TIBCO®

PDTool – Regression Automated Test Framework

An Open Source Asset for use with TIBCO® Data Virtualization

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

Revision History

Version	Date	Author	Comments
1.0	06/11/2015	Mike Tinius	Initial revision for Regression Module Automated Test Framework
1.1	10/13/2015	Mike Tinius	Added the "functional_as_is" test type capability.
4.0	12/14/2017	Mike Tinius	Initial revision with Tibco

Related Documents

Name	Author
PDTool Module – Regression.pdf	Mike Tinius

Supported Versions

Name	Version
TIBCO® Data Virtualization	7.0.4 or later

Table of Contents

1   Introduction

Purpose

The purpose of the Regression Module "***Regression Automated Test Framework***" is to make it easier to utilize the PDTool Regression Module testing capability by generating the necessary PDTool plans and module files. The focus of these scripts is on regression testing on a single Data Virtualization (DV) platform and performing comparisons from one DV code release to another.

Audience

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

- Architects

- Developers

- Administrators

- Operations personnel

Platform Support

The Automated Test Framework is only supported on Windows.

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

    o   Cisco Data Virtualization (DV)

    o   Composite Information Server (CIS)

2   PDTool Regression Automated Test Framework

Introduction

Regression testing is the act of testing a DV code release and comparing that release with the previous release.

This framework provides a way to accomplish automated testing for Data Virtualization 6.2 and 7.0 published resources including Views, Procedures and Web Services. This framework is based on the PDTool Deployment and Regression Testing tool.   Documentation for the PDTool can be found within the PDTool installation "docs" directory and the file "PDTool Module - Regression.pdf". There are several types of tests that can be performed with this framework including:

1. Functional Test (Generate and Execute):

   When the "**test type=functional**", this represents the current functionality which tests whether the published view, procedure or web service is functional or not. All that is necessary for this test is to execute a SELECT COUNT(1) cnt FROM <view> or SELECT COUNT(*) cnt FROM <procedure> to prove that the view or procedure is functional. The SQL input file is used as a guideline. When execute the queries are rewritten from the SQL input file and forced to use SELECT COUNT(1) for views and SELECT COUNT(*) for procedures. This type of test is also known as a "smoke" test.

   When the "**test type=functional_as_is**", this represents a capability whereby the SQL input file is used "as is" with no rewriting of the query. If the query contains a projection and where clauses, then those are used as is from the SQL input file.

2. Migration or Regresion Test (Data set and Log Comparison):

   The PD Tool Regression Module will help Administrators during Migrations from one version of DV to another by performing the regression testing on the new version of DV or by performing a Regression test against two releases of a project. These tests are functionally the same. The caveat to this test is that the result sets used for comparison must be derived from the same data sources. Any changes in data values may result in differences in result set and thus comparisons will not be equal.

3. Performance Test (Log Comparison):

   The PD Tool Regression Module can be used by QA to perform performance tests against views, procedures and web services. The two logs generated during the query execution are compared to determine success or failure based on a duration comparison.

4. Security Test (Users and Privileges):

   The PD Tool Regression Module can be used by QA to perform security tests. The security test correlates users with functional test queries. It provides a way to test the expected outcome for a given user and query to determine if the test results in a PASS or FAIL. All

tests are executed and an overall result of PASS or FAIL is awarded. The security test can be configured to throw an exception upon an overall status of FAIL.

Folder Structure

The following screenshot shows the "*Regression Automated Test Framework*" folder structure.   This testing framework provides a wrapper around the PDTool capability.   It allows for an automation of a DV Business Line/Business Area as defined by the user.

It provides a central place for documentation, logs, reports, output and PDTool scripts. This is a cookie-cutter (repeatable) approach to testing. As such there is a templates folder and a script that is used to generate the testing artifacts for a given Business Line/Business Area. The template scripts generate the "docs", "modules", and "plans" that PDTool uses. The "*Regression Automated Test Framework*" folder structure is as follows:
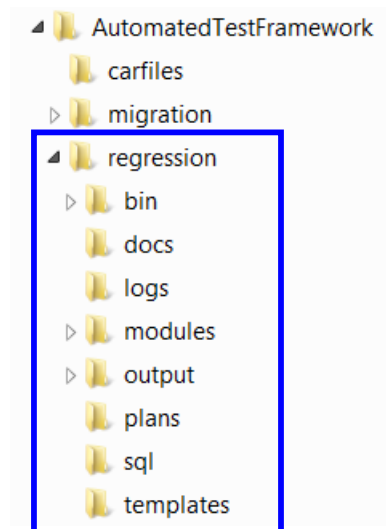

Image 2: Regression Automated Test Framework Folder Structure

The Scripts

The scripts in **"/bin"** provide the basis for the automation. The scripts will be defined in a later section.
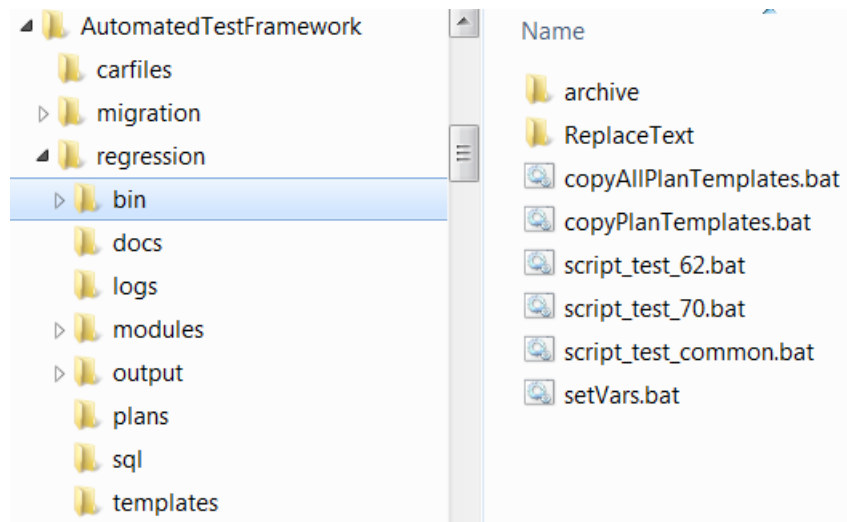
Image 3: Regression Automated Test Framework Scripts

Getting Started

This section discusses how to get started with the basics of testing.

Testing DV Published Resources:

- Requirements:

  - For DV 6.2, PDTool 6.2 is required to be installed.
  - For DV 7.0, PDTool 7.0.0 is required to be installed.

- Planning

  - It is important to plan ahead when using the "*Regression Automated Test Framework*". The following section will help the test user prepare: Planning

- Installation and Configuration Setup

  - Some post-installation configuration is required to use the "*Regression Automated Test Framework*". The following section provides the details: Installation and Configuration Setup
  - Details for installing the sample can be found here: Sample Installation

- Generate Test Scripts

  - Determine the name to give to the set of published views that you want to test. Typically this is known as the "Business Line / Business Area / Subject Area". The scripts will be based on this nomenclature. The name is required to be unique across all other names in use because it is used to generate the scrits.
  - Use the script "*copyPlanTemplates.bat*" to generate the necessary PDTool plans and module files. More information can be found in this section of this document: Script Framework – Template Generation Scripts.
  - Save a list of all of your names using "*copyAllPlanTemplates.bat*" so that you can regenerate all of plans and modules at once if required.

- Determine which DV environment "ENV" you are going to execute against such as:

  - Example: DEV, UAT, PROD.
  - These environments are defined by the DV Administrator.

- Determine which Business Line / Business Area you are testing.

  - Open the generated documentation in /docs for your Business Line /Business Area.
  - E.g. \docs\ documentation_MyProject1SubProj.txt

- Determine which type of test you want to execute.

