WHITE PAPER

Composite Data Virtualization

Composite PS Promotion and Deployment Tool

Rebind Module User Guide

Composite Professional Services

November 2014

TABLE OF CONTENTS

DOCUMENT CONTROL

Version History

Version	Date	Author	Description
1.0	8/15/2011	Jerry Joplin	Initial revision for Rebind Module User Guide
2.0	3/1/2013	Mike Tinius	Modified rebindResources and rebindFolder to be more flexible when rebinding a resource when the source resource does not exist.
3.0	8/21/2013	Mike Tinius	Updated docs with Cisco format.
3.1	2/18/2014	Mike Tinius	Prepare docs for open source.
3.2	3/24/2014	Mike Tinius	Changed references of XML namespace to www.dvbu.cisco.com
3.3	11/17/2014	Mike Tinius	Update license.

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

License

(c) 2014 Cisco and/or its affiliates. All rights reserved.

This software is released under the Eclipse Public License. The details can be found in the file LICENSE. Any dependent libraries supplied by third parties are provided under their own open source licenses as described in their own LICENSE files, generally named .LICENSE.txt. The libraries supplied by Cisco as part of the Composite Information Server/Cisco Data Virtualization Server, particularly csadmin-XXXX.jar, csarchive-XXXX.jar, csbase-XXXX.jar, csclient-XXXX.jar, cscommon-XXXX.jar, csext-XXXX.jar, csjdbc-XXXX.jar, csserverutil-XXXX.jar, csserver-XXXX.jar, cswebapi-XXXX.jar, and customproc-XXXX.jar (where -XXXX is an optional version number) are provided as a convenience, but are covered under the licensing for the Composite Information Server/Cisco Data Virtualization Server. They cannot be used in any way except through a valid license for that product.

This software is released AS-IS!. Support for this software is not covered by standard maintenance agreements with Cisco. Any support for this software by Cisco would be covered by paid consulting agreements, and would be billable work.

Purpose

The purpose of the Rebind Module User Guide is to demonstrate how to effectively use the Rebind Module and execute actions. As part of the Promotion and Deployment process, it is often necessary to rebind resources to different targets after importing into the target CIS server. For example, a popular scenario is to rebind resources that were pointing to a development data source in the DEV server and rebind to the test data source in the TEST server. This scenario occurs when the name of the data source is different in each environment. When using the Composite Software Best Practices for Folder structuring, it is possible to rebind all of the Views or Procedures in a folder from the old resource path to a new resource path using the “rebind folder” capability.

Audience

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

- Architects
- Developers
- Administrators.
- Operations personnel.

REBIND MODULE DEFINITION

Method Definitions and Signatures

1. **rebindResources**

   Change the binding of resources on the server by providing an explicit “from” resource path and a “to” resource path. It is not necessary for the source “from” resource to exist. However an exception will be thrown if the target “to” resource does not exist.

   ```
   @param serverId target server config name
   @param rebindIds list of comma separate rebind Ids
   @param pathToRebindXml the path to the rebinds XML
   @param pathToServersXML path to the server values xml
   @throws CompositeException

   public void rebindResources(String serverId, String rebindIds, String
   pathToRebindXml, String pathToServersXML) throws CompositeException;
   ```

2. **rebindFolder**

   Rebind Folder performs a rebind of all resources within a folder tree to a designated target. If both the source “from’ resource and target “to” resource exist then rebindFolder performs the explicit “rebindResource” operation. If the source “from” resource does not exist then it performs a textual rebind which does not require the “from” resources to exist. The target resources must of course exist. This type of rebind is very handy when all resources such as views or procedures are pointing to the same parent folder location like those developed using the Composite Best Practices layering approach.

   Note: Currently, this method only works on “Views” and “Procedures”.

   ```
   @param serverId target server config name
   @param rebindIds list of comma separated rebind Ids
   @param pathToRebindXml the path to the rebinds XML
   @param pathToServersXML path to the server values xml
   @throws CompositeException

   public void rebindFolder(String serverId, String rebindIds, String
   pathToRebindXml, String pathToServersXML) throws CompositeException;
   ```

3. **generateRebindXML**

   Generate an XML file with all the “rebinadable” resources starting at the “startPath” and traversing through the folder tree. It will generate the current rebind structure. This allows a user to go in and edit the XML and doing search and replace on old paths and replace with new paths. Once completed, this file becomes input to the rebindResources() method.

