PDTool Trigger 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	8/16/2011	Kevin O'Brien	Initial revision for Trigger Module User Guide
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	11/17/2014	Mike Tinius	Updated license.
3.4	3/4/2015	Mike Tinius	Updated table of contents to include methods and updated docs to Cisco format.
4.0	12/14/2017	Mike Tinius	Initial revision with Tibco

Related Documents

Name	Author
PDTool User's Guide.pdf	Mike Tinius

Supported Versions

Name	Version
TIBCO® Data Virtualization	7.0.4 or later

Table of Contents

1 Introduction

Purpose

The purpose of the Trigger Module User Guide is to demonstrate how to effectively use the Trigger Module and execute actions. Once Data Virtualization (DV) resources are imported into a target DV server during the deployment process, it is sometimes desirable to update the trigger configuration. The Trigger Module provides the mechanism to automate the configuration of trigger attributes. Using the "updateTriggers" method and pointing at an identifier within the TriggerModule.xml configuration file, the user of this tool will be able to execute a command-line or Ant script that will automatically connect to the target DV server and perform the update. Additionally, the user can automate enabling and disabling of a trigger on the target DV server. Finally, to make it easier on the developer or administrator who is configuring the deployment plan, they would use the "generateTriggersXML" to reach into the source server where the artifacts are being deployed from and generate the TriggerModule XML. Then, all they need to do is tweak a few configuration lines based on what the values are for the target DV server. The TriggerModule XML becomes part of the deployment plan that they will use during the automated deployment process.

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 Trigger Module Definition

Method Definitions and Signatures

1. updateTriggers

Update Trigger method updates trigger configurations based on the values identified by the trigger id in the TriggerXML and the target server.

```
@param serverId target server id from servers config xml
@param triggerIds list of comma separated trigger Ids
@param pathToTriggersXML path to the trigger xml
@param pathToServersXML path to the server values xml
@throws CompositeException


public void updateTriggers(String serverId, String triggerIds, String
pathToTriggersXML, String pathToServersXML) throws CompositeException;
```

2. enableTriggers

Enable Trigger method enables or disables triggers based on the setting identified by the trigger id in the TriggerXML and the target server.

```
@param serverId target server id from servers config xml
@param triggerIds list of comma separated trigger Ids
@param pathToTriggersXML path to the trigger xml
@param pathToServersXML path to the server values xml
@throws CompositeException


public void enableTriggers(String serverId, String triggerIds, String
pathToTriggersXML, String pathToServersXML) throws CompositeException;
```

3. generateTriggersXML

Generate the TriggerXML based on the starting path passed in and the target server information. Generate the XML to the file location passed in.

```
@param serverId target server id from servers config xml
@param startPath starting path of the resource e.g /shared
@param pathToTriggersXML path including name to the trigger xml which
needs to be created
@param pathToServersXML path to the server values xml
@throws CompositeException
```

```
generateTriggersXML(String serverId, String startPath, String
pathToTriggersXML, String pathToServersXML) throws CompositeException;
```

General Notes:

The arguments pathToTriggersXML and pathToServersXML are located in [PDTool/resources/modules]. The value passed into the methods is the fully qualified path. The paths get resolved when executing the property file and evaluating the $MODULE_HOME variable.

3 Trigger 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 TriggerModule XML provides a structure “TriggerModule” for “update, enable/disable and generateTriggerXML”. The global entry point node is called “TriggerModule” and contains a collection of “triggerList” and “scheduleList” nodes. The “triggerList” contains a collection of “trigger” nodes. Each “trigger” node has a “condition” and “action”. Where the “condition” is a timer event, a reference is made to a “schedule” node within “scheduleList”.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:TriggerModule xmlns:ns2="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <triggerList>

<!-- Example of trigger with timerEvent condition with executeProcedure action
-->

