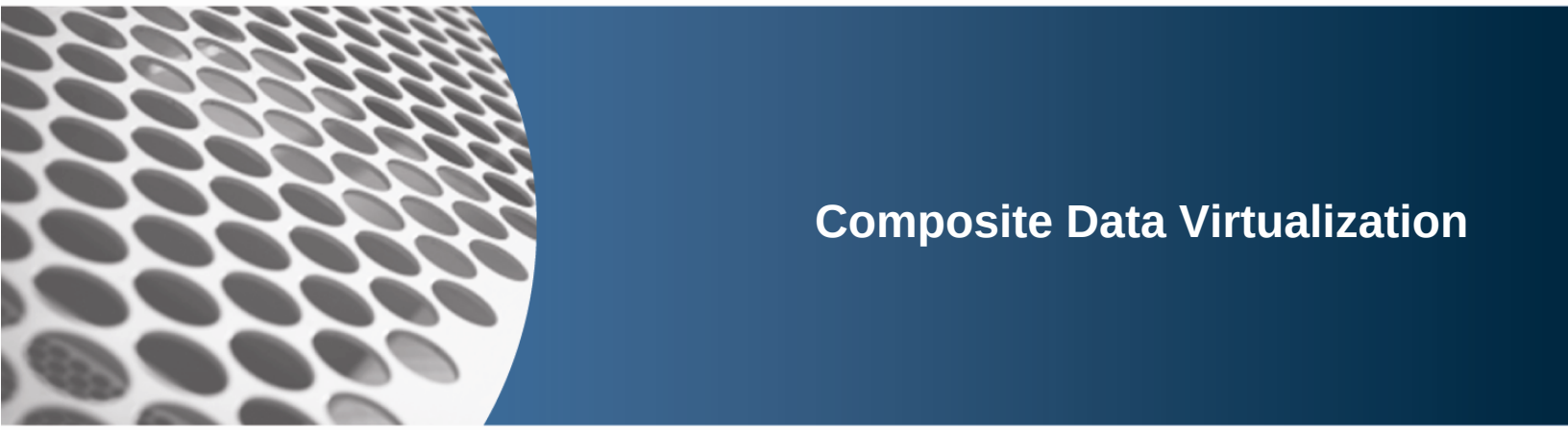
WHITE PAPER

COMPOSITE SOFTWARE



Composite PS Promotion and Deployment Tool

Resource Cache Module User Guide

Composite Professional Services

November 2014

TABLE OF CONTENTS

DOCUMENT CONTROL

Version History

Version	Date	Author	Description
1.0	6/6/2011	Mike Tinius	Initial revision for Resource Cache User Guide
1.0.1	8/1/2011	Mike Tinius	Revisions due to Architecture changes
3.0	8/21/2013	Mike Tinius	Updated docs to Cisco format
3.1	1/31/2014	Mike Tinius	Added updateResourceCacheEnabled to enable/disable all caches in a starting folder for the Resource Cache Module.
3.2	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

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 Resource Cache User Guide is to demonstrate how to effectively use the Resource Cache Module for managing cacheable resources. A cacheable resource can be either CIS procedures, data source tables or views. Managing a cacheable resource involves being able to update the cached resource parameters, clearing the cache, refreshing the cache and generating the XML property file from existing cached resources. Any changes made using this module will be reflected in Composite Studio after refreshing the resources.

Audience

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

- Architects
- Developers
- Administrators.
- Operations personnel.