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

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

@param pathToRebindXml path including name to the data source xml
which needs to be created

@param pathToServersXML path to the server values xml

@throws CompositeException


public void generateRebindXML(String serverId, String startPath,
String pathToRebindXML, String pathToServersXML) throws
CompositeException;
```

General Notes:

The arguments pathToRebindXML 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.

Method Detailed Description

Detailed Description of the usage of “rebindFolder” and “rebindResources”:

For both “rebindResources” and “rebindFolder”, the methods will attempt to use the explicit Composite API to “rebindResource” when both the source and target resources exist. When the target resource does not exist an exception is thrown. When only the source resource exists, then the following rules are applied and both “rebindResources” and “rebindFolder” will use the method to replace the path text within the table or procedure script. This only gets used when the "from" path does not exist in the target CIS server. This can happen when a car file is imported from one environment like DEV where the physical layer resources point to one data source and the target CIS server like TEST contains a different data source name. In that use case, it is not permitted to use the regular rebind as it will throw an error, therefore, the only recourse is to rebind the actual script or procedure text and use the individual methods for each sub-type to perform the update.

Textual Path Replacement Rules:

Caveat:

For textual path replacement, when the "from" resource does not exist and the SQL_TABLE (Views) and SQL_SCRIPT_PROCEDURE (Procedures) have models, the model is lost. The reason is there is no way to programmatically create a model in the API.   For parameterized queries this is unfortunate as there is no way to rebuild the model once removed. For Views, the model can be regenerated in most cases.

resourceType = 'TABLE'

subtype = 'SQL_TABLE' -- Get Regular View

procedureTextCurr = tableResource.getSqlText();

// Replace all of the matching old "fromFolder" paths with the new "toFolder" path

procedureText = procedureTextCurr.replaceAll(fromFolder, toFolder);

port.updateSqlTable(resourcePath, detailLevel, procedureText, model, isExplicitDesign, columns, annotation, attributes);

resourceType = 'PROCEDURE'

subtype = 'SQL_SCRIPT_PROCEDURE' -- Update Regular Procedure

procedureTextCurr = procedureResource.getScriptText();

// Replace all of the matching old "fromFolder" paths with the new "toFolder" path

procedureText = procedureTextCurr.replaceAll(fromFolder, toFolder);

port.updateSqlScriptProcedure(resourcePath, detailLevel, procedureText, model, isExplicitDesign, parameters, annotation, attributes);

subtype = 'EXTERNAL_SQL_PROCEDURE' -- Update Packaged Query Procedure

usedResourcePathCurr = procedureResource.getExternalDataSourcePath();

// Replace all of the matching old "fromFolder" paths with the new "toFolder" path

usedResourcePath = usedResourcePathCurr.replaceAll(fromFolder, toFolder);

port.updateExternalSqlProcedure(resourcePath, detailLevel, procedureText, usedResourcePath, parameters, annotation, attributes);

subtype = 'BASIC_TRANSFORM_PROCEDURE' -- Update XSLT Basic Transformation definition

usedResourcePathCurr = procedureResource.getTransformSourcePath();

// Replace all of the matching old "fromFolder" paths with the new "toFolder" path

usedResourcePath = usedResourcePathCurr.replaceAll(fromFolder, toFolder);

port.updateBasicTransformProcedure(resourcePath, detailLevel, usedResourcePath, resourceType, annotation, attributes);

subtype = 'XSLT_TRANSFORM_PROCEDURE' -- Update XSLT Transformation text

usedResourcePathCurr = procedureResource.getTransformSourcePath();

// Replace all of the matching old "fromFolder" paths with the new "toFolder" path

usedResourcePath = usedResourcePathCurr.replaceAll(fromFolder, toFolder);

port.updateXsltTransformProcedure(resourcePath, detailLevel, usedResourcePath, resourceType, procedureText, model, annotation, isExplicitDesign, parameters, attributes);

subtype = 'STREAM_TRANSFORM_PROCEDURE' -- Update XSLT Stream Transformation text

```
usedResourcePathCurr = procedureResource.getTransformSourcePath();
// Replace all of the matching old "fromFolder" paths with the new "toFolder" path
usedResourcePath = usedResourcePathCurr.replaceAll(fromFolder, toFolder);
port.updateStreamTransformProcedure(resourcePath, detailLevel, usedResourcePath,
resourceType, model, isExplicitDesign, parameters, annotation, attributes);
```

REBIND 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 RebindModule XML provides a structure “RebindModule” for rebindResources, rebindFolder and generateRebindXML. The global entry point node is called “RebindModule” and contains one or more “rebind” nodes.

