default)=No debug, Y=Debug for 2nd and 3rd tier procedures (deep debug).	CHAR(1)
IN	**inBypassErrors** – Bypass errors. Throw exception when paths not found. • N/Null (default) Do not bypass errors. • Y=bypass resource not found errors but report them.	CHAR(1)
OUT	**rowsProcessed** – number of rows processed from the spreadsheet	INTEGER
OUT	**logOutput** – The output log that gets printed to the command line console is also output.	LONGVARCHAR

Privilege Maintenance: Validate Privileges

3. **validatePrivilegesDriver –** This script is used to validate privileges. This procedure is used to validate privileges by reading the privileges from an excel spreadsheet and validate the resource path exists and the name (USER or GROUP) exists in the specified domain and the privilege is set according to the spreadsheet or database setting for the filter applied.

   Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/validatePrivilegesDriver

   3.1. Read the privileges from the following sheets:

   3.1.1. Privileges_shared - /shared privileges

   3.1.2. Privileges_databases - /services/databases privileges

   3.1.3. Privileges_webservices - /services/webservices privileges

Direction	Parameter Name	Parameter Type
IN	**Datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the	VARCHAR(255)

Direction	Parameter Name	Parameter Type
	default value is used from defaultValues.datasource. Possible values include: • EXCEL - Excel spreadsheet which gets uploaded to each DV server. • DB_LLE - SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD - SQL Server production database. This connection should always be used unless developing new functionality.	
IN	**Environment_Name** – REQUIRED FILTER: The Composite server environment in which to get/update privileges for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**Organization** – RECOMMENDED FILTER: The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**Project** – RECOMMENDED FILTER: The name of the project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given project. If left blank, all projects will be updated.	VARCHAR(255)
IN	**SubProject** – OPTIONAL FILTER*:* The name of the sub-project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given sub-project. If left blank, all rows for the project will be updated.	VARCHAR(255)
IN	**Sheet_Name** – OPTIONAL FILTER: The name of the Excel spreadsheet in which to get/update privileges from: [shared, databases, webservices]	VARCHAR(255)
IN	**Resource_Path** – OPTIONAL FILTER: The resource path in which to get/update privileges.	LONGVARCHAR
IN	**Resource_Type** – OPTIONAL FILTER: The resource type in which to get/update privileges. This will only be used when no "Resource_Path" or a single "Resource_Path" is provided. It is not used when a list of "Resource_Path" entries are provided. Example: • DATA_SOURCE - a published datasource or physical metadata datasource. • CONTAINER - a folder path, a catalog or schema path. • COLUMN - a column from a table • LINK - a published table or procedure. If it resides in the path /services and points to a TABLE or PROCEDURE then it is a LINK. • TABLE - a view in the /shared path. • PROCEDURE a procedure in the /shared path.	LONGVARCHAR

Direction	Parameter Name	Parameter Type
IN	**Name** – OPTIONAL FILTER: The user/group name in which to get/update privileges.	VARCHAR(255)
IN	**Name_Type** – OPTIONAL FILTER: Valid values are USER or GROUP.	VARCHAR(255)
IN	**Domain_Name** – OPTIONAL FILTER: The domain name in which to get/update privileges.	VARCHAR(255)
IN	**forceDomainAsComposite** – OPTIONAL PARAM: Y=true, N=false. For example, the LDAP group DV_Demo_Dev is automatically converted to the composite domain group dv_demo_dev. • Provides a way to force a conversion of the group from an LDAP domain name like "ldap" to the "composite" domain. • This is useful when setting up the privileges for testing with composite groups that mirror the LDAP groups. This way the spreadsheet does not have to be modified.	CHAR(1)
IN	**inDebug1** – N/Null=No debug, Y (default)=Debug for this procedure - basic messages.	CHAR(1)
IN	**inDebug1Console** – N/Null (default)=No debug, Y=Debug console. Determines whether to print to the Studio console window or not.	CHAR(1)
IN	**inDebug1CISLog** – N/Null (default)=No debug, Y=Debug log. Determines whether to print to the DV log or not.	CHAR(1)
IN	**inDebug1ReadOnly** – N/Null (default)=No debug, Y=Debug for READ only messages.	CHAR(1)
IN	**inDebug2** – N/Null (default)=No debug, Y=Debug for 2nd and 3rd tier procedures (deep debug).	CHAR(1)
IN	**inBypassErrors** – Bypass errors. Throw exception when paths not found. 1. N/Null (default) Do not bypass errors. 2. Y=bypass resource not found errors but report them.	CHAR(1)
OUT	**validPrivileges** – true=if all privileges are validated. false=if one or more privileges are invalid.	VARCHAR
OUT	**rowsProcessed** – number of rows processed from the spreadsheet	INTEGER
OUT	**invalidPrivilegeList** – Pipe separated list of invalid pivileges.	LONGVARCHAR
OUT	**invalidResourceList** – Pipe separated list of invalid resource	LONGVARCHAR

Direction	Parameter Name	Parameter Type
	paths that do not exist.	
OUT	**invalidNameList** – Pipe separated list of name, nameType, and domain combination that does not exist.	LONGVARCHAR
OUT	**logOutput** – The output log that gets printed to the command line console is also output.	LONGVARCHAR

Group Maintenance: Get Groups

4. **getGroups –** Construct a SQL statement based on filters passed in and retrieve the set of groups that match the criteria.

   Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/getGroups

Direction	Parameter Name	Parameter Type
IN	**Datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL - Excel spreadsheet which gets uploaded to each DV server. • DB_LLE - SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD - SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**Environment_Name** – REQUIRED FILTER: The Composite server environment in which to get/update privileges for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**Organization** – RECOMMENDED FILTER: The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**Project** – RECOMMENDED FILTER: The name of the project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given project. If left blank, all projects will be updated.	VARCHAR(255)
IN	**SubProject** – OPTIONAL FILTER*:* The name of the sub-project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given sub-project. If left blank, all rows for the project will be updated.	VARCHAR(255)
IN	**Group_Name** – OPTIONAL FILTER: The group name in which to get/update groups.	VARCHAR(255)

Direction	Parameter Name	Parameter Type
IN	**Domain_Name** – OPTIONAL FILTER: The domain name in which to get/update groups.	VARCHAR(255)
IN	**debug** – OPTIONAL: Y=debug on, N=debug off	CHAR(1)
OUT	DECLARE PUBLIC TYPE groupsRowType ROW (rownum INTEGER, sheetRownum INTEGER, datasource VARCHAR(255), sheetName VARCHAR(255), Organization VARCHAR(255), Project VARCHAR(255), SubProject VARCHAR(255), GroupName VARCHAR(255), Description VARCHAR(1024), "Domain" VARCHAR(255), "Env Type" VARCHAR(255), Access VARCHAR(1), Unlock VARCHAR(1), Config_R VARCHAR(1), Config_M VARCHAR(1), Resources_R VARCHAR(1), Resources_M VARCHAR(1), Status_R VARCHAR(1), Status_M VARCHAR(1), Users_R VARCHAR(1), Users_M VARCHAR(1), IsActive VARCHAR(1))	groupsRowType

Group Maintenance: Update Groups

5. **updateGroupsDriver –** This script is used to create or update groups. Reads the Composite Privileges spreadsheet and gets the list of groups from the sheet: Group_List

   Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/updateGroupsDriver

   5.1. Limitations

   5.1.1. If the domain is "composite" and the group does not exist then the group is created.

   5.1.2. If the domain is "composite" and the group does exist then the group is updated.

   5.1.3. If the domain is LDAP and the group does not exist then the group is NOT created and not updated.

   5.1.4. If the domain is LDAP and the group does exist then the group is updated.

Direction	Parameter Name	Parameter Type
IN	**Datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL - Excel spreadsheet which gets uploaded to each DV server. • DB_LLE - SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD - SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**Environment_Name** – REQUIRED FILTER: The Composite server environment in which to get/update privileges for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**Organization** – RECOMMENDED FILTER: The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**Project** – RECOMMENDED FILTER: The name of the project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given project. If left blank, all projects will be updated.	VARCHAR(255)
IN	**SubProject** – OPTIONAL FILTER*:* The name of the sub-project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given sub-project. If left blank, all rows for the project will be updated.	VARCHAR(255)
IN	**Group_Name** – OPTIONAL FILTER*:* The group name for which to get/update groups.	VARCHAR(255)
IN	**Domain_Name** – OPTIONAL FILTER: The domain name in which to get/update groups.	VARCHAR(255)
IN	**preview** – N or null(default)=Do not preview. Execute the privileges. Y=preview what will get set but don't actually execute the operation.	CHAR(1)
IN	**forceDomainAsComposite** – OPTIONAL PARAM: Y=true, N=false. For example, the LDAP group DV_Demo_Dev is automatically converted to the composite domain group dv_demo_dev. • Provides a way to force a conversion of the group from an LDAP domain name like "ldap" to the "composite" domain. • This is useful when setting up the privileges for testing with composite groups that mirror the LDAP groups.	CHAR(1)

Direction	Parameter Name	Parameter Type
	This way the spreadsheet does not have to be modified.	
IN	**inDebug1** – N/Null=No debug, Y (default)=Debug for this procedure - basic messages.	CHAR(1)
IN	**inDebug1Console** – N/Null (default)=No debug, Y=Debug console. Determines whether to print to the Studio console window or not.	CHAR(1)
IN	**inDebug1CISLog** – N/Null (default)=No debug, Y=Debug log. Determines whether to print to the DV log or not.	CHAR(1)
IN	**inDebug1ReadOnly** – N/Null (default)=No debug, Y=Debug for READ only messages.	CHAR(1)
OUT	**rowsProcessed** – number of rows processed from the spreadsheet	INTEGER
OUT	**logOutput** – The output log that gets printed to the command line console is also output.	LONGVARCHAR

Group Maintenance: Delete Groups

6. **deleteGroupsDriver –** This script is used to delete groups from DV. Reads the Composite Privileges spreadsheet and gets the list of groups from the sheet: Group_List

   Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/deleteGroupsDriver

   6.1. Limitations

   6.1.1. If the group does not exist then the group is NOT deleted.

   6.1.2. If the group does exist then the group is deleted no matter what domain the group is in.

Direction	Parameter Name	Parameter Type
IN	**Datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL - Excel spreadsheet which gets uploaded to each DV server. • DB_LLE - SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD - SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)

Direction	Parameter Name	Parameter Type
IN	**Environment_Name** – REQUIRED FILTER: The Composite server environment in which to get/update privileges for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**Organization** – RECOMMENDED FILTER: The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**Project** – RECOMMENDED FILTER: The name of the project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given project. If left blank, all projects will be updated.	VARCHAR(255)
IN	**SubProject** – OPTIONAL FILTER*:* The name of the sub-project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given sub-project. If left blank, all rows for the project will be updated.	VARCHAR(255)
IN	**Group_Name** – OPTIONAL FILTER*:* The group name for which to delete groups.	VARCHAR(255)
IN	**Domain_Name** – OPTIONAL FILTER: The domain name in which to delete groups.	VARCHAR(255)
IN	**preview** – N or null(default)=Do not preview. Execute the privileges. Y=preview what will get set but don't actually execute the operation.	CHAR(1)
IN	**forceDomainAsComposite** – OPTIONAL PARAM: Y=true, N=false. For example, the LDAP group DV_Demo_Dev is automatically converted to the composite domain group dv_demo_dev. • Provides a way to force a conversion of the group from an LDAP domain name like "ldap" to the "composite" domain. • This is useful when setting up the privileges for testing with composite groups that mirror the LDAP groups. This way the spreadsheet does not have to be modified.	CHAR(1)
IN	**inDebug1** – N/Null=No debug, Y (default)=Debug for this procedure - basic messages.	CHAR(1)
IN	**inDebug1Console** – N/Null (default)=No debug, Y=Debug console. Determines whether to print to the Studio console window or not.	CHAR(1)
IN	**inDebug1CISLog** – N/Null (default)=No debug, Y=Debug log. Determines whether to print to the DV log or not.	CHAR(1)
OUT	**rowsProcessed** – number of rows processed from the spreadsheet	INTEGER
OUT	**logOutput** – The output log that gets printed to the command line console is also output.	LONGVARCHAR

Group Maintenance: Validate Groups

7. **validateGroupsDriver –** This script is used to validate groups. Reads the Composite Privileges spreadsheet and validates whether the user exists or not from the sheet: Group_List

   Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/validateGroupsDriver

Direction	Parameter Name	Parameter Type
IN	**Datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL - Excel spreadsheet which gets uploaded to each DV server. • DB_LLE - SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD - SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**Environment_Name** – REQUIRED FILTER: The Composite server environment in which to get/update privileges for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**Organization** – RECOMMENDED FILTER: The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**Project** – RECOMMENDED FILTER: The name of the project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given project. If left blank, all projects will be updated.	VARCHAR(255)
IN	**SubProject** – OPTIONAL FILTER*:* The name of the sub-project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given sub-project. If left blank, all rows for the project will be updated.	VARCHAR(255)
IN	**Group_Name** – OPTIONAL FILTER*:* The group name for which to validate groups.	VARCHAR(255)
IN	**Domain_Name** – OPTIONAL FILTER: The domain name in which to validate groups.	VARCHAR(255)
IN	**forceDomainAsComposite** – OPTIONAL PARAM: Y=true, N=false. For example, the LDAP group DV_Demo_Dev is automatically converted to the composite domain group dv_demo_dev. • Provides a way to force a conversion of the group from an LDAP domain name like "ldap" to the "composite"	CHAR(1)

Direction	Parameter Name	Parameter Type
	domain. - This is useful when setting up the privileges for testing with composite groups that mirror the LDAP groups. This way the spreadsheet does not have to be modified.	
IN	**inDebug1** – N/Null=No debug, Y (default)=Debug for this procedure - basic messages.	CHAR(1)
IN	**inDebug1Console** – N/Null (default)=No debug, Y=Debug console. Determines whether to print to the Studio console window or not.	CHAR(1)
IN	**inDebug1CISLog** – N/Null (default)=No debug, Y=Debug log. Determines whether to print to the DV log or not.	CHAR(1)
IN	**inDebug2** – N/Null (default)=No debug, Y=Debug for 2nd and 3rd tier procedures (deep debug).	CHAR(1)
OUT	**validGroups** – true=if all groups are present (valid). false=if one or more groups are not present (invalid).	VARCHAR
OUT	**rowsProcessed** – number of rows processed from the spreadsheet	INTEGER
OUT	**logOutput** – The output log that gets printed to the command line console is also output.	LONGVARCHAR
OUT	**groupResult** – Result with pipe dlimited rows Format of output for a single row within the pipe: **Row=**row number from spreadsheet **SheetRow=**the number of row within the sheet **Sheet=**The name of the sheet from the spreadsheet "Group_List" **groupName=**the name of the group **domainName=**the name of the domain **exists=**YES or NO indicating whether the group exists or not. **studio_rights=**YES or NO indicating if the studio rights are valid if the row exists diff_rights=Provides a list of studio rights where each right contains [=,+,-] in front. The "=" means that the studio right from the source is the same as the actual set in DV. The "+" means that the studio right is set in DV but not supposed to be set as per the spreadsheet. The "-" means that the studio right is not set in DV but it is supposed to be set as per the spreadsheet. Example: =ACCESS_TOOLS +UNLOCK_RESOURCE +READ_ALL_CONFIG +MODIFY_ALL_CONFIG -READ_ALL_RESOURCES	LONGVARCHAR

Direction	Parameter Name	Parameter Type
	| = end of the row	

Group Maintenance: Get Users

8. **getUsers –** Construct a SQL statement based on filters passed in and retrieve the set of users that match the criteria.

Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/getUsers

Direction	Parameter Name	Parameter Type
IN	**Datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL - Excel spreadsheet which gets uploaded to each DV server. • DB_LLE - SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD - SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**Environment_Name** – REQUIRED FILTER: The Composite server environment in which to get/update privileges for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**Organization** – RECOMMENDED FILTER: The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**Project** – RECOMMENDED FILTER: The name of the project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given project. If left blank, all projects will be updated.	VARCHAR(255)
IN	**SubProject** – OPTIONAL FILTER*:* The name of the sub-project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given sub-project. If left blank, all rows for the project will be updated.	VARCHAR(255)
IN	**User_Name** – OPTIONAL FILTER: The user name in which to get/update users.	VARCHAR(255)
IN	**Domain_Name** – OPTIONAL FILTER: The domain name in which to get/update users.	VARCHAR(255)
IN	**debug** – OPTIONAL: Y=debug on, N=debug off	CHAR(1)
OUT	DECLARE PUBLIC TYPE userRowType ROW (	userRowType

Direction	Parameter Name	Parameter Type
	rownum INTEGER, sheetRownum INTEGER, datasource VARCHAR(255), sheetName VARCHAR(255), Organization VARCHAR(255), Project VARCHAR(255), SubProject VARCHAR(255), UserName VARCHAR(255), Description VARCHAR(1024), "Domain" VARCHAR(255), "Env Type" VARCHAR(255), Password VARCHAR(255), Force CHAR(1), GroupList VARCHAR(255), UserName_lowercase VARCHAR(255), IsActive VARCHAR(1))	

User Maintenance: Update Users

These scripts are used for rebinding the generation scripts from one folder to another.

9. **updateUsersDriver** – This script is used to create or update composite users only. It does not create or update LDAP users. Reads the Composite Privileges spreadsheet and gets the list of users from the sheet: User_List

   Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/updateUsersDriver

   9.1. Limitations:

   9.1.1. If the domain is "composite" and the user does not exist then the user is created.

   9.1.2. If the domain is "composite" and the user does exist then the user is updated.

   9.1.3. If the domain is LDAP no action is taken at all as users are not maintained in Composite.

Direction	Parameter Name	Parameter Type
IN	**Datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL - Excel spreadsheet which gets uploaded to each DV server. • DB_LLE - SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD - SQL Server production database. This	VARCHAR(255)

Direction	Parameter Name	Parameter Type
	connection should always be used unless developing new functionality.	
IN	**Environment_Name** – REQUIRED FILTER: The Composite server environment in which to get/update privileges for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**Organization** – RECOMMENDED FILTER: The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**Project** – RECOMMENDED FILTER: The name of the project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given project. If left blank, all projects will be updated.	VARCHAR(255)
IN	**SubProject** – OPTIONAL FILTER*:* The name of the sub-project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given sub-project. If left blank, all rows for the project will be updated.	VARCHAR(255)
IN	**User_Name** – OPTIONAL FILTER*:* The user name for which to create/update users.	VARCHAR(255)
IN	**Domain_Name** – OPTIONAL FILTER: The domain name in which to get/update users.	VARCHAR(255)
IN	**preview** – N or null(default)=Do not preview. Execute the privileges. Y=preview what will get set but don't actually execute the operation.	CHAR(1)
IN	**inDebug1** – N/Null=No debug, Y (default)=Debug for this procedure - basic messages.	CHAR(1)
IN	**inDebug1Console** – N/Null (default)=No debug, Y=Debug console. Determines whether to print to the Studio console window or not.	CHAR(1)
IN	**inDebug1CISLog** – N/Null (default)=No debug, Y=Debug log. Determines whether to print to the DV log or not.	CHAR(1)
OUT	**rowsProcessed** – number of rows processed from the spreadsheet	INTEGER
OUT	**logOutput** – The output log that gets printed to the command line console is also output.	LONGVARCHAR

User Maintenance: Delete Users

10. **deleteUsersDriver –** This script is used to delete composite users only. It does not create or update LDAP users. Reads the Composite Privileges spreadsheet and gets the list of users from the sheet: User_List.

    Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/deleteUsersDriver

10.1. Limitations:

10.1.1. If the domain is "composite" and the user does not exist then the user is NOT deleted.

10.1.2. If the domain is "composite" and the user does exist then the user is deleted.

10.1.3. If the domain is LDAP and the user does not exist then the user is NOT deleted.

10.1.4. If the domain is LDAP and the user does exist then the user is NOT deleted.

Direction	Parameter Name	Parameter Type
IN	**Datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL - Excel spreadsheet which gets uploaded to each DV server. • DB_LLE - SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD - SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**Environment_Name** – REQUIRED FILTER: The Composite server environment in which to get/update privileges for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**Organization** – RECOMMENDED FILTER: The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**Project** – RECOMMENDED FILTER: The name of the project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given project. If left blank, all projects will be updated.	VARCHAR(255)
IN	**SubProject** – OPTIONAL FILTER*:* The name of the sub-project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given sub-project. If left blank, all rows for the project will be updated.	VARCHAR(255)
IN	**User_Name** – OPTIONAL FILTER*:* The user name for which to delete users.	VARCHAR(255)
IN	**Domain_Name** – OPTIONAL FILTER: The domain name in which to delete users.	VARCHAR(255)
IN	**preview** – N or null(default)=Do not preview. Execute the privileges. Y=preview what will get set but don't actually execute the operation.	CHAR(1)

Direction	Parameter Name	Parameter Type
IN	**inDebug1** – N/Null=No debug, Y (default)=Debug for this procedure - basic messages.	CHAR(1)
IN	**inDebug1Console** – N/Null (default)=No debug, Y=Debug console. Determines whether to print to the Studio console window or not.	CHAR(1)
IN	**inDebug1CISLog** – N/Null (default)=No debug, Y=Debug log. Determines whether to print to the DV log or not.	CHAR(1)
OUT	**rowsProcessed** – number of rows processed from the spreadsheet	INTEGER
OUT	**logOutput** – The output log that gets printed to the command line console is also output.	LONGVARCHAR

User Maintenance: Validate Users

11. **validateUsersDrivers –** This script is used to validate users. Reads the Composite Privileges spreadsheet and validates whether the group exists or not using the sheet: User_List.

    Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/validateUsersDriver

Direction	Parameter Name	Parameter Type
IN	**Datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL – Excel spreadsheet which gets uploaded to each DV server. • DB_LLE – SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD – SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**Environment_Name** – REQUIRED FILTER: The Composite server environment in which to get/update privileges for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**Organization** – RECOMMENDED FILTER: The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**Project** – RECOMMENDED FILTER: The name of the project that is hosted on DV for this “Env Type”. This provides a filter for only applying privileges for the given project. If left blank, all projects	VARCHAR(255)

Direction	Parameter Name	Parameter Type
	will be updated.	
IN	**SubProject** – OPTIONAL FILTER*:* The name of the sub-project that is hosted on DV for this “Env Type”. This provides a filter for only applying privileges for the given sub-project. If left blank, all rows for the project will be updated.	VARCHAR(255)
IN	**User_Name** – OPTIONAL FILTER*:* The user name for which to validate users.	VARCHAR(255)
IN	**Domain_Name** – OPTIONAL FILTER: The domain name in which to validate users.	VARCHAR(255)
IN	**inDebug1** – N/Null=No debug, Y (default)=Debug for this procedure – basic messages.	CHAR(1)
IN	**inDebug1Console** – N/Null (default)=No debug, Y=Debug console. Determines whether to print to the Studio console window or not.	CHAR(1)
IN	**inDebug1CISLog** – N/Null (default)=No debug, Y=Debug log. Determines whether to print to the DV log or not.	CHAR(1)
IN	**inDebug2** – N/Null (default)=No debug, Y=Debug for 2nd and 3rd tier procedures (deep debug).	CHAR(1)
OUT	**validUsers** – true=if all users are present (valid). False=if one or more users are not present (invalid).	VARCHAR
OUT	**rowsProcessed** – number of rows processed from the spreadsheet	INTEGER
OUT	**logOutput** – The output log that gets printed to the command line console is also output.	LONGVARCHAR
OUT	**userResult** – Result with pipe delimited rows	LONGVARCHAR

Database Maintenance: Load Database Privileges

12. **loadPrivileges_EXCEL –** This procedure is the generic interface for inserting and updating privileges from the generic Resource_Privileges_LOAD_DB.xlsx spreadsheet. The location of the resource in DV is as follows: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/Metadata/Privileges_DS_EXCEL/Resource_Privileges_LOAD_DB.xlsx.

    It does not delete privilege rows. Any deletion of privileges should be done through “modify_06_COMPOSITE_PRIVILEGES”. In order to update a row, it uses the following columns to detect the existence of a row: ISACTIVE=Y, ORGANIZATION_NAME, PROJECT_NAME, SUBPROJECT_NAME, RESOURCE_PATH, RESOURCE_TYPE, DOMAIN, NAME_TYPE, NAME, and ENVIRONMENT_NAME. If no rows are detected then an insert is performed.

Location:
/shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/loadPrivileges_EXCEL

Direction	Parameter Name	Parameter Type
IN	**datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL – Excel spreadsheet which gets uploaded to each DV server. • DB_LLE – SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD – SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**inPerformInsertUpdate** – Y or N. Y=yes perform insert or update. N=no do not perform insert or update but display the same messages to see what will be executed.	CHAR(1)
OUT	**rowsProcessed** – Number of rows processed in total.	INTEGER
OUT	**rowsInserted** – Number of rows inserted into the privilege database.	INTEGER
OUT	**rowsUpdated** – Number of rows updated into the privilege database.	INTEGER
OUT	**rowsDeleted** – Number of rows deleted into the privilege database.	INTEGER
OUT	**rowsInactive** – Number of rows set to inactive into the privilege database.	INTEGER
OUT	**duplicateRows** – number of duplicate rows.	INTEGER
OUT	**duplicateRowMsg** – message for duplicate rows.	LONGVARCHAR
OUT	**message** – a general message	LONGVARCHAR

Database Maintenance: Load Database Groups

13. **loadGroups_EXCEL –** This procedure is the generic interface for inserting and updating groups from the generic Resource_Privileges_LOAD_DB.xlsx spreadsheet. The location of the resource in DV is as follows: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/Metadata/Privileges_DS_EXCEL/Resource_Privileges_LOAD_DB.xlsx.

    It does not delete group rows. Any deletion of groups should be done through “modify_07_COMPOSITE_STUDIO_RIGHTS”. In order to update a row, it uses the following columns to detect the existence of a row: ISACTIVE=Y, ORGANIZATION_NAME, PROJECT_NAME, GROUP_NAME, DOMAIN, ENVIRONMENT_NAME. If no rows are detected then an insert is performed.

Location:
/shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/loadGroups_EXCEL

Direction	Parameter Name	Parameter Type
IN	**datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL – Excel spreadsheet which gets uploaded to each DV server. • DB_LLE – SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD – SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**inPerformInsertUpdate** – Y or N. Y=yes perform insert or update. N=no do not perform insert or update but display the same messages to see what will be executed.	CHAR(1)

IN	**useDefaultConfig** – Y or N. Y=use the default config values for the following thus overriding the spreadsheet values.
	READ_ALL_CONFIG=null - this value unset
	MODIFY_ALL_CONFIG=null - this value unset
	READ_ALL_RESOURCES=null - this value unset
	MODIFY_ALL_RESOURCES=null- this value unset
		CHAR(1)

OUT	**rowsProcessed** – Number of rows processed in total.	INTEGER
OUT	**rowsInserted** – Number of rows inserted into the privilege database.	INTEGER
OUT	**rowsUpdated** – Number of rows updated into the privilege database.	INTEGER
OUT	**rowsDeleted** – Number of rows deleted into the privilege database.	INTEGER
OUT	**rowsInactive** – Number of rows set to inactive into the privilege database.	INTEGER
OUT	**duplicateRows** – number of duplicate rows.	INTEGER
OUT	**duplicateRowMsg** – message for duplicate rows.	LONGVARCHAR

Direction	Parameter Name	Parameter Type
OUT	**message** – a general message	LONGVARCHAR

Database Maintenance: Load Database Users

14. **loadUsers_EXCEL –** This procedure is the generic interface for inserting and updating users from the generic Resource_Privileges_LOAD_DB.xlsx spreadsheet. The location of the resource in DV is as follows: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/Metadata/Privileges_DS_EXCEL/Resource_Privileges_LOAD_DB.xlsx.

    It does not delete user rows. Any deletion of groups should be done through "modify_08_COMPOSITE_USER". In order to update a row, it uses the following columns to detect the existence of a row: ISACTIVE=Y, ORGANIZATION_NAME, PROJECT_NAME, SUBPROJECT_NAME, USER_NAME, DOMAIN, ENVIRONMENT_NAME. If no rows are detected then an insert is performed.

    Location:
    /shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/loadUsers_EXCEL

Direction	Parameter Name	Parameter Type
IN	**datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL – Excel spreadsheet which gets uploaded to each DV server. • DB_LLE – SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD – SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**inPerformInsertUpdate** – Y or N. Y=yes perform insert or update. N=no do not perform insert or update but display the same messages to see what will be executed.	CHAR(1)
OUT	**rowsProcessed** – Number of rows processed in total.	INTEGER
OUT	**rowsInserted** – Number of rows inserted into the privilege database.	INTEGER
OUT	**rowsUpdated** – Number of rows updated into the privilege database.	INTEGER
OUT	**rowsDeleted** – Number of rows deleted into the privilege	INTEGER

Direction	Parameter Name	Parameter Type
	database.	
OUT	**rowsInactive** – Number of rows set to inactive into the privilege database.	INTEGER
OUT	**duplicateRows** – number of duplicate rows.	INTEGER
OUT	**duplicateRowMsg** – message for duplicate rows.	LONGVARCHAR
OUT	**message** – a general message	LONGVARCHAR

Database Maintenance: Perform Supporting Table Maintenance

15. **performSupportingTableMaintenance –** This procedure is used to perform table maintenance on the supporting tables in the database.

    Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/performSupportingTableMaintenance

Table Name	Require Fields
ORGANIZATION_TYPE:	ORGANIZATION_NAME
PROJECT_TYPE:	ORGANIZATION_NAME
	PROJECT_NAME
SUBPROJECT_TYPE:	SUBPROJECT_NAME
	PROJECT_NAME
	ORGANIZATION_NAME
DOMAIN_TYPE:	DOMAIN_NAME
ENVIRONMENT_TYPE:	ENVIRONMENT_NAME

Rules:

1. performInsert:

When performInsert=Y, perform the insert of the new row for the given table.

When performInsert=N, do not perform the insert of the new row for the given table but display what will happen.