  - Smoke Test Generation
  - Smoke Test Execution
  - SmokeAsIs Test Eecution
  - Regression Test Execution
  - Regression Test Comparison
  - Performance Test Execution
  - Performance Test Comparison
  - Security Test Generation
  - Security Test Execution

- Refer to \docs\documentation_BusLineBusArea.txt or the example in this document.

  Reference this document: Script Framework – "Test Type" Execution Scripts
  Execute the test

    Refer to the parameters for script execution.   Set the RENAME_REL=false if you want a series of tests to go to the same release folder.

  Reference this document: Script Framework – Baseline Execution Scripts

    - **Example Scenario**: Test 2: Execute Regression Test for UAT 6.2 and MyProject using TOP SQL queries and force the test output to go into the existing "626R1" release folder without renaming it.

    script_test_62.bat UAT MyProject1SubProj_2Regression_exec.dp TOP false true

Script Framework – Template Generation Scripts

The scripts located in "**/bin**" provide the basis for generating the automation scripts from templates.

- ***copyPlanTemplates.bat*** – The purpose of this batch file is to make it easy to copy a *single* plan template for doing regression testing. The premise of this script is to prepare all of the necessary PDTool plans, modules and docs for a given Business Line / Business Area for all of the various PDTool Regression Module tests including: Smoke, Regression, Performance and Security.

1. copyPlanTemplates.bat [BusLineBusAreaSubjArea] [DATA_SOURCE_NAME] [RESOURCE_NAME] [false]

    a. BusLineBusAreaSubjArea - The BusLineBusAreaSubjArea is the portion of the deployment plan file name that occurs prior to the mandatory test type descriptor.
        i. Affix any prefix or postfix desired to BusLineBusAreaSubjArea such as prefix_BusLineBusAreaSubjArea_postfix
        ii. *Test type descriptor*: _1Smoke_gen.dp, _1Smoke_exec.dp, _1SmokeAsIs_exec.dp, _2Regression_exec.dp, _2Regression_compare.dp, _3Performance_exec.dp, _3Performance_compare.dp, _4Security_gen.dp, _4Security_exec.dp
        iii. *Example constructed Deployment plan*: prefix_MyProject1_MySubject_post_3Performance_exec.dp
        iv. |___BusLineBusAreaSubjArea____|
        v. |_____Deployment Plan File Name_____|

b. DATA_SOURCE_NAME - This is published DV data source to connect to for generating or executing queries.

c. RESOURCE_NAME - This may be CATALOG.SCHEMA.* or it may just be CATALOG.*. This is the filter based on Business Line and Business Area. The asterisk indicates a wildcard and is used to specify any views that are published in the location of the preceding resource name.

d. PROMPT_USER - when doing automated set PROMPT_USER=false so that it does not prompt the user to edit the Regression Module XML file.

2. Example1: *Test all views from all FooCat catalog schemas in the same framework*

a. Create a the regression test framework for a new Business Line / Business Area called "Foo" with a published catalog called FooCat in the "MY DB" database. The catalog contains multiple schemas FooSch1 and FooSch2 and the user wants to be able to test all views from all schemas from the same test framework.

b. copyPlanTemplates.bat Foo "MY DB" FooCat.* true

3. Example2: *Test views from each FooCat catalog schema separately in different frameworks.*

a. Create a the regression test framework for a new Business Line / Business Area called "Foo" with a published catalog called FooCat in the "MY DB" database. The catalog contains multiple schemas FooSch1 and FooSch2 and the user wants to be able to test the views for each schema independently of the other schema. To accomplish this we will setup two test frameworks

b. copyPlanTemplates.bat FooSch1 "MY DB" FooCat.FooSch1.* true

c. copyPlanTemplates.bat FooSch2 "MY DB" FooCat.FooSch2.* true

- ▪ ***copyAllPlanTemplates.bat*** – The purpose of this batch file is to make it easy to copy "ALL" of the regression test Business Line / Business Areas at one time specified in the "copyAllPlanTemplates.bat" file. This batch file will overwrite existing deployment plans (/plans), module XML files (/modules) and docs files (/docs).

Script Framework – Baseline Execution Scripts

The scripts located in "**/bin**" provide the basis for executing PDTool standard scripts.

These scripts provide a wrapper around the actual PDTool scripts.   They provide context for the environments [DEV, UAT, PROD]. They output the logs for each execution into /logs directory according to each type of test so that the log is retained across different types of tests.

Types of scripts

- ▪ ***script_test_62.bat*** – Provides an instance of execution for PDTool6.2. Requires its own installation of PDTool6.2. This script can only be run from a single command line window at one time. It cannot be run in parallel from two different command line windows.

- ▪ ***script_test_70.bat*** – Provides an instance of execution for PDTool7.0.0. Requires its own installation of PDTool7.0.0. This script can only be run from a single command line window at one time. It cannot be run in parallel from two different command line windows.

- ▪ Command Line:

- ▪ script_test_NN.bat ENV_TYPE DEPLOYMENT_PLAN [CUSTOM] [RENAME_REL] [PAUSE]

Parameters:

- *ENV_TYPE* – Example: [DEV|UAT|PROD]

  1. The valid values are defined as a result of the variable: VALID_ENV_CONFIG_PAIRS

- *DEPLOYMENT_PLAN* - The name of the deployment plan such as:

  1. BusLineBusArea_1Smoke_gen.dp
  2. BusLineBusArea_1Smoke_exec.dp
  3. BusLineBusArea_1SmokeAsIs_exec.dp
  4. BusLineBusArea_2Regression_exec.dp
  5. BusLineBusArea_2Regression_compare.dp
  6. BusLineBusArea_3Performance_exec.dp
  7. BusLineBusArea_3Performance_compare.dp
  8. BusLineBusArea_4Security_gen.dp
  9. BusLineBusArea_4Security_exec.dp

- *CUSTOM* – [optional] variable

  1. blank or "" – [default]. Generate or execute using SQL SELECT COUNT(1) cnt or SELECT COUNT(*) cnt.

  2. TOP - If the word TOP is provided then generate or execute using the SELECT TOP 1 * command. Top is a special type of CUSTOM.

  3. CUSTOM value - If any other word is used then execute the SQL file using this pattern and the value of the CUSTOM variable:

  - Example: Developer creates a custom SQL file where the custom name = MyQueries

  - Template: Example:

  - \sql\BusLineBusArea_RegressionTest_SQL_%CUSTOM%.txt = \sql\MyProject_RegressionTest_SQL_MyQueries.txt

  - \sql\BusLineBusArea_PerfTest_SQL_%CUSTOM%.txt   = \sql\MyProject_PerfTest_SQL_MyQueries.txt

  -

- *RENAME_REL* – [optional] variable. Default=true

  1. blank or "" or true - force a rename of the release output folders upon each execution of this script.

  2. false - disable the rename function and allow the results to go to the existing RELEASE_FOLDER1 directory.

    - Example. This can be useful when executing a series of tests for the same release such as the following: Smoke Test, Regression Test, Performance Test and Security Test.

    - Note: This script will automatically ignore renaming the output folder for the following and thus the RENAME_REL does not have to be set.

    - Generating Smoke Test, Compare Regression Test, Compare Performance Test and Generate Security Test.

- *PAUSE* – [optional] variable. Default=true

  1. blank or "" or true - pause during script execution.

  2. false - no pause during script execution.

Exiting a script

▪ **How to exit a script** – Use Control-C to exit a script. Type Y and return to exit. If the script is in the middle of executing and you press Control-C the prompt will not be visible. Type "Y" and press the return key to exit the script.