```
<?xml version="1.0"?>
<p1:RebindModule xmlns:p1="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <rebind>
             <!-- Rebind an individual resource of any type and apply one or more
rebind rules provided.  Both the from resource and to resource must exist. -->
        <rebindResource>
            <id>rebindResource-1</id>

<resourcePath>/shared/test/PhysicalProcedures/ds_orders/getCustomerById</resourcePat
h>
            <resourceType>PROCEDURE</resourceType>
            <rebindRules>
                <oldPath>/shared/test/PhysicalMetadata/ds_orders/customers</oldPath>
                <oldType>TABLE</oldType>
                <newPath>/shared/examples/ds_orders/customers</newPath>
                <newType>TABLE</newType>
            </rebindRules>
        </rebindResource>
    </rebind>
    <rebind>
        <rebindResource>
            <id>rebindResource-2</id>
            <resourcePath>/shared/test/PhysicalViews/customers</resourcePath>
            <resourceType>TABLE</resourceType>
            <rebindRules>
                <oldPath>/shared/test/PhysicalMetadata/ds_orders/customers</oldPath>
                <oldType>TABLE</oldType>
                <newPath>/shared/examples/ds_orders/customers</newPath>
                <newType>TABLE</newType>
            </rebindRules>
        </rebindResource>
    </rebind>
```

```
       <rebind>
             <!-- Rebind all views/procedures in a starting folder that match the
rebindFromFolder and rebind to the rebindToFolder -->
        <rebindFolder>
            <id>rebindFolder-1</id>
            <startingFolderPath>/shared/test/PhysicalProcedures</startingFolderPath>

<rebindFromFolder>/shared/test/PhysicalMetadata/ds_inventory</rebindFromFolder>
            <rebindToFolder>/shared/examples/ds_inventory</rebindToFolder>
        </rebindFolder>
    </rebind>
    <rebind>
       <rebindFolder>
            <id>rebindFolder-2</id>
            <startingFolderPath>/shared/test/PhysicalViews</startingFolderPath>

<rebindFromFolder>/shared/test/PhysicalMetadata/ds_inventory</rebindFromFolder>
            <rebindToFolder>/shared/examples/ds_inventory</rebindToFolder>
        </rebindFolder>
    </rebind>
</p1:RebindModule>
```

Attributes of Interest

Note: **rebindResource** and **rebindFolder** are choice of children which means that for any single iteration of "**rebind**" you can have one choice at that time.

rebindResource – Rebind Resource Type: The Rebind Resource Type provides a structure to define which resource to perform a rebind on. This resource may contain one or more resources that it uses (invokes). For example a SQL procedure may invoke another procedure and two views. If the underlying resource changes folders, then a rebind would need to be performed. The rebind rules type is used to provide iteration resources to rebind to. The rules identify the old resource path/type and the new resource path/type

> ***id*** – id is a unique name identifier within the RebindModule.xml list of resources to rebind.
>
> ***resourcePath*** – the CIS path of the resource to rebind.
>
> ***resourceType*** – the CIS type of resources   See ResourceTypeSimpleType in the Attribute Value Restrictions below for valid types.
>
> ***rebindRules*** – Rebind Rule Type: The Rebind Rule Type provides a way of identifying the old (source) path and type and the new (target) path and type.

oldPath – old resource path.

oldType – old resource type.

newPath – new (target) path.

newType – the new (target) type.

rebindFolder – Rebind Folder Type: The Rebind Folder Type provides an easy to configure interface where the user wants to rebind all procedures or views within the starting folder. Only views and procedures are supported. The rebindFromFolder is what the resource is currently pointing to. The rebindToFolder is where you want to rebind the resource to. Only the target resource need be present in the CIS for this operation to be successful.

startingFolder – the starting CIS folder path in which to locate resources to rebind. It is assumed that all resources in that folder are targeted to the same resource path.

rebindFromFolder – as a safeguard, the "rebindFolder()" method will only search and replace resources in the starting folder that contain this "rebindFromFolder" resource path..

rebindToFolder – if the "rebindFromFolder" path is found in a resource, it is replaced by this "rebindToFolder" resource path.

Attribute Value Restrictions

ResourceTypeSimpleType – a restriction list on the type of resources in 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>
```

