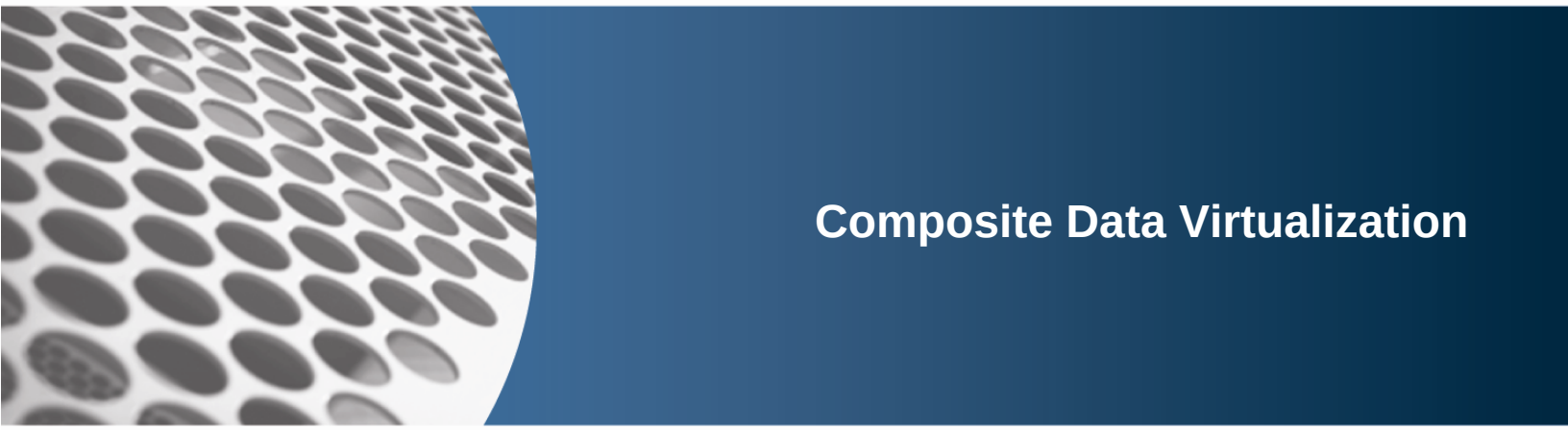
WHITE PAPER

COMPOSITE SOFTWARE



Composite PS Promotion and Deployment Tool

Template Module User Guide

Composite Professional Services

February 2014

TABLE OF CONTENTS

INTRODUCTION 4

Purpose 4
Audience 4

TEMPLATE MODULE DEFINITION 5

Method Definitions and Signatures 5

TEMPLATE MODULE XML CONFIGURATION 6

Description of the Module XML 6
Attributes of Interest 6
Attribute Value Restrictions 6

HOW TO EXECUTE 7

Script Execution 7
Ant Execution 8
Module ID Usage 10

EXAMPLES 11

Scenario 1 – description 11

EXCEPTIONS AND MESSAGES 12

CONCLUSION 13

Concluding Remarks 13
How you can help! 13

DOCUMENT CONTROL

Version History

Version	Date	Author	Description
1.0	6/6/2011	Author	Initial revision for Template Module User Guide
3.0	8/21/2013	Mike Tinius	Updated template to Cisco format.
3.1	2/18/2014	Mike Tinius	Prepare docs for open source.

Related Documents

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

Composite Products Referenced

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

INTRODUCTION

Purpose

The purpose of the Template Module User Guide is to demonstrate how to effectively use the Template Module and execute actions. More goes here…

Audience

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

- Architects
- Developers
- Administrators.
- Operations personnel.

TEMPLATE MODULE DEFINITION

Method Definitions and Signatures

1. **Method1**

   definition

```
@param arg1 - definition
@throws CompositeException


public void method1(String serverId, String moduleIds, String
pathToTemplateXML, String pathToServersXML, String additionalArgs)
throws CompositeException;
```

2. **Method2**

   definition

```
@param serverId target server id from servers XML property file
@param moduleIds list of comma separate module Ids
@param pathToTemplateXML path to the Template xml
@param pathToServersXML path to the server values xml
@throws CompositeException


public void method2(String serverId, String moduleIds, String
pathToTemplateXML, String pathToServersXML, String additionalArgs)
throws CompositeException;
```

General Notes:

The arguments pathToTemplateXML 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.

