PDTool Resource Module User Guide

An Open Source Asset for use with TIBCO® Data Virtualization

TIBCO Software empowers executives, developers, and business users with Fast Data solutions that make the right data available in real time for faster answers, better decisions, and smarter action. Over the past 15 years, thousands of businesses across the globe have relied on TIBCO technology to integrate their applications and ecosystems, analyze their data, and create real-time solutions. Learn how TIBCO turns data—big or small—into differentiation at www.tibco.com.

Project Name	AS Assets PDTool (Promotion and Deployment Tool)
Document Location	This document is only valid on the day it was printed. The source of the document will be found in the PDTool and PDToolRelease folder (https://github.com/TIBCOSoftware)
Purpose	User's Guide

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
1.0	6/13/2011	Mike Tinius	Initial revision for Resource Module User Guide
1.1	8/1/2011	Mike Tinius	Revision due to Architecture changes
1.2	5/7/2013	Mike Tinius	Added createFolder and createFolders methods
3.0	8/21/2013	Mike Tinius	Updated docs to 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	11/18/2014	Mike Tinius	Added support for printing out scalar or cursor output values for executeProcedure and executeConfiguredProcedure.
3.4	11/17/2014	Mike Tinius	Update license.
3.5	3/4/2015	Mike Tinius	Updated createFolder with ignoreErrors param and updated docs to Cisco format.
4.0	12/14/2017	Mike Tinius	Initial revision with Tibco
5.0	08/27/2020	Mike Tinius	Updated documentation
5.1	10/20/2020	Mike Tinius	Updated documentation

Related Documents

Name	Author
PDTool User's Guide.pdf	Mike Tinius

Supported Versions

Name	Version
TIBCO® Data Virtualization	7.0.8 or later

Table of Contents

1 Introduction

Purpose

The purpose of the Resource Module User Guide is to demonstrate how to effectively use the Resource Module and execute actions. The ResourceModule provides general purpose methods for managing resources such as copy, rename, move, delete, exists, lock, unlock and a specialized method for executing procedures with Data Virtualization (DV).

Audience

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

- Architects
- Developers
- Administrators
- Operations personnel

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)

2 Resource Module Definition

Method Definitions and Signatures

1. executeConfiguredProcedures

Execute published procedures associated with passed in procedure ids that reference arguments found in the resource XML.

This method only supports all scalar values or one cursor output.
This method does not support procedure calls with no output.
This method does not support procedure calls with mixed scalar and cursor output.
This method has been enhanced to print out the result scalar values or cursor output.

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

@param procedureIds procedure ids in the resourceXML

@param pathToResourceXML path to the resource xml

@param pathToServersXML path to the server values xml

@throws CompositeException if the execution of the procedure fails


public void executeConfiguredProcedures(String serverId, String
proceduresId, String pathToResourceXML, String pathToServersXML) throws
CompositeException;
```

2. executeProcedure

Execute a published procedure associated with passed in procedure name along with passed in arguments and arguments should be passed in the format 'arg1','arg2','arg3'.....

This method only supports all scalar values or one cursor output.
This method does not support procedure calls with no output.
This method does not support procedure calls with mixed scalar and cursor output.
This method has been enhanced to print out the result scalar values or cursor output.

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

@param procedureName published procedure name

@param dataServiceName data service name (equivalent to schema name)

@param pathToServersXML path to the server values xml

@param arguments string with arguments in the format
'arg1','arg2','arg3'....

@param outputReturnVariables [optional] true=(default) output the values
of the return variables from the procedure call, false=do not output the
return variable values.

@throws CompositeException if the execution of the procedure fails
```

```
Signature:
Option 1: (original method signature)
public void executeProcedure(String serverId, String procedureName,
String dataServiceName, String pathToServersXML,String arguments) throws
CompositeException;
option 2: (method signature that allows viewing method output)
public void executeProcedure(String serverId, String procedureName,
String dataServiceName, String pathToServersXML,String arguments,
outputReturnVariables) throws CompositeException;
```

3. deleteResource

Delete a resource associated with passed in resource path.

```
@param serverId target server id from servers config xml
@param resourcePath resource path
@param pathToServersXML path to the server values xml
@throws CompositeException if deletion of the resource fails

public void deleteResource(String serverId, String resourcePath, String
pathToServersXML) throws CompositeException;
```

