WHITE PAPER

COMPOSITE
SOFTWARE

Composite Data Virtualization

Composite PS Promotion and Deployment Tool

Privilege Module User Guide

Composite Professional Services

March 2014

TABLE OF CONTENTS

DOCUMENT CONTROL

Version History

Version	Date	Author	Description
1.0	8/5/2011	Mike Tinius	Initial revision for Privilege Module User Guide
1.1	3/9/2012	Mike Tinius	Added resourceType to the schema
1.2	8/20/2013	Mike Tinius	Fixed method signature documentation for generatePrivilegesXML
3.0	8/21/2013	Mike Tinius	Updated docs to Cisco format.
3.1	2/18/2014	Mike Tinius	Added PrivilegeModule.xml option "updateDepenciesRecursively". Prepare docs for open source.
3.2	3/24/2014	Mike Tinius	Changed references of XML namespace to www.dvbu.cisco.com

Related Documents

Document	File Name	Author
Composite PS Promotion and Deployment Tool User's Guide v1.0	*Composite PS Promotion and Deployment Tool User's Guide v1.0.pdf*	Mike Tinius

Composite Products Referenced

Composite Product Name	Version
Composite Information Server	5.1, 5.2, 6.0, 6.1, 6.2

INTRODUCTION

Purpose

The purpose of the Privilege Module User Guide is to demonstrate how to effectively use the Privilege Module and execute actions. Every resource in CIS has privileges assigned associated with it. The easiest way to demonstrate this is to use Composite Studio and right-mouse click on /shared and then select “Privileges” at the bottom of the selection list. The pop-up windows shows privileges for the composite domain and dynamic domain. It then shows privileges for Groups and Users within the domain. The types of privileges will vary for the resources but in general the list of privileges includes: READ, WRITE, EXECUTE, SELECT, UPDATE, INSERT, DELETE and GRANT. When assigning privileges, one should keep in mind that they can be assigned to the specific resource or they can be assigned recursively to child resources. Additionally, the user can force the privilege profile to look exactly like the privilege profile being set on the current resource.

Audience

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

- Architects
- Developers
- Administrators.
- Operations personnel.

PRIVILEGE MODULE DEFINITION

Method Definitions and Signatures

1. **updatePrivileges**

Update resource privileges for the resources identified by the privilegeIds list and the target CIS server.

The following rules apply to updating privileges. References are made to the PrivilegeModuleXML which is detailed out later in this document.

Only a user with GRANT privilege on a resource can modify the privileges for that resource. The owner of a resource always has GRANT privilege, as do users with the MODIFY_ALL_RESOURCES right.

When "mode" is 'OVERWRITE_APPEND", or is not supplied, privileges are applied on a per-user or per-group basis, so that updating privileges for one user or group does not alter privileges from any other user or group. The privileges applied for a user or group replaces the previous value for that user or group. When "mode" is "SET_EXACTLY", all privileges on the resource are made to look exactly like the provided privileges.

When "recurse" is "false", the privileges are applied only the specified resources. When it is "true", the privileges are recursively applied into any CONTAINER or DATA_SOURCE resource specified. When recursively applying privileges, the privilege change is ignored for any resource the user lacks owner privileges for.

Privileges that are not applicable for a given resource type are automatically stripped down to the set that is legal for each resource. TABLE resources support NONE, READ, WRITE, SELECT, INSERT, UPDATE, and DELETE. PROCEDURE resources support NONE, READ, WRITE, and EXECUTE. All other resource types only support NONE, READ, and WRITE.

The "combinedPrivileges" and "inheritedPrivileges" elements on each "privilege" node will be ignored and can be left unset.

Request Elements:

recurse: If "true", then all children of the given resources will recursively be updated with the privileges assigned to their parent.

privileges: A list of resource names, types, and the privileges.

mode (optional): determines whether privileges are merged with existing ones, default is "OVERWRITE_APPEND", which merges and does not update privileges for users or groups not mentioned. "SET_EXACTLY" makes privileges look exactly like those provided in the call.

```
@param serverId target server id from servers config xml

@param privilegeIds list of comma separate privilege Ids

@param pathToPrivilegeXML path to the privilege xml

@param pathToServersXML path to the server values xml

@throws CompositeException


public void updatePrivileges(String serverId, String privilegeIds,
String pathToPrivilegeXML, String pathToServersXML) throws
CompositeException;
```

2. **generatePrivilegesXML**

   Generate the privileges for resources starting at the specified CIS path and for the specified target CIS server.

   The returned privileges per user or group are the privileges specifically given to that user or group. In each "privilege" node as defined by the XML detailed out later in this document, the "combinedPrivileges" element contains the effective privileges for that user or group based on their membership in all other groups.

   In each "privilege" node, the "inheritedPrivileges" element only contains the privileges that were inherited due to group membership. Logically OR'ing the "privileges" and "inheritedPrivileges" is the same as the "combinedPrivileges".