Script Framework – "Test Type" Execution Scripts

The scripts provide the basis for executing PDTool standard scripts.

Script framework "test type" _single_ execution scripts

The section provides information on how to run a single specific type of test. The optional parameters at the end indicate whether to use the SQL queries containing CUSTOM queries ot not. The keyword TOP is classified as a special kind of CUSTOM query that allows the user to execute queries such as "SELECT TOP 1 * ".   The user may wish to specify other CUSTOM keywords that relate to SQL query files where the user has populated the file with their own queries.   If no CUSTOM option is provided, then the "SELECT COUNT(1)" queries are used.

Additionally there is an option for RENAME_REL and PAUSE set to true or false following CUSTOM. RENAME_REL informs the script to rename the release folder or not. If RENAME_REL=false then the user can force the output for a series of tests to be placed into the RELEASE_FOLDER1 which is defined within the script execution batch files [script_test_62.bat and script_test_70.bat].

PAUSE informs the scripts to pause before execution so that the user can see what variables are set before continuing.

▪ Test 1: Generate Smoke Test SQL for ENV

script_test_62.bat ENV BusLineBusArea_1Smoke_gen.dp "" ""   [PAUSE]

script_test_70.bat ENV BusLineBusArea_1Smoke_gen.dp "" ""   [PAUSE]

▪ Test 1: Execute Smoke Test for ENV

script_test_62.bat ENV BusLineBusArea_1Smoke_exec.dp "" [RENAME_REL]   [PAUSE]

script_test_70.bat ENV BusLineBusArea_1Smoke_exec.dp "" [RENAME_REL]   [PAUSE]

▪ Test 1: Execute SmokeAsIs Test for ENV

script_test_62.bat ENV BusLineBusArea_1SmokeAsIs_exec.dp "" [RENAME_REL]   [PAUSE]

script_test_70.bat ENV BusLineBusArea_1SmokeAsIs_exec.dp "" [RENAME_REL]   [PAUSE]

▪ Test 2: Execute Regression Test for ENV

script_test_62.bat ENV BusLineBusArea_2Regression_exec.dp [CUSTOM] [RENAME_REL] [PAUSE]

script_test_70.bat ENV BusLineBusArea_2Regression_exec.dp [CUSTOM] [RENAME_REL] [PAUSE]

▪ Test 2: Execute Regression Test Compare for ENV - Compare files and logs for difference

script_test_62.bat ENV BusLineBusArea_2Regression_compare.dp [CUSTOM] ""   [PAUSE]

script_test_70.bat ENV BusLineBusArea_2Regression_compare.dp [CUSTOM] ""   [PAUSE]

▪ Test 3: Execute Performance Test for ENV

script_test_62.bat ENV BusLineBusArea_3Performance_exec.dp [CUSTOM] [RENAME_REL] [PAUSE]

script_test_70.bat ENV BusLineBusArea_3Performance_exec.dp [CUSTOM] [RENAME_REL] [PAUSE]

- Test 3: Execute Performance Test Compare for ENV - Compare logs for difference

script_test_62.bat ENV BusLineBusArea_3Performance_compare.dp [CUSTOM] ""   [PAUSE]

script_test_70.bat ENV BusLineBusArea_3Performance_compare.dp [CUSTOM] ""   [PAUSE]

- Test 4: Generate Security Test for ENV

script_test_62.bat ENV BusLineBusArea_4Security_gen.dp "" ""   [PAUSE]

script_test_70.bat ENV BusLineBusArea_4Security_gen.dp "" ""   [PAUSE]

- Test 4: Execute Security Test for ENV

script_test_62.bat ENV BusLineBusArea_4Security_exec.dp "" [RENAME_REL] [PAUSE]

script_test_70.bat ENV BusLineBusArea_4Security_exec.dp "" [RENAME_REL] [PAUSE]

- **EXAMPLE 1**:

- Scenario: Test 2: Execute Regression Test for UAT 6.2 and MyProject using TOP SQL queries and force the test output to go into the existing "626R1" release folder without renaming it.

script_test_62.bat UAT MyProject1SubProj_2Regression_exec.dp TOP false true

- **EXAMPLE 2**:

- Scenario: Test 2: Execute Regression Test for UAT 6.2 and MyProject using cusom SQL queries and force the test output to go into a new "626R1" release folder.

script_test_62.bat UAT MyProject1SubProj_2Regression_exec.dp MyCustom ""

Script Framework – Documentation

The "**/docs**" folder contains all of the Business Line / Business Area generated documentation for all of the tests. It was prepared by the "**copyPlanTemplates.bat**" script. The file "**Regression Test Summary.xlsx**" provides a place to tally test results.
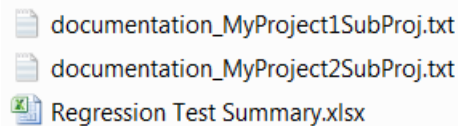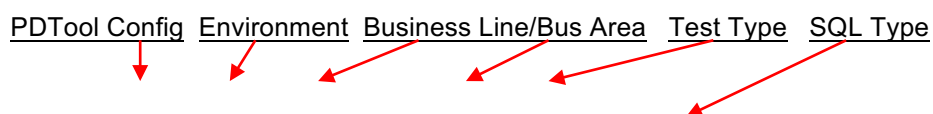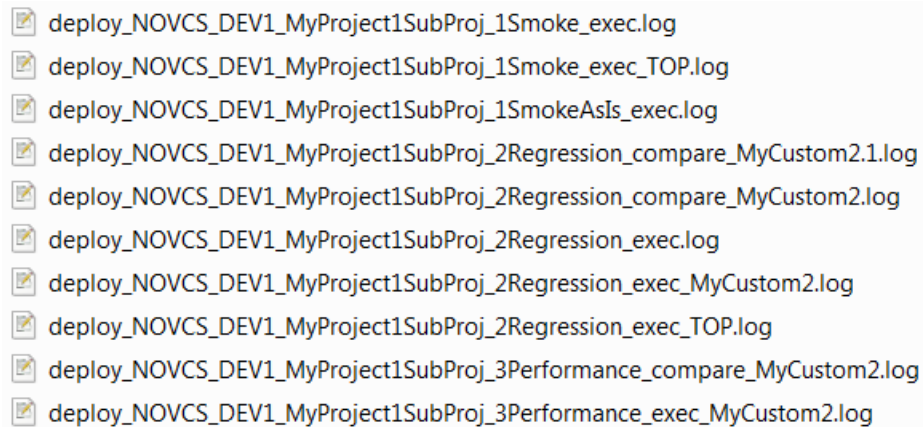
    documentation_MyProject1SubProj.txt
    documentation_MyProject2SubProj.txt
    Regression Test Summary.xlsx