RESOURCE CACHE MODULE DEFINITION

Method Definitions and Signatures

1. **updateResourceCache**

   Update the resource cache configuration for a procedure, table or view for a given set of resource Ids.

```
@param serverId - target server id from server XML configuration file

@param resourceIds - list of resource cache Ids (comma separated list
of Ids)

@param pathToResourceCacheXML - path to the resource cache module XML
configuration file.

@param pathToServersXML - path to the server XML configuration file.

@throws CompositeException


public void updateResourceCache(String serverId, String resourceIds,
String pathToResourceCacheXML, String pathToServersXML)  throws
CompositeException;
```

2. **clearResourceCache**

   Clear the cache for a given set of resource Ids.

```
@param serverId - target server id from server XML configuration file

@param resourceIds - list of resource cache Ids (comma separated list
of Ids)

@param pathToResourceCacheXML - path to the resource cache module XML
configuration file.

@param pathToServersXML - path to the server XML configuration file.

@throws CompositeException


public void clearResourceCache(String serverId, String resourceIds,
String pathToResourceCacheXML, String pathToServersXML)  throws
CompositeException;
```

3. **refreshResourceCache**

   Refresh the cache for a given set of resource Ids.

```
@param serverId - target server id from server XML configuration file

@param resourceIds - list of resource cache Ids (comma separated list
of Ids)

@param pathToResourceCacheXML - path to the resource cache module XML
configuration file.

@param pathToServersXML - path to the server XML configuration file.
```

```
@throws CompositeException


public void refreshResourceCache(String serverId, String resourceIds,
String pathToResourceCacheXML, String pathToServersXML)  throws
CompositeException;
```

4. **updateResourceCacheEnabled**

   Update the resource cache enable flag for a given resource or an entire folder of resources for a given set of resource Ids. If the resource type is a CONTAINER then recursively walk the starting resource path and interrogate resources that are configured for caching and set the enabled flag to either "true" (enabled) or "false" (disabled). If the resource type is anything other than CONTAINER, then only interrogate that resource to see if the cache is configured and set the enabled flag to "true" or "false".

```
@param serverId - target server id from server XML configuration file

@param resourceIds - list of resource cache Ids (comma separated list
of Ids)

@param pathToResourceCacheXML - path to the resource cache module XML
configuration file.

@param pathToServersXML - path to the server XML configuration file.
Only the "enabled" element is used.  All other elements do not need to
be present.

@throws CompositeException


public void updateResourceCacheEnabled(String serverId, String
resourceIds, String pathToResourceCacheXML, String pathToServersXML)
throws CompositeException;
```

5. **generatetResourceCacheXML**

   Generate an XML property file of all the resource cache configurations for a given starting path and the provided generation options.

```
@param serverId - target server id from server XML configuration file

@param startPath - starting path for the resources to generate

@param pathToResourceCacheXML - path including name to the resource
cache XML which will be generated

@param pathToServersXML - path to the server XML configuration file.

@param options – space or comma separated list of options [CONFIGURED
NONCONFIGURED TABLE PROCEDURE].

If left blank, the default is to produce an XML for all tables and
procedures whether they are cached configured or not

    CONFIGURED - generate XML for cache configured resources

    NONCONFIGURED - generate XML for non-configured cache resources
```

```
    TABLE - generate XML for a data source TABLE or VIEW
    PROCEDURE - generate XML for PROCEDURE
@throws CompositeException


public void generateResourceCacheXML(String serverId, String
startPath, String pathToResourceCacheXML, String pathToServersXML,
String options) throws CompositeException;
```

The following structure is an example of what is generated:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:ResourceCacheModule xmlns:ns2="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <resourceCache>
        <id>shared1</id>
        <resourcePath>/shared/test1/cache/customer cache</resourcePath>
        <resourceType>TABLE</resourceType>
        <cacheConfig>
            <configured>true</configured>
            <enabled>true</enabled>
            <storage>
                <mode>DATA SOURCE</mode>