4. deleteResources

Delete resources associated with passed in resource ids found in the resource XML.

```
@param serverId target server id from servers config xml
@param resourceIds comma separated list of resource ids from the resource
xml
@param pathToResourceXML path to the resource xml
@param pathToServersXML path to the server values xml
@throws CompositeException if deletion of the resource fails

public void deleteResources(String serverId, String resourceIds, String
pathToResourceXML,String pathToServersXML) throws CompositeException;
```

5. renameResource

Rename a resource associated with passed in resource path.

```
@param serverId target server id from servers config xml
@param resourcePath resource path
@param pathToServersXML path to the server values xml
@param newName the new name of the resource (this is not a path)
```

```
@throws CompositeException if deletion of the resource fails


void renameResource(String serverId, String resourcePath, String
pathToServersXML, String newName) throws CompositeException;
```

6. renameResources

Rename resources associated with passed in resource ids found in the resource XML.

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

@param resourceIds comma separated list of resource ids from the resource
xml

@param pathToResourceXML path to the resource xml

@param pathToServersXML path to the server values xml

@throws CompositeException if deletion of the resource fails


public void renameResources(String serverId, String resourceIds, String
pathToResourceXML,String pathToServersXML) throws CompositeException;
```

7. copyResource

Copy a resource associated with passed in resource path.

```
@param serverId target server config name

@param resourcePath resource path

@param pathToServersXML path to the server values xml

@param targetContainerPath the target CIS folder path to copy the
resource to

@param newName the new name of the resource being copied

@param copyMode the mode by which a copy is to be executed

"ALTER_NAME_IF_EXISTS" - If a resource of the same name and type of the
source resource already exists in the target container, then avoid
conflicts by  automatically generating a new name. Names are generated by
appending a number to the end of the provided name.

"FAIL_IF_EXISTS" - Fails if a resource of the same name and type already
exists in the target container. The resource will not be copied if this
occurs.

"OVERWRITE_MERGE_IF_EXISTS" - If a resource of the same name and type of
the source resource already exists in the target container, then
overwrite the resource in the target container. If the source resource is
a container, then merge the contents of the source container with the
corresponding resource in the target. All resources in the source
container will overwrite those resources with the same name in the
target, but child resources in the target with different names will not
be overwritten and remain unaltered.
```

```
"OVERWRITE_REPLACE_IF_EXISTS" - If a resource of the same name and type
of the source resource already exists in the target container, then
overwrite the resource in the target container. If the source resource is
a container, then replace the container within the target container with
the source container. This is equivalent to deleting the container in the
target before copying the source.

@throws CompositeException if resource copy fails


void copyResource(String serverId, String resourcePath, String
pathToServersXML, String targetContainerPath, String newName, String
copyMode) throws CompositeException;
```

8. copyResources

Copy resources associated with passed in resource ids found in the resource XML.

```
@param serverId target server config name

@param resourceIds comma separated list of resource ids from the resource
xml

@param pathToResourceXML path to the resource xml

@param pathToServersXML path to the server values xml

@throws CompositeException if resource copy fails


void copyResources(String serverId, String resourceIds, String
pathToResourceXML, String pathToServersXML) throws CompositeException;
```

9. moveResource

Move a resource associated with passed in resource path.

```
@param serverId target server config name

@param resourcePath resource path

@param pathToServersXML path to the server values xml

@param targetContainerPath the target CIS folder path to copy the
resource to

@param newName the new name of the resource being copied

@throws CompositeException if resource move fails


void moveResource(String serverId, String resourcePath, String
pathToServersXML, String targetContainerPath, String newName) throws
CompositeException;
```

10. moveResources

Move resources associated with passed in resource ids found in the resource XML.

```
@param serverId target server config name
```

```
@param resourceIds comma separated list of resource ids from the resource
xml
@param pathToResourceXML path to the resource xml
@param pathToServersXML path to the server values xml
@throws CompositeException if resource move fails


void moveResources(String serverId, String resourceIds, String
pathToResourceXML, String pathToServersXML) throws CompositeException;
```

11. doResourceExist

Checks for existence of a resource associated with passed in resource path. Throw an exception if any resource is not found.

```
@param serverId target server id from servers config xml
@param resourcePath path to the CIS resource
@param pathToServersXML path to the server values xml
@throws CompositeException if resource cannot be found


public void doResourcesExist(String serverId, String resourcePath, String
pathToServersXML) throws CompositeException;
```

12. doResourcesExist

Checks for existence of a resource associated with passed in resource ids. Throw an exception if any resources in the list are not found.

