



PDTool User 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	6/10/2011	Mike Tinius	Initial revision for User Module User Guide
1.0.1	8/1/2011	Mike Tinius	Revisions due to architecture changes
1.1	3/9/2012	Mike Tinius	Explained Generate User some more
1.2	10/1/2012	Mike Tinius	Fixed doc issue with privilege list
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
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	4
	Purpose	4
	Audience.....	4
	References	4
2	User Module Definition	5
	Method Definitions and Signatures	5
	1. createOrUpdateUsers.....	5
	2. deleteUsers	5
	3. generateUsersXML.....	5
3	User Module XML Configuration	7
	Description of the Module XML	7
	Attributes of Interest	8
	Attribute Value Restrictions	8
4	How To Execute	9
	Script Execution	9
	Ant Execution	10
	Module ID Usage.....	12
5	PDTool Examples.....	13
	Scenario 1 – Generate User XML	13
	Scenario 2 – delete a user	14
	Scenario 3 – create or update a user	15
6	Exceptions and Messages	17
7	Conclusion	18
	Concluding Remarks	18
	How you can help!	18

1 Introduction

Purpose

The purpose of the User Module User Guide is to demonstrate how to effectively use the User Module and execute actions. Users are managed within the browser-based Data Virtualization (DV) Manager. The User Module will allow the automation of creating, updating, deleting users as well as generating the User Module XML property file.

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 User Module Definition

Method Definitions and Signatures

1. createOrUpdateUsers

Create or update a CIS user based on the list of user Ids passed into this method. The method will check to see if a user exists. If it does not exist, the createUsers() is invoked otherwise updateUsers() is invoked. Whatever attributes are set in the XML will be updated for that user.

```
@param serverId - target server name
@param userIds - comma separated list of user names to create or update.
@param pathToUsersXML - path including name to the users XML containing a
list of UserIds to execute against
@param pathToServersXML - path to the server values XML
@return void
@throws CompositeException

public void createOrUpdateUsers(String serverId, String userIds, String
pathToUsersXML, String pathToServersXML) throws CompositeException;
```

2. deleteUsers

Delete a user from the CIS server instance. If the user does not exist, no action is taken and no error is thrown.

```
@param serverId - target server name
@param userIds - comma separated list of user names to be deleted.
@param pathToUsersXML - path including name to the users XML containing a
list of UserIds to execute against
@param pathToServersXML - path to the server values XML
@return void
@throws CompositeException

public void deleteUsers(String serverId, String userIds, String
pathToUsersXML, String pathToServersXML) throws CompositeException;
```

3. generateUsersXML

Generate the UserModule XML property file based on a domain and the CIS server instance referenced. Unlike other modules that provide a starting path, this module takes

in a domain name (i.e. composite). The method generateUsersXML does not export or generate the user name from DV.

```
@param serverId - target server name
@param doaminName - domain name from which to get a list of users. If
null get all users from all domains.
@param pathToUsersXML - path including name to the users XML which will
get created
@param pathToServersXML - path to the server values XML
@return void
@throws CompositeException

public void generateUsersXML(String serverId, String doaminName, String
pathToUsersXML, String pathToServersXML) throws CompositeException;
```

General Notes:

The arguments pathToUsersXML 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 User 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 UserModule XML provides a structure “user” for creating, updating, deleting and generating the user XML. The global entry point node is called “UserModule” and contains one or more “user” nodes.

```
<?xml version="1.0"?>
<p1:UserModule xmlns:p1="http://www.dvbu.cisco.com/ps/deploytool/modules">
  <user>
    <id>user1</id>
    <userName>user1</userName>
    <encryptedPassword>password1</encryptedPassword>
    <forcePassword>true</forcePassword>
    <domainName>composite</domainName>
    <groupMembershipList>
      <groupName>all</groupName>
      <groupDomain>composite</groupDomain>
    </groupMembershipList>
    <groupMembershipList>
      <groupName>group1</groupName>
      <groupDomain>composite</groupDomain>
    </groupMembershipList>
    <groupMembershipList>
      <groupName>group2</groupName>
      <groupDomain>composite</groupDomain>
    </groupMembershipList>
    <privilege>ACCESS_TOOLS</privilege>
    <annotation>user1</annotation>
  </user>
  <user>
    <id>user2</id>
    <userName>user2</userName>
    <encryptedPassword>Encrypted:7F6324FFD300BE8F</encryptedPassword>
    <forcePassword>false</forcePassword>
    <domainName>composite</domainName>
    <groupMembershipList>
      <groupName>all</groupName>
      <groupDomain>composite</groupDomain>
    </groupMembershipList>
    <groupMembershipList>
      <groupName>group2</groupName>
      <groupDomain>composite</groupDomain>
    </groupMembershipList>
    <privilege>ACCESS_TOOLS MODIFY_ALL_CONFIG MODIFY_ALL_RESOURCES MODIFY_ALL_STATUS
MODIFY_ALL_USERS READ_ALL_CONFIG READ_ALL_RESOURCES READ_ALL_STATUS READ_ALL_USERS
UNLOCK_RESOURCE</privilege>
  </user>
</p1:UserModule>
```