<storageDataSourcePath>/shared/examples/ds_orders</storageDataSourcePath>
                <storageTargets>
                    <targetName>result</targetName>
                    <path>/shared/examples/ds orders/customers cache</path>
                    <type>TABLE</type>
                </storageTargets>
            </storage>
            <refresh>
                <mode>SCHEDULED</mode>
                <schedule>
                    <startTime>2011-06-08T00:00:00.000Z</startTime>
                    <refreshPeriod>
                        <period>HOUR</period>
                        <count>1</count>
                    </refreshPeriod>
                </schedule>
            </refresh>
            <expirationPeriod>
                <period>DAY</period>
                <count>1</count>
            </expirationPeriod>
            <clearRule>NONE</clearRule>
        </cacheConfig>
    </resourceCache>
</ns2:ResourceCacheModule>
```

General Notes:

The arguments pathToResourceCacheXML 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.

RESOURCE CACHE 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 ResourceCacheModule XML provides a structure “resourceCache” for updating an existing resource cache and generating the XML from a given set of cacheable resources.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:ResourceCacheModule
xmlns:ns2="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <resourceCache>
        <id>shared1</id>
        <resourcePath>/shared/test1/cache/customer_cache</resourcePath>
        <resourceType>TABLE</resourceType>
        <cacheConfig>
            <configured>true</configured>
            <enabled>true</enabled>
            <storage>
                <mode>DATA_SOURCE</mode>

      <storageDataSourcePath>/shared/examples/ds_orders</storageDataSourcePat
h>
                <storageTargets>
                    <targetName>result</targetName>
                    <path>/shared/examples/ds_orders/customers_cache</path>
                    <type>TABLE</type>
                </storageTargets>
            </storage>
            <refresh>
                <mode>SCHEDULED</mode>
                <schedule>
                    <startTime>2011-06-08T00:00:00.000Z</startTime>
                    <refreshPeriod>
                        <period>HOUR</period>
                        <count>1</count>
                    </refreshPeriod>
                </schedule>
            </refresh>
            <expirationPeriod>
                <period>DAY</period>
                <count>1</count>
            </expirationPeriod>
            <clearRule>NONE</clearRule>
        </cacheConfig>
    </resourceCache>
</ns2:ResourceCacheModule>
```

Attributes of Interest

configured – Configured is a boolean (true or false) value that informs CIS whether this cache is configured or not. If a cache is configured and this value is set to false and an updateResourceCache() is performed, the cache is disabled and destroyed and therefore, no longer configured.

enabled – Enabled is a boolean (true or false) value that informs CIS whether this cache is enabled or not. A cache may be configured but not enabled. Modifying this value allows the user to control when caching is turned on and off.

startTime – this is a timestamp value which informs CIS when the caching should be started. Notice the "T" in between the data and time. The format is "YYYY-MM-DDTHH24:mm:ss.dddZ". Year=YYYY, Month=MM, Day=DD, T=XML separator, Hour=HH24 (24 hour military time), Minute=mm, Second=ss, Second Fraction=.ddd (optional), Timezone=Z

storageTargets.targetName – this value is generally set to "result" which signifies that the result cursor from the Composite View or Table is being used for column projections in the cache.

storageTargets.path – this is the path to the Composite data source table that is being used for caching of the specified resource.

storageTargets.type – this value is always TABLE which signifies that a relational table is being used.

Attribute Value Restrictions

mode (storage) – this mode defines what type of storage will be used for caching.

- AUTOMATIC=this is the default behavior when a cache is first created. It automatically uses an internal file system file as the cache.
- DATA_SOURCE=this is configured to use a relational database table for the cache. The data source and table must exist. The table that is created uses the projection of the Composite View or Table.

```
<xs:element name="mode">
  <xs:annotation>
      <xs:documentation xml:lang="en">…</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
      <xs:restriction base="xs:string">
          <xs:enumeration value="AUTOMATIC"/>
          <xs:enumeration value="DATA_SOURCE"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

clearRule – an optional element that may contain one of NONE, ON_LOAD or ON_FAILURE. Normally old cache data is cleared on expiration and when a cache refresh successfully completes. In the latter case the old cache data is replaced by the new cached data. If this element is not present, the enable setting will be left unaltered

- NONE=when user clears it manually. Behavior: normal behavior will be used.

- ON_LOAD=when refresh begins. Behavior: in addition to the normal behavior the old cache data will be cleared when a refresh is started.
- ON_FAILURE=when refresh fails. Behavior: in addition to the normal behavior the old cache data will be cleared when a refresh fails.