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 RebindModule.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-Rebind.dp

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

Properties File (UnitTest-Rebind.dp):

Property File Rules:

```
# ----------------------------
# UnitTest-Rebind.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:
# -----------------------------------------
PASS   FALSE  ExecuteAction   rebindResources    $SERVERID "*"
"$MODULE_HOME/RebindModule.xml" "$MODULE_HOME/servers.xml"
#
PASS   FALSE  ExecuteAction   generateRebindXML   $SERVERID
"/shared/test/DataAbstractionSample"    "$MODULE_HOME/RebindModule_Gen.xml"
"$MODULE_HOME/servers.xml"
#
#PASS  FALSE  ExecuteAction   rebindFolder     SERVERID "*"
"$MODULE_HOME/RebindModule.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-Rebind.xml

Unix: ./ExecutePDTool.sh -ant ../resources/ant/build-Rebind.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="rebindIds"                   value="rebindResource-1"/>
  <property name="pathToGenRebindXML"          value="${MODULE_HOME}/getRebindModule.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">
  <executeJavaAction description="Generate"  action="generateRebindXML"
        arguments="${SERVERID}^/shared/test00^${pathToGenRebindXML}^${pathToServersXML}"
        endExecutionOnTaskFailure="TRUE"/>

  <!-- Windows or UNIX
  <executeJavaAction description="Generate"  action="generateRebindXML"
        arguments="${SERVERID}^/shared/test00^${pathToGenRebindXML}^${pathToServersXML}"
        endExecutionOnTaskFailure="TRUE"/>

  <executeJavaAction description="Rebind" action="rebindResources"
        arguments="${SERVERID}^${rebindIds}^${pathToRebindXML}^${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 “rebindIds”.

- 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   rebindResources    $SERVERID "*"
"$MODULE_HOME/RebindModule.xml" "$MODULE_HOME/servers.xml"
```

```
PASS   FALSE  ExecuteAction       rebindResources    $SERVERID
"rebind1,rebind2"          "$MODULE_HOME/RebindModule.xml"
"$MODULE_HOME/servers.xml"
```

Example Ant build file

```
<executeJavaAction description="Rebind"        action="rebindResources"
arguments="${SERVERID}^rebind1,rebind2^${pathToRebindXML}^${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       rebindResources $SERVERID "*"
       "$MODULE_HOME/RebindModule.xml" "$MODULE_HOME/servers.xml"
```

Example Ant build file

```
<executeJavaAction description="Rebind"        action="rebindResources"
arguments="${SERVERID}^*^${pathToRebindXML}^${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       rebindResources $SERVERID "-rebind3,rebind4"
              "$MODULE_HOME/RebindModule.xml" "$MODULE_HOME/servers.xml"
```

Example Ant build file

```
<executeJavaAction description="Rebind"       action="rebindResources"
arguments="${SERVERID}^-
rebind3,rebind4^${pathToRebindXML}^${pathToServersXML}"
```

EXAMPLES

The following are common scenarios when using the RebindModule.

Scenario 1 – Generate Rebind XML

Description:

Generate the "rebindResource" choice XML for all of the "rebindable" resources starting at the given path "/shared/test00/Views".

XML Configuration Sample:

Not applicable for this example.