Image 4: Documentation for Business Line / Business Area

Script Framework – Log

The "**/logs**" folder contains output for a given DV version, environment, Business Line/Area, Test Type and SQL test type.   In the screenshot below each part of the log file path is described. This allows for a high degree of naming so that the logs can be preserved across different test types. However, if the exact same test is run again then it will overwrite an existing log of the same name.
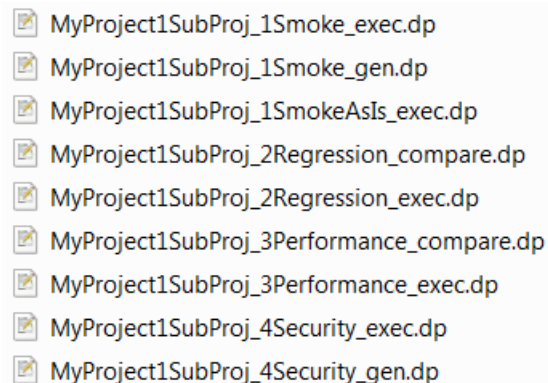
PDTool Config   Environment   Business Line/Bus Area   Test Type   SQL Type

deploy_NOVCS_DEV1_MyProject1SubProj_1Smoke_exec.log
deploy_NOVCS_DEV1_MyProject1SubProj_1Smoke_exec_TOP.log
deploy_NOVCS_DEV1_MyProject1SubProj_1SmokeAsIs_exec.log
deploy_NOVCS_DEV1_MyProject1SubProj_2Regression_compare_MyCustom2.1.log
deploy_NOVCS_DEV1_MyProject1SubProj_2Regression_compare_MyCustom2.log
deploy_NOVCS_DEV1_MyProject1SubProj_2Regression_exec.log
deploy_NOVCS_DEV1_MyProject1SubProj_2Regression_exec_MyCustom2.log
deploy_NOVCS_DEV1_MyProject1SubProj_2Regression_exec_TOP.log
deploy_NOVCS_DEV1_MyProject1SubProj_3Performance_compare_MyCustom2.log
deploy_NOVCS_DEV1_MyProject1SubProj_3Performance_exec_MyCustom2.log

Image 5: Log Naming Conventions

Script Framework – Plans

The "**/plans**" folder contains all of the deployment plan files for all of the tests. It was prepared by the copyAllPlanTemplates.bat script.
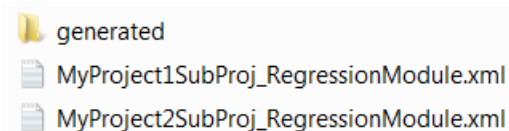
MyProject1SubProj_1Smoke_exec.dp
MyProject1SubProj_1Smoke_gen.dp
MyProject1SubProj_1SmokeAsIs_exec.dp
MyProject1SubProj_2Regression_compare.dp
MyProject1SubProj_2Regression_exec.dp
MyProject1SubProj_3Performance_compare.dp
MyProject1SubProj_3Performance_exec.dp
MyProject1SubProj_4Security_exec.dp
MyProject1SubProj_4Security_gen.dp

Image 6: Deployment Plan Files

Script Framework – Modules

The "**/modules**" folder contains all of the deployment module files for all of the tests. It was prepared by the copyAllPlanTemplates.bat script.

generated
MyProject1SubProj_RegressionModule.xml
MyProject2SubProj_RegressionModule.xml

Image 7: Deployment Module Files

Script Framework – SQL Queries

The "**/sql**" folder contains all of the SQL query files for all of the tests. The Smoke test files were generated.   The Regression and Performance files were prepared manually so that the tester can

control what they want to test. Custom queries can be provided or the tester can copy and paste from the Smoke Test query set. For custom files add a custom name following "SQL_" as shown below in a file name such as "MyProject1SubProj_RegressionTest_SQL_**MyCustom**.txt". Not "_TOP" is a special kind of custom file that contains "select TOP 1 *" queries.
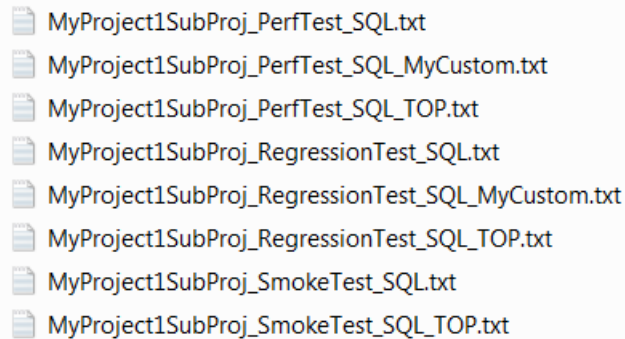
MyProject1SubProj_PerfTest_SQL.txt
MyProject1SubProj_PerfTest_SQL_MyCustom.txt
MyProject1SubProj_PerfTest_SQL_TOP.txt
MyProject1SubProj_RegressionTest_SQL.txt
MyProject1SubProj_RegressionTest_SQL_MyCustom.txt
MyProject1SubProj_RegressionTest_SQL_TOP.txt
MyProject1SubProj_SmokeTest_SQL.txt
MyProject1SubProj_SmokeTest_SQL_TOP.txt

Image 8: SQL Query Files

Script Framework – Output

The "**/output**" folder contains a folder for each Business Line / Business Area. That subfolder contains the result logs for each test type as well as subfolders for the regression tests which contain the data files for each query.
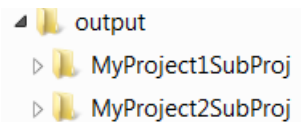
output
MyProject1SubProj
MyProject2SubProj

Image 9: Output Folders

For example, let's use *MyProject1SubProj* as a guide. The screen shot below shows several release folders for both 6.2 and 7.0.   The folder "701R1" is the RELEASE_FOLDER1 and "701R2" is RELEASE_FOLDER2 as specified in the script execution batch files. The folders with ".0001" and so on represent the archive of the "701R2" folders:

626R1
626R2
626R2.0001
626R2.0002
701R1
701R2
701R2.0001
701R2.0002

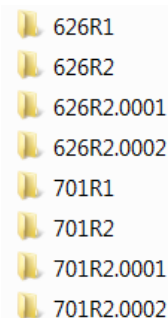Image 10: Contents of a Business Line / Business Area Folder

The screen shot below shows an example of the content of a release folder such as "701R1" or "701R2". The release folder contains one or more logs and sub-folders where the query output is

stored for each test type segregated by environment type. All log files are prefixed with the environment type so that any given release can be tracked across multiple DV environments.
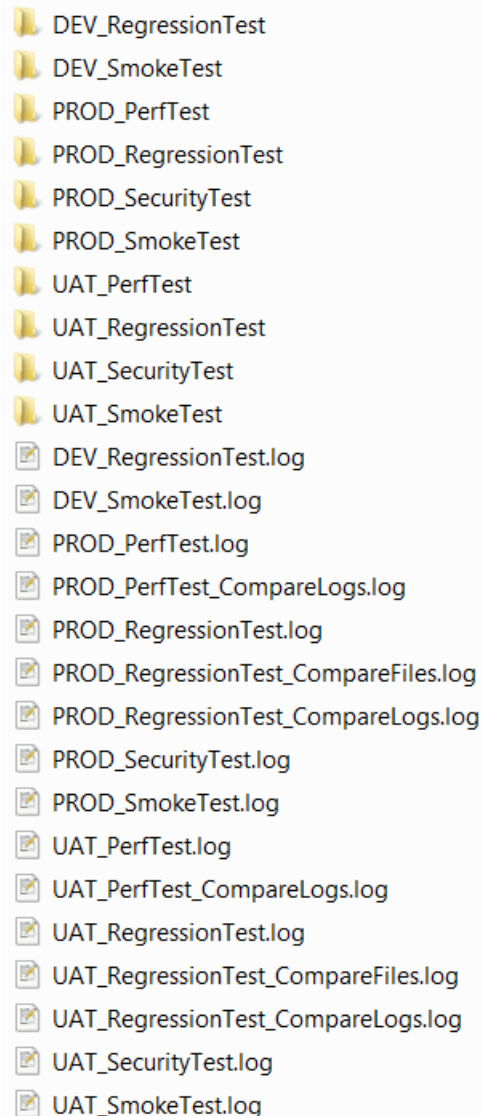

DEV_RegressionTest
DEV_SmokeTest
PROD_PerfTest
PROD_RegressionTest
PROD_SecurityTest
PROD_SmokeTest
UAT_PerfTest
UAT_RegressionTest
UAT_SecurityTest
UAT_SmokeTest
DEV_RegressionTest.log
DEV_SmokeTest.log
PROD_PerfTest.log
PROD_PerfTest_CompareLogs.log
PROD_RegressionTest.log
PROD_RegressionTest_CompareFiles.log
PROD_RegressionTest_CompareLogs.log
PROD_SecurityTest.log
PROD_SmokeTest.log
UAT_PerfTest.log
UAT_PerfTest_CompareLogs.log
UAT_RegressionTest.log
UAT_RegressionTest_CompareFiles.log
UAT_RegressionTest_CompareLogs.log
UAT_SecurityTest.log
UAT_SmokeTest.log

Image 11: Contents a release folder such as "701R1" or "701R2"

Description of the various log files

ENV_PerfTest.log or ENV_PerfTestTOP.log – This is the performance test log for a given release [701R1] as a result of executing regular queries or TOP-n queries. It logs the duration of execution for each thread along with the query executed over a defined period of time. It logs any errors that occur.