```
<xs:element name="clearRule" minOccurs="0">
    <xs:annotation>
        <xs:documentation xml:lang="en">…</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="NONE"/>
            <xs:enumeration value="ON_LOAD"/>
            <xs:enumeration value="ON_FAILURE"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

resourceType – The resource type is a standard Composite type for defining what type of resource is being accessed. The only valid cacheable resource types are “TABLE” and “PROCEDURE”. Other types are described below but are not allowed to be cacheable.

```
  <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.

    (Basic CIS folder)
    * CONTAINER / FOLDER_CONTAINER - A Composite folder.
    * CONTAINER / DIRECTORY_CONTAINER - A Composite directory.
    (Database)
    * CONTAINER / CATALOG_CONTAINER - A Composite catalog folder under a data source.
    * CONTAINER / SCHEMA_CONTAINER - A Composite schema container.
    (Web Services)
    * CONTAINER / SERVICE_CONTAINER - A web service container for the service.
    * CONTAINER / OPERATIONS_CONTAINER - A web service container for the operations
    * CONTAINER / PORT_CONTAINER - A Composite web service container for port.
    (Connectors)
    * CONTAINER / CONNECTOR_CONTAINER - A Composite container for connectors.
    * CONNECTOR / JMS - A Composite JMS Connector.
    * CONNECTOR / HTTP - A Composite HTTP Connector.
    * 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 Definition set.
    * DEFINITION_SET / WSDL_DEFINITION_SET - A Composite WSDL Definition set.
    * DEFINITION_SET / ABSTRACT_WSDL_DEFINITION_SET - A Composite Abstract WSDL Definition set
    * 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.
    (Custom procedures)
    * PROCEDURE / JAVA_PROCEDURE - A Composite java data source procedure.
    (Database procedures)
    * PROCEDURE / EXTERNAL_SQL_PROCEDURE - A Composite Packaged Query.
    * PROCEDURE / DATABASE_PROCEDURE - A database stored procedure.
    (XML procedures)
    * PROCEDURE / BASIC_TRANSFORM_PROCEDURE - A Composite Basic XSLT Transformation procedure.
    * PROCEDURE / XSLT_TRANSFORM_PROCEDURE - A Composite XSLT Transformation procedure.
    * PROCEDURE / STREAM_TRANSFORM_PROCEDURE - A Composite XSLT Streaming Transformation
procedure.
    * PROCEDURE / XQUERY_TRANSFORM_PROCEDURE - A Composite XQUERY Transformation Procedure.
    * PROCEDURE / OPERATION_PROCEDURE - A Composite web service or HTTP procedure operation.

    * TABLE / SQL_TABLE - A Composite View.
    * 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.
     </xs:documentation>
     </xs:annotation>
     <xs:restriction base="xs:string">
          <xs:enumeration value="CONTAINER"/>
          <xs:enumeration value="CONNECTOR"/>
          <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 (refresh) – This mode defines what type of refresh will occur on the cache. The default behavior is to have a manual cache refresh. The refresh can be configured to be scheduled.

- MANUAL=(default). Manually refresh the cache.
- SCHEDULED=configure the resource to refresh the cache on a scheduled basis.

```
<xs:element name="mode">
  <xs:annotation>
      <xs:documentation xml:lang="en">…</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
        <xs:enumeration value="MANUAL"/>
        <xs:enumeration value="SCHEDULED"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

period (refreshPeriod / expirationPeriod) – This is used by the refresh period and expiration period to set the period of time. This element coincides with another element "count" which provides the amount of time for the period qualifier.