Attributes of Interest

username – this value is the actual server attribute path and is unique across all server attribute configurations. The same path describes the server attribute and the server attribute definition.

encryptedPassword – may contain a clear text or encrypted password. If clear text, the User Module will encrypt the password prior to creating or updating a user. Optionally, the user may execute the Deploy Tool script “ExecuteEncrypt.bat or .sh” to encrypt all “encryptedPassword” values in the specified file. E.g. ExecuteEncrypt.bat /path/to/file.xml

It should be noted that “generatUserXML()” does not generate the password or export the password from DV.

forcePassword – this Boolean (true or false) value is used during update of a user. If the user exists and forcePassword=true, then the encryptedPassword value is passed into the updateUser method. If the user does not exist, then this value is ignored.

domainName – this value is tells the system which “valid” domain the user belongs to.

groupMembershipList – this is a list of group name/domain name pairs. If the user does not exist, then the user is assigned to whatever groups are in the list. If the user does exist, then the user is updated with whatever values are in the list.

annotation – this value is provides a description (annotation) about the user.

Attribute Value Restrictions

privilege – A space separated list of Privilege Access Rights that may include 1 or more of [ACCESS_TOOLS MODIFY_ALL_CONFIG MODIFY_ALL_RESOURCES MODIFY_ALL_STATUS MODIFY_ALL_USERS READ_ALL_CONFIG READ_ALL_RESOURCES READ_ALL_STATUS READ_ALL_USERS UNLOCK_RESOURCE]

Schema validation uses the following set:

```
<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>
```


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 UserModule.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/properties/UnitTest-User.dp

Unix: ./ExecutePDTool.sh -exec ../resources/properties/UnitTest-User.dp

Properties File (UnitTest-User.properties):