        <trigger>
            <id>TR-1</id>
            <resourcePath>/shared/test1/test 1/trigger1</resourcePath>
            <isEnabled>false</isEnabled>
            <maxEventsQueued>1</maxEventsQueued>
            <annotation>trigger 1 annotation</annotation>
            <condition>
                <timerEvent>
                    <scheduleId>TR-1-SCH-1</scheduleId>
                </timerEvent>
            </condition>
            <action>
                <executeProcedure>
                    <resourcePath>/shared/test1/test 1/proc 1</resourcePath>
                    <parameterValues>13421234.34,23243,2011-02-
02,8.710345879797E8,8098709.23454,2011-02-02 12:10:59.102,'this, is a different
string'</parameterValues>
                </executeProcedure>
            </action>
        </trigger>
    </triggerList>
    <scheduleList>
        <schedule>
            <scheduleId>TR-1-SCH-1</scheduleId>
            <mode>NONE</mode>
            <fromTimeInADay>-1</fromTimeInADay>
            <endTimeInADay>-1</endTimeInADay>
            <recurringDay>-1</recurringDay>
            <isCluster>true</isCluster>
        </schedule>
    </scheduleList>
```

```
</ns2:TriggerModule>
```

Attributes of Interest

id – The unique identifier within the TriggerModule.xml file that is used to identify a trigger resource configuration.

resourcePath – the DV path to a resource.

isEnabled – determines whether the trigger is enabled or disabled.

maxEventsQueued – number of events to store in queue.

condition – the condition which causes the trigger to fire. This is described by child nodes “timerEvent”, “jmsEvent”, “userDefinedEvent” or “systemEvent”.

action – the action to perform once the condition has fired. This is described by child nodes “executeProcedure”, “sendEmail”, “reintrospectDatasource”, or “gatherStatistics”.

scheduleId – when the condition type is “timerEvent”, this links to the specific “scheduleId” of a “schedule” configured in “scheduleList”.

mode – determines whether the schedule fires on a repeating basis. “NONE” means no repeating. “PERIODIC” means repeating information is provided in “startTime”, “period”, “count”, “fromTimeInADay”, “endTimeInADay” and “recurringDay”.

fromTimeInADay – the starting time from which timer firings are honored. Specified as the number of minutes in a 24hr period e.g. 1380 = 11pm. -1 means no recurrence restriction.

endTimeInADay – the ending after which timer firings are ignored. Specified as the number of minutes in a 24hr period e.g. 1380 = 11pm. -1 means no recurrence restriction.

recurringDay – any specific days on which timer firings should be honored. Works in conjunction with “fromTimeInADay” and “endTimeInADay”. Sunday = 1. Monday = 2. Tuesday = 4. Wednesday = 8. Thursday = 16. Friday = 32. Saturday = 64. Supply the value as a combination of days e.g. Tuesday + Wednesday = 12.

isCluster – determines whether this schedule timer should fire once per cluster.

Attribute Value Restrictions

TriggerConditionSystemEventValidationList – Defines the list of valid values for condition type SystemEvent. Element is restricted by the following list:

```
<xs:simpleType name="TriggerConditionSystemEventValidationList">
      <xs:annotation>
            <xs:documentation xml:lang="en">
                  Validation List for Trigger Condition System Events
```

```
                </xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:string">
                <xs:enumeration value="CacheRefreshFailure"/>
                <xs:enumeration value="CacheRefreshSuccess"/>
                <xs:enumeration value="DataSourceDown"/>
                <xs:enumeration value="DataSourceUp"/>
                <xs:enumeration value="RequestFailure"/>
                <xs:enumeration value="RequestInactive"/>
                <xs:enumeration value="RequestRunForTooLong"/>
                <xs:enumeration value="ResourceLock"/>
                <xs:enumeration value="ResourceUnlock"/>
                <xs:enumeration value="RequestsSpike"/>
                <xs:enumeration value="ErrorsSpike"/>
                <xs:enumeration value="FailedLoginSpike"/>
                <xs:enumeration value="StatisticsGatheringFailure"/>
                <xs:enumeration value="ServerStart"/>
                <xs:enumeration value="ServerStop"/>
                <xs:enumeration value="TransactionFailure"/>
            </xs:restriction>
        </xs:simpleType>
```

TriggerModeValidationList – defines the recurring behavior of a schedule associated with a TimerEvent condition. Element is restricted by the following list:

```
        <xs:simpleType name="TriggerModeValidationList">
            <xs:annotation>
                <xs:documentation xml:lang="en">
                    NONE = Exactly Once in DV.  PERIODIC = Periodic in DV
[encompasses INTERVAL and CALENDAR]
                </xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:string">
            <xs:enumeration value="NONE"/>
            <xs:enumeration value="PERIODIC"/>
            </xs:restriction>
        </xs:simpleType>
```

TriggerPeriodValidationList – defines the granularity of recurrence of a schedule associated with a TimerEvent condition. Element is restricted by the following list:

```
        <xs:simpleType name="TriggerPeriodValidationList">
            <xs:annotation>
                <xs:documentation xml:lang="en">
                    MINUTE equates to INTERVAL.  HOUR through YEAR equates to
CALENDAR
                </xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:string">
                <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>
```

TriggerConditionTypeValidationList – defines the set of valid trigger condition types. Element is restricted by the following list:

```
	<xs:simpleType name="TriggerConditionTypeValidationList">
		<xs:annotation>
			<xs:documentation xml:lang="en">
				Trigger Condition Type Validation List:  An action can be 1
and only 1 of [SYSTEM_EVENT,USER_DEFINED,JMS,TIMER]
			</xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:string">
			<xs:enumeration value="SYSTEM_EVENT"/>
			<xs:enumeration value="USER_DEFINED"/>
			<xs:enumeration value="JMS"/>
			<xs:enumeration value="TIMER"/>
		</xs:restriction>
	</xs:simpleType>
```

TriggerActionTypeValidationList – defines the set of valid trigger action types. Element is restricted by the following list:

```
	<xs:simpleType name="TriggerActionTypeValidationList">
		<xs:annotation>
			<xs:documentation xml:lang="en">
				Trigger Action Type Validation List:  An action can be 1 and
only 1 of [PROCEDURE, STATISTICS, REINTROSPECT, EMAIL]
			</xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:string">
			<xs:enumeration value="PROCEDURE"/>
			<xs:enumeration value="STATISTICS"/>
			<xs:enumeration value="REINTROSPECT"/>
			<xs:enumeration value="EMAIL"/>
		</xs:restriction>
	</xs:simpleType>
```

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 TriggerModule.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-Trigger.dp

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

Properties File (UnitTest-Trigger.dp):

Property File Rules:

```
# ----------------------------
# UnitTest-Trigger.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 distinguising 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,
PDToolautomatically 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 preceeding 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:
# ----------------------------------------
# Quick Test
#PASS  FALSE  ExecuteAction generateTriggersXML $SERVERID "/shared/test1"
      "$MODULE_HOME/getTriggerModule.xml"  "$MODULE_HOME/servers.xml"
#
#PASS  FALSE  ExecuteAction   enableTriggers   $SERVERID "TR-1"
      "$MODULE_HOME/TriggerModule.xml"  "$MODULE_HOME/servers.xml"
#
#PASS  FALSE  ExecuteAction   updateTriggers   $SERVERID "TR-1,TR-2,TR-3,TR-4"
      "$MODULE_HOME/TriggerModule.xml"  "$MODULE_HOME/servers.xml"
#
#PASS  FALSE  ExecuteAction   enableTriggers   $SERVERID "TR-5"
      "$MODULE_HOME/TriggerModule.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-Trigger.xml
```

Unix: ./ExecutePDTool.sh -ant ../resources/ant/build-Trigger.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="triggerIds" value="TR-1"/>
	<property name="pathToGenTriggerXML" value="${MODULE_HOME}/getTriggerModule.xml"/>

	<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 -->
        <executeJavaAction description="Generate" action="generateTriggersXML"
        arguments="${SERVERID}^/shared/test00^${pathToGenTriggerXML}^${pathToServersXML}"
        endExecutionOnTaskFailure="TRUE"/>

        <!-- Windows or UNIX: Entire list of actions
        <executeJavaAction description="Generate" action="generateTriggersXML"
        arguments="${SERVERID}^/shared/test00^${pathToGenTriggerXML}^${pathToServersXML}"
        endExecutionOnTaskFailure="TRUE"/>

        <executeJavaAction description="Enable" action="enableTriggers"
arguments="${SERVERID}^${triggerIds}^${pathToTriggerXML}^${pathToServersXML}"
        endExecutionOnTaskFailure="TRUE"/>

        <executeJavaAction description="Update" action="updateTriggers"
arguments="${SERVERID}^${triggerIds}^${pathToTriggerXML}^${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 “triggerIds”.

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

    Example command-line property file

    ```
    PASS   FALSE  ExecuteAction       updateTriggers $SERVERID "tr1,tr2"
           "$MODULE_HOME/TriggerModule.xml" "$MODULE_HOME/servers.xml"
    ```

    Example Ant build file

    ```
    <executeJavaAction description="Update"        action="updateTriggers"
    arguments="${SERVERID}^tr1,tr2^${pathToTriggersXML}^${pathToServersXML}"
    ```

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

    Example command-line property file

    ```
    PASS   FALSE  ExecuteAction       updateTriggers $SERVERID "*"
           "$MODULE_HOME/TriggerModule.xml" "$MODULE_HOME/servers.xml"
    ```

    Example Ant build file

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

- 3. ***Exclusion List*** - CSV string with '-' or whatever is configured to indicate exclude resources as prefix like “-id1,id2”

- PDToolwill ignore passed in resources and process the rest of the identifiers in the module XML file.

Example command-line property file

```
PASS   FALSE  ExecuteAction       updateTriggers $SERVERID "-tr3,tr4"
       "$MODULE_HOME/TriggerModule.xml" "$MODULE_HOME/servers.xml"
```

Example Ant build file

```
<executeJavaAction description="Update"        action="updateTriggers"
arguments="${SERVERID}^-tr3,tr4^${pathToTriggersXML}^${pathToServersXML}"
```

5 PDTool Examples

The following are common scenarios when using the TriggerModule.

Scenario 1 – Generate Trigger Module XML

Description:

Generate the Trigger Module XML configuration file for a specific DV project folder. This is useful for a developer/administrator to get the initial configuration from the development server. Once the TriggerModule.xml file is generated, the developer or administrator can tweak the parameters for the target DV server. This new file will be part of the deployment plan of the target DV server.

Execution Sample:

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

```
# ----------------------------------------
# Begin task definition list:
# ----------------------------------------
# Generate the list of Triggers
PASS FALSE ExecuteAction generateTriggersXML $SERVERID /shared/test00
    "$MODULE_HOME/getTriggerModule.xml" "$MODULE_HOME/servers.xml"
```

Results Expected:

PDTool executed and generated the getTriggerModule.xml file in the PDTool/resources/modules directory. An example of the output can be seen in the section "**Description of the Module XML**".

Next the developer or administrator would rename this file and edit the property, attributes values to align with values used on the target DV server.

Finally the user would execute scenario 2 below to update the trigger attributes.

Scenario 2 – Update Triggers

Description:

Update the triggers on the target DV server using the generated Trigger Module XML configuration file. This provides the administrator with a way to automate the deployment process and affect change on the target DV server.

XML Configuration Sample:

This is an example of a TriggerModule.xml configuration for a trigger with a jmsEvent condition and a reintrospectDatasource action.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:TriggerModule xmlns:ns2="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <triggerList>
        <trigger>
            <id>TR-3</id>
            <resourcePath>/shared/test/ResourceTrigger/trigger3</resourcePath>
            <isEnabled>true</isEnabled>
            <maxEventsQueued>1</maxEventsQueued>
            <annotation>trigger 3 annotation</annotation>
            <condition>
                <jmsEvent>
                    <connector>JmsConnector</connector>
                    <destination>JmsDestination</destination>
                    <selector>JmsSelector</selector>
                </jmsEvent>
            </condition>
            <action>
                <reintrospectDatasource>

<resourcePath>/shared/test1/data_sources/ds_orders</resourcePath>
                    <emailTo>to@compositesw.com</emailTo>
                    <emailCC></emailCC>
                    <emailBCC></emailBCC>
                    <emailReplyTo></emailReplyTo>
                    <emailSubject></emailSubject>
                    <emailBody></emailBody>
                    <skipIfNoResults>false</skipIfNoResults>
                    <noCommit>true</noCommit>
                </reintrospectDatasource>
            </action>
        </trigger>
    </triggerList>
</ns2:TriggerModule>
```