   A user with GRANT privilege or with READ_ALL_RESOURCES right will receive all privilege information for all users for that resource. Other users will only receive their own privilege information.

```
@param serverId target server id from servers config xml

@param startPath starting path of the resource e.g /shared

@param pathToPrivilegeXML path including name to the privilege xml
which needs to be created

@param pathToServersXML path to the server values xml

@param filter - a filter to return all or restrict the types of
resources that are returned.

The filter list is a space or comma separated list which may include
one or more of the following [ALL CONTAINER DATA_SOURCE DEFINITION_SET
LINK PROCEDURE TABLE TREE TRIGGER]

If the list contains ALL anywhere in the list then ALL resource types
are returned and the rest of the list is ignored.


@param options specify behavior.  Options are separated by spaces or
commas

       Note: admin privileges are never generated and never updated
```

```
  nameType
        USER - generate nameType=USER privileges for a given resource
       GROUP - generate nameType=GROUP privileges for a given resource
(Default behavior if nothing is specified)
       Note: Set both USER,GROUP to generate both


  System vs. NonSystem nameTypes
       SYSTEM -    generate SYSTEM nameTypes
                   group=all
                   users=anonymous, monitor
       NONSYSTEM - generate NONSYSTEM nameTypes (all users and groups
that are not SYSTEM) (Default behavior if nothing is specified)
       Note: Set both SYSTEM,NONSYSTEM to generate both groups


  Path hierarchy
       PARENT    - generate only the parent (starting path) according
to the filter (Default behavior if nothing is specified)
       CHILD     - generate privileges for all children of the
starting path according to the filter
       Note: Set both PARENT,CHILD if you want to generate the Parent
along with its children


@param domainList a space or comma separate list of domains for which
to generate privileges for (Default=composite)
@throws CompositeException


public void generatePrivilegesXML(String serverId, String startPath,
String pathToPrivilegeXML, String pathToServersXML, String filter,
String options, String domainList) throws CompositeException;
```

General Notes:

The arguments pathToPrivilegeXML and pathToServersXML will be located in [PDTool/resources/modules). The value passed into the methods will be the fully qualified path. The paths get resolved when executing the property file and evaluating the $MODULE_HOME variable.

PRIVILEGE MODULE XML CONFIGURATION

A full description of the PDToolModule XML Schema can be found by reviewing /docs/PDToolModules.xsd.html.

Description of the Module XML

The PrivilegeModule XML provides a structure “PrivilegeModule” for “updatePrivileges and generatePrivilegesXML”. The global entry point node is called “PrivilegeModule” and contains zero or more “resourcePrivileges” nodes.

```
<ns2:PrivilegeModule xmlns:ns2="http://www.dvbu.cisco.com/ps/deploytool/modules">

<!—Example setting privileges for type=USER:

    <resourcePrivilege>
        <id>priv1</id>
        <resourcePath>/shared/test00</resourcePath>
         <resourceType>CONTAINER</resourceType>
        <recurse>true</recurse>
        <mode>SET_EXACTLY</mode>

        <privilege>
            <name>user2</name>
            <nameType>USER</nameType>
            <domain>composite</domain>
            <privileges>READ WRITE EXECUTE</privileges>
       </privilege>

    </resourcePrivilege>

<!—Example of setting privileges for type=GROUP and two group entries:

    <resourcePrivilege>
        <id>priv2</id>
        <resourcePath>/services/webservices/testWebService</resourcePath>
         <resourceType>DATA_SOURCE</resourceType>
        <recurse>true</recurse>
       <updateDependenciesRecursively>false</updateDependenciesRecursively>
       <mode>SET_EXACTLY</mode>

        <privilege>
            <name>group1</name>
            <nameType>GROUP</nameType>
            <domain>composite</domain>
            <privileges>READ WRITE EXECUTE</privileges>
            <combinedPrivileges>NONE</combinedPrivileges>
            <inheritedPrivileges>NONE</inheritedPrivileges>
        </privilege>

        <privilege>
            <name>group2</name>
            <nameType>GROUP</nameType>
            <domain>composite</domain>
            <privileges>READ WRITE EXECUTE SELECT UPDATE INSERT DELETE GRANT</privileges>
            <combinedPrivileges>NONE</combinedPrivileges>
            <inheritedPrivileges>NONE</inheritedPrivileges>
        </privilege>
    </resourcePrivilege>

</ns2:PrivilegeModule>
```

Attributes of Interest

id **–** The unique identifier in the PrivilegeModule.xml file which identifies a configuration for a resource. It may in fact identify a resource for a path which gets applied recursively.

resourcePath **–**The CIS resource path to apply privileges to

resourceType **–**The CIS resource type to apply privileges to

recurse **–** [true|false] Indicates whether to apply this privilege profile recursively to all underlying child resources.

updateDependenciesRecursively **–** If "true", then all dependencies of the given resources will recursively be updated with the privileges assigned to their parent.

mode **–** Mode determines whether privileges are merged with existing ones. "OVERWRITE_APPEND" (default) merges and does not update privileges for users or groups not mentioned. "SET_EXACTLY" makes privileges look exactly like those provided in the call.

privilege **–** Grouping of one or more privileges containing the following elements:

- ***name*** **–** The name of the user or group for which to set privileges.
- ***nameType*** **–** The type of name. Type=USER or GROUP.
- ***domain*** **–** The composite domain in which the specified name is a member of.
- ***privileges*** **–** A space, separated, uppercase list of one or more of the following privileges: READ WRITE EXECUTE SELECT UPDATE INSERT DELETE GRANT

  combinedPrivileges **–** The "combinedPrivileges" element contains the effective privileges for that user or group based on their membership in all other groups.
- ***inheritedPrivileges*** **–** The privileges that were inherited due to group membership. Logically OR'ing the "privileges" and "inheritedPrivileges" is the same as the "combinedPrivileges".