ENV_PerfTest_CompareLogs.log or ENV_PerfTestTOP_CompareLogs.log – Provides a comparison between two different releases [701R1 and 701R2] for a given DV version of *performance* logs for regular or TOP-n queries to determine [for the same query] the following:

▪ Duration matched (same)

- Duration improved (performance improved)

- Duration was within acceptable range (user defined)

- Duration exceeded range

ENV_RegressionTest.log or ENV_RegressionTestTOP.log – This is the regression test log for a given release [701R1] as a result of executing regular queries or TOP-n queries. It only executes the query once. It logs duration of execution for each query along with the query executed. It logs any errors that occur.

ENV_RegressionTest_CompareFiles.log or **ENV_RegressionTestTOP_CompareFiles.log** – Provides a comparison between two different releases [701R1 and 701R2] for a given DV version of *data* files for regular or TOP-n queries to determine [for the same query] whether the files are an exact match or not. If the row-by-row checksums match then the query is marked with "SUCCESS" otherwise it is marked with "FAILURE". The key to this test is to test against the same data set for both releases [701R1 and 701R2].

ENV_RegressionTest_CompareLogs.log or **ENV_RegressionTestTOP_CompareLogs.log** – Provides a comparison between two different releases [701R1 and 701R2] for a given DV version of *regression* logs for regular or TOP-n queries to determine [for the same query] the following:

- Duration matched (same)

- Duration improved (performance improved)

- Duration was within acceptable range (user defined)

- Duration exceeded range

Regression Test Data Files:

The screen shot below demonstrates how a data file is created. A subfolder is created to represent the type of test and version such as "701R1\ENV_RegressionTest_TOP" so that tests can be run against either 6.2 or 7.0. Underneath that folder is another with the same name as the DV Data Source being tested such as "MY DB". Underneath "MY DB" are the data files for each query and named according to Catalog.Schema.Table name.
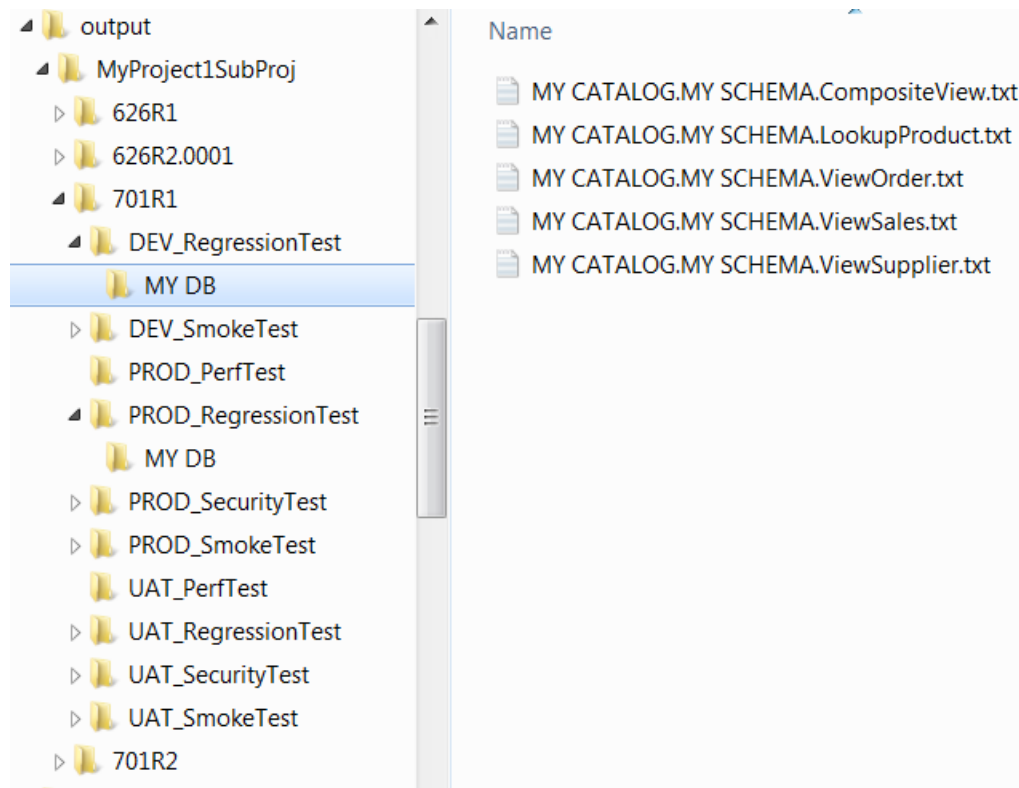
Image 12: Regression Test Data Files

Performance Test Data Files

Even though a subfolder is created for performance test such as "701R1\PerfTest_TOP" there are no data files created.

Security Test Configuration

The security test correlates users with functional test queries and provides a way to test the expected outcome for a given user/query combination to determine if the test results in a PASS or FAIL. The security test does not use a separate SQL input file like the other tests use. Instead all of the users, queries and correlated tests are defined within the Regression Module XML file. Since this is originally generated from a template, it will not have the correct values to start out with. There is a process involved to generate the queries and then configure them to meet the test requirements. Let's consider this high-level overview of the process involved to configure a security test:

1. **Configure** the list of **users** in the **regression module XML** file that was derived from the template "BusLineBusAreaSubjArea_RegressionModule.xml".

2. **Generate** the **security queries** for the desired Business Line / Area / Subject Area.

3. **Copy generated security queries** into the regression module XML file

4. **Edit expected results** and **assign** the appropriate **PASS/FAIL response**.

Scenario: The development and/or QA team wants to validate that the privileges are set correctly on the various folder structures. When it comes to security, the following are guidelines or best practices:

- o   There are representative test resources in each of the folders to be tested.

- o   The development team has setup groups and a corresponding test user in each of the groups.

- o   Privileges are assigned to groups which are applied to folders recursively.

Here is a high-level summary of the steps for the "Security Test Process":

Note: Refer to the PDTool documentation "**PDTool Module - Regression.pdf**" found in the PDTool/docs directory. Refer to the section titled "**4. Security Test Process:**" for the details.

- o   Step 1 – Regression Module XML for generating the regression security XML file.

- o   Step 2 – Deployment plan for generating the regression security XML file.

- o   Step 3 – Configure user list.

- o   Step 4 – Execute the Regression Security Generation plan

- o   Step 5 – Copy generated XML

- o   Step 6 – Regression Module XML for executing a Security test

- o   Step 7 – Deployment plan for "Executing the security test"

- o   Step 8 – Execute the Security Test plan.

- o   Step 9 – Review the results of the output log file.

