PDTool DataSource 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	8/5/2011	Mike Tinius	Initial version for DataSource Module User Guide
1.1	5/8/2013	Mike Tinius	Added support for generic attributes of type ValueArray, ValueList and ValueMap
1.2	8/19/2013	Mike Tinius	Added new methods and modified the XML Schema. New methods: introspectDataSources, generateDataSourceAttributeDefs, generateDataSourceAttributeDefsByDataSourceType, generateDataSourceTypes, and generateDataSourcesResourceListXML.
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	03/04/05	Mike Tinius	Updated table of contents to include methods.
3.3	3/4/2015	Mike Tinius	Updated docs to Cisco format.
3.4	5/7/2015	Mike Tinius	Updated docs to explain the interaction between updateAllIntrospectedResources and planEntry settings.
4.0	12/14/2017	Mike Tinius	Initial revision with Tibco
5.0	08/27/2020	Mike Tinius	Updated documentation
5.1	10/20/2020	Mike Tinius	Updated documentation

Related Documents

Name	Author
PDTool User's Guide.pdf	Mike Tinius

Supported Versions

Name	Version
TIBCO® Data Virtualization	7.0.8 or later

Table of Contents

1 Introduction

Purpose

The purpose of the DataSource Module User Guide is to demonstrate how to effectively use the DataSource Module and execute actions. Once Data Virtualization (DV) resources are imported into a target DV server during the deployment process, it is typically necessary to update the data source configuration. Usually, the user and password need updating. In some cases, the URL might have to change due to different ports. Whatever the reason, the DataSource Module provides the mechanism to automate the configuration of data source attributes. Using the "*updateDataSources*" method and pointing at an identifier within the DataSourceModule.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 re-introspecting a data source 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 "*generateDataSourcesXML*" to reach into the source server where the artifacts are being deployed from and generate the DataSourceModule 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 DataSourceModule 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 How To Create a DataSource XML Entry

The following section describes the workflow for setting up a Data Source Module XML entry to update, enable, re-introspect or introspect a data source. Essentially, there is a process by which a developer goes about creating an entry in the DataSourceModule.xml. One approach is to create entries from scratch. Even if you decided to start from scratch, it is easier to copy an existing entry from the samples. Another approach is to generate a template and then modify it. This section is more about the process of creating entries. Full descriptions for the XML structure and methods will be provided in later sections of this document.

Update, Enable or Re-introspect a Data Source

These actions are grouped together because they all use the same XML structure. Suffice it to say that the XML provides a choice of children. For these actions, the choice may be the “Relational Data Source XML” structure or the “Generic Data Source XML” structure.

1. **Generate a Template** – this approach is used when the user wants to point at the development environment and generate a template of the existing data sources. This is the recommended approach as there is less editing involved and less chances for mistakes.

   1.1. **Configure a generate data source deployment plan**

   The following example shows how to “generateDataSourcesXML” for a given target “$SERVERID”, a starting path of “/shared/test00” and output the XML into the file “getDataSourceModule.xml” in the directory "$MODULE_HOME/generated”.

   Entry in the deployment plan “myds.dp”

   ```
   PASS FALSE ExecuteAction  generateDataSourcesXML $SERVERID /shared/test00
   "$MODULE_HOME/generated/getDataSourceModule.xml"
   "$MODULE_HOME/servers.xml"
   ```

   1.2. **Execute the generate data source deployment plan**

   The following shows how to execute the file:

   ```
   ExecutePDTool -exec ../resources/plans/myds.dp
   ```

   1.3. **Rename the file or copy the generated data source module XML entries**

   Rename getDataSourceModule.xml to DataSourceModule.xml or whatever name fits your environment. Move it from the generated directory to the main /modules directory. This is done so that the next time you generate the data sources, you won’t overwrite the modified file.

1.4. **Modify the entries for the target deployment environment**

Modify Connection Details

Typically the developer will modify the connection information such as user and password. The other attributes generally do not change between environments.

Modify Attributes

If attributes are different between environments then make adjustments as needed. Review the “Helpful Tips For Adding Attributes” section.

2. **Create from Scratch** – this approach is used when the user wants to create the data source module XML by hand. There is more editing involved in this approach.

2.1. **Create a Data Source Module XML template (copy the entries from a sample)**

Copy the DataSourceModule.xml sample and give it a name appropriate for you environment.

2.2. **Modify the entries for the target deployment environment**

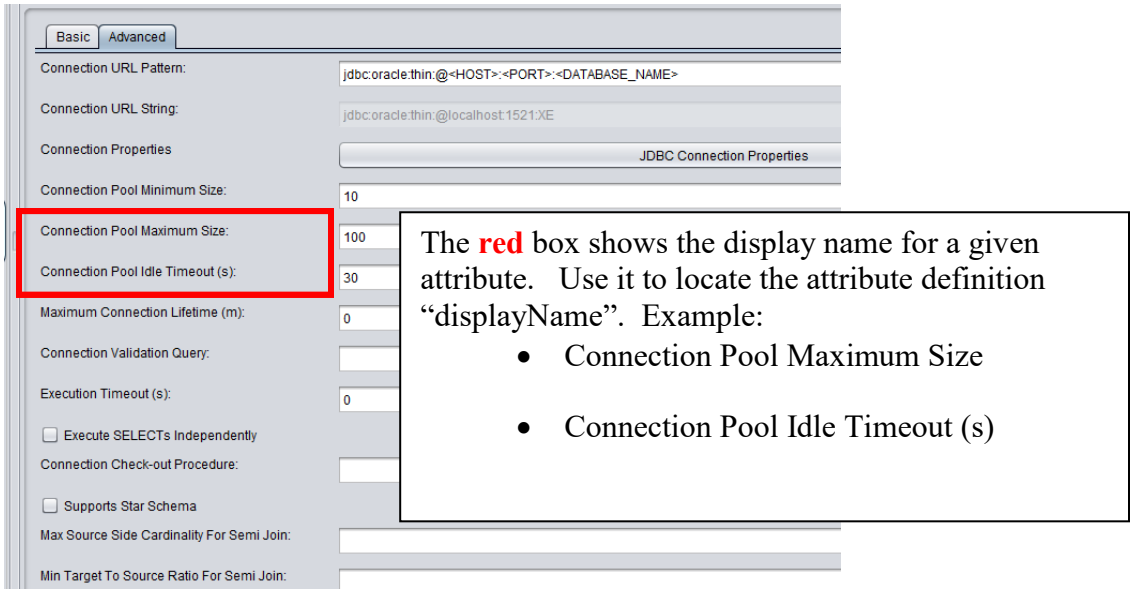
Modify Connection Details and Attributes

The developer will need to modify the connection information for all of the attributes. Review the “Helpful Tips For Adding Attributes” section.

3. **Helpful Tips For Adding Attributes** – This provides a helpful tip for finding the valid list of attributes for a particular type of data source:

3.1. **Find the Display Name for an Attribute – use DV Studio**

The display name can be found to the left of the edit box on the basic and advanced tabs of the data source. The display name will be used to search for the attribute definition “displayName” which then allows the developer to locate the attribute “name” and “type” that they will use to modify the DataSourceModule.xml attribute.



3.2. **Find an attribute for a data source – generateDataSourceAttributeDefs.xml**

Method: generateDataSourceAttributeDefs – This method takes in a starting path and will locate all of the data sources within that path and export the list of attributes for each data source.

Method: generateDataSourceAttributeDefsByDataSourceType – This method takes in a specific data source type and exports the list of attributes for that type of data source. The data source type can be acquired by executing the method "generateDataSourceTypes" or "generateDataSourcesXML" for a targeted list.