```
@param serverId target server id from servers config xml
@param resourceIds resource ids from the resource xml
@param pathToResourceXML path to the resource xml
@param pathToServersXML path to the server values xml
@throws CompositeException if resource cannot be found


public void doResourcesExist(String serverId, String resourceIds, String
pathToResourceXML,String pathToServersXML) throws CompositeException;
```

13. lockResource

Lock a resource associated with passed in resource path.

```
@param serverId target server config name
@param resourcePath resource path
@param pathToServersXML path to the server values xml
```

```
@throws CompositeException if resource deletion fails


void lockResource(String serverId, String resourcePath, String
pathToServersXML) throws CompositeException;
```

14. lockResources

Lock resource associate with passed in resource ids found in the resource XML.

```
@param serverId target server config name

@param resourceIds comma separated list of resource ids from the resource
xml

@param pathToResourceXML path to the resource xml

@param pathToServersXML path to the server values xml

@throws CompositeException if resource deletion fails


void lockResources(String serverId, String resourceIds, String
pathToResourceXML, String pathToServersXML) throws CompositeException;
```

15. unlockResource

Unlock a resource associated with passed in resource path.

```
@param serverId target server config name

@param resourcePath resource path

@param pathToServersXML path to the server values xml

@param comment description/comment for the unlock action

@throws CompositeException if resource deletion fails


void unlockResource(String serverId, String resourcePath, String
pathToServersXML, String comment) throws CompositeException;
```

16. unlockResources

Unlock resource associate with passed in resource ids found in the resource XML.

```
@param serverId target server config name

@param resourceIds comma separated list of resource ids from the resource
xml

@param pathToResourceXML path to the resource xml

@param pathToServersXML path to the server values xml

@throws CompositeException if resource deletion fails


void unlockResources(String serverId, String resourceIds, String
pathToResourceXML, String pathToServersXML) throws CompositeException;
```

17. createFolder

Create all folders in the path associated with passed in resource path.

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

@param resourcePath resource path

@param pathToServersXML path to the server values xml

@param recursive true=create all folders recursively, false=only create
the folder specified (Note: all intermediate folders must exist for the
operation to be successful when recursive=false)

@throws CompositeException if creation of the resource fails


void createFolder(String serverId, String resourcePath, String
pathToServersXML, String recursive) throws CompositeException;
```

18. createFolders

Create all folders in the path associated with the passed in resource ids.

```
@param serverId target server config name

@param resourceIds comma separated list of resource ids from the resource
xml

@param pathToResourceXML path to the resource xml

@param pathToServersXML path to the server values xml

@throws CompositeException if resource creation fails


void createFolders(String serverId, String resourceIds, String
pathToResourceXML, String pathToServersXML) throws CompositeException;
```

General Notes:

The arguments pathToResourceXML 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.

3 Resource 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 ResourceModule XML provides a structure "resource" for "executeConfiguredProcedures, deleteResources, renameResources, and doResourcesExist". The global entry point node is called "ResourceModule" and contains one or more "resource" nodes.

```
<?xml version="1.0"?>
<p1:ResourceModule xmlns:p1="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <resource>
        <id>string</id>
        <resourcePath>/shared/test1/proc1</resourcePath>
        <targetContainerPath>/shared/test</targetContainerPath>
        <recursive>true</recursive>
        <newName>PROC2</newName>
        <copyMode>OVERWRITE_REPLACE_IF_EXISTS</copyMode>
        <comment>description</comment>
        <dataServiceName>TEST</dataServiceName>
        <argument>arg1</argument>
        <argument>arg2</argument>
        <argument>3</argument>
        <outputReturnVariables>true</outputReturnVariables>
        <ignoreErrors>true</ignoreErrors>
    </resource>
</p1:ResourceModule>
```

Attributes of Interest

id – a unique resource identifier within the ResourceModule.xml.

resourcePath – the CIS resource path is used for all ResourceModule operations. A resource path can be found in the info tab of a given resource within Studio. Multiple resource paths may be provided for methods such as doResourcesExists and createFolder.

recursive – used for ***creating folders*** [true|false] true=recursively create all of the folders in the path and false=on create the last specified folder in the path.

targetContainerPath – used for ***moving*** or ***copying*** a resource to a new location

newName – used for ***renaming*** or ***copying*** a resource to a new name.

copyMode – used for ***copying*** a resource to another location.

comment – used to provide an annotation, description or comment when ***unlocking*** a resource.