Property File Rules:

```
# -----
# UnitTest-User.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:
# -----
# Generate
PASS  FALSE  ExecuteAction generateUsersXML          $SERVERID "composite"
$MODULE_HOME/getUserModule.xml $MODULE_HOME/servers.xml
# Delete
PASS  FALSE  ExecuteAction  deleteUsers              $SERVERID "user1"
$MODULE_HOME/UserModule.xml $MODULE_HOME/servers.xml
# Create or Update
PASS  FALSE  ExecuteAction  createOrUpdateUsers      $SERVERID "user1"
$MODULE_HOME/UserModule.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 -exec ../resources/ant/build-User.xml

Unix: ./ExecutePDTool.sh -exec ../resources/ant/build-User.xml

Build File:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="PDTool" default="default" basedir=".">

    <description>description</description>
```

```

<!-- Default properties -->
<property name="SERVERID"                value="localhost"/>
<property name="noarguments"              value="&quot;&quot;"/>

<!-- Default Path properties -->
<property name="RESOURCE_HOME"            value="${PROJECT_HOME}/resources"/>
<property name="MODULE_HOME"              value="${RESOURCE_HOME}/modules"/>
<property name="pathToServersXML"          value="${MODULE_HOME}/servers.xml"/>
<property name="pathToArchiveXML"          value="${MODULE_HOME}/ArchiveModule.xml"/>
<property name="pathToDataSourcesXML"      value="${MODULE_HOME}/DataSourceModule.xml"/>
<property name="pathToGroupsXML"           value="${MODULE_HOME}/GroupModule.xml"/>
<property name="pathToPrivilegeXML"        value="${MODULE_HOME}/PrivilegeModule.xml"/>
<property name="pathToRebindXML"           value="${MODULE_HOME}/RebindModule.xml"/>
<property name="pathToRegressionXML"       value="${MODULE_HOME}/RegressionModule.xml"/>
<property name="pathToResourceXML"         value="${MODULE_HOME}/ResourceModule.xml"/>
<property name="pathToResourceCacheXML"    value="${MODULE_HOME}/ResourceCacheModule.xml"/>
<property name="pathToServerAttributeXML"  value="${MODULE_HOME}/ServerAttributeModule.xml"/>
<property name="pathToTriggerXML"          value="${MODULE_HOME}/TriggerModule.xml"/>
<property name="pathToUsersXML"            value="${MODULE_HOME}/UserModule.xml"/>
<property name="pathToVCSModuleXML"        value="${MODULE_HOME}/VCSModule.xml"/>

<!-- Custom properties -->
<property name="userIds"                  value="user1,user2"/>
<property name="pathToGenUserXML"          value="${MODULE_HOME}/getUserModule.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.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="CreateOrUpdate" action="createOrUpdateUsers"
    arguments="${SERVERID}^${userIds}^${pathToUsersXML}^${pathToServersXML}"
    endExecutionOnTaskFailure="TRUE" />

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

<executeJavaAction description="CreateOrUpdate" action="createOrUpdateUsers"
    arguments="${SERVERID}^${userIds}^${pathToUsersXML}^${pathToServersXML}"
    endExecutionOnTaskFailure="TRUE" />

```

```

<executeJavaAction description="Delete"          action="deleteUsers"
  arguments="${SERVERID}^user1^${pathToUsersXML}^${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 “userIds”.

- 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 createOrUpdateUsers $SERVERID "user1,user2"
$MODULE_HOME/UserModule.xml $MODULE_HOME/servers.xml

```

Example Ant build file

```

<executeJavaAction description="CreateOrUpdate" action="createOrUpdateUsers"
  arguments="${SERVERID}^user1,user2^${pathToUsersXML}^${pathToServersXML}"
  endExecutionOnTaskFailure="TRUE" />

```

- 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 createOrUpdateUsers $SERVERID "*"
$MODULE_HOME/UserModule.xml $MODULE_HOME/servers.xml

```

Example Ant build file

```

<executeJavaAction description="CreateOrUpdate" action="createOrUpdateUsers"
  arguments="${SERVERID}^*^${pathToUsersXML}^${pathToServersXML}"
  endExecutionOnTaskFailure="TRUE" />

```

- 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 createOrUpdateUsers $SERVERID "-user1,user2"
$MODULE_HOME/UserModule.xml $MODULE_HOME/servers.xml

```

Example Ant build file

```

<executeJavaAction description="CreateOrUpdate" action="createOrUpdateUsers"
  arguments="${SERVERID}^~user1,user2^${pathToUsersXML}^${pathToServersXML}"
  endExecutionOnTaskFailure="TRUE" />

```

5 PDTool Examples

The following are common scenarios when using the User Module.

Scenario 1 – Generate User XML

Description:

Generate the user xml property file based on the domain “composite”.

XML Configuration Sample:

Not applicable for this example.

Execution Sample:

Unix: `./ExecutePDTool.sh -exec ../resources/properties/UnitTest-User.dp`
 Property file setup for UnitTest-User.properties:

```
# -----
# Begin task definition list for UNIX:
# -----
# Generate
PASS FALSE ExecuteAction generateUsersXML $SERVERID "composite"
$MODULE_HOME/getUserModule.xml $MODULE_HOME/servers.xml
```

Results Expected:

The file `getUserModule.xml` is produced with only users from the “composite” domain.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:UserModule xmlns:ns2="http://www.dvbu.cisco.com/ps/deploytool/modules">
  <user>
    <id>user1</id>
    <userName>monitor</userName>
  ... xml removed
  </user>
  <user>
    <id>user2</id>
    <userName>anonymous</userName>
  ... xml removed
  </user>
  <user>
    <id>user3</id>
    <userName>nobody</userName>
  ... xml removed
  </user>
  <user>
    <id>user4</id>
```