Security Test Details
Step 1 – [*NO ACTION REQUIRED*] Regression Module XML was generated for a given project (Business Line / Area / Subject Area).
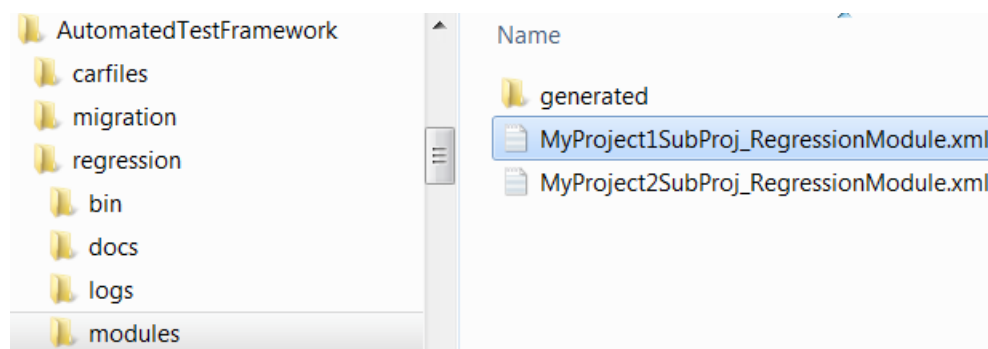

Image 13: Regression Module XML generated files

Step 2 – [*NO ACTION REQUIRED*] Deployment plan for "Generating the security XML" was generated for a given project (Business Line / Area / Subject Area).
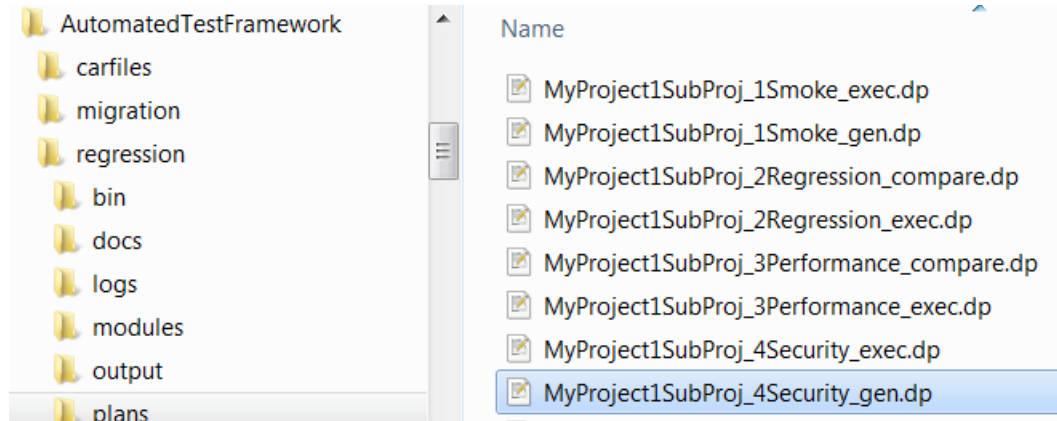

Image 14: Deployment plan generated files

Example deployment plan file entry for MyProject1SubProj_4Security_gen.dp:

```
PASS    TRUE    ExecuteAction  generateRegressionSecurityXML $SERVERID
        Test4.1_$SCRIPT_CIS_VERSION
        "$REGRESSION_TEST_MODULES/%BusLineBusAreaSubjArea%_RegressionModule.xml
        "  "$MODULE_HOME/servers.xml"
```

Step 3 – Configure user list

- Edit the Regression Module XML.

- Add a list of users that you want to use for testing security. These can be DV users or ldap users. Provide the user name, password and domain. Passwords may be encrypted after this part of the exercise is complete.

- There are two separate XML identifiers for DV 6.2 and DV 7.0. Configure one or both sections depending on which version of DV is being tested.

```
<regressionTest>
    <id>Test4.1_6.2</id> OR <id>Test4.1_7.0</id>
    <securityGenerationOptions>
        <!-- A security user default encrypted password. It will be encrypted when the
            ExecutePDTool.bat -encrypt ..\modules\RegressionModule.xml is executed.-->
        <encryptedPassword>password</encryptedPassword>
        <!-- Determines what DV users to generate.  Wildcards (*) may be used.  -->
        <userFilter>user1,user2</userFilter>
        <!-- Provides a way of specifying what domain the userFilter should be applied to.  -->
        <domainFilter>composite</domainFilter>
    </securityGenerationOptions>
</regressionTest>
```

Step 4 – Generate the Regression Security XML.
- Execute the Regression Security Generation plan which will execute the method "generateRegressionSecurityXML" to generate users, queries and security plan tests.

  Example Execute Line:

  script_test_70.bat UAT MyProject1SubProj_4Security_gen.dp "" "false"

Step 5 – Modify original Regression Module XML
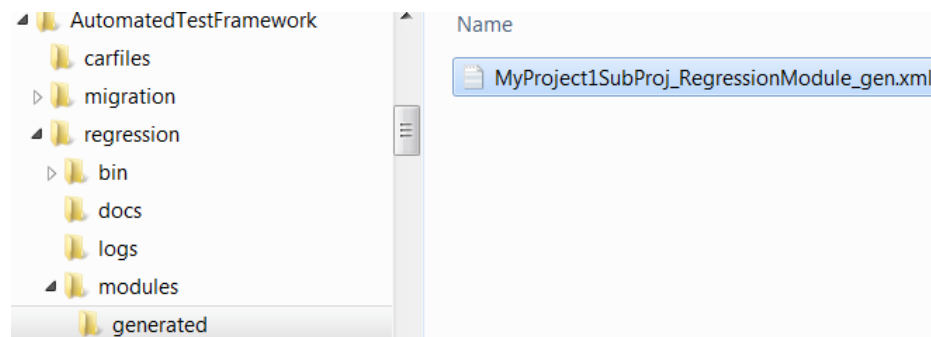- Copy the generated regression XML file section: "<regressionSecurity>" to the original Regression Module XML.

Image 15: Generated Regression Module XML

Image 16: Example of XML section that is to be copied and pasted

- Edit user list and supply correct password if necessary.

Example Regression Module XML (MyProject1SubProj_RegressionModule.xml):

```
<regressionSecurity>
   <!-- List of users to be used for security testing. -->
   <regressionSecurityUsers>
      <!—Iteration of users for project: MyProject1SubProj -->
      <regressionSecurityUser>
         <id>rsu1</id>
         <userName>user1</userName>
         <encryptedPassword>password</encryptedPassword>
         <domain>composite</domain>
      </regressionSecurityUser>
      <regressionSecurityUser>
         <id>rsu2</id>
         <userName>user3</userName>
         <encryptedPassword>password</encryptedPassword>
         <domain>composite</domain>
      </regressionSecurityUser>
   </regressionSecurityUsers>
</regressionSecurity>
```

- Re-encrypt password if necessary

```
ExecutePDTool.bat -encrypt
..\AutomatedTestFramework\regression\modules\MyProject1SubProj_RegressionModule.xml
```

- Validate Regression Security Plan tests – Validate the expected outcome for each plan test to insure that it is the correct outcome [PASS or FAIL]. This is a manual step performed by the developer, tester or administrator.

- Make a backup copy of the XML file in case you decide to regenerate the project files in the future.