TEMPLATE MODULE XML CONFIGURATION

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

Description of the Module XML

The TemplateModule XML provides a structure "TemplateModule" for "list of actions" and generating the user XML. The global entry point node is called "TemplateModule" and contains one or more "xyz" nodes.

```
Paste any XML Schema bits here
```

Attributes of Interest

element – element description.

Attribute Value Restrictions

elementWithRestriciton – description. Element is restricted by the following list:

```
this is an example - paste the actual element definition from the schema
(Stylus Studio creates this color scheme)
      <xs:element name="privilege" maxOccurs="unbounded" minOccurs="0">
          <xs:simpleType>
              <xs:restriction base="xs:string">
                  <xs:enumeration value="ACCESS_TOOLS"/>
                  <xs:enumeration value="MODIFY_ALL_CONFIG"/>
                  <xs:enumeration value="MODIFY_ALL_RESOURCES"/>
                  <xs:enumeration value="MODIFY_ALL_STATUS"/>
                  <xs:enumeration value="MODIFY_ALL_USERS"/>
                  <xs:enumeration value="READ_ALL_CONFIG"/>
                  <xs:enumeration value="READ_ALL_RESOURCES"/>
                  <xs:enumeration value="READ_ALL_STATUS"/>
                  <xs:enumeration value="READ_ALL_USERS"/>
                  <xs:enumeration value="UNLOCK_RESOURCE"/>
              </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 TemplateModule.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-Template.dp

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

Properties File (UnitTest-Template.dp):

Property File Rules:

```
# ----------------------------
# UnitTest-Template.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. $MODULE_HOME) gets resolved on
invocation CisDeployTool.
#                       Paths containing spaces must be enclosed in double quotes:
#                              ANSWER: "$MODULE_HOME/LabVCSModule.xml"
#                       Given that MODULE_HOME=C:/dev/Cis Deploy
Tool/resources/modules, CisDeployTool 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:
# ----------------------------------------

Place your properties here
```

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-Template.xml

Unix: ./ExecutePDTool.sh -ant ../resources/ant/build-Template.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 -->
  Place your property names here:
  <property name="moduleIds"                 value="id1,id2"/>
  <property name="pathToGenTemplateXML"      value="${MODULE_HOME}/getTemplateModule.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 -->
Place actions to execute here:

    <!-- Windows or UNIX: Entire list of actions
Place all your actions here:
    -->
```

```
  </target>
</project>
```

Module ID Usage

The following explanation provides a general pattern for module identifiers. The module identifier for this module is "moduleIds".

[Note: In this section replace updateDataSources with your primary update action example in the command-line and Ant sections below:]

- Possible values for the module identifier:
- 1. ***Inclusion List*** - CSV string like "id1,id2"
    - CisDeployTool will process only the passed in identifiers in the specified module XML file.

  Example command-line property file

```
PASS   FALSE  ExecuteAction       updateDataSources $SERVERID "ds1,ds2"
       "$MODULE_HOME/DataSourceModule.xml" "$MODULE_HOME/servers.xml"
```

  Example Ant build file

```
<executeJavaAction description="Update"        action="updateDataSources"
arguments="${SERVERID}^ds1,ds2^${pathToDataSourcesXML}^${pathToServersXML}"
```

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

  Example command-line property file

```
PASS   FALSE  ExecuteAction       updateDataSources $SERVERID "*"
       "$MODULE_HOME/DataSourceModule.xml" "$MODULE_HOME/servers.xml"
```

  Example Ant build file

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

- 3. ***Exclusion List*** - CSV string with '-' or whatever is configured to indicate exclude resources as prefix like "-id1,id2"
    - CisDeployTool 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       updateDataSources $SERVERID "-ds3,ds4"
       "$MODULE_HOME/DataSourceModule.xml" "$MODULE_HOME/servers.xml"
```

  Example Ant build file

```
<executeJavaAction description="Update"        action="updateDataSources"
arguments="${SERVERID}^-ds3,ds3^${pathToDataSourcesXML}^${pathToServersXML}"
```

EXAMPLES

The following are common scenarios when using the TemplateModule.

[Note: Add scenario blocks as needed by copying and pasting]

Scenario 1 – description

Description:

Description of scenario.

XML Configuration Sample:

Description or Not applicable for this example.

```
TemplateModule XML excerpt goes in here
```

Execution Sample:

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

Property file setup for UnitTest-Template.dp:

```
# ----------------------------------------
# Begin task definition list:
# ----------------------------------------
# Action
Place single line of execution here:
```

Results Expected:

Description of what to expect.

```
XML goes in here if applicable
```

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