dataServiceName – used when ***executing*** a procedure. The dataServiceName is the CIS JDBC database name.

argument – an unbounded iteration of arguments used when ***executing*** a ***configured procedure***. These fields are optional so if the action does not require procedures, they are not necessary to include. The ResourceModule will automatically format the procedures as follows: 'arg1','arg2','arg3'..... Additionally, the ResourceModule will automatically type cast the arguments to the CIS procedure being invoked.

outputReturnVariables – may be set when invoking executeProcedure(). true=(default) output the values of the return variables from the procedure call, false=do not output the return variable values.

ignoreErrors – this parameter is used in conjunction with the createFolders method. false=(default) do not ignore errors. true=ignore any errors and produce warning messages.

Attribute Value Restrictions

copyMode – restriction description:

- "ALTER_NAME_IF_EXISTS" - If a resource of the same name and type of the source resource already exists in the target container, then avoid conflicts by automatically generating a new name. Names are generated by appending a number to the end of the provided name.
- "FAIL_IF_EXISTS" - Fails if a resource of the same name and type already exists in the target container. The resource will not be copied if this occurs.
- "OVERWRITE_MERGE_IF_EXISTS" - If a resource of the same name and type of the source resource already exists in the target container, then overwrite the resource in the target container. If the source resource is a container, then merge the contents of the source container with the corresponding resource in the target. All resources in the source container will overwrite those resources with the same name in the target, but child resources in the target with different names will not be overwritten and remain unaltered.
- "OVERWRITE_REPLACE_IF_EXISTS" - If a resource of the same name and type of the source resource already exists in the target container, then overwrite the resource in the target container. If the source resource is a container, then replace the container within the target container with the source container. This is equivalent to deleting the container in the target before copying the source.

```
<xs:element name="copyMode" minOccurs="0">
    <xs:annotation>
        <xs:documentation xml:lang="en">...</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
         <xs:restriction base="xs:string">
             <xs:enumeration value="ALTER_NAME_IF_EXISTS"/>
             <xs:enumeration value="FAIL_IF_EXISTS"/>
             <xs:enumeration value="OVERWRITE_MERGE_IF_EXISTS"/>
             <xs:enumeration value="OVERWRITE_REPLACE_IF_EXISTS"/>
         </xs:restriction>
```

```
            </xs:simpleType>
        </xs:element>
```

