Welcome

Ken Leonard, Director
ITS Joint Program Office
Ken.Leonard@dot.gov

www.pcb.its.dot.gov
T308: Acceptance Testing for Advanced Transportation Controller (ATC) Application Programming Interface (API) Software
Instructor

Ralph W. Boaz

President
Pillar Consulting, Inc.
San Diego, CA, USA
Learning Objectives

- Explain the purpose of the API Validation Suite (APIVS) Software
- Use the API Reference Implementation (APIRI) test documentation to specify acceptance testing
- Use the APIVS Software to test the API Software
- Interpret and report results of testing API Software
Learning Objective 1

Explain the purpose of the API Validation Suite (APIVS) Software
Quick Review of Advanced Transportation Controllers (ATCs)

- A transportation controller is a computer.
- Traditional controllers run a single application program.
- Application Programming Interface (API) Software allows many application programs to run simultaneously.
- Application programs may come from different vendors than the ATC unit’s manufacturer.
- Working, consistent and tested API Software is essential.
Traditional controller units typically perform a single application

**Traditional Applications**

- Data Collection Application
  - OR
- Traffic Signal Application
  - OR
- Ramp Meter Application

Graphic: Ralph W. Boaz
ATC units can perform numerous applications simultaneously

Example Applications for ATCs

- Traffic Signal Control/Traffic Management
- Transit/Light Rail Priority
- Emergency Management
- Lane Use
- Red Light Enforcement
- Speed Monitoring/Enforcement
- Access Control
- Advanced Traveler Information Systems (ATIS)
- Data Collection Systems
- Connected Vehicle (CV) Applications

Graphics: Ralph W. Boaz
Elements of API Software

- Made up of **three software libraries:**
  - Front Panel User Interface (FPUI)
  - Field I/O (FIO)
  - Time of Day (TOD)

- **Two** resource management **programs:**
  - Front Panel Manager
  - Field I/O Manager

- Allows **application developers to write programs** that **safely share the controller**
API Software Testing in the Context of ATC Unit Testing

Example of the Front Panel Manager Window

Graphic: Ralph W. Boaz
API Software Testing in the Context of ATC Unit Testing

Unit Testing

Traditional controller unit testing:
- **Tests the controller hardware** (may include the operating system)
- **Tests the application program** running on the controller
API Validation Suite (APIVS) Software tests API Software

Typical of non-ATC 2070 Units

Typical of non-ATC 2070 & NEMA Units

Typical of ATC units with API Software

Graphics: Ralph W. Boaz
Architecture of the APIVS Software

Background

- Four methods of software validation: inspection, demonstration, analysis, and test
- Must validate that the API Software conforms to ATC 5401 Application Programming Interface Standard
- API Validation Suite (APIVS) Software is used for testing
- Testing involves initiating a test and comparing the result to a known correct result
- Must be repeatable
Architecture of the APIVS Software

Testing takes place inside the ATC unit

- **Computational power** of the ATC unit allows **internal testing** of the API Software

- **Front panel and field I/O devices** are **emulated**

- “Loopback Drivers” cause the API Software’s **responses** to be **routed** back **to the APIVS Software**

- APIVS Software **captures** the **test results** in files and **compares them** to known correct results
Architecture of the APIVS Software

Recall the layered ATC software architecture

Graphic: Ralph W. Boaz
Architecture of the APIVS Software

Modified architecture for the APIVS Software

Graphics: Ralph W. Boaz
Detailed architecture of the APIVS Software

Test Scripts (XML) → Validation Suite Engine (VSE) → Expected Result Files (Flat Files) → API Conformance Report (XML)

Front Panel Manager (FPM) → API Libraries (FPUI, FIO, TOD) → Field I/O Manager (FIOM)

Virtual-LoopbackAsync → Virtual-LoopbackSync

APPLICATION LAYER

API SOFTWARE LAYER

ATC BOARD SUPPORT PACKAGE LAYER
Features of the APIVS Software

Command-Line Interface (CLI) of the APIVS Software

```
# ATC login: root
# Password:
#
# vse -L 2 -c ./VS_config_1.txt -i C1420_in.xml -o C1420_log.xml
```
Features of the APIVS Software

Command-Line Interface (CLI) of the APIVS Software


Where:

- **vse** – Name of the VSE executable program.
- **-L [1-3]** – (required) Level of output for the conformance report.
- **-c configuration-file** – (optional) File that specifies a series of VSE configurable items. If this file is omitted, default values are used.
- **-i APIVSXML-file** – (optional) Path to the input XML test script to use. If –i is not present, the input will be read from stdin.
- **-o output-file** – (optional) Path of where to place the generated output XML file. If –o is not present, the output will be placed on stdout.
Features of the APIVS Software

Command-Line Interface (CLI) of the APIVS Software

```bash
[-n test_suite_name] [-R count] [-H] [-C]
```

Where:

- `vse` – Name of the VSE executable program.
- `-c configuration-file` – (optional) File that specifies a series of VSE configurable items. If this file is omitted, default values are used.
- `-i APIVSXML-file` – (optional) Path to the input XML test script to use. If `-i` is not present, the input will be read from stdin.
- `-o output-file` – (optional) Path of where to place the generated output XML file. If `-o` is not present, the output will be placed on stdout.
Features of the APIVS Software

Command-Line Interface (CLI) of the APIVS Software


Where:

- **vse** – Name of the VSE executable program.
- **-L [1-3]** – (required) Level of output for the conformance report.
- **-c configuration-file** – (optional) File that specifies a series of VSE configurable items. If this file is omitted, default values are used.
- **-i APIVSXML-file** – (optional) Path to the input XML test script to use. If –i is not present, the input will be read from stdin.
- **-o output-file** – (optional) Path of where to place the generated output XML file. If –o is not present, the output will be placed on stdout.
Features of the APIVS Software

Command-Line Interface (CLI) of the APIVS Software


Where:

- **vse** – Name of the VSE executable program.
- **-L [1-3]** – (required) Level of output for the conformance report.
- **-c configuration-file** – (optional) File that specifies a series of VSE configurable items. If this file is omitted, default values are used.
- **-i APIVSXML-file** – (optional) Path to the input XML test script to use. If –i is not present, the input will be read from stdin.
- **-o output-file** – (optional) Path of where to place the generated output XML file. If –o is not present, the output will be placed on stdout.
Features of the APIVS Software

Command-Line Interface (CLI) of the APIVS Software


Where:

- **vse** – Name of the VSE executable program.
- **-L [1-3]** – (required) Level of output for the conformance report.
- **-c configuration-file** – (optional) File that specifies a series of VSE configurable items. If this file is omitted, default values are used.
- **-i APIVSXML-file** – (optional) Path to the input XML test script to use. If –i is not present, the input will be read from stdin.
- **-o output-file** – (optional) Path of where to place the generated output XML file. If –o is not present, the output will be placed on stdout.
Features of the APIVS Software

Command line interface for APIVS Software (cont.)