- generateDataSourceTypes – Generate the entire list of data source types available in the DV Server.
- generateDataSourcesXML – Generate a list of all data sources and their associated data source types.

The red box shows what to look for. Search for your resource path to find the data source

```
<datasource>
    <relationalDataSource>
        <id>ds2</id>
        <resourcePath>/shared/test00/DataSources/cisOraDemoDS</resourcePath>
        <resourceType>DATA_SOURCE</resourceType>
        <subType>RELATIONAL_DATA_SOURCE</subType>
        <dataSourceType>Oracle 10g (Thin Driver)</dataSourceType>
```

These methods are useful when the developer needs to add an attribute to their Data Source Module XML but does not know what the attribute name is. Using DV Studio, locate the display name in studio, find the correct data source entry and then locate the display name for that data source entry. A data source entry is identified by the "resourcePath". Create a deployment plan to generate data source attribute definitions to a file.

```
<datasource>
    <attributeDefsDataSource>
        <id>ds8</id>
        <resourcePath>/shared/test00/DataSources/cisOraDemoDS</resourcePath>
        <resourceType>DATA_SOURCE</resourceType>
        <subtype>RELATIONAL_DATA_SOURCE</subtype>
        <childCount>1</childCount>
        <dataSourceType>Oracle 10g (Thin Driver)</dataSourceType>
        <attributeDefs>
            <attributeDef>
                <name>connPoolMaxSize</name>
                <type>INTEGER</type>
                <updateRule>READ_WRITE</updateRule>
                <required>true</required>
                <displayName>Connection Pool Maximum Size</displayName>
                <visible>true</visible>
            </attributeDef>
            <attributeDef>
                <name>connPoolMinSize</name>
                <type>INTEGER</type>
                <updateRule>READ_WRITE</updateRule>
                <required>true</required>
                <displayName>Connection Pool Minimu
                <visible>true</visible>
            </attributeDef>
            <attributeDef>
                <name>connPoolTimeout</name>
                <type>INTEGER</type>
                <updateRule>READ_WRITE</updateRule>
                <required>true</required>
                <displayName>Connection Pool Idle Timeout (s)</displayName>
                <visible>true</visible>
            </attributeDef>
            <attributeDef>
                <name>connProperties</name>
                <type>MAP</type>
                <updateRule>READ_WRITE</updateRule>
                <required>false</required>
                <displayName>Connection Properties</displayName>
                <visible>true</visible>
            </attributeDef>
```

The **red** box shows what to look for. The **blue** box shows what to put in the DataSourceModule.xml.

3.3. Generate a Data Source Module template – generateDataSourcesXML.xml

generateDataSourcesXML – This method takes in a starting path and will locate all of the data sources within that path and generate a Data Sources Module template. This template can then be renamed or copied to create the master "DataSourceModule.xml". Create a deployment plan to execute this method. Steps 3.2 and 3.3 can be executed in one deployment plan.

```
<datasource>
    <relationalDataSource>
        <id>ds2</id>
        <resourcePath>/shared/test00/DataSources/cisOraDemoDS</resourcePath>
        <resourceType>DATA_SOURCE</resourceType>
        <subType>RELATIONAL_DATA_SOURCE</subType>
        <dataSourceType>Oracle 10g (Thin Driver)</dataSourceType>
        <hostname>localhost</hostname>
        <port>1521</port>
        <databaseName>XE</databaseName>
        <login>cisorademo</login>
        <encryptedPassword>Encrypt
        <valQuery></valQuery>
        <genericAttribute>
            <name>connPoolMaxSize
            <type>INTEGER</type>
            <value>100</value>
        </genericAttribute>
        <genericAttribute>
            <name>connPoolMinSize</name>
            <type>INTEGER</type>
            <value>10</value>
        </genericAttribute>
        <genericAttribute>
            <name>connPoolTimeout</name>
            <type>INTEGER</type>
            <value>30</value>
        </genericAttribute>
```

rd>

The **blue** box shows what was edited from the output of the file **generateDataSourceAttributeDefs.xml**

Introspect a Data Source

The introspect action uses the "Introspect Data Sources" choice of children XML structure. The use case for this is to add, update or remove data source children. For example, if the data source schema or catalog name in development is different than the one in test, UAT or production, then this deployment method would be used.

1. **Create from Scratch** – this approach is used when the user wants to create the data source module XML by hand.

    1.1. **Create a Data Source Module XML template (copy the entries from a sample)**

    The sample file is located at /resources/modules/DataSourceModule.xml. Copy this file to a master file for the target environment.

    If the master file already exists, then copy a sample entry from the DataSourceModule.xml as shown below.

```
<datasource>
    <introspectDataSource>...
</datasource>
```

    1.2. **Modify the entries for the target deployment environment**

Open the new file and edit it. Delete entries that are not applicable. Add new entries as needed. Add attributes as necessary. Add plan entries as necessary. The following screen shot shows an example of what an introspect XML looks like:

```
<datasource>
    <introspectDataSource>
        <id>ds7</id>
        <resourcePath>/shared/test00/DataSources/cisOraDemoDS</resourcePath>
        <!--Element runInBackgroundTransaction is optional-->
        <runInBackgroundTransaction>false</runInBackgroundTransaction>
        <reportDetail>SIMPLE_COMPRESSED</reportDetail>
        <plan>
            <updateAllIntrospectedResources>true</updateAllIntrospectedResources>
            <failFast>true</failFast>
            <commitOnFailure>true</commitOnFailure>
            <autoRollback>true</autoRollback>
            <scanForNewResourcesToAutoAdd>false</scanForNewResourcesToAutoAdd>
            <!--Element planEntry is optional-->
            <planEntry>
                <resourceId>
                    <resourcePath>CISORADEMO/EMPLOYEES</resourcePath>
                    <resourceType>TABLE</resourceType>
                    <subtype>DATABASE_TABLE</subtype>
                </resourceId>
                <action>ADD_OR_UPDATE</action>
            </planEntry>
            <planEntry>...
            <planEntry>...
            <planEntry>...
        </plan>
    </introspectDataSource>
</datasource>
```

1.3. **Guidance**

- Relative Paths – The planEntry/resourceId/resourcePath is a relative path from the data source “resourcePath”. Any level of relative path can be specified. For example, for a relational database with both a catalog, schema and tables, any of those three levels may be specified. The higher up in the path the more time it will take to introspect and thus more objects will be brought into DV. The lower in the relative path you go, the more targeted you get and less time will be needed for introspection.

- Actions – The actions include “REMOVE”, “ADD_OR_UPDATE”, or “ADD_OR_UPDATE_RECURSIVELY”.

    - In the above example, CISORADEMO is the schema name and EMPLOYEES is a database table. Here are some examples of using the different actions:

        - To remove a specific table set the action

```
<planEntry>
    <resourceId>
        <resourcePath>CISORADEMO/EMPLOYEES</resourcePath>
        <resourceType>TABLE</resourceType>
        <subtype>DATABASE_TABLE</subtype>
    </resourceId>
    <action>REMOVE</action>
</planEntry>
```

- To add or update a specific database table object for a schema container set plan entry as follows:

```
<planEntry>
    <resourceId>
        <resourcePath>CISORADEMO/EMPLOYEES</resourcePath>
        <resourceType>TABLE</resourceType>
        <subtype>DATABASE_TABLE</subtype>
    </resourceId>
    <action>ADD_OR_UPDATE</action>
</planEntry>
```

- To add or update all objects for a schema container set plan entry as follows:

```
<planEntry>
    <resourceId>
        <resourcePath>CISORADEMO</resourcePath>
        <resourceType>CONTAINER</resourceType>
        <subtype>SCHEMA_CONTAINER</subtype>
    </resourceId>
    <action>ADD_OR_UPDATE_RECURSIVELY</action>
</planEntry>
```

- Locating resourceType and subtype
  - Each plan entry contains a “resourceId” grouping that specifies the relative path to the resource “resourcePath”, the type of resource “resourceType” and the sub type “subtype”.
  - The list shown below is not a complete list but shows the most popular type/subtype combinations.