Step 6 – [*NO ACTION REQUIRED*] Regression Module XML for "Executing the security test" was generated for a given project (Business Line / Area / Subject Area).
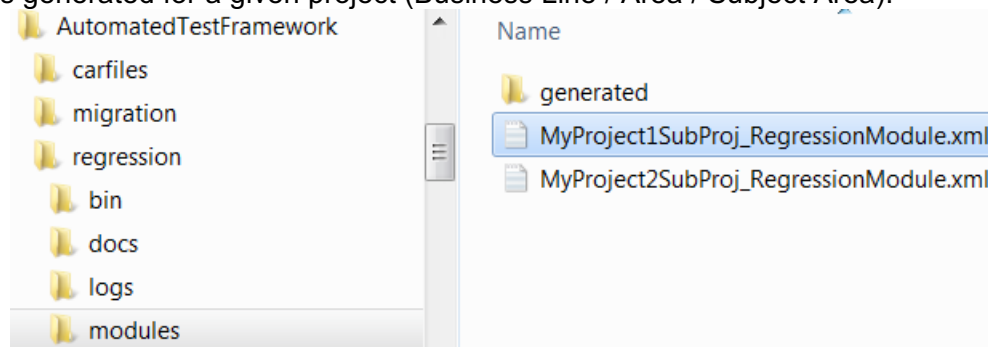
Image 17: Regression Module XML generated files

Step 7 – [*NO ACTION REQUIRED*] Deployment plan for "Executing the regression security test" was generated for a given project (Business Line / Area / Subject Area).
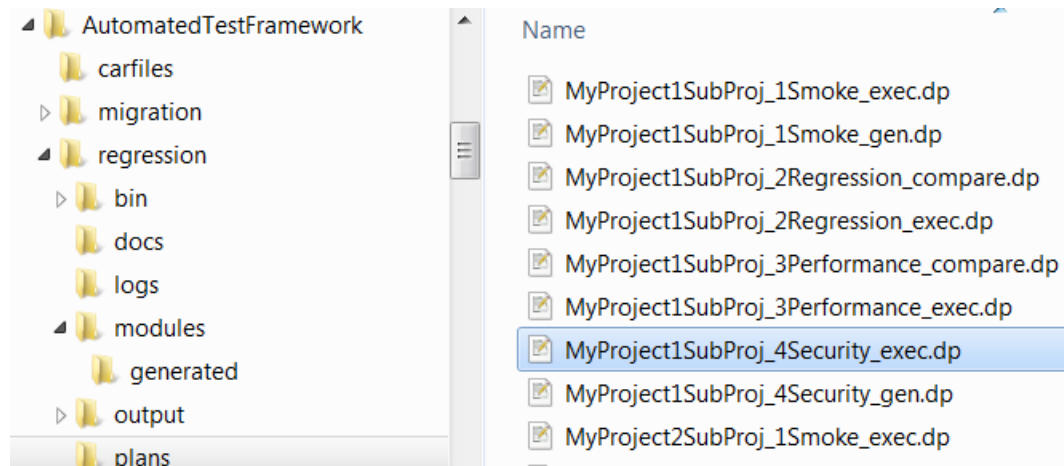

Image 18: Deployment Plan for Security Execution

- Example deployment plan file entry for MyProject1SubProj_4Security_exec.dp:

```
PASS TRUE ExecuteAction executeSecurityTest $SERVERID
Test4.1_$SCRIPT_CIS_VERSION
"$REGRESSION_TEST_MODULES/%BusLineBusAreaSubjArea%_RegressionModule.xml"
"$MODULE_HOME/servers.xml"
```

Step 8 – Execute the Regression Security Test plan.
- Executing the regression test is accomplished by running the Automated Test Framework scripts.

- This regression security test will use the queries generated in the regression security XML and execute them against DV. It will log the results into a summary execution file in the file system. It will use several filters that you previously set up in the regression XML file. For example, you can turn on and off execution of the three categories, queries, procedures and web services. You can also filter on which datasources you execute tests for. The input file may contain a broad list of datasources and queries but you don't have to run all the tests.

Example Execute Line:
script_test_70.bat UAT MyProject1SubProj_4Security_exec.dp   ""   "false"

Step 9 – Review the results of the output file.

The results of the security test are saved the Automated Test Framework output folder under the project name and release sub-folders. The actual log file is called SecurityTest.log.

The following screen shot shows the location of the log file.
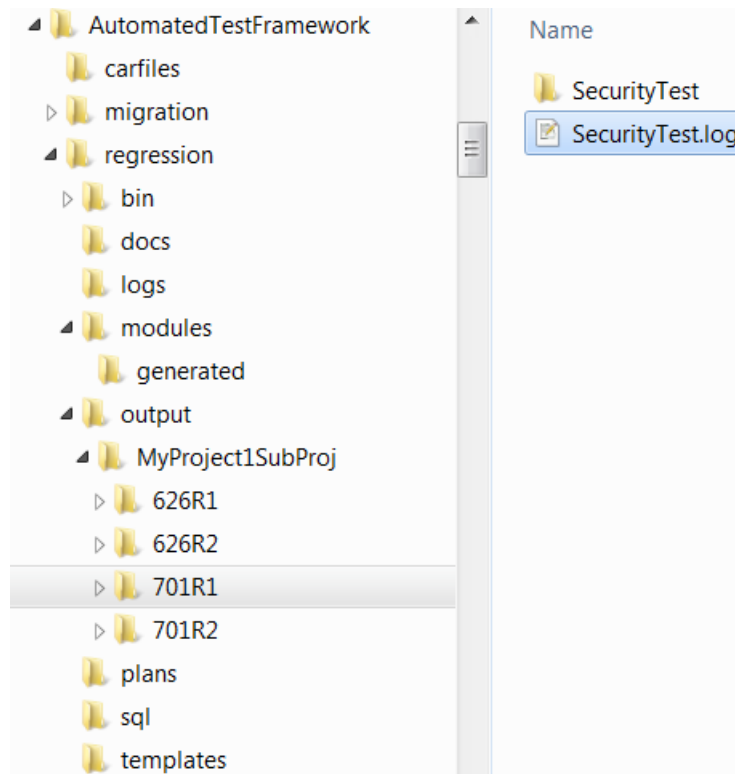

Image 19: Security Test Log Output Location – SecurityTest.log

The following screen shot shows the content of the SecurityTest.log file:

Image 20: Security Test Log Output
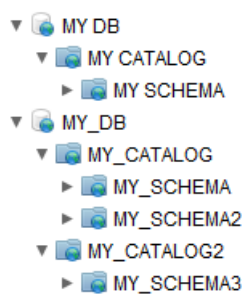
3   Installation and Configuration

Introduction

This section describes how to install and configure the "***Regression Automated Test Framework***".

Planning

The section helps the test user to plan for how they want to use the testing framework.

Planning worksheet:

1. Identify Published Resources to test: _____

    a. Example published data source, catalogs and schemas

   ▼ 🌐 MY DB
     ▼ 📁 MY CATALOG
       ▶ 📁 MY SCHEMA
   ▼ 🌐 MY_DB
     ▼ 📁 MY_CATALOG
       ▶ 📁 MY_SCHEMA
       ▶ 📁 MY_SCHEMA2
     ▼ 📁 MY_CATALOG2
       ▶ 📁 MY_SCHEMA3

2. Assign a name to the published resources: _____

    a. Known as the Business Line or Business Area or Subject Area

3. Determine the DV environment names: _____

    a. Example: DEV, UAT PROD

4. Determine the PDTool config file mappings: _____

    a. This is how a PDTool configuration property file maps to a given DV environment.

    b. Example:

        i. <u>ENV</u>   <u>PDTool Configuration Property File Name</u>

        ii. DEV   = deploy_7.0_DEV.properties

        iii. UAT   = deploy_7.0_UAT.properties

        iv. PROD   = deploy_7.0_PROD.properties

5. PDTool 6.2 directory location: _____

6. PDTool 7.0 directory location: _____

Installation and Configuration Setup

The section provides the necessary steps for setting up the batch scripts used for testing.