4 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 ResourceModule.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 "*PDTool User's Guide.pdf*". The abridged version is as follows:
Windows: ExecutePDTool.bat -exec ../resources/plans/UnitTest-Resource.dp
Unix: ./ExecutePDTool.sh -exec ../resources/plans/UnitTest-Resource.dp
Properties File (UnitTest-Resource.dp):
Property File Rules:

```
# ----------------------------
# UnitTest-Resource.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:
# ----------------------------------------
# Execute a DV procedure
PASS   TRUE   ExecuteProcedure    executeConfiguredProcedures $SERVERID testproc-success
                  $MODULE_HOME/ResourceModule.xml $MODULE_HOME/servers.xml
# Execute a Configured Procedure (defined in ResourceModule.xml
PASS   TRUE   ExecuteProcedure    executeProcedure    $SERVERID testproc
      TEST   $MODULE_HOME/servers.xml "'myname','0','12.3','3.141592653589793','2000-
02-01','23:59:01','1923-03-06 23:59:31','','1'"
# Delete a resource at a given path
PASS   FALSE  ExecuteAction       deleteResource      $SERVERID /shared/test/f1/f2/p2
      $MODULE_HOME/servers.xml
# Delete resources configured in ResourceModule.xml
PASS   FALSE  ExecuteAction       deleteResources     $SERVERID "delete1,delete2"
      $MODULE_HOME/ResourceModule.xml $MODULE_HOME/servers.xml
# Rename a resource at a given path
#PASS  FALSE  ExecuteAction       renameResource      $SERVERID /shared/test/f1/f2/p2
      $MODULE_HOME/servers.xml p2copy
# Rename resources configured in ResourceModule.xml
PASS   FALSE  ExecuteAction       renameResources     $SERVERID "rename1,rename2"
      $MODULE_HOME/ResourceModule.xml $MODULE_HOME/servers.xml
# Copy a resource at a given path
#PASS  FALSE  ExecuteAction       copyResource        $SERVERID /shared/test/f1/f2/p2
      $MODULE_HOME/servers.xml /shared/test/f1/f2 p2copy OVERWRITE_REPLACE_IF_EXISTS
```

```
# Copy resources configured in ResourceModule.xml
#PASS  FALSE  ExecuteAction       copyResources       $SERVERID "copy1"
             $MODULE_HOME/ResourceModule.xml $MODULE_HOME/servers.xml
# Move a resource at a given path
#PASS  FALSE  ExecuteAction       moveResource        $SERVERID /shared/test/f1/f2/p2
      $MODULE_HOME/servers.xml /shared/test/f1/f2 p2move OVERWRITE_REPLACE_IF_EXISTS
# Move resources configured in ResourceModule.xml
#PASS  FALSE  ExecuteAction       moveResources       $SERVERID "move1"
             $MODULE_HOME/ResourceModule.xml $MODULE_HOME/servers.xml
# Check for existence of a resource specified by the resource path
PASS   FALSE  ExecuteAction       doResourceExist     $SERVERID /shared/test00/f1/p1
      $MODULE_HOME/servers.xml
# Check for existence of resources configured in ResourceModule.xml
PASS   FALSE  ExecuteAction       doResourcesExist    $SERVERID "exist1,exist2"
      $MODULE_HOME/ResourceModule.xml $MODULE_HOME/servers.xml
# Lock a resource at a given path
PASS   FALSE  ExecuteAction       lockResource        $SERVERID /shared/test/f1/p1
      $MODULE_HOME/servers.xml
# Lock a resource at a given path
PASS   FALSE  ExecuteAction       lockResources       $SERVERID "lock1,lock2"
      $MODULE_HOME/ResourceModule.xml $MODULE_HOME/servers.xml
# Unlock a resource at a given path
#PASS  FALSE  ExecuteAction       unlockResource      $SERVERID /shared/test/f1/p1
      $MODULE_HOME/servers.xml
# Unlock resource configured in ResourceModule.xml
PASS   FALSE  ExecuteAction       unlockResources     $SERVERID "lock1,lock2"
      $MODULE_HOME/ResourceModule.xml $MODULE_HOME/servers.xml
# Create folder resource at a given path
PASS   FALSE  ExecuteAction       createFolder              $SERVERID
/shared/test00/_test1                   "$MODULE_HOME/servers.xml" true
# Create folder resource configured in ResourceModule.xml
PASS   FALSE  ExecuteAction       createFolders             $SERVERID "createFolder1"
             "$MODULE_HOME/ResourceModule.xml" "$MODULE_HOME/servers.xml"
```

Ant Execution

The full details on build file setup and ant execution can be found in the document "*PDTool User's Guide.pdf*". The abridged version is as follows:
Windows: ExecutePDTool.bat -ant ../resources/ant/build-Resource.xml
Unix: ./ExecutePDTool.sh -ant ../resources/ant/build-Resource.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="&amp;quot;&amp;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="resourceIds"               value="id1,id2"/>
  <property name="arguments-pass" value="'myname','0','12.3','3.141592653589793','2000-02-
01','23:59:01','1923-03-06 23:59:31','','1'"/>
  <property name="arguments-fail" value="'myname','5','12.3','3.141592653589793','2000-02-
01','23:59:01','1923-03-06 23:59:31','','1'"/>
  <property name="noarguments" value="&amp;quot;&amp;quot;"/>

  <!-- 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.tibco.ps.deploytool.ant.CompositeAntTask" classpathref="project.class.path"/>

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

    <!-- Execute Line Here -->
Place actions to execute here:

    <!-- Windows or UNIX: Entire list of actions
# Execute a DV procedure
   <executeJavaAction action="executeProcedure"
zrguments="${SERVERID}^testproc^TEST^${pathToServersXML}^${arguments-pass}"
endExecutionOnTaskFailure="TRUE" />

# Execute a Configured Procedure (defined in ResourceModule.xml
   <executeJavaAction action="executeConfiguredProcedures"
arguments="${SERVERID}^testproc^${pathToResourceXML}^${pathToServersXML}"
endExecutionOnTaskFailure="TRUE" />
```

```
# Delete a resource at a given path
   <executeJavaAction action="deleteResource"
arguments="${SERVERID}^/shared/test/f1/f2/p2^${pathToServersXML}" endExecutionOnTaskFailure="TRUE"
/>

# Delete resources configured in ResourceModule.xml
   <executeJavaAction action="deleteResources"
arguments="${SERVERID}^delete1,delete2^${pathToResourceXML}^${pathToServersXML}"
endExecutionOnTaskFailure="TRUE" />