Where:

- **-n test_suite_name** – (optional) Specific “test suite” named in the input XML file that is to be run. If omitted, all test suites contained in the file will be run.

- **-R count** – (optional) Repeat test load count times, or indefinitely if count is 0.

- **-H** – (optional) Halt on error when running in Repeat mode.

- **-C** – (optional) Capture mode. Displays and command messages stored into files for use in subsequent tests.
Features of the APIVS Software

Command line interface for APIVS Software (cont.)

```
```

Where:

- `-n test_suite_name` – (optional) Specific “test suite” named in the input XML file that is to be run. If omitted, all test suites contained in the file will be run.

- `-R count` – (optional) Repeat test load count times, or indefinitely if count is 0.

- `-H` – (optional) Halt on error when running in Repeat mode.

- `-C` – (optional) Capture mode. Displays and command messages stored into files for use in subsequent tests.
Features of the APIVS Software

Command line interface for APIVS Software (cont.)

[-n test_suite_name] [-R count] [-H] [-C]

Where:

- `-n test_suite_name` – (optional) Specific “test suite” named in the input XML file that is to be run. If omitted, all test suites contained in the file will be run.

- `-R count` – (optional) Repeat test load count times, or indefinitely if count is 0.

- `-H` – (optional) Halt on error when running in Repeat mode.

- `-C` – (optional) Capture mode. Displays and command messages stored into files for use in subsequent tests.
Features of the APIVS Software

Command line interface for APIVS Software (cont.)


Where:

- `-n test_suite_name` – (optional) Specific “test suite” named in the input XML file that is to be run. If omitted, all test suites contained in the file will be run.

- `-R count` – (optional) Repeat test load count times, or indefinitely if count is 0.

- `-H` – (optional) Halt on error when running in Repeat mode.

- `-C` – (optional) Capture mode. Displays and command messages stored into files for use in subsequent tests.
What type of controller software is NOT traditionally tested by agencies?

Answer Choices

a) Data Collection Software
b) Signal Control Software
c) Application Programming Interface Software
d) Ramp Meter Software
Review of Answers

a) Data Collection Software

Incorrect. Data Collection is an application. Agencies usually have methods for testing their applications.

b) Signal Control Software

Incorrect. Signal Control is an application. Agencies usually have methods for testing their applications.

c) Application Programming Interface Software

Correct! Until recently, it was not possible to test API software. The API Validation Suite discussed in this module provides this ability.

d) Ramp Meter Software

Incorrect. Ramp Meter is an application. Agencies usually have methods for testing their applications.
Learning Objectives

Explain the purpose of the API Validation Suite (APIVS) Software

Use the API Reference Implementation (APIRI) test documentation to specify acceptance testing
Learning Objective 2

Use the API Reference Implementation (APIRI) test documentation to specify acceptance testing
API Reference Implementation (APIRI) Project

Software Vendors

Open Source APIRI & APIVS

Consultants

Manufacturers

Agencies

Graphics: Ralph W. Boaz (U)

Graphics: (L) Intelight, McCain, Econolite, Peek
API Reference Implementation (APIRI) Project

- USDOT funded the APIRI Project, which was completed in October 2016

- Produced an open source software (OSS) implementation of ATC 5401 Standard v02 called the APIRI Software
  - https://github.com/apiriadmin/APIRI

- Produced OSS APIVS Software to test API Software
  - https://github.com/apiriadmin/APIVS

- Formal Verification and Validation process that can be used for testing any API Software implementation

- APIRI Project Test documentation conforms to IEEE 829-2008
Benefits

- **Consistent with** the Linux O/S open source concept
- Promotes **collaboration** of developers across industry
- Provides **forum for users** to express ideas and concerns
- Promotes **quick bug fixes** and **alternative solutions** to issues
- Facilitates **introduction of new application developers**
- **Incorporated on ATC units** by manufacturers
- Provides best opportunity for **consistent API Software behavior**
## APIRI Documents Used to Specify Testing
(IEEE Std 829-1998)

<table>
<thead>
<tr>
<th>Test Document</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test Plan</strong></td>
<td>Specifies scope and approach for testing. Identifies the features to be tested by the Test Plan and, in the APIRI Project, includes the Test Design Specifications.</td>
</tr>
<tr>
<td><strong>Test Design Specification (TDS)</strong></td>
<td>Specifies refinements of the test approach in the test plan and identifies the features to be tested by this design and the associated tests. There is a TDS in the Test Plan for each of the FPUI, FIO and TOD libraries.</td>
</tr>
<tr>
<td><strong>Test Case Specification (TCS)</strong></td>
<td>Defines the information needed as it pertains to inputs and outputs from the software being tested. The APIRI project produced about 40 Test Case Specifications.</td>
</tr>
<tr>
<td><strong>Test Procedure Specification (TPS)</strong></td>
<td>Specifies the steps for executing the test cases on the APIRI software. There are TPSs for testing using the APIVS software and TPSs for doing other methods of validation.</td>
</tr>
</tbody>
</table>
APIRI Documents Used to Specify Testing
(IEEE Std 829-1998)