Resource Type	Sub Type	Definition
DATA_SOURCE	RELATIONAL_DATA_SOURCE	A relational database source
DATA_SOURCE	XML_FILE_DATA_SOURCE	A comma separate file data source
DATA_SOURCE	WSDL_DATA_SOURCE	An XML file data source
DATA_SOURCE	XML_HTTP_DATA_SOURCE	A DV web service data source

DATA_SOURCE	NONE	A custom java procedure data source
CONTAINER	FOLDER_CONTAINER	A DV folder
CONTAINER	DIRECTORY_CONTAINER	A DV directory
CONTAINER	CATALOG_CONTAINER	A DV catalog folder under a data source
CONTAINER	SCHEMA_CONTAINER	A DV schema container under a data source
CONTAINER	SERVICE_CONTAINER	A web service container for the service
CONTAINER	OPERATIONS_CONTAINER	A web service container for the operations
CONTAINER	PORT_CONTAINER	A DV web service container for port
CONTAINER	CONNECTOR_CONTAINER	A DV container for connectors
TABLE	DATABASE_TABLE	A DV database table
PROCEDURE	DATABASE_PROCEDURE	A database stored procedure
TREE	XML_FILE_TREE	The XML tree associated with a file-XML data source

- Determine the exact resource type and subtype for a resource.

  generateDataSourcesResourceListXML – This method takes in a starting path and will locate all of the data sources within that path and generate a data source resource list including children. The image below shows an example of a relational data source. As you can see, the relevant information regarding path, resource type and sub type are present. This provides enough information to create a plan entry.

```
<datasource>
    <genericDataSource>
        <id>ds635</id>
        <resourcePath>/shared/test00/DataSources/cisOraDemoDS</resourcePath>
        <resourceType>DATA_SOURCE</resourceType>
        <subType>RELATIONAL_DATA_SOURCE</subType>
        <dataSourceType>Oracle 10g (Thin Driver)</dataSourceType>
    </genericDataSource>
</datasource>
<datasource>
    <genericDataSource>
        <id>ds636</id>
        <resourcePath>/shared/test00/DataSources/cisOraDemoDS/CISORADEMO</resourcePath>
        <resourceType>CONTAINER</resourceType>
        <subType>SCHEMA_CONTAINER</subType>
    </genericDataSource>
</datasource>
<datasource>
    <genericDataSource>
        <id>ds637</id>
        <resourcePath>/shared/test00/DataSources/cisOraDemoDS/CISORADEMO/CUSTOMERS</resourcePath>
        <resourceType>TABLE</resourceType>
        <subType>DATABASE_TABLE</subType>
    </genericDataSource>
</datasource>
```

3 DataSource Module Definition

Method Definitions and Signatures

1. updateDataSources

Update Data Source method updates data source configurations based on the values identified by the data source id in the DataSourceXML and the target server.

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

@param dataSourceIds list of data sources Ids(comma separated data source Ids)

@param pathToDataSourceXML path to the data source xml

@param pathToServersXML path to the server values xml

@throws CompositeException

public void updateDataSources(String serverId, String dataSourceIds, String pathToDataSourceXML, String pathToServersXML) throws CompositeException;
```

2. enableDataSources

Enable a data source for access.

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

@param dataSourceIds list of data sources Ids(comma separated data source Ids)

@param pathToDataSourceXML path to the data source xml

@param pathToServersXML path to the server values xml

@throws CompositeException

public void enableDataSources(String serverId, String dataSourceIds, String pathToDataSourceXML, String pathToServersXML) throws CompositeException;
```

3. reIntrospectDataSources

Re-introspect a data source.

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

@param dataSourceIds list of comma separate data sources Ids

@param pathToDataSourceXML path to the data source xml

@param pathToServersXML path to the server values xml

@throws CompositeException
```

```
public void reIntrospectDataSources(String serverId, String
dataSourceIds, String pathToDataSourceXML, String pathToServersXML)
throws CompositeException;
```

4. introspectDataSources

Introspect a data source to add, update or remove children resources. This method is useful when the catalog or schema name in your target deployment environment is different than what was used in the development environment. This allows the deployment administrator to add resource data base tables to a different schema name at deployment time.

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

@param dataSourceIds list of comma separate data sources Ids

@param pathToDataSourceXML path to the data source xml

@param pathToServersXML path to the server values xml

@throws CompositeException

public void introspectDataSources(String serverId, String dataSourceIds,
String pathToDataSourceXML, String pathToServersXML) throws
CompositeException;
```

5. generateDataSourcesXML

Generate the DataSourceXML 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 pathToDataSourceXML path including name to the data source xml
which needs to be created

@param pathToServersXML path to the server values xml

@throws CompositeException

public void generateDataSourcesXML(String serverId, String startPath,
String pathToDataSourceXML, String pathToServersXML) throws
CompositeException;
```

6. generateDataSourceAttributeDefs

Generate Data Source Attribute Definitions for the passed in starting path and the target server Id. The method allows the invoker generate a file of data source attribute definitions. This method will be useful when the user wants to determine what valid attributes are available for a given data source. This method will search for all data sources within the given “startPath” and export the attributes for those data sources found.

The content of this method is useful when creating the attributes for the DataSourceModule.xml. For example, let’s say that you are in DV Studio and want to determine how to set the “Connection Pool Maximum Size”. That string is known as the

"displayName" in the attribute definition below. The first thing that you would do is search for your specific data source path. Within that node of the XML, you would then search for the "displayName". Once you find that the "name" attribute defines the attribute name to be set in the DataSourceModule.xml. The "type" defines the type to be set. In order to create a new attribute in the DataSourceModule.xml, it must have an update rule of READ_WRITE. An example attribute definition is shown below:

```
<attributeDef>
   <name>connPoolMaxSize</name>
   <type>INTEGER</type>
   <updateRule>READ_WRITE</updateRule>
   <required>true</required>
   <displayName>Connection Pool Maximum Size</displayName>
   <visible>true</visible>
</attributeDef>
```

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

@param startPath starting path of the resource e.g /shared

@param pathToDataSourceAttrDefs path to the data attribute definitions
xml

@param pathToServersXML path to the server values xml

@throws CompositeException

public void generateDataSourceAttributeDefs(String serverId, String
startPath, String pathToDataSourceAttrDefs, String pathToServersXML)
throws CompositeException;
```

7. generateDataSourceAttributeDefsByDataSourceType

Generate Data Source Attribute Definitions for the passed in data source type and the target server Id. The method allows the invoker generate a file of data source attribute definitions. This method will be useful when the user wants to determine what valid attributes are available for a specific data source. This method will only return the attributes for the single data source type passed in.