- Refresh=when used for refresh, it defines the period of time between cache refreshes.
- Expiration=when used with expiration, it defines the expiration period for the cache.

```
<xs:element name="period" minOccurs="1">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="SECOND"/>
            <xs:enumeration value="MINUTE"/>
            <xs:enumeration value="HOUR"/>
            <xs:enumeration value="DAY"/>
            <xs:enumeration value="WEEK"/>
            <xs:enumeration value="MONTH"/>
            <xs:enumeration value="YEAR"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

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 ResourceCacheModule.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-ResourceCache.dp

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

Properties File (UnitTest-ResourceCache.dp):

Property File Rules:

```
# ----------------------------
# UnitTest-ResourceCache.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 for UNIX:
# ----------------------------------------
#
PASS   FALSE  ExecuteAction generateResourceCacheXML   $SERVERID "/shared/test1"
       $MODULE_HOME/getResourceCacheModule.xml $MODULE_HOME/servers.xml "CONFIGURED
TABLE PROCEDURE"
#
PASS   FALSE  ExecuteAction   updateResourceCache     $SERVERID "shared1"
       $MODULE_HOME/ResourceCacheModule.xml $MODULE_HOME/servers.xml
#
PASS   FALSE  ExecuteAction   clearResourceCache      $SERVERID "shared1"
       $MODULE_HOME/ResourceCacheModule.xml $MODULE_HOME/servers.xml
#
PASS   FALSE  ExecuteAction   updateResourceCacheEnabled     $SERVERID "shared1"
              $MODULE_HOME/ResourceCacheModule.xml $MODULE_HOME/servers.xml
#
PASS   FALSE  ExecuteAction refreshResourceCache       $SERVERID "shared1"
       $MODULE_HOME/ResourceCacheModule.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-ResourceCache.xml

Unix: ./ExecutePDTool.sh -ant ../resources/ant/build-ResourceCache.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="resourceCacheIds"          value="shared1"/>
  <property name="pathToGenResCacheXML"
       value="${MODULE_HOME}/getResourceCacheModule.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="generateResourceCacheXML"
       arguments="${SERVERID}^/shared/test1^${pathToGenResCacheXML}^${pathToServersXML}^
CONFIGURED,TABLE,PROCEDURE"   endExecutionOnTaskFailure="TRUE" />

    <!-- Windows or UNIX: Entire list of actions
  <executeJavaAction description="Generate"  action="generateResourceCacheXML"
       arguments="${SERVERID}^/shared/test1^${pathToGenResCacheXML}^${pathToServersXML}^
CONFIGURED,TABLE,PROCEDURE"   endExecutionOnTaskFailure="TRUE" />

  <executeJavaAction description="Update"    action="updateResourceCache"
       arguments="${SERVERID}^${resourceCacheIds}^${pathToResourceCacheXML}^${pathToServersXML}
"              endExecutionOnTaskFailure="TRUE" />

  <executeJavaAction description="Clear"      action="clearResourceCache"
       arguments="${SERVERID}^${resourceCacheIds}^${pathToResourceCacheXML}^${pathToServersXML}
"              endExecutionOnTaskFailure="TRUE" />

  <executeJavaAction description="Refresh"   action="refreshResourceCache"
       arguments="${SERVERID}^${resourceCacheIds}^${pathToResourceCacheXML}^${pathToServersXML}
"              endExecutionOnTaskFailure="TRUE" />

  <executeJavaAction description="Update"    action="updateResourceCacheEnabled"
       arguments="${SERVERID}^${resourceCacheIds}^${pathToResourceCacheXML}^${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 “resourceCacheIds”.

- 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   updateResourceCache     $SERVERID "rc1,rc2"
                  $MODULE_HOME/ResourceCacheModule.xml $MODULE_HOME/servers.xml
    ```

    Example Ant build file

    ```
    <executeJavaAction description="Update"        action="updateResourceCache"
    arguments="${SERVERID}^rc1,rc2^${pathToResourceCacheXML}^${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   updateResourceCache     $SERVERID "*"
       $MODULE_HOME/ResourceCacheModule.xml $MODULE_HOME/servers.xml
```

  Example Ant build file