<table>
<thead>
<tr>
<th>Test Document</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Plan</td>
<td><strong>Specifies scope and approach for testing.</strong> Identifies the features to be tested by the Test Plan and, in the APIRI Project, includes the Test Design Specifications.</td>
</tr>
<tr>
<td>Test Design Specification (TDS)</td>
<td><strong>Specifies refinements of the test approach</strong> in the test plan and identifies the features to be tested by this design and the associated tests. There is a TDS in the Test Plan for each of the FPUI, FIO and TOD libraries.</td>
</tr>
<tr>
<td>Test Case Specification (TCS)</td>
<td><strong>Defines the information needed as it pertains to inputs and outputs from the software being tested.</strong> The APIRI project produced about 40 Test Case Specifications.</td>
</tr>
<tr>
<td>Test Procedure Specification (TPS)</td>
<td><strong>Specifies the steps for executing the test cases on the APIRI software.</strong> There are TPSs for testing using the APIVS software and TPSs for doing other methods of validation.</td>
</tr>
</tbody>
</table>
## APIRI Documents Used to Specify Testing (IEEE Std 829-1998)

<table>
<thead>
<tr>
<th>Test Document</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Plan</td>
<td>Specifies scope and approach for testing. Identifies the features to be tested by the Test Plan and, in the APIRI Project, includes the Test Design Specifications.</td>
</tr>
<tr>
<td>Test Design Specification (TDS)</td>
<td>Specifies refinements of the test approach in the test plan and identifies the features to be tested by this design and the associated tests. There is a TDS in the Test Plan for each of the FPUI, FIO and TOD libraries.</td>
</tr>
<tr>
<td>Test Case Specification (TCS)</td>
<td>Defines the information needed as it pertains to inputs and outputs from the software being tested. The APIRI project produced about 40 Test Case Specifications.</td>
</tr>
<tr>
<td>Test Procedure Specification (TPS)</td>
<td>Specifies the steps for executing the test cases on the APIRI software. There are TPSs for testing using the APIVS software and TPSs for doing other methods of validation.</td>
</tr>
</tbody>
</table>
## Organization and Content of the APIRI Test Documentation

### APIRI Documents Used to Specify Testing (IEEE Std 829-1998)

<table>
<thead>
<tr>
<th>Test Document</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Plan</td>
<td>Specifies scope and approach for testing. Identifies the features to be tested by the Test Plan and, in the APIRI Project, includes the Test Design Specifications.</td>
</tr>
<tr>
<td>Test Design Specification (TDS)</td>
<td>Specifies refinements of the test approach in the test plan and identifies the features to be tested by this design and the associated tests. There is a TDS in the Test Plan for each of the FPUI, FIO and TOD libraries.</td>
</tr>
<tr>
<td>Test Case Specification (TCS)</td>
<td><strong>Defines the</strong> information needed as it pertains to <strong>inputs and outputs from the software being tested</strong>. The APIRI project produced about 40 Test Case Specifications.</td>
</tr>
<tr>
<td>Test Procedure Specification (TPS)</td>
<td>Specifies the steps for executing the test cases on the APIRI software. There are TPSs for testing using the APIVS software and TPSs for doing other methods of validation.</td>
</tr>
</tbody>
</table>
## APIRI Documents Used to Specify Testing (IEEE Std 829-1998)

<table>
<thead>
<tr>
<th>Test Document</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Plan</td>
<td>Specifies scope and approach for testing. Identifies the features to be tested by the Test Plan and, in the APIRI Project, includes the Test Design Specifications.</td>
</tr>
<tr>
<td>Test Design Specification (TDS)</td>
<td>Specifies refinements of the test approach in the test plan and identifies the features to be tested by this design and the associated tests. There is a TDS in the Test Plan for each of the FPUl, FIO and TOD libraries.</td>
</tr>
<tr>
<td>Test Case Specification (TCS)</td>
<td>Defines the information needed as it pertains to inputs and outputs from the software being tested. The APIRI project produced about 40 Test Case Specifications.</td>
</tr>
<tr>
<td>Test Procedure Specification (TPS)</td>
<td>Specifies the steps for executing the test cases on the APIRI software. There are TPSs for testing using the APIVS software and TPSs for doing other methods of validation.</td>
</tr>
</tbody>
</table>
Organization and Content of the APIRI Test Documentation

APIRI Documents Used to Specify Testing
(IEEE Std 829-2008)

APIRI Test Plan

FPUI Test Design

FPUI Test Cases

FPUI Test Procedures

FIO Test Design, etc.

TOD Test Design, etc.

Test Execution

Graphic: Ralph W. Boaz
Organization and Content of the APIRI Test Documentation

APIRI Documents Used to Specify Testing (IEEE Std 829-2008)

APIRI Test Plan

FPUI Test Design, etc.

FIO Test Design

TOD Test Design, etc.

FIO Test Cases

FIO Test Procedures

Test Execution

Graphic: Ralph W. Boaz
APIRI Documents Used to Specify Testing (IEEE Std 829-2008)

APIRI Test Plan

FIO Test Design, etc.

FPUI Test Design, etc.

TOD Test Design

TOD Test Cases

TOD Test Procedures

Test Execution

Graphic: Ralph W. Boaz
APIRI Test Plan Outline

1. Introduction
2. Test Items
3. Features to Be Tested
4. Features Not to Be Tested
5. Approach
6. Item Pass/Fail Criteria
7. Suspension Criteria and Resumption Requirements
8. Test Deliverables
9. Testing Tasks
10. Environmental Needs
### Features to Be Tested