```

        <userName>system</userName>
... xml removed
    </user>
    <user>
        <id>user5</id>
        <userName>admin</userName>
... xml removed
    </user>
    <user>
        <id>user6</id>
        <userName>user2</userName>
        <encryptedPassword></encryptedPassword>
        <forcePassword>false</forcePassword>
        <domainName>composite</domainName>
        <groupMembershipList>
            <groupName>all</groupName>
            <groupDomain>composite</groupDomain>
        </groupMembershipList>
        <groupMembershipList>
            <groupName>group2</groupName>
            <groupDomain>composite</groupDomain>
        </groupMembershipList>
        <privilege>ACCESS_TOOLS MODIFY_ALL_CONFIG MODIFY_ALL_RESOURCES
MODIFY_ALL_STATUS MODIFY_ALL_USERS READ_ALL_CONFIG READ_ALL_RESOURCES
READ_ALL_STATUS READ_ALL_USERS UNLOCK_RESOURCE</privilege>
        <annotation>user1</annotation>
    </user>
</ns2:UserModule>

```

Scenario 2 – delete a user

Description:

Delete a user from the CIS server.

XML Configuration Sample:

In preparation for this test, use the browser-based DV Manager to add a new user called “user1”. Use the UserModule XML file and make sure it has an entry that looks like this:

```

<?xml version="1.0"?>
<p1:UserModule xmlns:p1="http://www.dvbu.cisco.com/ps/deploytool/modules">
    <user>
        <id>user1</id>
        <userName>user1</userName>
        <encryptedPassword>password1</encryptedPassword>
        <forcePassword>true</forcePassword>
        <domainName>composite</domainName>
    </user>
</p1:UserModule>

```

Execution Sample:

Unix: `./ExecutePDTool.sh -exec ../resources/properties/UnitTest-User.dp`
 Property file setup for UnitTest-User.dp:

```
# -----
# Begin task definition list for UNIX:
# -----
# Delete a user
PASS FALSE ExecuteAction deleteUsers $SERVERID "user1"
$MODULE_HOME/UserModule.xml $MODULE_HOME/servers.xml
```

Results Expected:

The script will report “PASS” for the execution of this action. Open DV Manager and review the list of users. The user “user” should be deleted.

Scenario 3 – create or update a user**Description:**

Create a user if it does not exist or update it if it does.

XML Configuration Sample:

In preparation for this test, use the browser-based DV Manager to add two new groups: group1 and group2. Use the UserModule XML file and make sure it has an entry that looks like this:

```
<?xml version="1.0"?>
<p1:UserModule xmlns:p1="http://www.dvbu.cisco.com/ps/deploytool/modules">
  <user>
    <id>user1</id>
    <userName>user1</userName>
    <encryptedPassword>password</encryptedPassword>
    <forcePassword>true</forcePassword>
    <domainName>composite</domainName>
    <groupMembershipList>
      <groupName>all</groupName>
      <groupDomain>composite</groupDomain>
    </groupMembershipList>
    <groupMembershipList>
      <groupName>group1</groupName>
      <groupDomain>composite</groupDomain>
    </groupMembershipList>
    <groupMembershipList>
      <groupName>group2</groupName>
      <groupDomain>composite</groupDomain>
    </groupMembershipList>
    <privilege>ACCESS_TOOLS</privilege>
    <annotation>user1</annotation>
```

```
</user>  
</p1:UserModule>
```

Execution Sample:

Unix: `./ExecutePDTool.sh -exec ../resources/properties/UnitTest-User.properties`
Property file setup for UnitTest-User.properties:

```
# -----  
# Begin task definition list for UNIX:  
# -----  
# Create or Update a user  
ExecuteAction    createOrUpdateUsers    $SERVERID "user1"  
$MODULE_HOME/UserModule.xml $MODULE_HOME/servers.xml
```

Results Expected:

The script will report “PASS” for the execution of this action. Open DV Manager and review the list of users. The user “user1” should exist now and it should be assigned to the groups: All, group1 and group2.

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.

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"***.