

Test Case Subscription Capability
for
Contract DG133W-05-CQ-1067
Advanced Weather Interactive Processing System (AWIPS)
Operations & Maintenance

AWP.TE.SWCTR/TO10-0021

Prepared for:

U.S. Department of Commerce
NOAA/NWS Acquisition Management Division
SSMC2, Room 11220
1325 East-West Highway
Silver Spring, MD 20910

Prepared by:

Raytheon Company
STC Office
6825 Pine Street
Omaha, NE 68106

6 February 2009

This document includes data that shall not be duplicated, used, or disclosed – in whole or in part – outside the Government for any purpose other than to the extent provided in contract DG133W-05-CQ-1067. However, the Government shall have the right to duplicate, use, or disclose the data to the extent provided in the contract. This restriction does not limit the Government's right to use information contained in this data if it is obtained from another source without restriction. The data subject to this restriction are contained in all sheets.

HARD COPY UNCONTROLLED

Submitted By:

Test Engineer

Date

Approved By:

Program Manager

Date

Mission Assurance Quality

Date

Change History

Revision	Date	Affected Pages	Explanation of Change
Draft	21 Nov. 2008	ALL	Initial Draft
1	10 Jan. 2009	ALL	Result of NWS comments and PDT.
2	6 Feb. 2009	iii, 4, 5	Result of DT

Table of Contents

	<i>Page</i>
1.0 SCOPE	1
2.0 APPLICABLE DOCUMENTS	2
2.1 Source Documents	2
2.2 Reference Documents	2
3.0 TEST CASE DESCRIPTION.....	3
3.1 Assumptions, Constraints and Preconditions.....	3
3.2 Recommended Hardware.....	3
3.3 Test Inputs.....	3
3.4 Test Outputs	3
4.0 TEST SCENARIO	4
5.0 REQUIREMENTS VERIFICATION TRACEABILITY MATRIX (RVTM).....	4

1.0 SCOPE

See TO10 Software Test Plan.

2.0 APPLICABLE DOCUMENTS

2.1 Source Documents

- None.

2.2 Reference Documents

- TO10 Software Test Plan for the Advanced Weather Interactive Processing System Project, Contract #DG133W-05-CQ-1067, January 2009.
- The Silver Spring NWS AWIPS 1 test bed application.
- Rational RequisitePro.

3.0 TEST CASE DESCRIPTION

This test case illustrates the AWIPS II subscription capability that replaces the AWIPS I database trigger capability; the test is accomplished via inspection of the file system and/or logs.

3.1 Assumptions, Constraints, and Preconditions

- TO10 software has been installed successfully.
- CAVE, EDEX and pgAdmin III are running.
- Data has been ingested.
- Actions, Results, and Requirements highlighted in gray indicate requirements and/or capabilities to be included in the scope of future task orders. They are included here for purposes of continuity and traceability with the original AWIPS I test case documents.

3.2 Recommended Hardware

See TO10 Software Test Plan.

3.3 Test Inputs

Section 4.0 contains the test procedures for this test case. Sections 2.2 – 2.9 of the TO10 Software Test Plan contain general test inputs applicable to all TO10 test cases.

3.4 Test Outputs

The results outlined in section 4.0 are met.

4.0 TEST SCENARIO

Step #	Action	Result	Pass/Fail
1.	Launch a terminal window.	Terminal window is at user prompt.	
2.	ssh root@awips-xxx	You are prompted for password.	
3.	Enter password to log in. cd /awips/fxa/bin	You are at the awips-int1 # sign prompt in the directory with the command line interfaces scripts.	
4.	Open the PG Admin database tool. Connect to the testbed database and select the metadata database. Expand the database tree until the tables are displayed by selecting schema, subscription, and tables.	The database tables are displayed.	
5.	Query the subscriptions table.	The data displayed are the subscriptions stored in the database.	
Scenario 1: Create subscript for timer, product, and LDAD driven scripts.			
6.	In the terminal window execute the following: ./subscription -o add -t timer -r python -p "0 **** ?" < src/data/HelloWorld.py	Identifies the subscription operation (add) the type of trigger (timer), the uEngine script runner (python), and the pattern (Quartz cron expression) of the trigger script execution. You will see a status message something like "Database insert was successful."	
7.	In the terminal window execute the following: ./subscription -o add -t data -r python -p "/obs*/KOMA/*" < src/data/EchoTrigger.py	Identifies the subscription operation (add), the type of trigger (data), the uEngine script runner (python), and the pattern (P5 regex that matches product data uris) of the trigger data to match. You will see a status message something like "Database insert was successful."	
8.	In the terminal window execute the following: ./subscription -o add -t ldad -r ldad -p CCCNNNXXX -f tester.sh	Identifies the subscription operation (add) the type of trigger (ldad), the uEngine script runner (ldad), the script to run (tester.sh), and the pattern (AFOS PIL) of the trigger data to match. You will see a status message something like "Database insert was successful."	
9.	Query the subscriptions table to retrieve data added in the step above.	The data displayed shows the subscriptions added to the database.	
Scenario 2: querying the database for lists of subscriptions			
10.	In the terminal window execute the following: ./subscription -o read -t timer	Requests a listing of all subscriptions having timer triggers. You should get at least one script at this time.	
11.	In the terminal window execute the following: ./subscription -o read -t data	Requests a listing of all subscriptions having data triggers. You should get at least one script at this time.	
12.	In the terminal window execute the following: ./subscription -o read -t ldad	Requests a listing of all subscriptions having ldad triggers. You should get at least one script at this time.	

Step #	Action	Result	Pass/Fail
13.	In the terminal window execute the following: ./subscription -o read -r python	Requests a listing of all subscriptions using the Python Micro Engine. You should get at least two scripts at this time.	
14.	In the terminal window execute the following: ./subscription -o read -p CCCNNNXXX	Requests a listing of all subscriptions having the specified trigger. You should get at least one script at this time.	
Scenario 3: Updating database entries			
15.	In the terminal window execute the following: ./subscription -o read -t data	Gets a list of data triggered subscriptions currently in the database. (There should be at least one.) Note the ID number for one of the subscriptions.	
16.	In the terminal window execute the following: ./subscription -o update -i <index> -u active false	Updates the subscription <index> changing the active flag to false. (Use the ID from step 15 in place of <index>)	
Scenario 4: Deleting a subscription from the subscription table.			
17.	In the terminal window execute the following: ./subscription -o read -t data	Gets a list of data triggered subscriptions currently in the database. (There should be at least one.) Note the ID number for one of the subscriptions.	
18.	In the terminal window execute the following: ./subscription -o delete -i <index>	Deletes the subscription having the specified index.(Use the ID from step 17 in place of <index>)	
19.	Query the subscriptions table to retrieve data modified in the steps above.	The data displayed shows the subscriptions modified in the database.	
Scenario 5: Force an error condition.			
20.	In the terminal window execute the following: ./subscription -l 1035 -u active true	Creates an error and returns the error message: Error: ArgError: 'For subscription maintenance; must specify operation'	
21.	Close the database window.	The database window is closed.	
22.	Exit to log out and close the terminal window.	The terminal window is closed.	
End of Test			

5.0 REQUIREMENTS VERIFICATION TRACEABILITY MATRIX (RVTM)

Number	Description	Test Step(s)
SYSR3124	The AWIPS system shall implement command line Interfaces to the subscription script runner.	ALL