# Rename a resource at a given path
   <executeJavaAction action="renameResource"
arguments="${SERVERID}^/shared/test/f1/f2/p2^${pathToServersXML}^p2copy"
endExecutionOnTaskFailure="TRUE" />

# Rename resources configured in ResourceModule.xml
   <executeJavaAction action="renameResources"
arguments="${SERVERID}^rename1,rename2^${pathToResourceXML}^${pathToServersXML}"
endExecutionOnTaskFailure="TRUE" />

# Copy a resource at a given path
   <executeJavaAction action="copyResource"
arguments="${SERVERID}^/shared/test/f1/f2/p2^${pathToServersXML}^/shared/test/f1/f2^p2copy^OVERWRIT
E_REPLACE_IF_EXISTS" endExecutionOnTaskFailure="TRUE" />

# Copy resources configured in ResourceModule.xml
   <executeJavaAction action="copyResources"
arguments="${SERVERID}^copy1^${pathToResourceXML}^${pathToServersXML}"
endExecutionOnTaskFailure="TRUE" />

# Move a resource at a given path
   <executeJavaAction action="moveResource"
arguments="${SERVERID}^/shared/test/f1/f2/p2^${pathToServersXML}^/shared/test/f1/f2^p2move"
endExecutionOnTaskFailure="TRUE" />

# Move resources configured in ResourceModule.xml
   <executeJavaAction action="moveResources"
arguments="${SERVERID}^move1^${pathToResourceXML}^${pathToServersXML}"
endExecutionOnTaskFailure="TRUE" />

# Check for existence of a resource specified by the resource path
   <executeJavaAction action="doResourceExist"
arguments="${SERVERID}^/shared/test00/f1/p1^${pathToServersXML}" endExecutionOnTaskFailure="TRUE"
/>

# Check for existence of resources configured in ResourceModule.xml
   <executeJavaAction action="doResourcesExist"
arguments="${SERVERID}^exist1,exist2^${pathToResourceXML}^${pathToServersXML}"
endExecutionOnTaskFailure="TRUE" />

# Lock a resource at a given path
   <executeJavaAction action="lockResource"
arguments="${SERVERID}^/shared/test/f1/p1^TEST^${pathToServersXML}"
endExecutionOnTaskFailure="TRUE" endExecutionOnScriptLaunch="TRUE"/>

# Lock resources configured in ResourceModule.xml
```

```
   <executeJavaAction action="lockResources"
arguments="${SERVERID}^lock1,lock2^${pathToResourceXML}^${pathToServersXML}"
endExecutionOnTaskFailure="TRUE" endExecutionOnScriptLaunch="TRUE"/>


# Unlock a resource at a given path
   <executeJavaAction action="unlockResource"
arguments="${SERVERID}^/shared/test/f1/p1^${pathToServersXML}" endExecutionOnTaskFailure="TRUE"
endExecutionOnScriptLaunch="TRUE"/>


# Unlock resource configured in ResourceModule.xml
   <executeJavaAction action="unlockResources"
arguments="${SERVERID}^lock1,lock2^${pathToResourceXML}^${pathToServersXML}"
endExecutionOnTaskFailure="TRUE" endExecutionOnScriptLaunch="TRUE"/>


# Create folder resource at a given path -->
<executeJavaAction action="createFolder"
arguments="${SERVERID}^/shared/test00/_test1^${pathToServersXML}^true"
endExecutionOnTaskFailure="TRUE" />
    -->
  </target>
</project>
```

Module ID Usage

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

- 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   executeConfiguredProcedures $SERVERID "r1,r2"
       "$MODULE_HOME/ResourceModule.xml" "$MODULE_HOME/servers.xml"
```

  Example Ant build file

```
<executeJavaAction description="Update" action="executeConfiguredProcedures"
arguments="${SERVERID}^r1,r2^${pathToResourceXML}^${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   executeConfiguredProcedures $SERVERID "*"
       "$MODULE_HOME/ResourceModule.xml" "$MODULE_HOME/servers.xml"
```

  Example Ant build file

```
<executeJavaAction description="Update" action="executeConfiguredProcedures"
arguments="${SERVERID}^*^${pathToResourceXML}^${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   executeConfiguredProcedures $SERVERID "-r3,r4"
       "$MODULE_HOME/ResourceModule.xml" "$MODULE_HOME/servers.xml"
```

Example Ant build file

```
<executeJavaAction description="Update" action="executeConfiguredProcedures"
arguments="${SERVERID}^-r3,r3^${pathToResourceXML}^${pathToServersXML}"
```

5 PDTool Examples

The following are common scenarios when using the ResourceModule.