2. organization

Used for tables ORGANIZATION_TYPE or PROJECT_TYPE

If not null, then test ORGANIZATION_TYPE.ORGANIZATION_NAME for existence.

If not exist then insert.

3. projectName

Used for table PROJECT_TYPE.ORGANIZATION_NAME and PROJECT_TYPE.PROJECT_NAME

If projectName and organization is not null then test PROJECT_TYPE.ORGANIZATION_NAME and PROJECT_TYPE.PROJECT_NAME for existence.

If not exist then insert.

4. subprojectList

Used for table SUBPROJECT_TYPE.SUBPROJECT_NAME and PROJECT_NAME and ORGANIZATION_NAME

This may be a comma separated list of subprojects.

For each subproject and project, if they are not null then test PROJECT_TYPE.SUBPROJECT_NAME and PROJECT_NAME and ORGANIZATION_NAME for existence.

If not exist then insert.

5. domainNameList

Used for tables DOMAIN_TYPE

This may be a comma separated list of domain names.

If not null, then test DOMAIN_TYPE.DOMAIN_NAME for existence.

If not exist then insert.

6. environmentNameList

Used for tables ENVIRONMENT_TYPE

This may be a comma separated list of environent names.

If not null, then test ENVIRONMENT_TYPE.ENVIRONMENT_NAME for existence.

If not exist then insert.

Direction	Parameter Name	Parameter Type
IN	**performInsert** – Y or N. Y=yes perform insert or update. N=no do not perform insert or update but display the same messages	CHAR(1)

Direction	Parameter Name	Parameter Type
	to see what will be executed.	
IN	**organization** – leave null unless you want to insert into ORGANIZATION_TYPE and/or PROJECT_TYPE. Examples (1 of): ORG1, ORG2, ORG3.	VARCHAR
IN	**projectName** – leave null unless you want to insert into PROJECT_TYPE and/or SUBPROJECT_TYPE. Examples (1 of): PROJECT1, PROJECT2.	VARCHAR
IN	**subprojectList** – leave null unless you want to insert into SUBPROJECT_TYPE. This may be a comma separated list of subprojects. Examples (1 of): SB1, SB2, SB3.	LONGVARCHAR
IN	**domainName List**– leave null unless you want to insert into DOMAIN_TYPE. This may be a comma separated list of domain names. Examples (1 of): ldap, composite, dynamic	LONGVARCHAR
IN	**environemntNameList** – leave null unless you want to insert into ENVIRONMENT_TYPE. This may be a comma separated list of environment names. Examples (1 of): DEV, UAT, PROD	LONGVARCHAR
OUT	**message** – a general message	LONGVARCHAR

Database Maintenance: Fix Privilege Table

16. **fix_PROD_ORA_COMPOSITE_PRIVILEGES –** This procedure is used to perform table maintenance on the COMPOSITE_PRIVILEGES Oracle PROD table to fix any issues.

16.1. **fix_PROD_SS_COMPOSITE_PRIVILEGES –** This procedure is used to perform table maintenance on the COMPOSITE_PRIVILEGES SQL Server PROD table to fix any issues.

16.2. The following columns should not contain any blanks, spaces, empty character or any characters other than [X,Y,N]. This procedure updates the COMPOSITE_PRIVILEGES table and sets the column to NULL if it does not X, Y and N. The column being updated include: INITIALIZE, READ_ACCESS, WRITE_ACCESS, EXECUTE_ACCESS, SELECT_ACCESS, UPDATE_ACCESS, INSERT_ACCESS, DELETE_ACCESS, GRANT_ACCESS.

Location:
/shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/TableMaintenance/fix_...

Direction	Parameter Name	Parameter Type
IN	**inOrganization** – RECOMMENDED FILTER: The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**inProject** – RECOMMENDED FILTER: The name of the project that is hosted on DV for this "Env Type". This provides a filter for only applying privileges for the given project. If left blank, all	VARCHAR(255)

Direction	Parameter Name	Parameter Type
	projects will be updated.	
OUT	**countINITIALIZE** – number of rows modified.	INTEGER
OUT	**countREAD_ACCESS** – number of rows modified.	INTEGER
OUT	**countWRITE_ACCESS** – number of rows modified.	INTEGER
OUT	**countEXECUTE_ACCESS** – number of rows modified.	INTEGER
OUT	**countSELECT_ACCESS** – number of rows modified.	INTEGER
OUT	**countUPDATE_ACCESS** – number of rows modified.	INTEGER
OUT	**countINSERT_ACCESS** – number of rows modified.	INTEGER
OUT	**countDELETE_ACCESS** – number of rows modified.	INTEGER
OUT	**countGRANT_ACCESS** – number of rows modified.	INTEGER
OUT	**countOWNER_NAME_DEV** – number of rows modified.	INTEGER
OUT	**countOWNER_NAME_UAT** – number of rows modified.	INTEGER
OUT	**countOWNER_NAME_PROD** – number of rows modified.	INTEGER
OUT	**countOWNER_DOMAIN_DEV** – number of rows modified.	INTEGER
OUT	**countOWNER_DOMAIN_UAT** – number of rows modified.	INTEGER
OUT	**countOWNER_DOMAIN_PROD** – number of rows modified.	INTEGER

Database Maintenance: Modify Organization Type

17. **modify_01_ORGANIZATION_TYPE –** This procedure is used to insert/update/delete/inactivate rows in the ORGANIZATION_TYPE table.

    Location:
    /shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/TableMaintenance/
    modify_01_ORGANIZATION_TYPE

    Examples:

        ORG1

        ORG2

Direction	Parameter Name	Parameter Type
IN	**datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL – Excel spreadsheet which gets uploaded to each DV server. • DB_LLE – SQL Server database for lower level environments. Basically it is used for testing purposes	VARCHAR(255)

Direction	Parameter Name	Parameter Type
	and developing new functionality. • DB_PROD – SQL Server production database. This connection should always be used unless developing new functionality.	
IN	**setInactive** – Set the ISACTIVE flag to N. Y=if the record exists set it inactive. Overrides performDelete. N/null=do not perform this operation.	VARCHAR(1)
IN	**performDelete** – Delete the row if it exists. Y=perform a delete using filters supplied. N/null=do not perform delete but perform insert or update instead.	VARCHAR(1)
IN	**inORGANIZATION_NAME** – The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
OUT	**message** – A general message	LONGVARCHAR

Database Maintenance: Modify Domain Type

18. **modify_02_DOMAIN_TYPE –** This procedure is used to insert/update/delete/inactivate rows in the DOMAIN_TYPE table. The two composite domains are "composite" and "dynamic". Additional domain names for LDAP will be whatever the administrator chooses such as "ldap".

Location:
/shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/TableMaintenance/modify_02_DOMAIN_TYPE

Examples:

ldap
composite
dynamic

Direction	Parameter Name	Parameter Type
IN	**datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL – Excel spreadsheet which gets uploaded to each DV server. • DB_LLE – SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD – SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)

Direction	Parameter Name	Parameter Type
IN	**setInactive** – Set the ISACTIVE flag to N. Y=if the record exists set it inactive. Overrides performDelete. N/null=do not perform this operation.	VARCHAR(1)
IN	**performDelete** – Delete the row if it exists. Y=perform a delete using filters supplied. N/null=do not perform delete but perform insert or update instead.	VARCHAR(1)
IN	**inDOMAIN_NAME** – The domain name to insert/update/delete/inactivate. If both setInactive and performDelete are null or N then perform the insert. If the row does not exist then insert. If the row already exists then do nothing.	VARCHAR(255)
OUT	**message** – A general message	LONGVARCHAR

Database Maintenance: Modify Environment Type

19. **modify_03_ENVIRONMENT_TYPE –** This procedure is used to insert/update/delete/inactivate rows in the ENVIRONMENT_TYPE table. The environment will match the LDAP server environment. In some organizations, they will have an environment for each deployment area such as DEV, UAT and PROD. Other organizations may only have a single LDAP environment such as PROD in which all DV instances are connected to. The name is an abstract name but descriptive of the LDAP instance that the DV domain is connecting to.