Execution Sample:

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

Property file setup for UnitTest-Rebind.dp:

```
# -----------------------------------------
# Begin task definition list:
# -----------------------------------------
# Generate Rebind XML
PASS  FALSE  ExecuteAction  generateRebindXML  $SERVERID
"/shared/test00/Views" "$MODULE_HOME/getRebindModule.xml"
"$MODULE_HOME/servers.xml"
```

Results Expected:

The XML is generated with four "rebind" nodes representing the four views found in the /shared/test00/Views folder.

```
<ns2:RebindModule xmlns:ns2="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <rebind>
        <rebindResource>
            <id>rebindResource-1</id>
            <resourcePath>/shared/test00/Views/Customers</resourcePath>
            <resourceType>TABLE</resourceType>
            <rebindRules>
                <oldPath>/shared/examples/ds_orders/customers</oldPath>
                <oldType>TABLE</oldType>

<newPath>/shared/examples/ds_orders/customers_CHANGEME</newPath>
                <newType>TABLE</newType>
            </rebindRules>
        </rebindResource>
    </rebind>
...
</ns2:RebindModule>
```

Scenario 2 – Rebind a Resource (old and new must exist)

Description:

In this example, we will use the XM generated in scenario 1. The XML is modified slightly to provide a new target path pointing to a different data source.

XML Configuration Sample:

RebindModule.xml configuration file generated in scenario 1.

```
XML Generated (BEFORE CHANGES):
    <rebind>
        <rebindResource>
            <id>rebindResource-1</id>
            <resourcePath>/shared/test00/Views/Customers</resourcePath>
            <resourceType>TABLE</resourceType>
            <rebindRules>
                <oldPath>/shared/examples/ds_orders/customers</oldPath>
                <oldType>TABLE</oldType>

<newPath>/shared/examples/ds_orders/customers_CHANGEME</newPath>
                <newType>TABLE</newType>
            </rebindRules>
        </rebindResource>
    </rebind>

XML (AFTER CHANGES):
    <rebind>
        <rebindResource>
            <id>rebindResource-1</id>
            <resourcePath>/shared/test00/Views/Customers</resourcePath>
            <resourceType>TABLE</resourceType>
            <rebindRules>

<oldPath>/shared/test00/DataSources/ds_orders/customers</oldPath>
                <oldType>TABLE</oldType>
                <newPath>/shared/examples/ds_orders/customers</newPath>
                <newType>TABLE</newType>
            </rebindRules>
        </rebindResource>
    </rebind>
```

Execution Sample:

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

Property file setup for UnitTest-Rebind.dp:

```
# ----------------------------------------
# Begin task definition list:
# -----------------------------------------
# Rebind Resource
PASS  FALSE  ExecuteAction  rebindResources  $SERVERID "rebindResource-1"
"$MODULE_HOME/getRebindModule.xml"  "$MODULE_HOME/servers.xml"
```

Results Expected:

The View pointed to “/shared/test00/DataSources/ds_orders/customers” prior to the rebind. After the rebind the Customer view not points to “/shared/examples/ds_orders/customers”. This is demonstrated in the text box below:

```
VIEW TEXT BEFORE REBIND
SELECT
    *
FROM
    /shared/test00/DataSources/ds_orders/customers customers

VIEW TEXT AFTER REBIND
SELECT
    *
FROM
    /shared/examples/ds_orders/customers customers
```

Scenario 3 – Rebind an entire folder of Views

Description:

When using the Composite Software Best Practices for folder structure, the rebindFolder() method makes it really easy to simply point at the starting folder and have the PD Tool rebindFolder introspect all the resources in the folder and change the “rebind from path” to the “rebind to path”.

XML Configuration Sample:

RebindModule.xml configuration example.

```
<p1:RebindModule xmlns:p1="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <rebind>
       <rebindFolder>
            <id>rebindFolder-2</id>
            <startingFolderPath>/shared/test00/Views</startingFolderPath>

<rebindFromFolder>/shared/test00/DataSources/ds_orders</rebindFromFolder>
            <rebindToFolder>/shared/examples/ds_orders</rebindToFolder>
        </rebindFolder>
    </rebind>
...
</ns2:RebindModule>
```

Execution Sample:

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

Property file setup for UnitTest-Rebind.dp:

```
# -----------------------------------------
# Begin task definition list:
# -----------------------------------------
# Rebind Folder
PASS   FALSE  ExecuteAction rebindFolder $SERVERID "rebindFolder-1"
       "$MODULE_HOME/RebindModule.xml"   "$MODULE_HOME/servers.xml"
```

Results Expected:

All of the view in "/shared/test00/Views" are now rebound to "/shared/examples/ds_orders"

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***".