Scenario 1 – execute a CIS JDBC procedure

Description:

This scenario describes how to execute a CIS JDBC procedure where the procdure returns any output from OUT variables to the command line window and log file. This capability allows a user to write CIS SQL Script procedures to perform customizations within CIS. Execute Configured Procedures allows the user to easily execute those procedures and pass in arguments from the command or Ant scripts.

XML Configuration Sample:

The following XML provides an example of how to setup the ResourceModule.xml file for executing a configured procedure. The resourcePath is used for the procedure name. The dataServiceName is the CIS JDBC database name. For each parameter in the procedure, there is an argument defined in the XML. The procedure does type conversions from the argument to the procedure parameter.

```
<?xml version="1.0"?>
<p1:ResourceModule xmlns:p1="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <resource>
        <id>testproc</id>
        <resourcePath>testproc</resourcePath>
        <dataServiceName>TEST</dataServiceName>
        <argument>Test Proc success</argument>
        <argument>0</argument>
        <argument>12.3</argument>
        <argument>3.141592653589793</argument>
        <argument>2000-02-01</argument>
        <argument>23:59:01</argument>
        <argument>1923-03-06 23:59:31</argument>
        <argument></argument>
        <argument>0</argument>
        <outputReturnVariables>true</outputReturnVariables>
    </resource>
</p1:ResourceModule>
```

Execution Sample:

Unix: ./ExecutePDTool.sh -exec ../resources/plans/UnitTest-Resource.dp
Property file setup for UnitTest-Resource.dp:

```
# ----------------------------------------
```

```
# Begin task definition list for UNIX:
# ----------------------------------------
# Execute Configured Procedure
PASS   TRUE   ExecuteProcedure    executeProcedure $SERVERID testproc
       $MODULE_HOME/ResourceModule.xml $MODULE_HOME/servers.xml
```

Alternative Style of execution using "executeProcedure":
Unix: ./ExecutePDTool.sh -exec ../resources/plans/UnitTest-Resource.dp
Property file setup for UnitTest-Resource.dp:

```
# ----------------------------------------
# Begin task definition list for UNIX:
# ----------------------------------------
# Execute Configured Procedure
PASS   TRUE   ExecuteAction  executeProcedure $SERVERID testproc TEST00
"$MODULE_HOME/servers.xml" "'myname','0','12.3','3.141592653589793','2000-02-
01','23:59:01','1923-03-06 23:59:31','','1'"  true


# Note: true at the end of the line informs executeProcedure to output the value
of any variables.
```

Results Expected:
The script will report "PASS" for the execution of this action.

Scenario 2 – resource exists in CIS folder

Description:
This scenario describes how to check for existence of a CIS resource. This can be useful to insure that certain key resources are present in the target CIS instance before proceeding with deployment.

XML Configuration Sample:
The following XML provides an example of how to setup the ResourceModule.xml file for executing a configured procedure. The resourcePath is used for the procedure name. The dataServiceName is the CIS JDBC database name. For each parameter in the procedure, there is an argument defined in the XML. The procedure does type conversions from the argument to the procedure parameter.

```
<?xml version="1.0"?>
<p1:ResourceModule xmlns:p1="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <resource>
        <id>exist1</id>
        <resourcePath>/shared/test/f1/p1</resourcePath>
    </resource>
</p1:ResourceModule>
```

Execution Sample:
Unix: ./ExecutePDTool.sh -exec ../resources/plans/UnitTest-Resource.dp

Property file setup for UnitTest-Resource.dp:

```
# ----------------------------------------
# Begin task definition list for UNIX:
# ----------------------------------------
# Does resource exist
#PASS  FALSE  ExecuteAction       doResourcesExist    $SERVERID "exist1"
       $MODULE_HOME/ResourceModule.xml $MODULE_HOME/servers.xml
```

Results Expected:

The script will report “PASS” for the execution of this action.

6 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.

Copy Resource Faults:

DuplicateName: If a resource in the target container exists with the same type as the source, same name as newName, and the copy mode is FAIL_IF_EXISTS.

IllegalArgument: If the any of the given paths or types are malformed, or if the copyMode is not one of the legal values.

IllegalState: If the source resource is not allowed to be copied. Resources in /services/databases/system, /services/webservices/system, or within any physical data source may not be copied.