    Location:
    /shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/TableMaintenance/modify_03_ENVIRONMENT_TYPE

    Examples:

    DEV
    UAT
    PROD

Direction	Parameter Name	Parameter Type
IN	**datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL – Excel spreadsheet which gets uploaded to each DV server. • DB_LLE – SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD – SQL Server production database. This connection should always be used unless developing	VARCHAR(255)

Direction	Parameter Name	Parameter Type
	new functionality.	
IN	**setInactive** – Set the ISACTIVE flag to N. Y=if the record exists set it inactive. Overrides performDelete. N/null=do not perform this operation.	VARCHAR(1)
IN	**performDelete** – Delete the row if it exists. Y=perform a delete using filters supplied. N/null=do not perform delete but perform insert or update instead.	VARCHAR(1)
IN	**inENVIRONMENT_NAME** – The environment name to insert/update/delete/inactivate. If both setInactive and performDelete are null or N then perform the insert. If the row does not exist then insert. If the row already exists then do nothing.	VARCHAR(255)
OUT	**message** – A general message	LONGVARCHAR

Database Maintenance: Modify Project Type

20. **modify_04_PROJECT_TYPE –** This procedure is used to insert/update/delete/inactivate rows in the PROJECT_TYPE table. This procedure performs a cascade modify for the ORGANIZATION_TYPE table also. However, it does not perform the cascade inactivate. That must be done explicitly on each table.

    Location:
    /shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/TableMaintenance/modify_04_PROJECT_TYPE

    Examples:

PROJECT_NAME	ORGANIZATION_NAME
PROJECT1	ORG1
PROJECT2	ORG2

Direction	Parameter Name	Parameter Type
IN	**datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL – Excel spreadsheet which gets uploaded to each DV server. • DB_LLE – SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD – SQL Server production database. This connection should always be used unless developing	VARCHAR(255)

Direction	Parameter Name	Parameter Type
	new functionality.	
IN	**setInactive** – Set the ISACTIVE flag to N. Y=if the record exists set it inactive. Overrides performDelete. N/null=do not perform this operation.	VARCHAR(1)
IN	**performDelete** – Delete the row if it exists. Y=perform a delete using filters supplied. N/null=do not perform delete but perform insert or update instead.	VARCHAR(1)
IN	**inPROJECT** – The Composite server environment in which to update studio rights for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**inORGANIZATION_NAME** – The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
OUT	**message** – A general message	LONGVARCHAR

Database Maintenance: Modify SubProject Type

21. **modify_05_SUBPROJECT_TYPE –** This procedure is used to insert/update/delete/inactivate rows in the SUBPROJECT_TYPE table. This procedure performs a cascade modify for the PROJECT_TYPE table also. This procedure performs a cascade modify for the ORGANIZATION_TYPE table also. However, it does not perform the cascade inactivate. That must be done explicitly on each table.

    Location:
    /shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/TableMaintenance/modify_05_SUBPROJECT_TYPE

    Examples:

SUBPROJECT_NAME	PROJECT_NAME
SB1	PROJECT1
SB2	PROJECT2

Direction	Parameter Name	Parameter Type
IN	**datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL – Excel spreadsheet which gets uploaded to each DV server. • DB_LLE – SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD – SQL Server production database. This connection should always be used unless developing	VARCHAR(255)

Direction	Parameter Name	Parameter Type
	new functionality.	
IN	**setInactive** – Set the ISACTIVE flag to N. Y=if the record exists set it inactive. Overrides performDelete. N/null=do not perform this operation.	VARCHAR(1)
IN	**performDelete** – Delete the row if it exists. Y=perform a delete using filters supplied. N/null=do not perform delete but perform insert or update instead.	VARCHAR(1)
IN	**inSUBPROJECT_NAME** – The subproject name to insert/update/delete/inactivate. If both setInactive and performDelete are null or N then perform the insert. If the row does not exist then insert. If the row already exists then do nothing.	VARCHAR(255)
IN	**inPROJECT** – The Composite server environment in which to update studio rights for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**inORGANIZATION_NAME** – The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
OUT	**message** – A general message	LONGVARCHAR

Database Maintenance: Modify Composite Privileges

22. **modify_06_COMPOSITE_PRIVILEGES –** This procedure is used to insert/update/delete/inactivate rows in the COMPOSITE_PRIVILEGES table. These are rows that come from the Privileges_shared, Privileges_databases, and Privileges_webservices sheets in the Resource_Privileges_LOAD_DB.xlsx spreadsheet.

    For deleting records the following fields are used to create the where clause: PRIVIELGE_ID - if present then it only uses this to delete that 1 record.

    The following list of fields are used for constructing a dynamic where clause. If a field contains a null then it is not used for the where clause. Therefore it provides a lot of flexibility for deleting as much or little as you wish. inENVIRONMENT_NAME, inORGANIZATION_NAME, inPROJECT_NAME, inSUBPROJECT_NAME, inRESOURCE_PATH, inNAME, inNAME_TYPE, inDOMAIN_NAME

    Location:
    /shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/TableMaintenance/modify_06_COMPOSITE_PRIVILEGES

    Example:

PRIVILEGE_ID:	2
ORGANIZATION_NAME:	ORG1
PRIVILEGE_TYPE:	SHARED

```
PROJECT_NAME:             Common
SUBPROJECT_NAME:          [NULL]
RESOURCE_PATH:            /shared/examples
RESOURCE_TYPE:            CONTAINER
RECURSE_DEPENDENCIES:     [NULL]
RECURSE_DEPENDENTS:       [NULL]
RECURSE_CHILD:            YM
REVOKE_ALL:               N
NAME:                     group1
NAME_TYPE:                GROUP
DOMAIN_NAME:              composite
ENVIRONMENT_NAME:         DEV
READ_ACCESS:              X
WRITE_ACCESS:             [NULL]
EXECUTE_ACCESS:           X
SELECT_ACCESS:            X
UPDATE_ACCESS:            [NULL]
INSERT_ACCESS:            [NULL]
DELETE_ACCESS:            [NULL]
GRANT_ACCESS:             [NULL]
INITIALIZE:               X
OWNER_NAME:               [NULL]
OWNER_DOMAIN:             [NULL]
COMMENTS:                 initialize
CREATEDBY:                CMPPRV
CREATEDDATE:              2017-05-17 15:31:13
UPDATEDBY:                CMPPRV
UPDATEDDATE:              2017-05-17 15:31:13
ISACTIVE:                 Y
```

Direction	Parameter Name	Parameter Type
IN	**datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL – Excel spreadsheet which gets uploaded to each DV server. • DB_LLE – SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality.	VARCHAR(255)