```
<executeJavaAction description="Update"        action="updateResourceCache"
arguments="${SERVERID}^*^${pathToResourceCacheXML}^${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   updateResourceCache     $SERVERID "-rc3,rc4"
              $MODULE_HOME/ResourceCacheModule.xml $MODULE_HOME/servers.xml
```

  Example Ant build file

```
<executeJavaAction description="Update"        action="updateResourceCache"
arguments="${SERVERID}^-rc3,rc3^${pathToResourceCacheXML}^${pathToServersXML}"
```

EXAMPLES

The following are common scenarios when using the ResourceCacheModule.

Scenario 1 – Generate Resource Cache XML

Description:

Generate the resource cache XML property file for /examples using the "CONFIGURED TABLE PROCEDURE" options.

XML Configuration Sample:

Not applicable for this example.

Execution Sample:

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

Property file setup for UnitTest-ResourceCache.dp:

```
# ----------------------------------------
# Begin task definition list for UNIX:
# ----------------------------------------
# Generate resource cache entries to ResourceCacheModule2.xml
PASS   FALSE  ExecuteAction generateResourceCacheXML   $SERVERID
"/shared/examples"  %MODULE_HOME%/ResourceCacheModule2.xml
%MODULE_HOME%/servers.xml "CONFIGURED TABLE PROCEDURE"
```

Results Expected:

The file getResourceCacheModule.xml is produced with only resourceCache nodes populated from the path /shared/examples.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:ResourceCacheModule
xmlns:ns2="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <resourceCache>
        <id>shared1</id>
        <resourcePath>/shared/examples/ds_orders/orders</resourcePath>
        <resourceType>TABLE</resourceType>
        <cacheConfig>
            <configured>true</configured>
            <enabled>true</enabled>
            <storage>
                <mode>DATA_SOURCE</mode>

<storageDataSourcePath>/shared/examples/ds_orders</storageDataSourcePath>
                <storageTargets>
                    <targetName>result</targetName>
                    <path>/shared/examples/ds_orders/orders_cache</path>
                    <type>TABLE</type>
```

```
                </storageTargets>
            </storage>
            <refresh>
                <mode>MANUAL</mode>
            </refresh>
            <expirationPeriod>
                <period>SECOND</period>
                <count>0</count>
            </expirationPeriod>
            <clearRule>NONE</clearRule>
        </cacheConfig>
    </resourceCache>
</ns2:ResourceCacheModule>
```

Scenario 2 – Clear and Refresh the cache

Description:

Clear and Refresh the cache.

XML Configuration Sample:

Use the Resource Cache XML file that was generated in scenario 1. Copy the "shared1" resourceCache entry and paste it directly below the last </resourceCache> closing element. Modify the id to be "shared2". Remove the entire element <cacheConfig>...</cacheConfig>. It should look like this:

```
    <resourceCache>
        <id>shared2</id>
        <resourcePath>/shared/examples/ds_orders/orders</resourcePath>
        <resourceType>TABLE</resourceType>
    </resourceCache>
```

Execution Sample:

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

Property file setup for UnitTest-ResourceCache.dp:

```
# ----------------------------------------
# Begin task definition list for UNIX:
# ----------------------------------------
PASS   FALSE  ExecuteAction   clearResourceCache      $SERVERID "shared2"
              %MODULE_HOME%/ResourceCacheModule2.xml %MODULE_HOME%/servers.xml
PASS   FALSE  ExecuteAction   refreshResourceCache    $SERVERID "shared2"
              %MODULE_HOME%/ResourceCacheModule2.xml %MODULE_HOME%/servers.xml
```

Results Expected:

The script will report “PASS” for the execution of this action. Open Composite Studio and review the changes for Studio. The cache status should now be “green” and “UP”.

Scenario 3 – Update Cache Configuration (Enable/Disable)

Description:

Enable/Disable the cache for a given resource.