The content of this method is useful when creating the attributes for the DataSourceModule.xml. For example, let's say that you are in DV Studio and want to determine how to set the "Connection Pool Maximum Size". That string is known as the "displayName" in the attribute definition below. The first thing that you would do is search for your specific data source path. Within that node of the XML, you would then search for the "displayName". Once you find that the "name" attribute defines the attribute name to be set in the DataSourceModule.xml. The "type" defines the type to be set. In order to create a new attribute in the DataSourceModule.xml, it must have an update rule of READ_WRITE. An example attribute definition is shown below:

```
<attributeDef>
   <name>connPoolMaxSize</name>
   <type>INTEGER</type>
   <updateRule>READ_WRITE</updateRule>
   <required>true</required>
```

```
    <displayName>Connection Pool Maximum Size</displayName>
    <visible>true</visible>
  </attributeDef>
```

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

@param dataSourceType a valid data source type which can be found in
"getDataSourceTypes" output

@param pathToDataSourceAttrDefs path to the data attribute definitions
xml

@param pathToServersXML path to the server values xml

@throws CompositeException

public void generateDataSourceAttributeDefsByDataSourceType(String
serverId, String dataSourceType, String pathToDataSourceAttrDefs, String
pathToServersXML) throws CompositeException;
```

8. generateDataSourceTypes

Generate Data Source Types for the DV Server. The method allows the invoker generate a file of data source types for the DV Server. The value of “dataSourceType” element can be used as input to the method “generateDataSourceAttributeDefsByDataSourceType”. An example data source type XML is shown below:

```
<dataSourceType>
  <name>Oracle 11g (Thin Driver)</name>
  <type>Oracle</type>
  <attribute>
    <name>driverClass</name>
    <type>STRING</type>
    <value>oracle.jdbc.OracleDriver</value>
  </attribute>
  <attribute>
    <name>urlPattern</name>
    <type>STRING</type>
    <value>jdbc:oracle:thin:@&lt;HOST&gt;:&lt;PORT&gt;:&lt;DATABASE_NAME&gt;</value>
  </attribute>
</dataSourceType>
```

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

@param pathToDataSourceTypesXML path to the data source types xml

@param pathToServersXML path to the server values xml

@throws CompositeException

public void generateDataSourceTypes(String serverId, String
pathToDataSourceTypesXML, String pathToServersXML) throws
CompositeException;
```

9. generateDataSourcesResourceListXML

Generate Data Sources Resource List for a starting path and target server Id will export all of the Data Sources found and their children. This will be useful to know the type and subtype for a particular data source child when constructing the "plan" entries for the "introspectDataSources" method. An example SQL Server relational database resource

list XML is shown below. The developer can use path, type and subtype to determine how best to construct the plan entries.

```
    <genericDataSource>
      <id>ds1</id>
      <resourcePath>/shared/test00/DataSources/MyDS</resourcePath>
      <resourceType>DATA_SOURCE</resourceType>
      <subType>RELATIONAL_DATA_SOURCE</subType>
      <dataSourceType>Microsoft SQL Server 2008</dataSourceType>
    </genericDataSource>
  </datasource>
  <datasource>
    <genericDataSource>
      <id>ds2</id>
      <resourcePath>/shared/test00/DataSources/MyDS/MyCatalog</resourcePath>
      <resourceType>CONTAINER</resourceType>
      <subType>CATALOG_CONTAINER</subType>
    </genericDataSource>
  </datasource>
  <datasource>
    <genericDataSource>
      <id>ds3</id>
      <resourcePath>/shared/test00/DataSources/MyDS/MyCatalog/dbo</resourcePath>
      <resourceType>CONTAINER</resourceType>
      <subType>SCHEMA_CONTAINER</subType>
    </genericDataSource>
  </datasource>
  <datasource>
    <genericDataSource>
      <id>ds4</id>
      <resourcePath>/shared/test00/DataSources/MyDS/MyCatalog/dbo /Customer</resourcePath>
      <resourceType>TABLE</resourceType>
      <subType>DATABASE_TABLE</subType>
    </genericDataSource>
  </datasource>
```

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

@param startPath starting path of the resource e.g /shared

@param pathToDataSourceResourceListXML path including name to the data
source resource list xml which needs to be created

@param pathToServersXML path to the server values xml

@throws CompositeException

void generateDataSourcesResourceListXML(String serverId, String
startPath, String pathToDataSourceResourceListXML, String
pathToServersXML) throws CompositeException;
```

General Notes:

The arguments pathToDataSourceXML 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.

4 DataSource 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 DataSourceModule XML provides a structure “DatasourceModule”.

The global entry point node is called “DataSourceModule” and contains one or more “datasource” nodes. The XML is constructed as a “Choice of Children”

```
<!—Choice of Children:

<xs:complexType name="DataSourceChoiceType">
	<xs:annotation>
		<xs:documentation xml:lang="en">
			Data Source Choice Type: This selection provides a choice between the
relational source, generic source, introspection, attribute definition and data source types.
		</xs:documentation>
	</xs:annotation>
	<xs:choice>
		<xs:element name="relationalDataSource" maxOccurs="1" minOccurs="0"
type="ns:RelationalDataSourceType"/>
		<xs:element name="genericDataSource" minOccurs="0" type="ns:GenericDataSourceType"
maxOccurs="1"/>
		<xs:element name="introspectDataSource" minOccurs="0"
type="ns:IntrospectDataSourceType" maxOccurs="1"/>
		<xs:element name="attributeDefsDataSource" minOccurs="0"
type="ns:AttributeDefsDataSourceType" maxOccurs="1"/>
		<xs:element name="dataSourceTypesDataSource" minOccurs="0"
type="ns:DataSourceTypesType" maxOccurs="1"/>
	</xs:choice>
</xs:complexType>
```

The various operations are shown below with the choice of children indicated

- enable – enable a data source
  - use Choice of relationalDataSource or genericDataSource
- reintrospect – update existing data source resources
  - use Choice of relationalDataSource or genericDataSource
- introspect – add, update, remove resources from a data source
  - use Choice of introspectDataSource
- updateDataSource – Modify attributes for a data source
  - use Choice of relationalDataSource or genericDataSource

- generateDataSourceXML – generate the DataSourceModule.xml template from an existing data source.
    - use Choice of relationalDataSource or genericDataSource
- generateDataSourceAttributeDefs – generate data source attribute definitions
    - use Choice of attributeDefsDataSource
- generateDataSourceAttributeDefsByDataSourceType – generate data source attribute definitions
    - use Choice of attributeDefsDataSource
- generateDataSourceTypes – generate data source types
    - use Choice of dataSourceTypesDataSource

The following chart shows an example of a “relationalDataSource” structure:

```
<p1:DatasourceModule xmlns:p1="http://www.dvbu.cisco.com/ps/deploytool/modules">

<!—Example of a Relational specific attributes only for configuration:
       <datasource>
           <relationalDataSource>
               <id>ds1</id>
               <resourcePath>/shared/examples/ds_orders</resourcePath>
               <hostname>localhost</hostname>
               <port>9408</port>
               <databaseName>orders</databaseName>
               <login>tutorial</login>
               <encryptedPassword>tutorial</encryptedPassword>
               <valQuery>select 1 from dual</valQuery>
           </relationalDataSource>
       </datasource>

<!—Example of a File data source using generic attributes for configuration:
       <datasource>
         <genericDataSource>

           <id>ds2</id>
           <resourcePath>/shared/test1/ServerAttributeDefinitions</resourcePath>

           <genericAttribute>
               <name>filters</name>
               <type>STRING</type>
               <value>*.xml</value>
           </genericAttribute>

            <genericAttribute>
                <name>root</name>
                <type>STRING</type>
                <value>$PROJECT_HOME\resources\modules</value>
            </genericAttribute>
         </genericDataSource>
       </datasource>

<!—Example of a Relational data source using specific relation attributes and
generic attributes for configuration:
       <datasource>
```