Direction	Parameter Name	Parameter Type
	- DB_PROD – SQL Server production database. This connection should always be used unless developing new functionality.	
IN	**performInsertUpdate** – Y or N. Y=yes perform insert or update. N=no do not perform insert or update but display the same messages to see what will be executed.	VARCHAR(1)
IN	**setInactive** – Set the ISACTIVE flag to N. Y=if the record exists set it inactive. Overrides performDelete. N/null=do not perform this operation.	VARCHAR(1)
IN	**performDelete** – Delete the row if it exists. Y=perform a delete using filters supplied. N/null=do not perform delete but perform insert or update instead.	VARCHAR(1)
IN	**inPRIVILEGE_ID** – This is used only when setting a row inactive or deleting a specific row by its generated PRIVILEGE_ID.	DECIMAL(38,0)
IN	**inORGANIZATION_NAME** – The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**inPROJECT_NAME** – The project name to insert/update/delete/inactivate. If both setInactive and performDelete are null or N then perform the insert. If the row does not exist then insert. If the row already exists then do nothing.	VARCHAR(255)
IN	**inSUBPROJECT_NAME** – The subproject name to insert/update/delete/inactivate.	VARCHAR(255)
IN	**inRESOURCE_PATH** – The DV resource path.	VARCHAR(4000)
IN	**inRESOURCE_TYPE** – The DV resource type such as DATA_SOURCE, CONTAINER, TABLE, COLUMN or etc.	VARCHAR(255)
IN	**inRECURSE_**DEPENDENCIES – N or blank = Do not recurse. Y = Apply recursively to depenendencies	VARCHAR(1)
IN	**inRECURSE_DEPENDENTS** – N or blank = Do not recurse. Y = Apply recursively to depenents	VARCHAR(1)
IN	**inRECURSE_CHILD** – N or blank = do not recurse. Set specified privileges for the resource only. YM = Only apply modification recursively YC = Make child resources look like this resource recursively Concept 1: Only select YM to recurse child resources when the resource being set is at the lowest level at which it is safe to recursively set privileges for a given USER or GROUP. If the resource is an intermediate level folder with other branches offshoot from it then only apply the privilege modification to the resource folder using "N" unless all branches are required to have the same privilege settings. "YC" should only be used when	VARCHAR(2)

Direction	Parameter Name	Parameter Type
	you want to reset all privileges for other USERS or GROUPS to "NONE" except for the current USER or GROUP being set. Concept 2: If the higher level resource folder is set with "YC" then it is not necessary to set this USER or GROUP for branch-level resources.	
IN	**inREVOKE_ALL** – Revoke all privileges for all users and all groups prior to setting the specified privileges. N or blank=Do not revoke all privileges Y=Revoke all privileges This only needs to be done once per Resource Path group. If it user chooses to revoke all, then it must only be done on the first entry for a given Resource Path as it will set the privileges to "NONE" for all users and groups. If this value is set to Y on subsequent calls, it will reset any privileges previously set.	VARCHAR(1)
IN	**inNAME** – The user or group name.	VARCHAR(255)
IN	**inNAME_TYPE** – This is either USER or GROUP depending on the NAME field.	VARCHAR(255)
IN	**inDOMAIN_NAME** – The domain the group exists in.	VARCHAR(255)
IN	**inENVIRONMENT_NAME** – The Composite server environment in which to update studio rights for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**inREAD_ACCESS** – null or X to indicate access.	VARCHAR(1)
IN	**inWRITE_ACCESS** – null or X to indicate access.	VARCHAR(1)
IN	**inEXECUTE_ACCESS** – null or X to indicate access.	VARCHAR(1)
IN	**inSELECT_ACCESS** – null or X to indicate access.	VARCHAR(1)
IN	**inUPDATE_ACCESS** – null or X to indicate access.	VARCHAR(1)
IN	**inINSERT_ACCESS** – null or X to indicate access.	VARCHAR(1)
IN	**inDELETE_ACCESS** – null or X to indicate access.	VARCHAR(1)
IN	**inGRANT_ACCESS** – null or X to indicate access.	VARCHAR(1)
IN	**inINITIALIZE** – null or X to indicate access. This field should be ignored.	VARCHAR(1)
IN	**inOWNER_NAME** – The owner of the resource which gets set at the time the privilege is applied. For ENVIRONMENT_NAME=DEV, the owner should be null so that it does not get set. For other environments, it may be any valid user for that environment such as "admin".	VARCHAR(255)
IN	**inOWNER_DOMAIN** – The owner domain of the resource which gets set at the time the privilege is applied. For ENVIRONMENT_NAME=DEV, the owner domain should be null so that it does not get set. For other environments, it may be	VARCHAR(255)

Direction	Parameter Name	Parameter Type
	any valid domain in which OWNER_NAME is contained for that environment such as “composite”.	
IN	**inCOMMENTS** – a general comment.	VARCHAR(1024)
OUT	**actionType** – I=insert, U=update, D=delete, S=set inactive	CHAR(1)
OUT	**rowsProcessed** – Number of rows process for the action type	INTEGER
OUT	**duplicateRows** – Number of duplicate rows found	INTEGER
OUT	**duplicateRowMsg** – message for duplicate rows	LONGVARCHAR
OUT	**message** – a general message	LONGVARCHAR

Database Maintenance: Modify Composite Studio Rights

23. **modify_07_COMPOSITE_STUDIO_RIGHTS –** This procedure is used to insert/update/delete/inactivate rows in the COMPOSITE_STUDIO_RIGHTS table. These are rows that come from the Group_List sheet in the Resource_Privileges_LOAD_DB.xlsx spreadsheet.

    Location:
    /shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/TableMaintenance/
    modify_07_COMPOSITE_STUDIO_RIGHTS

    Example:

STUDIO_RIGHTS_ID:	2
ORGANIZATION_NAME:	ORG1
PROJECT_NAME:	Demo
GROUP_NAME:	Demo_Arch
GROUP_DESC:	Demo Architect
DOMAIN_NAME:	composite
ENVIRONMENT_NAME:	DEMO
ACCESS_FLAG:	X
UNDO_LOCK:	[NULL]
READ_ALL_CONFIG:	[NULL]
MODIFY_ALL_CONFIG:	[NULL]
READ_ALL_RESOURCES:	[NULL]
MODIFY_ALL_RESOURCES:	[NULL]
READ_ALL_STATUS:	X
MODIFY_ALL_STATUS:	[NULL]
READ_ALL_USER:	[NULL]
MODIFY_ALL_USER:	[NULL]

CREATEDBY:	CMPPRVP
CREATEDDATE:	2015-06-16 12:11:46
UPDATEDBY:	CMPPRVP
UPDATEDDATE:	2015-06-16 12:11:46
ISACTIVE:	Y

Direction	Parameter Name	Parameter Type
IN	**datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL – Excel spreadsheet which gets uploaded to each DV server. • DB_LLE – SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD – SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**performInsertUpdate** – Y or N. Y=yes perform insert or update. N=no do not perform insert or update but display the same messages to see what will be executed.	VARCHAR(1)
IN	**setInactive** – Set the ISACTIVE flag to N. Y=if the record exists set it inactive. Overrides performDelete. N/null=do not perform this operation.	VARCHAR(1)
IN	**performDelete** – Delete the row if it exists. Y=perform a delete using filters supplied. N/null=do not perform delete but perform insert or update instead.	VARCHAR(1)
IN	**inSTUDIO_RIGHTS_ID** – This is used only when setting a row inactive or deleting a specific row by its generated STUDIO_RIGHTS_ID.	DECIMAL(38,0)
IN	**inORGANIZATION_NAME** – The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**inPROJECT_NAME** – The project name to insert/update/delete/inactivate. If both setInactive and performDelete are null or N then perform the insert. If the row does not exist then insert. If the row already exists then do nothing.	VARCHAR(255)
IN	**inSUBPROJECT_NAME** – The subproject name to insert/update/delete/inactivate.	VARCHAR(255)
IN	**inGROUP_NAME** – The name of the group to be modified.	VARCHAR(255)