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Document Name</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIRI.TDS.2001</td>
<td>APIRI Test Design Spec 1</td>
<td>Test All APIRI FPUI Required Features</td>
</tr>
<tr>
<td>APIRI.TDS.3001</td>
<td>APIRI Test Design Spec 2</td>
<td>Test All APIRI FIO Required Features</td>
</tr>
<tr>
<td>APIRI.TDS.4001</td>
<td>APIRI Test Design Spec 3</td>
<td>Test All APIRI TOD Required Features</td>
</tr>
<tr>
<td>APIRI.TCS.2010</td>
<td>APIRI Test Case Spec 1</td>
<td>FPUI Text UI Virtual Displays</td>
</tr>
<tr>
<td>APIRI.TCS.2020</td>
<td>APIRI Test Case Spec 2</td>
<td>FPUI Front Panel Manager</td>
</tr>
<tr>
<td>APIRI.TCS.2030</td>
<td>APIRI Test Case Spec 3</td>
<td>FPUI Character Set and Screen Attribs</td>
</tr>
<tr>
<td>APIRI.TCS.2040</td>
<td>APIRI Test Case Spec 4</td>
<td>FPUI Reading and Writing Data</td>
</tr>
<tr>
<td>APIRI.TCS.2100</td>
<td>APIRI Test Case Spec 9</td>
<td>API Version Information (All Libraries)</td>
</tr>
<tr>
<td>APIRI.TCS.3010</td>
<td>APIRI Test Case Spec 10</td>
<td>General FIO Operations</td>
</tr>
<tr>
<td>APIRI.TCS.3020</td>
<td>APIRI Test Case Spec 11</td>
<td>FIO Inputs and Outputs</td>
</tr>
<tr>
<td>APIRI.TCS.3030</td>
<td>APIRI Test Case Spec 12</td>
<td>FIO Channel Mapping</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

APIRI Test Plan Outline (cont.)

11 Responsibilities
12 Staffing and Training Needs
13 Schedule
14 Risks and Contingencies
15 Approvals
16 Appendices

16.1 FPUI Library Requirements to Validation Description Matrix
16.2 FIO Library Requirements to Validation Description Matrix
16.3 TOD Library Requirements to Validation Description Matrix
16.4 APIRI Test Design Specifications
### FPUI Library Requirements to Validation Description Matrix

<table>
<thead>
<tr>
<th>Req ID</th>
<th>Req Description</th>
<th>ATC Function</th>
<th>APIRI SDD Design Narrative</th>
<th>Test Cases</th>
<th>Test Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIR3.1.1.2[10]</td>
<td>The API shall provide a function to read a queued character or key code from the input buffer of a window.</td>
<td>fpui_read_char</td>
<td>The implementation of the fpui_read_char() library function (Section 3.4.8) makes use of the Linux operating system call to return a single character from the input buffer of the FrontPanelDriver device interface (Section 3.3).</td>
<td>APIR.I.TCS. 2040</td>
<td>APIR.I.TPS. 1001</td>
</tr>
<tr>
<td>APIR3.1.1.2[11]</td>
<td>The API shall provide a function to write a character to the current cursor position of a window.</td>
<td>fpui_write_char</td>
<td>The implementation of the fpui_write_char() library function (Section 3.4.8) makes use of the Linux operating system call to write a single character to the output buffer of the FrontPanelDriver device interface (Section 3.3).</td>
<td>APIR.I.TCS. 2040</td>
<td>APIR.I.TPS. 1001</td>
</tr>
</tbody>
</table>


16.4.1 Test Design Specification 1 - Test All APIRI FPUI Features

16.4.1.1 Test Design Specification Identifier

The identifier for this Test Design Specification is APIRI.TDS.2001.

16.4.1.2 Features To Be Tested

This Test Design Specification will test all FPUI features of the API Reference Implementation (APIRI) which are subject to testing for validation …

16.4.1.3 Approach Refinements

All test cases will be tested using the general approach as defined in this test plan and as further refined in Test Procedure Specification APIRI.TPS.0001…
16.4.1.4 Test Identification

All test documents to be used by this Test Design Specification can be found in Section 3, Table 1.

16.4.1.5 Feature Pass/Fail Criteria

This Test Design Specification will be considered to have passed if and only if every individual test case passes according to its own pass/fail criteria as well as any pass/fail criteria associated with the test procedure used to execute the test case.
APIRI Test Case Outlines

2.6 Test Case Specification 4 – FPUI Reading and Writing Data

2.6.1 Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.2040.

2.6.2 Objective

The objective of this Test Case is to test the operation of the API functions used to write display data to and read keypresses from the Front Panel.

2.6.3 Test Items

... 

APIR3.1.1.2[13] The API shall provide a function to write a string to a window at the current cursor position.

APIR3.1.1.2[14] The API shall provide a function to write a string to a window at a starting position defined by column number and line number.
2.6.4 Input Specifications

This test case requires the following file(s) as input:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2040_in.xml</td>
<td>APIVXML test script (XML format)</td>
</tr>
<tr>
<td>Cxxxxx_key0.txt</td>
<td>keystroke file (Key ‘0’)</td>
</tr>
<tr>
<td>Cxxxxx_key1.txt</td>
<td>keystroke file (Key ‘1’)</td>
</tr>
<tr>
<td>Cxxxxx_keyESC.txt</td>
<td>keystroke file (Key ‘&lt;Esc&gt;’)</td>
</tr>
<tr>
<td>C2040_vd_1.txt</td>
<td>Virtual Display compare file (display 1)</td>
</tr>
<tr>
<td>VS_config_1.txt</td>
<td>VSE configuration file (for VSE command line)</td>
</tr>
</tbody>
</table>

2.6.5 Output Specifications

This test case produces the following file(s) as output:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2040_log.xml</td>
<td>Conformance report (XML format)</td>
</tr>
</tbody>
</table>
2.1 Test Procedure Specification 1 - Auto-Execute Selected APIVS Script(s)

2.1.1 Test Procedure Specification Identifier

The identifier for this Test Procedure Specification is APIVS.TPS.1001.