XML Configuration Sample:

Use the Resource Cache XML file that was generated in scenario 1. Copy the “shared1” resourceCache entry and paste it directly below the last </resourceCache> closing element. Modify the id to be “shared3”. Leave the <cacheConfig>...</cacheConfig> element. Remove all elements inside except <enabled>. It should look like this:

```
    <resourceCache>
        <id>shared3</id>
        <resourcePath>/shared/examples/ds_orders/orders</resourcePath>
        <resourceType>TABLE</resourceType>
        <cacheConfig>
            <enabled>false</enabled>
        </cacheConfig>
    </resourceCache>
```

Execution Sample:

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

Property file setup for UnitTest-ResourceCache.dp:

```
# ----------------------------------------
# Begin task definition list for UNIX:
# ----------------------------------------
PASS   FALSE  ExecuteAction   updateResourceCache     $SERVERID "shared3"
              %MODULE_HOME%/ResourceCacheModule2.xml %MODULE_HOME%/servers.xml
```

Results Expected:

The script will report “PASS” for the execution of this action. Open Composite Studio and review the changes for Studio. The cache should now be disabled and the status should now be “gray” and “DISABLED”.

Scenario 4 – Update Cache Configuration (Enable/Disable) for all cached resources in a folder and sub-folders.

Description:

Recursively walk the given folder and sub-folders interrogating resources for a configured cache and “enable” or “disable” the cache for all cached resources found within the boundaries of the starting folder.

XML Configuration Sample:

Use the Resource Cache XML file that was generated in scenario 1. Copy the “shared3” resourceCache entry and paste it directly below the last </resourceCache> closing element. Modify the id to be “shared4”. Modify the resource path to be “/shared/examples” and the resource type to be “CONTAINER”. Leave the <cacheConfig>...</cacheConfig> element. Remove all elements inside except <enabled>. It should look like this:

```
    <resourceCache>
        <id>shared4</id>
        <resourcePath>/shared/examples</resourcePath>
        <resourceType>CONTAINER</resourceType>
        <cacheConfig>
            <enabled>false</enabled>
        </cacheConfig>
    </resourceCache>
```

Execution Sample:

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

Property file setup for UnitTest-ResourceCache.dp:

```
# ----------------------------------------
# Begin task definition list for UNIX:
# ----------------------------------------
PASS   FALSE  ExecuteAction    updateResourceCacheEnabled     $SERVERID
"shared4"           %MODULE_HOME%/ResourceCacheModule2.xml
%MODULE_HOME%/servers.xml
```

Results Expected:

The script will report “PASS” for the execution of this action. Open Composite Studio and review the changes for Studio. The cache should now be disabled and the status should now be “gray” and “DISABLED”.

This method will output the following to the PDTool log:

```
2014-01-27 10:24:35,721 main INFO
[com.cisco.dvbu.ps.deploytool.services.ResourceCacheManagerImpl] - <processing action
ENABLE_DISABLE on resource cache /shared/examples>
2014-01-27 10:24:41,525 main INFO
[com.cisco.dvbu.ps.deploytool.services.ResourceManagerImpl] - <Resource exists? [true]
/shared/examples on server localhost9420>
2014-01-27 10:24:42,587 main INFO
[com.cisco.dvbu.ps.deploytool.services.ResourceCacheManagerImpl] - <Cache operation
(enable)=false  prevStatus=true  currStatus=false  resourceType=TABLE
resourcePath=/shared/examples/ds_orders/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***".

ABOUT COMPOSITE SOFTWARE

Composite Software, Inc. ® is the only company that focuses solely on data virtualization.

Global organizations faced with disparate, complex data environments, including ten of the top 20 banks, six of the top ten pharmaceutical companies, four of the top five energy firms, major media and technology organizations as well as government agencies, have chosen Composite's proven data virtualization platform to fulfill critical information needs, faster with fewer resources.

Scaling from project to enterprise, Composite's middleware enables data federation, data warehouse extension, enterprise data sharing, real-time and cloud computing data integration.

Founded in 2002, Composite Software is a privately held, venture-funded corporation based in Silicon Valley. For more information, please visit www.compositesw.com.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**

Printed in USA CXX-XXXXXX-XX 10/11