Privilege Definition

Privileges indicate who can view, modify, or perform an action on a resource using the Composite suite of products. Privileges such as Read, Write, Execute, Select, Update, Insert, Delete, and Grant are set on folders, data sources, views, and procedures to secure proper access for specified groups and users. Privilege specification provides a comprehensive security layer to safeguard access to containers, folders, and objects defined within CIS.

Default Privileges on a new resource

By default the creator of a folder, container, or resource (the default "resource owner") gets all privileges associated with a new object definition. All other users (except for those with admin rights) get **no** privileges by default for new resources.

Restrictive default initial security settings for newly defined resources protect resources from inadvertent exposure by forcing explicit privilege assignment.

Default Read privileges are given for all example resources, system resources, and parent containers to members of the All Group. Dynamic users who belong to registered LDAP groups also gain limited Read and Execute privileges to access initially installed example resources and globally available system resources. By default dynamic users and anonymous users are disabled in the CIS installation and must be explicitly allowed by the implementation to enable access.

View the Privileges set on any resource within Studio

Right-select any named resource in the Composite Studio resource tree and select the “Privileges” option to view the privileges granted to groups and users of that resource.

Some of the access privileges apply to data modeling while others apply to data manipulation. Read, Write, and Grant privileges apply to the resources in the metadata repository whereas Execute, Select, Insert, and Update, and Delete privileges apply to the data in the source. The Read and Write privileges are design-time privileges that apply when using Studio and other Composite utilities. Modification of an existing resource definition requires user possession of the Write privilege on that resource, whereas simply viewing or incorporating a view as a building block for another view will require Read privileges on the resource. At runtime when using resources contained with resources the Read privilege must be present up the chain in all parent containers for the end-user to exercise runtime privileges on a resource.

The Read privilege on a resource grants the ability to see that the resource exists.

The Write privilege on a resource enables modification of the Composite resource definition that defines what and how the native resource may be used.

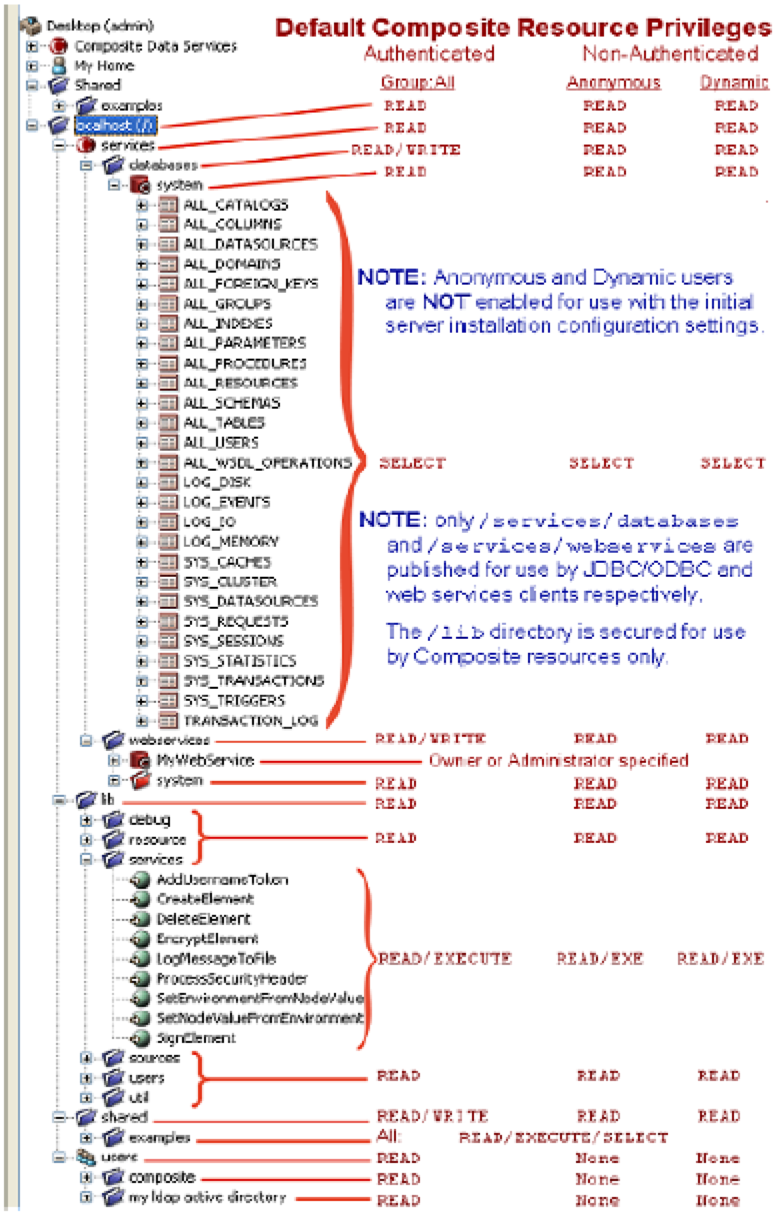
Run-time resource privileges must be set on the object resource to enable Select or Execute privileges view or use by end-users. Additionally all enclosing containers (parent or folder objects) must give the user Read privileges so that the resource may be accessible.