2.1.2 Purpose

This procedure runs the Validation Suite Engine (VSE) using the source test script and runtime options as associated with one or more specific Test Case Specifications. This execution will run from beginning to end with only limited human intervention…
2.1.3 Special Requirements

This procedure requires the editing of text files and the movement of files between a host computer Hard Disk Drive and a USB Flash Drive and is intended to be run by an operator with a reasonable technical knowledge of personal computer (PC) file systems…

2.1.4 Procedure Steps

*Subsections contained in this section: Log, Setup, Start, Proceed, Measure, Shutdown, Restart, Stop, Wrap Up, and Contingencies*
Files Needed for Executing the Test Cases

<table>
<thead>
<tr>
<th>Test Document</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIRI Test Scripts</td>
<td>Written in <strong>XML</strong> (extensible markup language), which <strong>allows testers to exercise the API software</strong> without having to write C programs. These Test Scripts are input to the VSE.</td>
</tr>
<tr>
<td>Flat Files</td>
<td>Files that are used to <strong>configure the device emulators</strong> in the APIVS software and files that <strong>represent known correct outputs</strong> of the API software for given test cases.</td>
</tr>
<tr>
<td>Validation Suite Engine (VSE)</td>
<td>Allows testers to set various <strong>system options</strong> for APIVS software such as the file paths, screen size, and setting the ports for the loopback device driver software.</td>
</tr>
<tr>
<td>Configuration File</td>
<td></td>
</tr>
<tr>
<td>Linux Shell Scripts</td>
<td>Allows the testers to <strong>run successive executions</strong> of the VSE <strong>without typing</strong> them in a line at a time.</td>
</tr>
<tr>
<td>Output Files</td>
<td><strong>Output from the VSE in XML format</strong>, which allows various tools to be used for analyzing test results.</td>
</tr>
</tbody>
</table>
## Files Needed for Executing the Test Cases

<table>
<thead>
<tr>
<th>Test Document</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIRI Test Scripts</td>
<td>Written in <strong>XML</strong> (extensible markup language), which <strong>allows testers to exercise the API software</strong> without having to write C programs. These Test Scripts are input to the VSE.</td>
</tr>
<tr>
<td>Flat Files</td>
<td>Files that are used to <strong>configure the device emulators</strong> in the APIVS software and files that <strong>represent known correct outputs</strong> of the API software for given test cases.</td>
</tr>
<tr>
<td>Validation Suite Engine (VSE)</td>
<td>Allows testers to <strong>set</strong> various <strong>system options</strong> for APIVS software such as the file paths, screen size, and setting the ports for the loopback device driver software.</td>
</tr>
<tr>
<td>Configuration File</td>
<td></td>
</tr>
<tr>
<td>Linux Shell Scripts</td>
<td>Allows the testers to <strong>run successive executions</strong> of the VSE <strong>without typing</strong> them in a line at a time.</td>
</tr>
<tr>
<td>Output Files</td>
<td><strong>Output</strong> from the VSE in <strong>XML format</strong>, which allows various tools to be used for analyzing test results.</td>
</tr>
</tbody>
</table>
## Files Needed for Executing the Test Cases

<table>
<thead>
<tr>
<th>Test Document</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>APIRI Test Scripts</strong></td>
<td>Written in <strong>XML</strong> (extensible markup language), which <strong>allows testers to exercise the API software</strong> without having to write C programs. These Test Scripts are input to the VSE.</td>
</tr>
<tr>
<td><strong>Flat Files</strong></td>
<td>Files that are used to <strong>configure the device emulators</strong> in the APIVS software and files that <strong>represent known correct outputs</strong> of the API software for given test cases.</td>
</tr>
<tr>
<td><strong>Validation Suite Engine (VSE) Configuration File</strong></td>
<td>Allows testers to set various <strong>system options</strong> for APIVS software such as the file paths, screen size, and setting the ports for the loopback device driver software.</td>
</tr>
<tr>
<td><strong>Linux Shell Scripts</strong></td>
<td>Allows the testers to <strong>run successive executions</strong> of the VSE <strong>without typing</strong> them in a line at a time.</td>
</tr>
<tr>
<td><strong>Output Files</strong></td>
<td><strong>Output</strong> from the VSE in <strong>XML format</strong>, which allows various tools to be used for analyzing test results.</td>
</tr>
</tbody>
</table>
# Files Needed for Executing the Test Cases

<table>
<thead>
<tr>
<th>Test Document</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIRI Test Scripts</td>
<td>Written in XML (extensible markup language), which allows testers to exercise the API software without having to write C programs. These Test Scripts are input to the VSE.</td>
</tr>
<tr>
<td>Flat Files</td>
<td>Files that are used to configure the device emulators in the APIVS software and files that represent known correct outputs of the API software for given test cases.</td>
</tr>
<tr>
<td>Validation Suite Engine (VSE)</td>
<td>Allows testers to set various system options for APIVS software such as the file paths, screen size, and setting the ports for the loopback device driver software.</td>
</tr>
<tr>
<td>Linux Shell Scripts</td>
<td>Allows the testers to run successive executions of the VSE without typing them in a line at a time.</td>
</tr>
<tr>
<td>Output Files</td>
<td>Output from the VSE in XML format, which allows various tools to be used for analyzing test results.</td>
</tr>
</tbody>
</table>
## Files Needed for Executing the Test Cases

<table>
<thead>
<tr>
<th>Test Document</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIRI Test Scripts</td>
<td>Written in <strong>XML</strong> (extensible markup language), which allows testers to exercise the API software without having to write C programs. These Test Scripts are input to the VSE.</td>
</tr>
<tr>
<td>Flat Files</td>
<td>Files that are used to <strong>configure the device emulators</strong> in the APIVS software and files that represent <strong>known correct outputs</strong> of the API software for given test cases.</td>
</tr>
<tr>
<td>Validation Suite Engine (VSE)</td>
<td>Allows testers to set various <strong>system options</strong> for APIVS software such as the file paths, screen size, and setting the ports for the loopback device driver software.</td>
</tr>
<tr>
<td>Linux Shell Scripts</td>
<td>Allows the testers to <strong>run successive executions</strong> of the VSE <strong>without typing</strong> them in a line at a time.</td>
</tr>
<tr>
<td>Output Files</td>
<td>Output from the VSE in <strong>XML format</strong>, which allows various tools to be used for analyzing test results.</td>
</tr>
</tbody>
</table>
## Files Needed for Executing the Test Cases