```
        <relationalDataSource>
          <id>ds3</id>
          <resourcePath>/shared/test00/DataSources/ds_orders</resourcePath>
          <hostname>localhost</hostname>
          <port>9408</port>
          <databaseName>orders</databaseName>
          <login>tutorial</login>
          <encryptedPassword>Encrypted:A49A5A0FAFF13F4A</encryptedPassword>
          <valQuery></valQuery>

          <genericAttribute>
              <name>autoAddChildren</name>
              <type>BOOLEAN</type>
              <value>true</value>
          </genericAttribute>
          <genericAttribute>
              <name>commitOnFetchDone</name>
              <type>BOOLEAN</type>
              <value>false</value>
          </genericAttribute>
          <genericAttribute>
              <name>connPoolMaxSize</name>
              <type>INTEGER</type>
              <value>100</value>
          </genericAttribute>
          <genericAttribute>
              <name>connPoolMinSize</name>
              <type>INTEGER</type>
              <value>10</value>
          </genericAttribute>
          <genericAttribute>
              <name>connPoolTimeout</name>
              <type>INTEGER</type>
              <value>30</value>
          </genericAttribute>
          <genericAttribute>
              <name>execTimeout</name>
              <type>INTEGER</type>
              <value>0</value>
          </genericAttribute>
          <genericAttribute>
              <name>isPassThrough</name>
              <type>STRING</type>
              <value>Disabled</value>
          </genericAttribute>
          <genericAttribute>
              <name>persistPassword</name>
              <type>BOOLEAN</type>
              <value>true</value>
          </genericAttribute>
          <genericAttribute>
              <name>streamingResults</name>
              <type>BOOLEAN</type>
              <value>true</value>
          </genericAttribute>
          <genericAttribute>
              <name>txnIsolationLevel</name>
              <type>STRING</type>
              <value>Read Committed</value>
          </genericAttribute>
          <genericAttribute>
              <name>urlPatternStr</name>
              <type>STRING</type>
              <value>jdbc:mysql://&lt;HOST&gt;:&lt;PORT&gt;/&lt;DATABASE_NAME&gt;</value>
          </genericAttribute>
      </relationalDataSource>
  </datasource>
```

```
<!—Example of a web service data source automating import of keystore
certificates:
  <datasource>
    <genericDataSource>
      <id>ds11</id>
      <resourcePath>/shared/test00/DataSources/testWebService</resourcePath>
      <!--Element keystore is optional-->
      <keystore>
        <!--Element keystoreAttributes is optional, maxOccurs=unbounded-->
        <keystoreAttributes>
          <keystoreFilePath>$PROJECT_HOME_PHYSICAL/resources/certs/cert.jks</keystoreFilePath>
          <keystoreType>jks</keystoreType>
          <keystorePassword>changeit</keystorePassword>
        </keystoreAttributes>
      </keystore>
    </genericDataSource>
  </datasource>
</p1:DatasourceModule>
```

Relational Attributes of Interest

id – The unique identifier within the DataSourceModule.xml file that is used to identify a data source resource configuration.

resourcePath – the DV path to the data source resource.

hostname – the qualified data base hostname.

port – the database port.

databaseName – The database name or SID for oracle.

login – the database users login.

encryptedPassword – the password for the specified data source. Use the following syntax to encrypt the DataSourceModule.xml file:

ExecutePDTool.[bat|.sh] -encrypt ../resources/modules/DataSourceModule.xml

valQuery – a SQL validation query in the specific syntax of the underlying data source.

keystore – The name of the attribute. The method updateDatasources will use this to automatically import and update a datasource that requires certificates. The three key variables is the path to the certificate file, the type .jks [type=jks] or .pfx [type=pkcs12], and the keystore password.

```
      <keystore>
        <!--Element keystoreAttributes is optional, maxOccurs=unbounded-->
        <keystoreAttributes>
          <keystoreFilePath>$PROJECT_HOME_PHYSICAL/resources/certs/cert.jks</keystoreFilePath>
          <keystoreType>jks</keystoreType>
          <keystorePassword>changeit</keystorePassword>
        </keystoreAttributes>
      </keystore>
```

genericAttribute – zero or more iterations of the Generic Attributes type defined in the section below.

Generic Attributes of Interest

name – The name of the attribute. The method generateDataSourcesXML will generate a file that will contain all of the writable generic attributes. This allows the user to simply modify values rather than having to know all of the names.

type – The type of the attribute as defined by the AttributeTypeSimpleType described in the attribute value restrictions section below.

Value is one of value or valueArray or valueList or valueMap.

value – (optional) The value for the attribute name.

```
<genericAttribute>
    <name>LowerBoundForDataShip</name>
    <type>LONG</type>
    <value>100</value>
</genericAttribute>
```

valueArray – (optional) The value array for the attribute name.

```
<genericAttribute>
    <name>valueArrayName</name>
    <type>STRING_ARRAY</type>
   <valueArray>
       <!--Element item is optional, maxOccurs=unbounded-->
       <item>value1</item>
       <item>value2</item>
       <item>value3</item>
   </valueArray>
</genericAttribute>
```

valueList – (optional) The value list for the attribute name.

```
<genericAttribute>
    <name>valueListName</name>
    <type>LIST</type>
   <valueList>
       <!--Element item is optional, maxOccurs=unbounded-->
       <item>
           <!--Element type is optional-->
           <type>STRING</type>
           <!--Element value is optional-->
           <value>string</value>
       </item>
       <item>
           <!--Element type is optional-->
           <type>STRING</type>
           <!--Element value is optional-->
           <value>string</value>
       </item>
   </valueList>
</genericAttribute>
```

valueMap – (optional) The value map for the attribute name.

```
<genericAttribute>
```

```
                <name>DBLinkList</name>
                <type>MAP</type>
                <valueMap>
                    <entry>
                        <key>
                            <type>STRING</type>
                            <value>COMPOSITE_$ENV</value>
                        </key>
                        <value>
                            <type>STRING</type>
                            <value>/shared/Common/Org/Physical/Metadata/Warehouse/DB1</value>
                        </value>
                    </entry>
                </valueMap>
            </genericAttribute>
```

Custom and System Properties

The DataSourceModule has implemented a capability whereby each field except the Password fields may use properties that are defined either in deploy.dp (custom properties), Java Environment properties (-DVAR=val) or System Environment properties (Windows:set VAR=val or UNIX:export VAR=val). For example, the default Java Environment properties PROJECT_HOME=<your installed path> is set automatically upon execution. This allows you to reference the $PROJECT_HOME path dynamically and place relative path references in the DataSourceModule.xml file if needed. Another example would be to set MYPATH=/dev/mypath as a custom property in the deploy.dp file (PDTool/resources/config/deploy.dp) and then reference the variable $MYPATH in DataSourceModule.xml. An example is shown below

```
            <genericAttribute>
                <name>root</name>
                <type>STRING</type>
                <value>$PROJECT_HOME\resources\modules</value>
            </genericAttribute>
```

Attribute Value Restrictions

type (AttributeTypeSimpleType) – The type of generic attribute being configured is defined by the following restriction list:

```
    <xs:simpleType name="AttributeTypeSimpleType">
        <xs:annotation>
            <xs:documentation xml:lang="en">
                Attribute Type: This simple type is used when assigning the "type" for
any attribute and for any module.
            </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="BOOLEAN"/>
            <xs:enumeration value="BOOLEAN_ARRAY"/>
            <xs:enumeration value="BYTE"/>
            <xs:enumeration value="BYTE_ARRAY"/>
            <xs:enumeration value="DATE"/>
            <xs:enumeration value="DATE_ARRAY"/>
            <xs:enumeration value="DOUBLE"/>
            <xs:enumeration value="DOUBLE_ARRAY"/>
            <xs:enumeration value="FILE_PATH_STRING"/>
            <xs:enumeration value="FLOAT"/>
            <xs:enumeration value="FLOAT_ARRAY"/>
            <xs:enumeration value="FOLDER"/>
            <xs:enumeration value="INT_ARRAY"/>
            <xs:enumeration value="INTEGER"/>
            <xs:enumeration value="LIST"/>
```

```
            <xs:enumeration value="LONG"/>
            <xs:enumeration value="LONG_ARRAY"/>
            <xs:enumeration value="MAP"/>
            <xs:enumeration value="NULL"/>
            <xs:enumeration value="OBJECT"/>
            <xs:enumeration value="PASSWORD_STRING"/>
            <xs:enumeration value="PATH_STRING"/>
            <xs:enumeration value="SET"/>
            <xs:enumeration value="SHORT"/>
            <xs:enumeration value="SHORT_ARRAY"/>
            <xs:enumeration value="STRING"/>
            <xs:enumeration value="STRING_ARRAY"/>
            <xs:enumeration value="UNKNOWN"/>
        </xs:restriction>
    </xs:simpleType>
```

Introspect Attributes of Interest

```
<!—Example of an Introspect data source for introspection:
<datasource>
       <introspectDataSource>
              <id>ds6</id>
              <resourcePath>/shared/test00/DataSources/ds_orders</resourcePath>
              <!--Element runInBackgroundTransaction is optional-->
              <runInBackgroundTransaction>false</runInBackgroundTransaction>
              <reportDetail>SIMPLE</reportDetail>
              <plan>
                  <!-- TRUE=Updates all resources regardless of targeted planEntry section
                       FALSE=Recommended when using planEntry setting. -->
                     <updateAllIntrospectedResources>true</updateAllIntrospectedResources>
                     <failFast>true</failFast>
                     <commitOnFailure>true</commitOnFailure>
                     <autoRollback>true</autoRollback>
                     <scanForNewResourcesToAutoAdd>false</scanForNewResourcesToAutoAdd>
                     <!--Element planEntry is optional.
                        Set updateAllIntrospectedResources to FALSE when using this section.
                        -->
                     <planEntry>
                            <resourceId>
                                   <resourcePath>shippingmethods</resourcePath>
                                   <resourceType>TABLE</resourceType>
                                   <subtype>DATABASE_TABLE</subtype>
                            </resourceId>
                            <action>ADD_OR_UPDATE</action>
                     </planEntry>
                     <planEntry>
                            <resourceId>
                                   <resourcePath>customers</resourcePath>
                                   <resourceType>TABLE</resourceType>
                                   <subtype>DATABASE_TABLE</subtype>
                            </resourceId>
                            <action>REMOVE</action>
                     </planEntry>
              </plan>
       </introspectDataSource>
</datasource>
```

id – The unique identifier within the DataSourceModule.xml file that is used to identify a data source resource configuration.

resourcePath – the DV path to the data source resource.

runInBackgroundTransaction – PDTool only supports foreground processing. Value must be set to false.

reportDetail – This is one of SUMMARY, SIMPLE, or FULL. The report detail determines how much of the report to print out to the log. In all cases, if there is an error, print out the error.

SUMMARY: print out the introspection report summary.

SIMPLE: print out the resource identifier and status fields only.

SIMPLE_COMPRESSED: print out the resource identifier and status fields only as one line with no blank lines following.

FULL: print out the full report including messages and detail. When tables are added, there is a message for each column added.

plan – provides the details for the introspection plan.

updateAllIntrospectedResources – If the plan's updateAllIntrospectedResources option is TRUE, then update ALL of the introspected resources upon completion. The implication is that even if you are targeting specific resources in the “planEntry” section, this setting has the effect of ignoring those when set to TRUE. The recommendation is to set this parameter to FALSE when specifying specific entries in the “planEntry” section described below.

failFast – If the plan's failFast option is TRUE, then the introspection will fail when the first error occurs. Otherwise the plan will run to completion as a best effort. The default is FALSE.

commitOnFailure – If the plan's commOnFail option is TRUE, then the introspection commits whatever it can. fastFail is also TRUE, then only the successfully introspected resources, up to that point, will be committed. The default is FALSE.

autoRollback – If the plan's autoRollback option is TRUE, then the introspection task will rollback back rather than committing. This supersedes all commit options. This allows you to perform a dry run of resource introspection. The "introspectionResourcesResult" operation is usable if autoRollback is TRUE. If autoRollback is FALSE or unset, then the introspection will not automatically be rolled back.

scanForNewResourcesToAutoAdd – If the plan's scanForNewResourcesToAutoAdd option is TRUE, then the introspection task will for native resources that have been newly added to the data source. If newly added resources are found and their parent container has the "autoAddChildren" introspection set, then that child will automatically be introspected.

planEntry – Introspect Data Source Plan Entry Type: Provides an iteration of detailed entries for introspection. Set **updateAllIntrospectedResources** to FALSE when using this section.

resourceId – Introspect Data Source Plan Entry Resource Id Type: The path, type, and subtype of the resource to be introspected. Resource paths are relative to the data source. An empty path (i.e. "") identifies the data source itself.

resourcePath – The relative path to the data source child. Do not include a "/" in front of the path. Include catalog and schema names when referencing tables.

resourceType – The type of resource identified by the resourcePath. E.g. DATA_SOURCE, CONTAINER, PROCEDURE, TABLE.

subtype – The subtype identified by the resourcePath. E.g. RELATIONAL_DATA_SOURCE, SCHEMA_CONTAINER, DATABASE_TABLE.

action – action: This is one of ADD_OR_UPDATE or REMOVE. If ADD_OR_UPDATE is specified, then the resource will be added if it does not already exist. Otherwise it will be updated. If REMOVE is specified, then the resource will be removed if it exists. Data sources may not be removed. Use the destroyResource operation to remove a data source.

genericAttribute – Refer to the above section labeled "**Generic Attributes of Interest**"

genericAttribute – Refer to the above section labeled "**Generic Attributes of Interest**"

5 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 DataSourceModule.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-DataSource.dp

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

Properties File (UnitTest-DataSource.dp):

Property File Rules:

```
# ----------------------------
# UnitTest-DataSource.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:
# ----------------------------------------
# Generate the list of Datasources
PASS   FALSE  ExecuteAction       generateDataSourcesXML     $SERVERID /shared/test00
              "$MODULE_HOME/getDataSourceModule.xml" "$MODULE_HOME/servers.xml"
#
# Update Datasource
#PASS  FALSE  ExecuteAction       updateDataSources          $SERVERID "ds1"
              "$MODULE_HOME/DataSourceModule.xml" "$MODULE_HOME/servers.xml"
#
# Enable Datasource
#PASS  FALSE  ExecuteAction       enableDataSources          $SERVERID "ds1,ds2"
              "$MODULE_HOME/DataSourceModule.xml" "$MODULE_HOME/servers.xml"
#
# Reintrospect Datasource
#PASS  FALSE  ExecuteAction       reIntrospectDataSources    $SERVERID "ds1,ds2"
              "$MODULE_HOME/DataSourceModule.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-DataSource.xml