The Select privilege allows submission of SQL selects to retrieve data.

The Execute privilege enables execution of a procedure.

The Insert, Update, and Delete privileges enable change of table data.

At runtime, Read privileges are used for folders and their contents, but Read is not used at all on Tables or Procedures. The intention is that a user with just Select privileges on a particular table and without explicitly assigned Read privileges can select from that table. Here is a pictorial summary of default privileges assigned to system resources.



Default Composite Resource Privileges

Resource	Authenticated	Non-Authenticated	
	Group:All	Anonymous	Dynamic
Desktop (admin)			
Composite Data Services			
My Home			
Shared			
examples	READ	READ	READ
localhost (/)	READ	READ	READ
services	READ/WRITE	READ	READ
databases	READ	READ	READ
system			
ALL_CATALOGS, ALL_COLUMNS, ALL_DATASOURCES, ALL_DOMAINS, ALL_FOREIGN_KEYS, ALL_GROUPS, ALL_INDEXES, ALL_PARAMETERS, ALL_PROCEDURES, ALL_RESOURCES, ALL_SCHEMAS, ALL_TABLES, ALL_USERS, ALL_WSDL_OPERATIONS, LOG_DISK, LOG_EVENTS, LOG_IO, LOG_MEMORY, SYS_CACHES, SYS_CLUSTER, SYS_DATASOURCES, SYS_REQUESTS, SYS_SESSIONS, SYS_STATISTICS, SYS_TRANSACTIONS, SYS_TRIGGERS, TRANSACTION_LOG	SELECT	SELECT	SELECT
webservices	READ/WRITE	READ	READ
MyWebService	Owner or Administrator specified		
system	READ	READ	READ
lib	READ	READ	READ
debug, resource, services	READ	READ	READ
AddUsernameToken, CreateElement, DeleteElement, EncryptElement, LogMessageToFile, ProcessSecurityHeader, SetEnvironmentFromNodeValue, SetNodeValueFromEnvironment, SignElement	READ/EXECUTE	READ/EXE	READ/EXE
sources, users, util	READ	READ	READ
shared	READ/WRITE	READ	READ
examples	All: READ/EXECUTE/SELECT		
users	READ	None	None
composite	READ	None	None
my ldap active directory	READ	None	None

NOTE: Anonymous and Dynamic users are NOT enabled for use with the initial server installation configuration settings.

NOTE: only /services/databases and /services/webservices are published for use by JDBC/ODBC and web services clients respectively.

The /lib directory is secured for use by Composite resources only.

Anonymous users are disallowed from access by default CIS config setting.

LDAP/iPlanet users are non-authenticated dynamic users.

Getting the Read privilege only means that the user may see the resource exists given access to either a client that connects with CIS or if the user is assigned Composite Access Tools right then resources are visible with any Composite utility like Composite Studio.

Attribute Value Restrictions

resourceType (ResourceTypeSimpleType) – provides a list of resource types within CIS.

```
<xs:simpleType name="ResourceTypeSimpleType">
  <xs:annotation>
  <xs:documentation>
```

TYPES / SUBTYPES:
=================
The following resource types/subtypes are supported by this operation. Resources cannot be created under "/services" unless otherwise noted, and cannot be created within a physical data source.

(Datasource table columns)
* COLUMN / n/a - The column type is only used when updating privileges on a table column.

(Basic CIS folder)
* CONTAINER / FOLDER_CONTAINER - A Composite folder. Cannot be created anywhere under /services except in another FOLDER under /services/webservices.
* CONTAINER / DIRECTORY_CONTAINER - A Composite directory.
(Database)
* CONTAINER / CATALOG_CONTAINER - A Composite catalog folder under a data source. Can only be created within a data source under /services/databases.
* CONTAINER / SCHEMA_CONTAINER - A Composite schema container. Can only be created within a CATALOG that is under /services/databases.
(Web Services)
* CONTAINER / SERVICE_CONTAINER - A web service container for the service. Can only be created within a Composite Web Services data source that is under /services/webservices.
* CONTAINER / OPERATIONS_CONTAINER - A web service container for the operations
* CONTAINER / PORT_CONTAINER - A Composite web service container for port. Can only be created within a SERVICE under /services/webservices.
(Connectors)
* CONTAINER / CONNECTOR_CONTAINER - A Composite container for connectors.

* CONNECTOR / JMS - A Composite JMS Connector. Created with no connection information
* CONNECTOR / HTTP - A Composite HTTP Connector. Created with no connection information

* DATA_SOURCE / RELATIONAL_DATA_SOURCE - A relational database source.
* DATA_SOURCE / FILE_DATA_SOURCE - A comma separate file data source.
* DATA_SOURCE / XML_FILE_DATA_SOURCE - An XML file data source.
* DATA_SOURCE / WSDL_DATA_SOURCE - A Composite web service data source.
* DATA_SOURCE / XML_HTTP_DATA_SOURCE - An HTTP XML data source.
* DATA_SOURCE / NONE - A custom java procedure data source.