Direction	Parameter Name	Parameter Type
IN	**inGROUP_DESC** – The the description of the group.	VARCHAR(255)
IN	**inDOMAIN_NAME** – The domain the group exists in.	VARCHAR(255)
IN	**inENVIRONMENT_NAME** – The Composite server environment in which to update studio rights for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**inACCESS_FLAG** – null or X to indicate access.	VARCHAR(1)
IN	**inUNDO_LOCK** – null or X to indicate access.	VARCHAR(1)
IN	**inREAD_ALL_CONFIG** – null or X to indicate access.	VARCHAR(1)
IN	**inMODIFY_ALL_CONFIG** – null or X to indicate access.	VARCHAR(1)
IN	**inREAD_ALL_RESOURCES** – null or X to indicate access.	VARCHAR(1)
IN	**inMODIFY_ALL_RESOURCES** – null or X to indicate access.	VARCHAR(1)
IN	**inREAD_ALL_STATUS** – null or X to indicate access.	VARCHAR(1)
IN	**inMODIFY_ALL_STATUS** – null or X to indicate access.	VARCHAR(1)
IN	**inREAD_ALL_USER** – null or X to indicate access.	VARCHAR(1)
IN	**inMODIFY_ALL_USER** – null or X to indicate access.	VARCHAR(1)
OUT	**actionType** – I=insert, U=update, D=delete, S=set inactive	CHAR(1)
OUT	**rowsProcessed** – Number of rows process for the action type	INTEGER
OUT	**duplicateRows** – Number of duplicate rows found	INTEGER
OUT	**duplicateRowMsg** – message for duplicate rows	LONGVARCHAR
OUT	**message** – a general message	LONGVARCHAR

Database Maintenance: Modify Composite User

24. **modify_08_COMPOSITE_USER –** This procedure is used to insert/update/delete/inactivate rows in the COMPOSITE_USER table. These are rows that come from the User_List sheet in the Resource_Privileges_LOAD_DB.xlsx spreadsheet.

    Location:
    /shared/ASAssets/BestPractices_v81/PrivilegeScripts/ModifyDBPrivileges/TableMaintenance/modify_08_COMPOSITE_USER

    Example:

USERID:	1
ORGANIZATION_NAME:	ORG1
PROJECT_NAME:	Common
SUBPROJECT_NAME:	SB1
USER_NAME:	user1
USER_DESC:	user desc

DOMAIN_NAME:	composite
ENVIRONMENT_NAME:	DEV
USER_PWD:	password
FORCE_USER:	1
GROUP_LIST:	group1|composite
USER_NAME_LOWERCASE:	user1
CREATEDBY:	CMPPRV
CREATEDDATE:	2017-05-17 16:57:13
UPDATEDBY:	CMPPRV
UPDATEDDATE:	2017-05-17 16:57:13
ISACTIVE:	Y

Direction	Parameter Name	Parameter Type
IN	**datasource** – RECOMMENDED FILTER: the datasource where the privileges are stored. If not parameter is provided the default value is used from defaultValues.datasource. Possible values include: • EXCEL – Excel spreadsheet which gets uploaded to each DV server. • DB_LLE – SQL Server database for lower level environments. Basically it is used for testing purposes and developing new functionality. • DB_PROD – SQL Server production database. This connection should always be used unless developing new functionality.	VARCHAR(255)
IN	**performInsertUpdate** – Y or N. Y=yes perform insert or update. N=no do not perform insert or update but display the same messages to see what will be executed.	VARCHAR(1)
IN	**setInactive** – Set the ISACTIVE flag to N. Y=if the record exists set it inactive. Overrides performDelete. N/null=do not perform this operation.	VARCHAR(1)
IN	**performDelete** – Delete the row if it exists. Y=perform a delete using filters supplied. N/null=do not perform delete but perform insert or update instead.	VARCHAR(1)
IN	**inUSERID** – This is used only when setting a row inactive or deleting a specific row by its generated USERID.	DECIMAL(38,0)
IN	**inORGANIZATION_NAME** – The name of the organization such as ORG1 or ORG2.	VARCHAR(255)
IN	**inPROJECT_NAME** – The project name to insert/update/delete/inactivate. If both setInactive and performDelete are null or N then perform the insert. If the row does not exist then insert. If the row already exists then do	VARCHAR(255)

Direction	Parameter Name	Parameter Type
	nothing.	
IN	**inSUBPROJECT_NAME** – The subproject name to insert/update/delete/inactivate.	VARCHAR(255)
IN	**inUSER_NAME** – The name of the user to be modified.	VARCHAR(255)
IN	**inUSER_DESC** – The the description of the user.	VARCHAR(255)
IN	**inDOMAIN_NAME** – The domain the group exists in.	VARCHAR(255)
IN	**inENVIRONMENT_NAME** – The Composite server environment in which to update studio rights for: [DEV, UAT, PROD].	VARCHAR(255)
IN	**inUSER_PWD** – The user password.	VARCHAR(255)
IN	**inFORCE_USER** – 0 or 1 to indicate access.	VARCHAR(1)
IN	**inGROUP_LIST** – Pipe separated list of group|domain. It may contain a list of pairs separated by spaces. It may not exceed 2048 charaacters. Below the list contains a space separate list of group|domain pairs: E.g. G1|D1 G2|D1 G3|D1 group1|composite ^---GROUP_NAME|DOMAIN_NAME	VARCHAR(2048)
OUT	**actionType** – I=insert, U=update, D=delete, S=set inactive	CHAR(1)
OUT	**rowsProcessed** – Number of rows process for the action type	INTEGER
OUT	**duplicateRows** – Number of duplicate rows found	INTEGER
OUT	**duplicateRowMsg** – message for duplicate rows	LONGVARCHAR
OUT	**message** – a general message	LONGVARCHAR

Group Clean-up: Remove Groups

25. **remove_groups –** This procedure is used to remove the existence of groups from DV using the free-form list "group_list_string". Edit the group_list_string here: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/CleanupGroups/group_list_string

    Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/CleanupGroups/remove_groups

Direction	Parameter Name	Parameter Type
IN	**indomainName** – Provide the Composite domain in which to validate the group list. This may be a composite domain or ldap domain. e.g. composite, ldap or ldap_im or whatever the domain name is where the groups reside.	VARCHAR(255)
IN	**inExceptionList** – [OPTINOAL] Add an item to the exclusion list to keep from deleting that group. Each item must be appended	LONGVARCHAR

Direction	Parameter Name	Parameter Type
	with a comma including the last one.	
IN	**inInclusionList** – [OPTIONAL] Add an item to the inclusion list to test one or more specific groups. Each item must be appended with a comma including the last one.	LONGVARCHAR
OUT	**result** – The result of this operation returns a cursor of rows	PIPE (GroupName VARCHAR(255), DomainName VARCHAR(255), Status VARCHAR(255))

Group Clean-up: Validate Groups

26. **validate_groups –** This procedure is used to validate the existence of groups from DV using the free-form list "group_list_string". Edit the group_list_string here: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/CleanupGroups/group_list_string

    Location: /shared/ASAssets/BestPractices_v81/PrivilegeScripts/CleanupGroups/validate_groups

Direction	Parameter Name	Parameter Type
IN	**indomainName** – Provide the Composite domain in which to validate the group list. This may be a composite domain or ldap domain. e.g. composite, ldap or ldap_im or whatever the domain name is where the groups reside.	VARCHAR(255)
IN	**inExceptionList** – [OPTINOAL] Add an item to the exclusion list to keep from deleting that group. Each item must be appended with a comma including the last one.	LONGVARCHAR
IN	**inInclusionList** – [OPTIONAL] Add an item to the inclusion list to test one or more specific groups. Each item must be appended with a comma including the last one.	LONGVARCHAR
OUT	**result** – The result of this operation returns a cursor of rows	PIPE (GroupName VARCHAR(255), DomainName VARCHAR(255), Status VARCHAR(255))