<table>
<thead>
<tr>
<th>Test Document</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIRI Test Scripts</td>
<td>Written in <strong>XML</strong> (extensible markup language), which <strong>allows testers to exercise the API software</strong> without having to write C programs. These Test Scripts are input to the VSE.</td>
</tr>
<tr>
<td>Flat Files</td>
<td>Files that are used to <strong>configure the device emulators</strong> in the APIVS software and files that <strong>represent known correct outputs</strong> of the API software for given test cases.</td>
</tr>
<tr>
<td>Validation Suite Engine (VSE) Configuration File</td>
<td>Allows testers to <strong>set various system options</strong> for APIVS software such as the file paths, screen size, and setting the ports for the loopback device driver software.</td>
</tr>
<tr>
<td>Linux Shell Scripts</td>
<td>Allows the testers to <strong>run successive executions</strong> of the VSE <strong>without typing</strong> them in a line at a time.</td>
</tr>
<tr>
<td>Output Files</td>
<td><strong>Output</strong> from the VSE in <strong>XML format</strong>, which allows various tools to be used for analyzing test results.</td>
</tr>
</tbody>
</table>
Files Needed for Executing the Test Cases

APIRI Test Scripts in XML

ATC 5401 API Reference Implementation Project
  Filename: C2040_in.xml
  File Type: APIVSXML test script (XML format)
  Test Case: APIRI.TCS.2040
  Description: FPUI Reading and Writing Data
  TC XML: begins on Line 1187

Test Case Narrative
  open an FPUI connection
  put the app in focus, wait for confirmation
  write to the VD using all write methods
  (APIR3.1.1.2[15])
  (APIR3.1.1.2[16])
  (APIR3.1.1.2[11])
  ...

Files Needed for Executing the Test Cases

APIRI Test Scripts in XML (cont.)

<!-- write to the VD using all write method -->
<Set var="$write_buf" value="$C2040"/>
<Set var="$write_chr" value="$J"/>
<Set var="$write_len" value="$1"/>
<Set var="$row" value="$4"/>
<Set var="$column" value="$6"/>

<!-- (APIR3.1.1.2[15]) -->
<Call ref="fpui_write" setUp="API_Init_Variables"/>

<!-- (APIR3.1.1.2[16]) -->
<Call ref="fpui_write_at" setUp="API_Init_Variables"/>

<!-- (APIR3.1.1.2[11]) -->
<Call ref="fpui_write_char" setUp="API_Init_Variables"/>

...
Files Needed for Executing the Test Cases

Expected Result Flat Files

# -- Virtual Display and Global Variable Dump --
# Date: 20160713 19:55:26 -
#
# Display Rows:
#0  1  2  3  4
#234567890123456789012345678901234567890123456789

FRONT PANEL MANAGER

SELECT WINDOW [0-F] SET DEFAULT * [0-F]
0 C1160_00       1 C1160_01
2 C1160_02       3 C1160_03
4 C1160_04       5 C1160_05
6 C1160_06       7 C1160_07
8 C1160_08       9 C1160_09

[UP/DN ARROW] [CONFIG INFO- NEXT]

#----------------------------------------
Files Needed for Executing the Test Cases

Expected Result Flat Files (cont.)

# ATC 5401 API Reference Implementation Project
#
# Filename: C3020_cmd55a.txt
# File Type: APIVS flat file (text format)
# Test Case: APIRI.TCS.3xxx
# Description: file load of Command Frame 55, test outputs set
#
# Date  Revision  Description
# 2/24/16  1.0     initial release

0x37 0x05 0x50 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Validation Suite Engine (VSE) Configuration File

...# Filename: VS_config_1.txt
# File Type: VSE configuration file
# Test Case: many
# Description:
#
# Date      Revision    Description
# 10/21/15   1.0         initial release
(XMLInputFilePath = ./
XMLOutputFilePath = /tmp/
SetFilePath = ./
ScreenWidth = 40
ScreenHeight = 8
FPUICompareFilePath = ./
FPUIInputFilePath = ./
...
What document is used to specify the inputs and outputs for a particular test of the API Software?

Answer Choices

a) Test Design Specification
b) Test Procedure Specification
c) Test Plan
d) Test Case Specification
Review of Answers

a) Test Design Specification

Incorrect. The TDS specifies refinements of the test approach in the test plan.

b) Test Procedure Specification

Incorrect. The TPS specifies the steps for executing the test cases on the API software.

c) Test Plan

Incorrect. The test plan specifies the scope and approach for testing and identifies the features to be tested. In the APIRI Project, it also contains the Test Design Specifications.

d) Test Case Specification

Correct! In the APIRI Project, all of the tests cases are contained in a single document.
Learning Objectives

Explain the purpose of the API Validation Suite (APIVS) Software

Use the API Reference Implementation (APIRI) test documentation to specify acceptance testing

Use the APIVS Software to test the API Software
Learning Objective 3

Use the APIVS Software to test the API Software
Open Source Software (OSS) Environment of the APIVS Software

APIVS Repository – https://github.com/apiriadmin/APIVS

apiadmin / APIVS

Code | Issues 2

APIVS Repository

123 commits | 1 branch | 0 releases | 5 contributors | GPL-2.0

Clone or download

apiadmin committed on GitHub: Merge pull request #18 from gallagjm/master

docs Add project test documentation. 3 months ago
include Remove older copyright notices. 2 months ago
modules Add APIVS source code. a year ago
samples APIVS update with updated demo test file and package script a year ago
src Remove older copyright notices. 2 months ago
support Add full set of test cases. 2 months ago
tests Add run scripts and support files for miscellaneous non-automated tests. 2 months ago
Preparations for Testing

Equipment Required for Testing

- **Basic Equipment**
  - ATC unit with operational API Software
  - PC with 1GB available hard drive storage and USB port
  - **VSE executable provided by your ATC vendor** (or compiled by you using vendor’s tool chain)

- If using **CLI Method** add
  - Serial or Ethernet cable to connect the PC to the ATC unit

- If using **USB Test Package Method** add
  - 1GB USB Flash Drive (minimum), formatted with a FAT16 or FAT32 file system
Preparations for Testing

Command-Line Interface (CLI) Method

- APIVS Software has CLI designed to run in a Linux “shell”
- **Allows complete control** for each execution of a test
- Best method if tester is doing a lot of variations on individual tests or creating new tests
- **Tester** must be **comfortable working in** a **Linux** environment
Preparations for Testing