* DEFINITION_SET / SQL_DEFINITION_SET - A Composite SQL Definition set.
* DEFINITION_SET / XML_SCHEMA_DEFINITION_SET - A Composite XML Schema Defintion set.
* DEFINITION_SET / WSDL_DEFINITION_SET - A Composite WSDL Definition set.
* DEFINITION_SET / ABSTRACT_WSDL_DEFINITION_SET - A Composite Abstract WSDL Definition set such as the ones imported from Designer.
* DEFINITION_SET / SCDL_DEFINITION_SET - A Composite SCA composite Definition set imported from Designer.

* LINK / sub-type unknown - Used to link a Composite Data Service to a Composite resource such as a view or sql procedure.

(CIS procedures)
* PROCEDURE / SQL_SCRIPT_PROCEDURE - A Composite SQL Procedure. Created with a simple default script body that is runnable.
(Custom procedures)
* PROCEDURE / JAVA_PROCEDURE - A Composite java data source procedure. Created from a java data source (jar file).
(Database procedures)
* PROCEDURE / EXTERNAL_SQL_PROCEDURE - A Composite Packaged Query. Created with no SQL text, so it is not runnable.
* PROCEDURE / DATABASE_PROCEDURE - A database stored procedure.
(XML procedures)
* PROCEDURE / BASIC_TRANSFORM_PROCEDURE - A Composite Basic XSLT Transformation procedure. Created with no target procedure and no output columns, so it is not runnable.
* PROCEDURE / XSLT_TRANSFORM_PROCEDURE - A Composite XSLT Transformation procedure. Created with no target procedure and no output columns, so it is not runnable.
* PROCEDURE / STREAM_TRANSFORM_PROCEDURE - A Composite XSLT Streaming Transformation procedure. Created with no target procedure and no output columns, so it is not runnable.
* PROCEDURE / XQUERY_TRANSFORM_PROCEDURE - A Composite XQUERY Transformation Procedure. Created with no target schema and no model, so it is not runnable.
(Misc procedures)
* PROCEDURE / OPERATION_PROCEDURE - A Composite web service or HTTP procedure operation.

* TABLE / SQL_TABLE - A Composite View. Created with no SQL text or model, so it is not runnable.
* TABLE / DATABASE_TABLE - A Composite database table.
* TABLE / DELIMITED_FILE_TABLE - A Composite delimited file table
* TABLE / SYSTEM_TABLE - A Composite system table view.

```
  * TREE / XML_FILE_TREE - The XML tree structure associated with a file-XML data source.

  * TRIGGER / NONE - A Composite trigger. Created disabled.
  </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="COLUMN"/>
    <xs:enumeration value="CONTAINER"/>
    <xs:enumeration value="DATA_SOURCE"/>
    <xs:enumeration value="DEFINITION_SET"/>
    <xs:enumeration value="LINK"/>
    <xs:enumeration value="PROCEDURE"/>
    <xs:enumeration value="TABLE"/>
    <xs:enumeration value="TREE"/>
    <xs:enumeration value="TRIGGER"/>
  </xs:restriction>
 </xs:simpleType>
```

mode (PrivilegeModeValidationList) – provides a list of valid privilege modes within CIS.

```
  <xs:simpleType name="PrivilegeModeValidationList">
    <xs:annotation>
      <xs:documentation xml:lang="en">
The Mode validation list including 1 of [OVERWRITE_APPEND SET_EXACTLY]. Can be null. Only used when updating a resource privilege.
When "mode" is 'OVERWRITE_APPEND", or is not supplied, privileges are applied on a per-user or per-group basis, so that updating privileges for one user or group does not alter privileges from any other user or group. The privileges applied for a user or group replace the previous value for that user or group.
When "mode" is "SET_EXACTLY", all privileges on the resource are made to look exactly like the provided privileges.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="OVERWRITE_APPEND"/>
      <xs:enumeration value="SET_EXACTLY"/>
   </xs:restriction>
  </xs:simpleType>
```

nameType (PrivilegeNameTypeValidationList) – provides a list of valid name types.

```
  <xs:simpleType name="PrivilegeNameTypeValidationList">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        The Name Type validation list including 1 of [USER GROUP]
      </xs:documentation>
    </xs:annotation> <xs:restriction base="xs:string">
      <xs:enumeration value="USER"/>
      <xs:enumeration value="GROUP"/>
   </xs:restriction>
  </xs:simpleType>
```

privileges, combinedPrivileges, inheritedPrivileges (PrivilegeList) – provides a list of valid privileges.

```
  <xs:simpleType name="PrivilegeList">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        A space separated list of Privileges that may include 1 or more of [NONE READ WRITE EXECUTE SELECT
```

```
UPDATE INSERT DELETE GRANT]
            </xs:documentation>
        </xs:annotation>
      <xs:list itemType="ns:PrivilegeValidationList"/>
    </xs:simpleType>

    <xs:simpleType name="PrivilegeValidationList">
        <xs:restriction base="xs:string">
            <xs:enumeration value="NONE"/>
            <xs:enumeration value="READ"/>
            <xs:enumeration value="WRITE"/>
            <xs:enumeration value="EXECUTE"/>
            <xs:enumeration value="SELECT"/>
            <xs:enumeration value="UPDATE"/>
            <xs:enumeration value="INSERT"/>
            <xs:enumeration value="DELETE"/>
            <xs:enumeration value="GRANT"/>
      </xs:restriction>
    </xs:simpleType>
```

HOW TO EXECUTE

The following section describes how to setup a property file for both command line and Ant and execute the script. This script will use the PrivilegeModule.xml that was described in the previous section.