Unix: ./ExecutePDTool.sh -ant ../resources/ant/build-DataSource.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="datasourceIds"     value="ds1,ds2,ds3"/>
  <property name="pathToGenDataSourceXML" value="${MODULE_HOME}/getDataSourceModule.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="Generate"    action="generateDataSourcesXML"
       arguments="${SERVERID}^/shared/test00^${pathToGenDataSourceXML}^${pathToServersXML}"
       endExecutionOnTaskFailure="TRUE"/>


       <!-- Windows or UNIX
        <executeJavaAction description="Generate"          action="generateDataSourcesXML"
       arguments="${SERVERID}^/shared/test00^${pathToGenDataSourceXML}^${pathToServersXML}"
       endExecutionOnTaskFailure="TRUE"/>
        <executeJavaAction description="Update"     action="updateDataSources"
       arguments="${SERVERID}^${datasourceIds}^${pathToDataSourcesXML}^${pathToServersXML}"
       endExecutionOnTaskFailure="TRUE"/>
        <executeJavaAction description="Enable"     action="enableDataSources"
       arguments="${SERVERID}^${datasourceIds}^${pathToDataSourcesXML}^${pathToServersXML}"
       endExecutionOnTaskFailure="TRUE"/>
        <executeJavaAction description="Reintrospect" action="reIntrospectDataSources"
       arguments="${SERVERID}^${datasourceIds}^${pathToDataSourcesXML}^${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 “dataSourceIds”.

- 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       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
    - PDTool 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”

- 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       updateDataSources $SERVERID "-ds3,ds4"
       "$MODULE_HOME/DataSourceModule.xml" "$MODULE_HOME/servers.xml"
```

Example Ant build file

```
<executeJavaAction description="Update"        action="updateDataSources"
arguments="${SERVERID}^-ds3,ds4^${pathToDataSourcesXML}^${pathToServersXML}"
```

6 PDTool Examples

The following are common scenarios when using the DataSourceModule.

Scenario 1 – Generate DataSource Module XML

Description:

Generate the DataSource 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 DataSourceModule.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-DataSource.dp
Property file setup for UnitTest-DataSource.dp:

```
# ----------------------------------------
# Begin task definition list for UNIX:
# ----------------------------------------
# Generate the list of Datasources
PASS   FALSE  ExecuteAction  generateDataSourcesXML   $SERVERID /shared/test00
              "$MODULE_HOME/getDataSourceModule.xml" "$MODULE_HOME/servers.xml"
```

Results Expected:

PDTool executed and generated the getDataSourceModule.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 data source attributes.

Scenario 2 – Update Data Source

Description:

Update the data sources on the target DV server using the generated DataSource 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 DataSourceModule.xml configuration for a relational data source.

```
<p1:DatasourceModule xmlns:p1="http://www.dvbu.cisco.com/ps/deploytool/modules">

<!—Example of a Relational specific attributes only for configuration:
       <datasource>
              <relationalDataSource>
                     <id>ds1</id>
                     <resourcePath>/shared/examples/ds_orders</resourcePath>
                     <hostname>myhost.domain.com</hostname>
                     <port>9418</port>
                     <databaseName>orders</databaseName>
                     <login>user1</login>
                     <encryptedPassword>Encrypted:A49A5A0FAFF13F4A</encryptedPassword>
                     <valQuery>select 1 from dual</valQuery>
              </relationalDataSource>
       </datasource>
```

Execution Sample:

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

```
# ----------------------------------------
# Begin task definition list:
# ----------------------------------------
# Update Datasource
PASS   FALSE  ExecuteAction   updateDataSources $SERVERID "ds1"
       "$MODULE_HOME/DataSourceModule.xml" "$MODULE_HOME/servers.xml"
```

Results Expected:

PDTool executed the "*updateDataSources*" for the data source identified by "**ds1**" and the target DV server identified by the variable **"$SERVERID**". All variables are defined in /resources/config/deploy.properties.

Scenario 3 – Introspect Data Source

Description:

Introspect the data sources on the target DV server. This provides the deployment manager the ability to add, update and remove database tables from a target data source during deployment.

XML Configuration Sample:

This is an example of a DataSourceModule.xml configuration for a relational data source.

```
<p1:DatasourceModule xmlns:p1="http://www.dvbu.cisco.com/ps/deploytool/modules">

<!—Example of a Introspection Data Source XML configuration:

   <datasource>
      <introspectDataSource>
             <id>ds5</id>
             <resourcePath>/shared/examples/ds_orders</resourcePath>
             <!--Element runInBackgroundTransaction is optional-->
             <runInBackgroundTransaction>false</runInBackgroundTransaction>
             <reportDetail>SUMMARY</reportDetail>
             <plan>
                 <updateAllIntrospectedResources>true</updateAllIntrospectedResources>
                 <failFast>true</failFast>
```

```
                    <commitOnFailure>true</commitOnFailure>
                        <autoRollback>true</autoRollback>
                    <scanForNewResourcesToAutoAdd>false</scanForNewResourcesToAutoAdd>
                    <!--Element planEntry is optional-->
                    <planEntry>
                        <resourceId>
                                <resourcePath>shippingmethods</resourcePath>
                                <resourceType>TABLE</resourceType>
                                <subtype>DATABASE_TABLE</subtype>
                        </resourceId>
                        <action>REMOVE</action>
                    </planEntry>
                </plan>
        </introspectDataSource>
    </datasource>
```

Execution Sample:

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

```
# ----------------------------------------
# Begin task definition list:
# ----------------------------------------
# Update Datasource
PASS   FALSE  ExecuteAction   introspectDataSources $SERVERID "ds5"
       "$MODULE_HOME/DataSourceModule.xml" "$MODULE_HOME/servers.xml"
```

Results Expected:

PDTool executed the “introspect*DataSources*” for the data source identified by “**ds5**” and the target DV server identified by the variable **“$SERVERID**”. All variables are defined in /resources/config/deploy.properties. The database table “shippingmethods” was removed from the data source.

7 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.

8 Introspection Report

The following sections show example output for the introspection report.

SUMMARY:

This example shows a SUMMARY report.

```
Introspection Report (SUMMARY):
        Status=SUCCESS
    Start Time=2013-08-18T07:56:33.334Z
      End Time=2013-08-18T07:56:33.381Z
         Added=0
       Removed=3
       Updated=0
       Skipped=19
     Completed=22
       Warning=3
        Errors=0
```

SIMPLE:

This example shows a SIMPLE report. The summary information is displayed. The resource and status information are displayed on separate lines with a blank line separating resources.

```
Introspection Report (SIMPLE):
        Status=SUCCESS
    Start Time=2013-08-19T20:24:16.341Z
      End Time=2013-08-19T20:24:16.403Z
         Added=2
       Removed=0
       Updated=0
       Skipped=0
     Completed=2
       Warning=0
        Errors=0

  RESOURCE:  Path=customers   Type=TABLE   Subtype=NONE
    STATUS:  Status=INFO   Action=ADD   Duration=0

  RESOURCE:  Path=shippingmethods   Type=TABLE   Subtype=NONE
    STATUS:  Status=INFO   Action=ADD   Duration=0
```

SIMPLE_COMPRESSED:

This example shows a SIMPLE_COMPRESSED report. The summary information is displayed. The resource path and status information are displayed all on one line.