Edit/Configure the **setVars.bat** batch file:

1. Location: \AutomatedTestFramework\regression\bin\setVars.bat

REM # Set environment variables for PDTool 6.2

REM # Set the location of PDTool 6.2

set PDTOOL_INSTALL_HOME_6=C:\PDTool_Test\PDTool6.2

REM List of valid Environments~Config property file name pairs.   Comma separated no space and no double quotes. Tilde separates pairs: ENV~ConfigFileName. These are the property file names configured in the PDTool6.2\resources\config folder minus the .properties extension.

set VALID_ENV_CONFIG_PAIRS_6=DEV~deploy_6.2_DEV,UAT~deploy_6.2_UAT,PROD~deploy_6.2_PROD

REM Set the release folders to indicate which version is being tested

REM   Release folder 1 is designated as the DV instance current folder. R1 designates it is a release 1 primary, current folder.

REM   Release folder 2 is designated as the DV instance previous folder. R2 designates it is a release 2 secondary, previous folder.

set RELEASE_FOLDER1_6=626R1

set RELEASE_FOLDER2_6=626R2

REM # Set environment variables for PDTool 7.0

REM Set the location of PDTool 7.0

set PDTOOL_INSTALL_HOME_7=C:\PDTool_Test\PDTool7.0.0

REM List of valid Environments~Config property file name pairs.   Comma separated no space and no double quotes. Tilde separates pairs: ENV~ConfigFileName. These are the property file names configured in the PDTool7.0.0\resources\config folder minus the .properties extension.

set VALID_ENV_CONFIG_PAIRS_7=DEV~deploy_7.0.1_DEV,UAT~deploy_7.0.1_UAT,PROD~deploy_7.0.1_PROD

REM Set the release folders to indicate which version is being tested.

REM   Release folder 1 is designated as the DV instance current folder. R1 designates it is a release 1 primary, current folder.

REM   Release folder 2 is designated as the DV instance previous folder. R2 designates it is a release 2 secondary, previous folder.

set RELEASE_FOLDER1_7=701R1

set RELEASE_FOLDER2_7=701R2

REM # Set environment variables for Automated Test Framework

REM Automated Test Framework Home. This folder may be independent of where PDTOOL_INSTALL_HOME is located.

set ATF_HOME= C:\PDTool_Test\PDTool6.2\AutomatedTestFramework\regression

REM # Used by copyPlanTemplates.bat

REM # Set JAVA_HOME to JRE7

if not defined JAVA_HOME set JAVA_HOME=C:\Program Files\Java\jre7

REM # Used by copyPlanTemplates.bat

REM # Use one or the other or provide your own text editor path.

REM # If you have notepad++ it is a much better editor than notepad.

REM # Do not put double quotes around path. The script takes care of that.

set EDITOR=C:\Program Files (x86)\Notepad++\notepad++.exe

rem set EDITOR=%windir%\system32\notepad.exe

REM Script Main Activity

set SCRIPT_ACTIVITY=Execute Regression Test

REM Debug=Y or N. Default=N

set DEBUG=N

Sample Installation

The section provides the necessary steps for setting up the sample.

Create groups on the target server where the sample will be imported:

1. Launch DV Manager and login

    a. For 6.2, select "Users → Group Management"

    b. For 7.0, select "Security → Group Management"

    c. Select "Add Group"

        i. Group name: **group1**

        ii. Click OK

    d. Select "Add Group"

        i. Group name: **group2**

        ii. Click OK

Create users on the target server where the sample will be imported:

1. Note: The users "user1" and "user2" use the password text "password" within the sample files. If you wish to change the password, then you will need to provide the same password in the sample files

    a. \AutomatedTestFramework\regression\modules

        i. \MyProject1SubProj_RegressionModule.xml

        ii. \MyProject2SubProj_RegressionModule.xml

    b. Encrypting passwords in files:

    ExecutePDTool.bat -encrypt
    ..\AutomatedTestFramework\regression\modules\MyProject1SubProj_RegressionModule.xml

    ExecutePDTool.bat -encrypt
    ..\AutomatedTestFramework\regression\modules\MyProject2SubProj_RegressionModule.xml

2. Launch DV Manager and login

    a. For 6.2, select "Users → User Management"

    b. For 7.0, select "Security → User Management"

    c. Select "Add Group"

        i. User name: **user1**

        ii. New password: **password**

        iii. Confirm password: **password**

        iv. Click OK

    d. Select "Add Group"

        i. User name: **user2**

        ii. New password: **password**

        iii. Confirm password: **password**

        iv. Click OK

Associate users with groups on the target server where the sample will be imported:

1. Launch DV Manager and login

    a. For 6.2, select "Users → User Management"

    b. For 7.0, select "Security → User Management"

    c. Check the box next to "**user1**" and click "Edit Group Membership"

        i. Check the box for "**group1**"

        ii. Click OK

    d. Check the box next to "**user2**" and click "Edit Group Membership"

        i. Check the box for "**group2**"

        ii. Click OK

Import the sample CAR file:

1. Launch DV Studio 6.2 or 7.0

2. Right click on "Desktop (user)" and select Import

3. Browse to the location of \AutomatedTestFramework\carfiles

4. Select MySampleDBs.car and click Open

5. Click Import>

6. Click Done

Verify Privileges:

1. Launch DV Studio 6.2 or 7.0

2. Right-click on "/services/databases/MY DB" and select "Privileges"

3. Select the middle radio button "Hide users and groups without explicit privileges" and click OK

4. The privileges should be as follows:

group1	☑	☐	☑	☑	☐	☐	☐	☐

5. Right-click on "/services/databases/MY_DB" and select "Privileges"

6. Select the middle radio button "Hide users and groups without explicit privileges" and click OK

7. The privileges should be as follows:

group2	☑	☐	☑	☑	☐	☐	☐	☐

4 Changing the PDTool Password

Introduction

This section describes how to change your PDTool password.

PDTool 6.2

1. Locate the PDTool 6.2 folder

   C:\Users\%USERNAME%\.compositesw\PDTool\PDTool6.2

2. Edit the variables file – setMyPrePDToolVars.bat

   a. Locate the variable CIS_PASSWORD

      set CIS_PASSWORD=Encrypted:6ABCABC9F46BAA2

   b. Modify the password by removing all text after the "=" including the "Encrypted:"

      set CIS_PASSWORD=

   c. Enter your new password.

   d. Save and close the file

3. Open a command line

   a. Change directories to the following location

      cd C:\Users\%USERNAME%\.compositesw\PDTool\PDTool6.2\bin

   b. Execute the following command

      ExecutePDTool.bat -encrypt ..\..\setMyPrePDToolVars.bat

4. Edit the variables file – setMyPrePDToolVars.bat

   a. Locate the variable CIS_PASSWORD and determine if the password is now encrypted.

PDTool 7.0

1. Locate the PDTool 7.0.0 folder

   C:\Users\%USERNAME%\.compositesw\PDTool\PDTool7.0.0

2. Edit the variables file – setMyPrePDToolVars.bat

   b. Locate the variable CIS_PASSWORD

      set CIS_PASSWORD=Encrypted:6ABCABC9F46BAA2

   c. Modify the password by removing all text after the "=" including the "Encrypted:"

      set CIS_PASSWORD=

      d.   Enter your new password.

      e.   Save and close the file

3.   Open a command line

      a.   Change directories to the following location

         cd C:\Users\%USERNAME%\.compositesw\PDTool\PDTool7.0.0\bin

      b.   Execute the following command

         ExecutePDTool.bat -encrypt ..\..\setMyPrePDToolVars.bat

4.   Edit the variables file – setMyPrePDToolVars.bat

      f.   Locate the variable CIS_PASSWORD and determine if the password is now encrypted.

5   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***".