Script Execution

The full details on property file setup and script execution can be found in the document "*Composite PS Promotion and Deployment Tool User's Guide v1.0.pdf*". The abridged version is as follows:

```
Windows: ExecutePDTool.bat -exec ../resources/plans/UnitTest-Privilege.dp

Unix: ./ExecutePDTool.sh -exec ../resources/plans/UnitTest- Privilege.dp
```

Properties File (UnitTest-Privilege.dp):

Property File Rules:

```
# ----------------------------
# UnitTest-Privilege.dp
# ----------------------------
#   1. All parameters are space separated.  Commas are not used.
#         a. Any number of spaces may occur before or after any parameter and are
trimmed.
#
#   2. Parameters should always be enclosed in double quotes according to these
rules:
#         a. when the parameter value contains a comma separated list:
#                                    ANSWER: "ds1,ds2,ds3"
#
#         b. when the parameter value contain spaces or contains a dynamic variable
that will resolve to spaces
#            i.   There is no distinguishing between Windows and Unix variables.
Both UNIX style variables ($VAR) and
#                 and Windows style variables (%VAR%) are valid and will be parsed
accordingly.
#            ii.  All parameters that need to be grouped together that contain
spaces are enclosed in double quotes.
#            iii. All paths that contain or will resolve to a space must be enclosed
in double quotes.
#                 An environment variable (e.g. $MODULE_HOME) gets resolved on
invocation PDTool.
#                       Paths containing spaces must be enclosed in double quotes:
#                              ANSWER: "$MODULE_HOME/LabVCSModule.xml"
#                       Given that MODULE_HOME=C:/dev/Cis Deploy
Tool/resources/modules, PDTool automatically resolves the variable to
#                       "C:/dev/Cis Deploy Tool/resources/modules/LabVCSModule.xml".
#
#         c. when the parameter value is complex and the inner value contains spaces
#                     i. In this example $PROJECT_HOME will resolve to a path that contains spaces such as C:/dev/Cis Deploy Tool
#                        For example take the parameter -pkgfile $PROJECT_HOME$/bin/carfiles/testout.car.
#                        Since the entire command contains a space it must be enclosed in double quotes:
#                               ANSWER: "-pkgfile $PROJECT_HOME/bin/carfiles/testout.car"
#
#   3. A comment is designated by a # sign preceding any other text.
#         a. Comments may occur on any line and will not be processed.
#
#   4. Blank lines are not processed
#         a. Blank lines are counted as lines for display purposes
#         b. If the last line of the file is blank, it is not counted for display purposes.
#
```

Property File Parameters:

```
# ----------------------------
# Parameter Specification:
# ----------------------------
# Param1=[PASS or FAIL]  :: Expected Regression Behavior.  Informs the script whether you expect the action to pass or fail.  Can be used for regression testing.
# Param2=[TRUE or FALSE] :: Exit Orchestration script on error
# Param3=Module Batch/Shell Script name to execute (no extension).  Extension is added by script.
# Param4=Module Action to execute
# Param5-ParamN=Specific space separated parameters for the action.  See Property Rules below.
```

Property File Example:

```
# -----------------------------------------
# Begin task definition list:
# -----------------------------------------
# Generate Privilege XML
# Param5=serverId          [localhost]
# Param6=startingPath      [/shared/test00]
# Param7=Path-to-PrivilegeModuule.xml    [$MODULE_HOME/getPrivilegeModule.xml]
# Param8=Path-to-Servers.xml [$MODULE_HOME/servers.xml]
#
# Param9=filter -          ALL] - return privileges for all resource types in the path
# containing a space or comma separated list of one or more filter resource types to generate privileges for [ALL CONTAINER DATA_SOURCE DEFINITION_SET LINK PROCEDURE TABLE TREE TRIGGER]
#  If the list contains ALL anywhere in the list then ALL resource types are returned and the rest of the list is ignored.
#
# param10=options -        [GROUP USER NONSYSTEM PARENT CHILD]
#      space or comma separated list of one or more options to generate privileges
for [USER GROUP SYSTEM NONSYSTEM PARENT CHILD]
#
#      USER=return privileges for users
#      GROUP=return privileges for groups.  This is the default if neither USER or
GROUP is specified.
#
#      SYSTEM=return privileges for system users (anonymous,monitor) and groups (all)
#      NONSYSTEM=return privileges for all non-system users and groups.  This is the
default if neither SYSTEM or NONSYSTEM is specified.
#
#      PARENT=return privileges for the parent starting path
#      CHILD=return privileges for all children of the starting path.  This is the
default if neither PARENT or CHILD is specified.
#
# param11=domainList – space or comma separated list of domains to generate
privileges for [composite]
#
PASS   FALSE  ExecuteAction generatePrivilegesXML             $SERVERID
"/shared/test00"    "$MODULE_HOME/getPrivilegeModule.xml" "$MODULE_HOME/servers.xml"
"ALL" "GROUP,USER,NONSYSTEM,PARENT,CHILD" "composite"
#
# Update Privileges
#PASS  FALSE  ExecuteAction   updatePrivileges               $SERVERID "priv1,priv2"
              "$MODULE_HOME/PrivilegeModule.xml" "$MODULE_HOME/servers.xml"
```