Execution Sample:

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

```
# ----------------------------------------
# Begin task definition list:
# ----------------------------------------
# Update trigger
PASS FALSE ExecuteAction updateTriggers $SERVERID "TR-3"
      "$MODULE_HOME/TriggerModule.xml" "$MODULE_HOME/servers.xml"
```

Results Expected:

PDTool executed the “updateTriggers” for the trigger identified by “TR-3” and the target DV server identified by “test9400”.

Scenario 3 – Enable Triggers

Description:
Enable or disable the triggers on the target DV server using the generated Trigger Module XML configuration file. This action only utilizes the isEnabled element. No other trigger attributes are modified by this action. This provides the administrator with a way to automate the deployment process and affect change on the target DV server.

XML Configuration Sample:

This is an example of a TriggerModule.xml configuration for a trigger with a timerEvent condition (with corresponding schedule described in scheduleList) and a sendEmail action.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:TriggerModule xmlns:ns2="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <triggerList>
        <trigger>
            <id>TR-2</id>
            <resourcePath>/shared/test/ResourceTrigger/trigger2</resourcePath>
            <isEnabled>false</isEnabled>
            <maxEventsQueued>2</maxEventsQueued>
            <annotation>trigger 2 annotation</annotation>
            <condition>
                <timerEvent>
                    <scheduleId>TR-2-SCH-2</scheduleId>
                </timerEvent>
            </condition>
            <action>
                <sendEmail>
                    <resourcePath>/shared/test/ResourceTrigger/proc
2</resourcePath>
                    <parameterValues></parameterValues>
                    <emailTo>mtinius@compositesw.com</emailTo>
                    <emailCC></emailCC>
                    <emailBCC></emailBCC>
                    <emailReplyTo></emailReplyTo>
                    <emailSubject></emailSubject>
                    <emailBody></emailBody>
                    <skipIfNoResults>true</skipIfNoResults>
                    <includeSummary>false</includeSummary>
                </sendEmail>
            </action>
        </trigger>
    </triggerList>
```

```
    <scheduleList>
        <schedule>
            <scheduleId>TR-2-SCH-2</scheduleId>
            <mode>PERIODIC</mode>
            <startTime>2011-06-01T15:55:37.000Z</startTime>
            <period>HOUR</period>
            <count>1</count>
            <fromTimeInADay>1200</fromTimeInADay>
            <endTimeInADay>1400</endTimeInADay>
            <recurringDay>2</recurringDay>
            <isCluster>true</isCluster>
        </schedule>
    </scheduleList>
</ns2:TriggerModule>
```

Execution Sample:

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