```
Introspection Report (SIMPLE_COMPRESSED):
        Status=SUCCESS
    Start Time=2013-08-18T07:56:33.334Z
      End Time=2013-08-18T07:56:33.381Z
         Added=0
       Removed=3
       Updated=0
       Skipped=19
     Completed=22
       Warning=3
        Errors=0
```

```
  RESOURCE:  Path=  Type=DATA_SOURCE  Subtype=NONE  [STATUS]:  Status=INFO  Action=SKIP  Duration=0
  RESOURCE:  Path=CISORADEMO  Type=CONTAINER  Subtype=SCHEMA_CONTAINER  [STATUS]:  Status=INFO
Action=SKIP  Duration=0
  RESOURCE:  Path=CISORADEMO/PROCCURSOREXAMPLE  Type=PROCEDURE  Subtype=DATABASE_PROCEDURE
[STATUS]:  Status=INFO  Action=SKIP  Duration=3
  RESOURCE:  Path=CISORADEMO/PROCONEINOUTPARAMETER  Type=PROCEDURE  Subtype=DATABASE_PROCEDURE
[STATUS]:  Status=INFO  Action=SKIP  Duration=3
  RESOURCE:  Path=CISORADEMO/PROCONEINPARAMETER  Type=PROCEDURE  Subtype=DATABASE_PROCEDURE
[STATUS]:  Status=INFO  Action=SKIP  Duration=3
  RESOURCE:  Path=CISORADEMO/PROCONEOUTPARAMETER  Type=PROCEDURE  Subtype=DATABASE_PROCEDURE
[STATUS]:  Status=INFO  Action=SKIP  Duration=3
  RESOURCE:  Path=CISORADEMO/PROCPRINTHELLOWORLD  Type=PROCEDURE  Subtype=DATABASE_PROCEDURE
[STATUS]:  Status=INFO  Action=SKIP  Duration=3
  RESOURCE:  Path=CISORADEMO/CUSTOMERS  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:  Status=INFO
Action=SKIP  Duration=2
  RESOURCE:  Path=CISORADEMO/EMPLOYEES  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:  Status=INFO
Action=SKIP  Duration=2
  RESOURCE:  Path=CISORADEMO/ORDERS  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:  Status=INFO
Action=SKIP  Duration=2
  RESOURCE:  Path=CISORADEMO/ORDERS_HISTORY  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:
Status=INFO  Action=SKIP  Duration=2
  RESOURCE:  Path=CISORADEMO/ORDER_DETAILS  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:  Status=INFO
Action=SKIP  Duration=2
  RESOURCE:  Path=CISORADEMO/ORDER_DETAILS_HISTORY  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:
Status=INFO  Action=SKIP  Duration=2
  RESOURCE:  Path=CISORADEMO/PRODUCT  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:  Status=INFO
Action=SKIP  Duration=2>
  RESOURCE:  Path=CISORADEMO/PRODUCT_CATEGORY  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:
Status=INFO  Action=SKIP  Duration=2
  RESOURCE:  Path=CISORADEMO/SUPPLIER  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:  Status=INFO
Action=SKIP  Duration=2
  RESOURCE:  Path=CISORADEMO/SUPPORTING_DOCUMENTS  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:
Status=INFO  Action=SKIP  Duration=2
  RESOURCE:  Path=CISORADEMO/cache_status  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:  Status=INFO
Action=SKIP  Duration=2
  RESOURCE:  Path=CISORADEMO/cache_tracking  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:  Status=INFO
Action=SKIP  Duration=2
  RESOURCE:  Path=CISORADEMO/Suppliers2  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:  Status=WARNING
Action=REMOVE  Duration=0
  RESOURCE:  Path=CISORADEMO/Suppliers1  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:  Status=WARNING
Action=REMOVE  Duration=0
  RESOURCE:  Path=CISORADEMO/Suppliers0  Type=TABLE  Subtype=DATABASE_TABLE  [STATUS]:  Status=WARNING
Action=REMOVE  Duration=0
```

FULL:

This example shows a FULL report. The summary information is displayed. With the full report any resources that are added will display messages for each column added.

```
Introspection Report (FULL):
        Status=SUCCESS
    Start Time=2013-08-19T20:26:40.713Z
      End Time=2013-08-19T20:26:40.760Z
        Added=1
      Removed=1
      Updated=0
      Skipped=0
    Completed=2
      Warning=0
       Errors=0

  RESOURCE:  Path=customers  Type=TABLE  Subtype=DATABASE_TABLE
    STATUS:  Status=INFO  Action=REMOVE  Duration=0

  RESOURCE:  Path=shippingmethods  Type=TABLE  Subtype=NONE
    STATUS:  Status=INFO  Action=ADD  Duration=0
  MESSAGES:  Severity=INFO  Message=Added column 'ShippingMethodID'.
```

MESSAGES: Severity=INFO Message=Added column 'ShippingMethod'.
MESSAGES: Severity=INFO Message=Added index 'PRIMARY'.

ERROR:

Errors will be displayed with any of the reporting levels. This example shows a report with an error. This error was a result of the plan entry path not being configured as a relative path. In fact, part of the path was duplicated thus DV could not locate the exact path. If an error is detected, the error will be thrown to PDTool.

Resource: Path=cisOraDemoDS/CISORADEMO/EMPLOYEES Type=TABLE Subtype=DATABASE_TABLE
Status: Status=ERROR Action=SKIP Duration=0
Messages: Severity=WARNING Message=Error occurred introspecting new resource. Resource not introspected
Messages: Detail=The resource " /shared/test00/DataSources/cisOraDemoDS/cisOraDemoDS/CISORADEMO " was not found during introspection. Parent containers of introspected resources need to be explicitly listed in the introspection plan if they do not already exist.
at com.compositesw.cdms.datasource.introspect.IntrospectionDriver.createResource(IntrospectionDriver.java:1223)
at com.compositesw.cdms.datasource.introspect.IntrospectionDriver.processResourceBatchToAddInternal(IntrospectionDriver.java:882)
at com.compositesw.cdms.datasource.introspect.IntrospectionDriver.access$200(IntrospectionDriver.java:64)
at com.compositesw.cdms.datasource.introspect.IntrospectionDriver$1.call(IntrospectionDriver.java:718)
at java.util.concurrent.FutureTask$Sync.innerRun(FutureTask.java:303)
at java.util.concurrent.FutureTask.run(FutureTask.java:138)
at java.util.concurrent.ThreadPoolExecutor$Worker.runTask(ThreadPoolExecutor.java:886)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:908)
at java.lang.Thread.run(Thread.java:662)
Messages: Severity=ERROR Code=9901556 Name=api Message=ADD Failed: The resource " /shared/test00/DataSources/cisOraDemoDS/cisOraDemoDS/CISORADEMO " was not found during introspection. Parent containers of introspected resources need to be explicitly listed in the introspection plan if they do not already exist.
Messages: Detail=The resource " /shared/test00/DataSources/cisOraDemoDS/cisOraDemoDS/CISORADEMO " was not found during introspection. Parent containers of introspected resources need to be explicitly listed in the introspection plan if they do not already exist.
at com.compositesw.cdms.datasource.introspect.IntrospectionDriver.createResource(IntrospectionDriver.java:1223)
at com.compositesw.cdms.datasource.introspect.IntrospectionDriver.processResourceBatchToAddInternal(IntrospectionDriver.java:882)
at com.compositesw.cdms.datasource.introspect.IntrospectionDriver.access$200(IntrospectionDriver.java:64)
at com.compositesw.cdms.datasource.introspect.IntrospectionDriver$1.call(IntrospectionDriver.java:718)
at java.util.concurrent.FutureTask$Sync.innerRun(FutureTask.java:303)
at java.util.concurrent.FutureTask.run(FutureTask.java:138)
at java.util.concurrent.ThreadPoolExecutor$Worker.runTask(ThreadPoolExecutor.java:886)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:908)
at java.lang.Thread.run(Thread.java:662)

9 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***".