Ant Execution

The full details on build file setup and ant execution can be found in the document "*Composite PS Promotion and Deployment Tool User's Guide v1.0.pdf*". The abridged version is as follows:

```
Windows: ExecutePDTool.bat -ant ../resources/ant/build-Privilege.xml

Unix: ./ExecutePDTool.sh -ant ../resources/ant/build-Privilege.xml
```

Build File:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="PDTool" default="default" basedir=".">

  <description>description</description>

  <!-- Default properties -->
  <property name="SERVERID"                 value="localhost"/>
  <property name="noarguments"              value="&quot;&quot;"/>

  <!-- Default Path properties -->
  <property name="RESOURCE_HOME"            value="${PROJECT_HOME}/resources"/>
  <property name="MODULE_HOME"              value="${RESOURCE_HOME}/modules"/>
  <property name="pathToServersXML"         value="${MODULE_HOME}/servers.xml"/>
  <property name="pathToArchiveXML"         value="${MODULE_HOME}/ArchiveModule.xml"/>
  <property name="pathToDataSourcesXML"      value="${MODULE_HOME}/DataSourceModule.xml"/>
  <property name="pathToGroupsXML"           value="${MODULE_HOME}/GroupModule.xml"/>
  <property name="pathToPrivilegeXML"        value="${MODULE_HOME}/PrivilegeModule.xml"/>
  <property name="pathToRebindXML"           value="${MODULE_HOME}/RebindModule.xml"/>
  <property name="pathToRegressionXML"       value="${MODULE_HOME}/RegressionModule.xml"/>
  <property name="pathToResourceXML"         value="${MODULE_HOME}/ResourceModule.xml"/>
  <property name="pathToResourceCacheXML"    value="${MODULE_HOME}/ResourceCacheModule.xml"/>
  <property name="pathToServerAttributeXML"  value="${MODULE_HOME}/ServerAttributeModule.xml"/>
  <property name="pathToTriggerXML"          value="${MODULE_HOME}/TriggerModule.xml"/>
  <property name="pathToUsersXML"            value="${MODULE_HOME}/UserModule.xml"/>
  <property name="pathToVCSModuleXML"        value="${MODULE_HOME}/VCSModule.xml"/>

  <!-- Custom properties -->
  <property name="privilegeIds"              value="priv1,prive2"/>
  <property name="pathToGenPrivilegeXML"     value="${MODULE_HOME}/getPrivilegeModule.xml"/>

  <!-- Default Classpath [Do Not Change] -->
  <path id="project.class.path">
       <fileset dir="${PROJECT_HOME}/lib"><include name="**/*.jar"/></fileset>
       <fileset dir="${PROJECT_HOME}/dist"><include name="**/*.jar"/></fileset>
       <fileset dir="${PROJECT_HOME}/ext/ant/lib"><include name="**/*.jar"/></fileset>
  </path>

  <taskdef name="executeJavaAction" description="Execute Java Action"
classname="com.cisco.dvbu.ps.deploytool.ant.CompositeAntTask"
classpathref="project.class.path"/>

       <!-- =================================
          target: default
         ================================= -->
       <target name="default" description="Update CIS with environment specific parameters">

     <!-- Execute Line Here -->
        <executeJavaAction description="Generate"    action="generatePrivilegesXML"
       arguments="${SERVERID}^/shared/test00^${pathToGenPrivilegeXML}^${pathToServersXML}^
CONTAINER^GROUP,NONSYSTEM,PARENT^composite"          endExecutionOnTaskFailure="TRUE"/>

       <!-- Windows or UNIX
        <executeJavaAction description="Generate"    action="generatePrivilegesXML"
       arguments="${SERVERID}^/shared/test00^${pathToGenPrivilegeXML}^${pathToServersXML}^
CONTAINER^GROUP,NONSYSTEM,PARENT^composite"          endExecutionOnTaskFailure="TRUE"/>

        <executeJavaAction description="Update"      action="updatePrivileges"
       arguments="${SERVERID}^${privilegeIds}^${pathToPrivilegeXML}^${pathToServersXML}"
       endExecutionOnTaskFailure="TRUE"/>
    -->
       </target>
</project>
```

Module ID Usage

The following explanation provides a general pattern for module identifiers. The module identifier for this module is “privilegeIds”.

- Possible values for the module identifier:

- 1. ***Inclusion List*** - CSV string like "id1,id2"
    - PDTool will process only the passed in identifiers in the specified module XML file.

Example command-line property file