```
# ----------------------------------------
# Begin task definition list:
# ----------------------------------------
# Update trigger enable flag
PASS FALSE ExecuteAction enableTriggers $SERVERID "TR-2"
      "$MODULE_HOME/TriggerModule.xml" "$MODULE_HOME/servers.xml"
```

Results Expected:

PDTool executed the “enableTriggers” for the trigger identified by “TR-2” and the target DV server identified by “test9400”. The outcome of this action is that the trigger is disabled. No other trigger attributes are changed.

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.

Error processing trigger id: <id>. JMS event condition property 'Connector' is null:

This may occur when you do not supply a value for the Connector property for a trigger condition type of jmsEvent.

Error processing trigger id: <id>. JMS event condition property 'Destination' is null:

This may occur when you do not supply a value for the Destination property for a trigger condition type of jmsEvent.

Error processing trigger id: <id>. System event condition property 'System Event Name' is empty

This may occur when you do not supply a value for the System Event Name property for a trigger condition type of systemEvent.

Error processing trigger id: <id>. System event condition property 'System Event Name' value '<value>' is invalid

This may occur when the value supplied for the System Event Name property for a trigger condition type of systemEvent is not one of the known values. Please see the definition of TriggerConditionSystemEventValidationList for the list of valid values.

Error processing trigger id: <id>. User defined event condition property 'User Defined Event Name' is empty

This may occur when you do not supply a value for the User Defined Event Name property for a trigger condition type of userDefinedEvent.

Error processing trigger. Referenced ScheduleId <id> is missing from TriggerModule.xml

This may occur when you reference a schedule id in a timerEvent condition, but that schedule id does not appear in scheduleList section of the trigger module XML file.

Error processing trigger schedule id: <id>. Schedule condition property 'Mode' is empty

This may occur when a defined schedule (corresponding with a trigger condition of type timerEvent) has an empty value for Mode. Please see the definition of TriggerModeValidationList for the list of valid values.

Error processing trigger schedule id: <id>. Schedule condition property 'Mode' value '<value>' is invalid

This may occur when a defined schedule (corresponding with a trigger condition of type timerEvent) has an invalid value for Mode. Please see the definition of TriggerModeValidationList for the list of valid values.

Error processing trigger schedule id: <id>. Schedule condition property 'Mode' has value 'PERIODIC' but schedule condition property 'count' is missing

This may occur when a defined schedule (corresponding with a trigger condition of type timerEvent) has an invalid value for count. Count determines the frequency of repetition of the timer schedule.

Error processing trigger schedule id: <id>. Schedule condition property 'Mode' has value 'PERIODIC' but schedule condition property 'count' is less than zero: <countValue>

This may occur when a defined schedule (corresponding with a trigger condition of type timerEvent) has a negative value for count. Count determines the frequency of repetition of the timer schedule and must be a positive value.

Error processing trigger schedule id: <id>. Schedule condition property 'Count' has value <countValue> which exceeds maximum allowed value of <Integer.MAX_VALUE>

This may occur when a defined schedule (corresponding with a trigger condition of type timerEvent) has too large a value for count. Integer.MAX_VALUE is platform specific for a given Java Virtual Machine. Count determines the frequency of repetition of the timer schedule.

Error processing trigger schedule id: <id>. Schedule condition property 'Period' has an invalid value: <periodValue>

This may occur when a defined schedule (corresponding with a trigger condition of type timerEvent) has an invalid value for period. Please see the definition of TriggerPeriodValidationList for the list of valid values.

Error processing trigger schedule id: <id>. Schedule condition property 'Recurring Day' value '<recurringDayValue>' is invalid. Valid values are between 1 and 127.

This may occur when a defined schedule (corresponding with a trigger condition of type timerEvent) has an invalid value for recurringDay. recurringDay must have a value between 1 and 127.

Error processing trigger schedule id: <id>. Schedule condition property 'FromTimeInADay' must have a value between 0 and 1440. Current value is: <value>

This may occur when a defined schedule (corresponding with a trigger condition of type timerEvent) has an invalid value for fromTimeInADay. fromTimeInADay is specified in minutes and must have a value between 0 and 1440.

Error processing trigger schedule id: <id>. Schedule condition property 'EndTimeInADay' must have a value between 0 and 1440. Current value is: <value>

This may occur when a defined schedule (corresponding with a trigger condition of type timerEvent) has an invalid value for endTimeInADay. endTimeInADay is specified in minutes and must have a value between 0 and 1440.

Error processing trigger schedule id: <id>. Schedule condition property 'FromTimeInADay' must have a value between 0 and 1440. Current value is: <value>

This may occur when a defined schedule (corresponding with a trigger condition of type timerEvent) has an invalid value for fromTimeInADay. fromTimeInADay is specified in minutes and must have a value between 0 and 1440.

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***".