NotAllowed: If the source resource is not allowed to exist within the target container. Resources cannot be copied into a physical data source. LINK resources can only be copied into RELATIONAL_DATA_SOURCEs, SCHEMAs, and PORTs under /services. Non-LINK resources cannot be copied into any location under /services.

NotFound: If the source resource or any portion of the new path does not exist.

Security: If the user does not have READ access on all items in the source path.

Security: If the user does not have READ access on the items in the targetContainerPath other than the last item.

Security: If the user does not have WRITE access to the last item in targetContainerPath.

Security: If the user does not have WRITE access to a resource that is to be overwritten in one of the overwrite modes.

Security: If the user does not have the ACCESS_TOOLS right.

Rename Resource Faults:

DuplicateName: If a resource already exists of this type with the new name.

IllegalArgument: If the path is malformed or the type is illegal.

IllegalState: If the resource is not allowed to be renamed. Resources within a physical data source cannot be renamed. The "/shared", "/services/databases", "/services/webservices", and user home folders cannot be renamed.

NotFound: If resource does not exist.

Security: If the user does not have READ access on all items in the path other than the last one.

Security: If the user does not have WRITE access to the last item in newPath.

Security: If the user does not have the ACCESS_TOOLS right.

DuplicateName: A resource named <"abc"> already exists at path <"/shared/xyz"> with the same type.

Move Resource Faults:

DuplicateName: If a resource in the target container exists with the same name and type as one of the source and the overwrite is "false".

IllegalArgument: If the any of the given paths are malformed or if any of the types are illegal.

IllegalState: If any of the source resources is not allowed to be moved. Resources in /services/databases/system, /services/webservices/system, or within any physical data source may not be moved.

NotAllowed: If any of the source resources is not allowed to exist within the target container. Resources cannot be moved into a physical data source.

LINK resources can only be moved into RELATIONAL_DATA_SOURCEs, SCHEMAs, and PORTs under /services. Non-LINK resources cannot be moved into any location under /services.

NotFound: If any of the source resources or any portion of the path to the target container do not exist.

Security: If the user does not have READ access on all items in the source paths.

Security: If the user does not have READ access on the items in the targetContainerPath other than the last item.

Security: If the user does not have WRITE access to the last item in targetContainerPath.

Security: If the user does not have WRITE access to a resource that is to be overwritten.

Security: If the user does not have the ACCESS_TOOLS right.

Delete Resource Faults:

IllegalArgument: If the path or type is malformed.

IllegalState: If the resource is not allowed to be destroyed.

NotFound: If the resource or any portion of the path does not exist.

Security: If the user does not have READ access on all items in the path other than the last one.

Security: If the user does not have WRITE access to the resource.

Security: If "force" is "false" and the resource is a container and the user does not have WRITE access to any resource within the container.

Security: If the user does not have the ACCESS_TOOLS right.

Lock Resource Faults:

IllegalArgument: If the path is malformed or the type or detail are illegal.

NotFound: If the resource or any portion of the path to the resource does not exist.

Security: If the user does not have READ access on all items in the path, except the last item in the path.

Security: If the user does not have WRITE access to the last item in the path and does not have the MODIFY_ALL_RESOURCES right.

Security: If the user does not have the ACCESS_TOOLS right.

IllegalStateException: Resource <xyz> is already locked by user <abc>

IllegalStateException: Resource <xyz> is not locked, so it cannot be unlocked

Unlock Resource Faults:

Faults:

IllegalArgument: If the path is malformed or the type or detail are illegal.

NotFound: If the resource or any portion of the path to the resource does not exist.

Security: If the user does not have WRITE access to the last item in the path and does not have the MODIFY_ALL_RESOURCES right.

Security: If the user does not have the ACCESS_TOOLS right.

Security: If the user is not the lock owner and does not have the UNLOCK_RESOURCE right.

Execute Procedure Faults:

IllegalArgument: If any elements are malformed.
RuntimeError: If an error occurs during execution.
RuntimeError: If the user does not have appropriate privileges on any resources referred to by the "sqlText".
Security: If the user omitted the "dataServiceName" and does not have the ACCESS_TOOLS right.

7 Conclusion

Concluding Remarks

The Promotion and Deployment Tool is a set of pre-built modules intended to provide a turn-key experience for promoting DV resources from one DV 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 Professional Services for the advancement of the "***Promotion and Deployment Tool***".