```
PASS   FALSE  ExecuteAction       updatePrivileges $SERVERID "priv1,priv2"
              "$MODULE_HOME/PrivilegeModule.xml" "$MODULE_HOME/servers.xml"
```

Example Ant build file

```
<executeJavaAction description="Update"        action=" updatePrivileges "
arguments="${SERVERID}^priv1,priv2^${pathToPrivilegeXML}^${pathToServersXML}"
```

- 2. ***Process All*** - '*' or whatever is configured to indicate all resources
    - PDTool will process all resources in the specified module XML file.

Example command-line property file

```
PASS   FALSE  ExecuteAction       updatePrivileges $SERVERID "*"
       "$MODULE_HOME/PrivilegeModule.xml" "$MODULE_HOME/servers.xml"
```

Example Ant build file

```
<executeJavaAction description="Update"        action="updatePrivileges"
arguments="${SERVERID}^*^${pathToPrivilegeXML}^${pathToServersXML}"
```

- 3. ***Exclusion List*** - CSV string with '-' or whatever is configured to indicate exclude resources as prefix like "-id1,id2"
    - PDTool will ignore passed in resources and process the rest of the identifiers in the module XML file.

Example command-line property file

```
PASS   FALSE  ExecuteAction       updatePrivileges $SERVERID "-priv3,priv4"
              "$MODULE_HOME/PrivilegeModule.xml" "$MODULE_HOME/servers.xml"
```

Example Ant build file

```
<executeJavaAction description="Update"        action="updatePrivileges"
arguments="${SERVERID}^-priv3,priv4^${pathToPrivilegeXML}^${pathToServersXML}"
```

EXAMPLES

The following are common scenarios when using the PrivilegeModule.

Scenario 1 – Generate Privileges

Description:

Generate the resource privileges for resources starting at path /shared/test00. Even though this will generate a rather large list of resources, the best practice is to set resources at a project level. In this case the project level is test00. Therefore upon update, we will only need to configure one node in the PrivilegeModule.xml.

Options: GROUP,NONSYSTEM,PARENT – Generate only group privileges for non-system groups and the parent starting path resource.

domainList: composite – only generate privileges for the “composite” domain

Execution Sample:

Unix: ./ExecutePDTool.sh -exec ../resources/plans/UnitTest-Privilege.dp

Property file setup for UnitTest-Privilege.dp:

```
# -----------------------------------------
# Begin task definition list:
# -----------------------------------------
# Generate XML
PASS   FALSE  ExecuteAction    generatePrivilegesXML   $SERVERID
"/shared/test00"           "$MODULE_HOME/getPrivilegeModule.xml"
"$MODULE_HOME/servers.xml" "ALL" "GROUP NONSYSTEM PARENT" "composite"
```

Results Expected:

The execution will generate the XML into the PDTool/resources/modules directory. An example of the XML out can be views in the previous section title “***Description of the Module XML***”.

Scenario 2 – Update Privileges

Description:

Update privileges recursively starting at the parent folder “/shared/test00”. The mode is set to “SET_EXACTLY” so that all privileges of underlying children are set exactly like the parent. The privileges will be set on the GROUP “group1” which will have the following privileges: READ WRITE EXECUTE SELECT.   No insert, update or delete privileges are granted to this group on the underlying data sources.

XML Configuration Sample:

```
<ns2:PrivilegeModule xmlns:ns2="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <resourcePrivilege>
        <id>priv1</id>
        <resourcePath>/shared/test00</resourcePath>
        <recurse>true</recurse>
        <mode>SET_EXACTLY</mode>

        <privilege>
            <name>group1</name>
            <nameType>GROUP</nameType>
            <domain>composite</domain>
            <privileges>READ WRITE EXECUTE SELECT</privileges>
       </privilege>
    </resourcePrivilege>
</ns2:PrivilegeModule>
```

Execution Sample:

Unix: ./ExecutePDTool.sh -exec ../resources/plans/UnitTest-Privilege.dp

Property file setup for UnitTest-Privilege.dp:

```
# -----------------------------------------
# Begin task definition list:
# -----------------------------------------
# Update Privileges
PASS   FALSE  ExecuteAction   updatePrivileges        $SERVERID "priv1"
       "$MODULE_HOME/PrivilegeModule.xml" "$MODULE_HOME/servers.xml"
```

Results Expected:

The execution will update the folder “/shared/test00” and all of its child resources and set the privileges to exactly shown in this privilege profile.

EXCEPTIONS AND MESSAGES

The following are common exceptions and messages that may occur.

Wrong Number of Arguments:

This may occur when you do not place double quotes around comma separated lists.

CONCLUSION

Concluding Remarks

The PS Promotion and Deployment Tool is a set of pre-built modules intended to provide a turn-key experience for promoting CIS resources from one CIS instance to another. The user only requires system administration skills to operate and support. The code is transparent to operations engineers resulting in better supportability. It is easy for users to swap in different implementations of a module using the Spring framework and configuration files.

How you can help!

Build a module and donate the code back to Composite Professional Services for the advancement of the "***PS Promotion and Deployment Tool***".