USB Test Package Method

- **Preconfigured tests can be downloaded** from the web to a USB flash drive
- Simply plug the **USB drive** into the ATC unit and **turn on the power**
- **Variations** (if desired) **made by simple edits** of the runAPIVS file on the USB drive
- **Windows or Linux environment**

Graphics: Ralph W. Boaz (L) and Thinkstock (R)
Using the USB Test Package Method

1. Download or clone the APIVS repository to a PC
2. Install a USB flash drive into your PC
3. Run package.sh (for Linux PCs) or package.bat (for Windows PCs) from a Linux or Windows shell, respectively
4. Copy the VSE executable and APIVS Loopback Drivers to the USB drive
5. (optional) Edit runAPIVS to modify tests
6. Install the USB flash drive in the ATC unit
7. Turn the ATC unit on
8. Wait for completion
9. Test results may be analyzed by reinstalling the USB flash drive on the PC and viewing the log files (*log.xml)
Using the USB Test Package Method

APIVS Repository

Download or Clone

Vendor Supplied VSE Executable

Copy Software

Run package.sh or package.bat

Install and Run

Happy Tester

Copy Results

Remove
Prepare the APIVS Software for Testing

Command-Line Interface (CLI) of the APIVS Software

```
```

Where:

- **vse** – Name of the VSE executable program.
- **-L [1-3]** – (required) Conformance level of the output desired.
- **-c configuration-file** – (optional) File that specifies a series of VSE configurable items. If this file is omitted, default values are used.
- **-i APIVSXML-file** – (optional) Path to the input XML test script to use. If –i is not present, the input will be read from stdin.
- **-o output-file** – (optional) Path of where to place the generated output XML file. If –o is not present, the output will be placed on stdout.
- **-R count** – (optional) Repeat test load count times, or indefinitely if count is 0.
Prepare the APIVS Software for Testing

Editing runAPIVS (optional)

- **runAPIVS** is a *Linux shell script* in the root of the USB Flash Drive

- **Defaults to running all tests cases** on the API software one time with Logging Level 1

- **Easy edits** to
  - Change the Logging Level
  - Increase iterations of particular tests
  - Run a subset of the test cases and/or change other VSE options
Editing the runAPIVS Linux Shell Script

... # Filename: runAPIVS
# File Type: Linux shell script
# Test Case: many
# Description: run VSE from USB at startup on specific test cases
#
# Date      Revision    Description
# 2/24/16   1.0         initial release
#
# start async loopback driver; add symbolic links
insmod /media/sda1/APIVS/bin/tty0tty.ko
ln -s /dev/tnt0 /dev/sp6_loopback_a
ln -s /dev/tnt1 /dev/sp6_loopback_b
...
# set the conformance level this run (1,2,3)
LEVEL=1
...
reset_modules
if [ "DELETE_LOGS" == TRUE ]; then rm C2020_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.2020... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2020_in.xml
   -o C2020_log.xml
print_test_result

reset_modules
if [ "DELETE_LOGS" == TRUE ]; then rm C2030_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.2030... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2030_in.xml
   -o C2030_log.xml
print_test_result

reset_modules
if [ "DELETE_LOGS" == TRUE ]; then rm C2040_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.2040... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2040_in.xml
   -o C2040_log.xml
print_test_result

...
reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C2020_log.xml; fi

clear_test_line; printf "Testing APIRI.TCS.2020... " >/dev/sp6
vse -L 3 -c ./VS_config_1.txt -i C2020_in.xml
   -o C2020_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C2030_log.xml; fi

clear_test_line; printf "Testing APIRI.TCS.2030... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2030_in.xml -R 10
   -o C2030_log.xml
print_test_result

# reset_modules
# if [ "$DELETE_LOGS" == TRUE ]; then rm C2040_log.xml; fi
# clear_test_line; printf "Testing APIRI.TCS.2040... " >/dev/sp6
# vse -L $LEVEL -c ./VS_config_1.txt -i C2040_in.xml
#   -o C2040_log.xml
# print_test_result

...
Running the USB Test Package

- Plug the USB drive into the ATC and turn it on
- Follow the screens that appear on the ATC unit front panel
Use the APIVS Software to Execute Tests

Running the USB Test Package

- Plug the USB drive into the ATC and turn it on
- Follow the screens that appear on the ATC unit front panel

ATC 5401 API Validation Suite v1.0
Running test session.
Testing APIRI.TCS.2010...

Test cases passed: 13 failed: 0
Use the APIVS Software to Execute Tests

Running the USB Test Package

- Plug the USB drive into the ATC and turn it on
- Follow the screens that appear on the ATC unit front panel

ATC 5401 API Validation Suite v1.0
Running test session.
Session complete.
Please remove USB drive and reboot.

Test cases passed: 40 failed: 0
ACTIVITY
What is **not** an appropriate reason to edit the runAPIVS shell script?

**Answer Choices**

a) Turn off all test output
b) Change the number of iterations on a test
c) Change the conformance report logging
d) Select a subset of the existing test cases
Review of Answers

a) Turn off all test output

*Correct! One cannot turn off all test output. A pass/fail is the most terse output available.*

b) Change the number of iterations on a test

*Incorrect. Selecting a particular test case is a good reason to edit runAPIVS.*

c) Change the conformance report logging

*Incorrect. Changing the conformance report logging is a good reason to edit runAPIVS.*

d) Select a subset of the existing test cases

*Incorrect. Selecting a subset of the existing test cases is a good reason to edit runAPIVS.*
Learning Objectives

- Explain the **purpose of the API Validation Suite (APIVS) Software**
- Use the **API Reference Implementation (APIRI) test documentation to specify acceptance testing**
- Use the **APIVS Software to test the API Software**
- Interpret and report results of testing API Software
Learning Objective 4

Interpret and report results of testing API Software
Analyze Results Using Off-the-Shelf Tools

- **Outputs** of the APIVS Software are in XML
- In the simplest case, all users are looking for is a PASS/FAIL indication
- Otherwise, use a tool. Examples:
  - **Notepad++** ([http://notepad-plus-plus.org](http://notepad-plus-plus.org))
    - General purpose editing tool for software-related files
    - Color coding and formatting of XML text files
  - **XML Differences** ([www.corefiling.com/opensource/xmldiff.html](http://www.corefiling.com/opensource/xmldiff.html))
    - Online comparison of XML files
  - **XmlGrid** ([http://xmlgrid.net](http://xmlgrid.net))
    - Online editor displays in formatted XML text or in grids (tables)
  - **XML Viewer** ([www.codebeautify.org/xmlviewer](http://www.codebeautify.org/xmlviewer))
    - Online editor displays in formatted XML text or in tree view
Analyze Results Using Off-the-Shelf Tools

```xml
<ApiVsRun date="2016-10-30 03:58:31 AM UTC" configuration="./VS_config_1.txt" input=""

  <TestSuite line="1320" timestamp="+0.147173" name="All_Test_Cases" description=""/>

  <TestSuite line="1187" timestamp="+2.149510" name="Case_C2040" description=""

    <Sleep line="1188" timestamp="+2.149722" time="5" />
    <Call line="1200" timestamp="+7.150038" ref="fpui_open" >
      <Function line="615" timestamp="+7.150131" funcName="fpui_open" return">
        <Call line="626" timestamp="+7.150819" ref="Success_Handler" />
      </Function>
    </Call>

    <Sleep line="1206" timestamp="+7.151509" time="1" />
    <Call line="1207" timestamp="+8.151738" ref="fpui_get_focus" >
      <Function line="491" timestamp="+8.151835" funcName="fpui_get_focus" >
        <Call line="502" timestamp="+8.153056" ref="Success_Handler" />
      </Function>
    </Call>

    <Call line="1217" timestamp="+8.153195" ref="fpui_write" >
      <Function line="1061" timestamp="+8.153271" funcName="fpui_write" return">
        <Call line="1076" timestamp="+8.153548" ref="Success_Handler" />
      </Function>
    </Call>

```
Analyze Results Using Off-the-Shelf Tools

```xml
<?xml version="1.0" encoding="utf-8"?>

<ApiVsRun
    configuration='./VS_config_1.txt'
    - date='2016-10-31 06:07:53 AM UTC'
    + date='1970-10-29 12:13:19 AM UTC'
    input='./C2040_in.xml'
    level='trace'
    output='./tmp/C2040_log.xml'
    testSuite='ALL_TESTS'
>

<Define
    line='70'
    - timestamp='+0.045790'
    + timestamp='+0.045831'
    type='fpui_handle'
    var='$fpui_handle'
/>

<Define
    line='71'
    - timestamp='+0.046147'
    + timestamp='+0.046198'
    type='fpui_aux_handle'
    var='$aux_handle'
/>
```
Analyze Results Using Off-the-Shelf Tools

![XML Grid.net screenshot](image-url)
Analyze Results Using Off-the-Shelf Tools

XML VIEWER

Result: XML Tree View

```
<apivsrun>
  <date>2016-10-30 03:58:31 AM UTC</date>
  <configuration>.\VS_config_1.txt</configuration>
  <input>.\C2040_in.xml</input>
  <output>/tmp/C2040_log.xml</output>
  <testSuite>ALL_TESTS</testSuite>
  <level>summary</level>
<apivsrun>

<testsuite>
  <id>1320</id>
  <timestamp>+0.147173</timestamp>
  <name>All_Test_Cases</name>
  <description>Run All Test Cases</description>
<testsuite>

<testcase>
  <line>1187</line>
  <timestamp>+2.149510</timestamp>
  <name>Case_C2040</name>
  <description>C2040: FPUI Reading and Writing Data</description>
<testcase>

<sleep>
  <line>1188</line>
  <timestamp>+2.149722</timestamp>
  <time>%5</time>
sleep
```
Create Test Reports Using the APIVS

Conformance report options

- Testers may include test logs in their test reports

- **Level 1** – Conformance/nonconformance indication only
  - 304 lines of output – about 16 minutes

- **Level 2** – Conformance/nonconformance indication and summary result
  - 9,693 lines of output – about 16 minutes

- **Level 3** – Conformance/nonconformance indication, summary result, and all logs and traces
  - 73,066 lines of output – about 22 minutes

Graphic: Thinkstock
1 **Introduction**
   1.1 Document identifier
   1.2 Scope
   1.3 References

2 **Details**
   2.1 Overview of test results
   2.2 Detailed test results
   2.3 Rationale for decisions
   2.4 Conclusions and recommendations

3 **General**
   3.1 Glossary
   3.2 Document change procedures and history
2.2 Detailed Test Results

...
True or False: It is a good idea to always log as much information as possible on all tests.

Answer Choices

a) True
b) False
Review of Answers

a) True

Incorrect. If a tester wants to view full logging, it is better to do this on selected tests.

b) False

Correct! Logging all of the data creates a voluminous report and makes understanding the results difficult. While full logging on all tests can be done, it is advised that testers repeat their test with full logging for tests that failed previously.
Module Summary

- Explain the purpose of the API Validation Suite (APIVS) Software
- Use the API Reference Implementation (APIRI) test documentation to specify acceptance testing
- Use the APIVS Software to test the API Software
- Interpret and report results of testing API Software
ATC Curriculum Completed To Date

**Module A207a/b**: Building an ITS Infrastructure Based on the Advanced Transportation Controller (ATC) 5201 Standard

**Module A208**: Using the ATC 5401 API Standard to Leverage ITS Infrastructures

**Module A307a**: Understanding User Needs for Advanced Transportation Controllers (ATC) Based on ATC 5201 Standard v06

**Module A307b**: Understanding Requirements for Advanced Transportation Controllers (ATC) Based on ATC 5201 Standard v06

**Module T307**: Applying Your Test Plan to the Advanced Transportation Controller (ATC) Based on ATC 5201 Standard v06

**Module T308**: Acceptance Testing for Advanced Transportation Controller Application Programming Interface Software
Thank you for completing this module.

Feedback
Please use the Feedback link below to provide us with your thoughts and comments about the value of the training.

Thank you!