STANDARDS ITS TRAINING

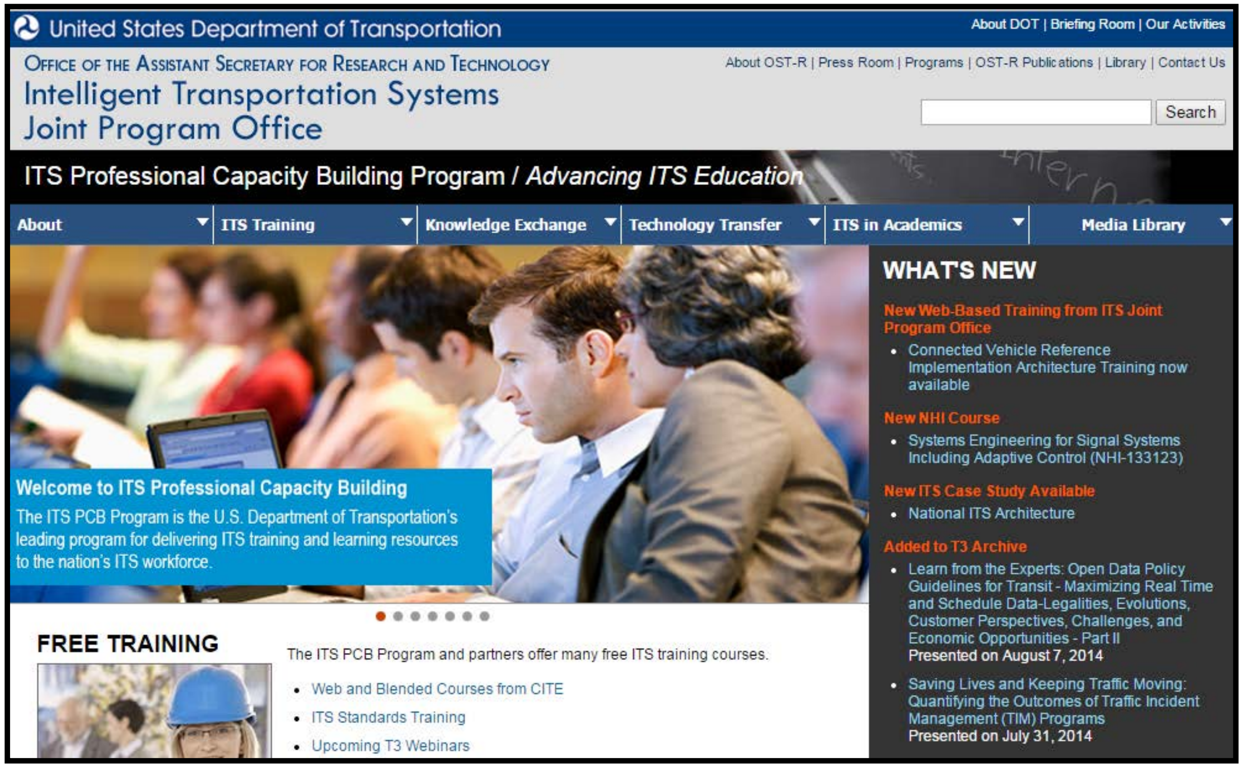
WELCOME

U.S. Department of Transportation

Office of the Assistant Secretary for Research and Technology

Welcome

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



United States Department of Transportation

About DOT | Briefing Room | Our Activities

Office of the Assistant Secretary for Research and Technology

Intelligent Transportation Systems
Joint Program Office

About OST-R | Press Room | Programs | OST-R Publications | Library | Contact Us

Search

ITS Professional Capacity Building Program / *Advancing ITS Education*

About | ITS Training | Knowledge Exchange | Technology Transfer | ITS in Academics | Media Library

Welcome to ITS Professional Capacity Building

The ITS PCB Program is the U.S. Department of Transportation's leading program for delivering ITS training and learning resources to the nation's ITS workforce.

FREE TRAINING

The ITS PCB Program and partners offer many free ITS training courses.

- Web and Blended Courses from CITE
- ITS Standards Training
- Upcoming T3 Webinars

WHAT'S NEW

New Web-Based Training from ITS Joint Program Office

- Connected Vehicle Reference Implementation Architecture Training now available

New NHI Course

- Systems Engineering for Signal Systems Including Adaptive Control (NHI-133123)

New ITS Case Study Available

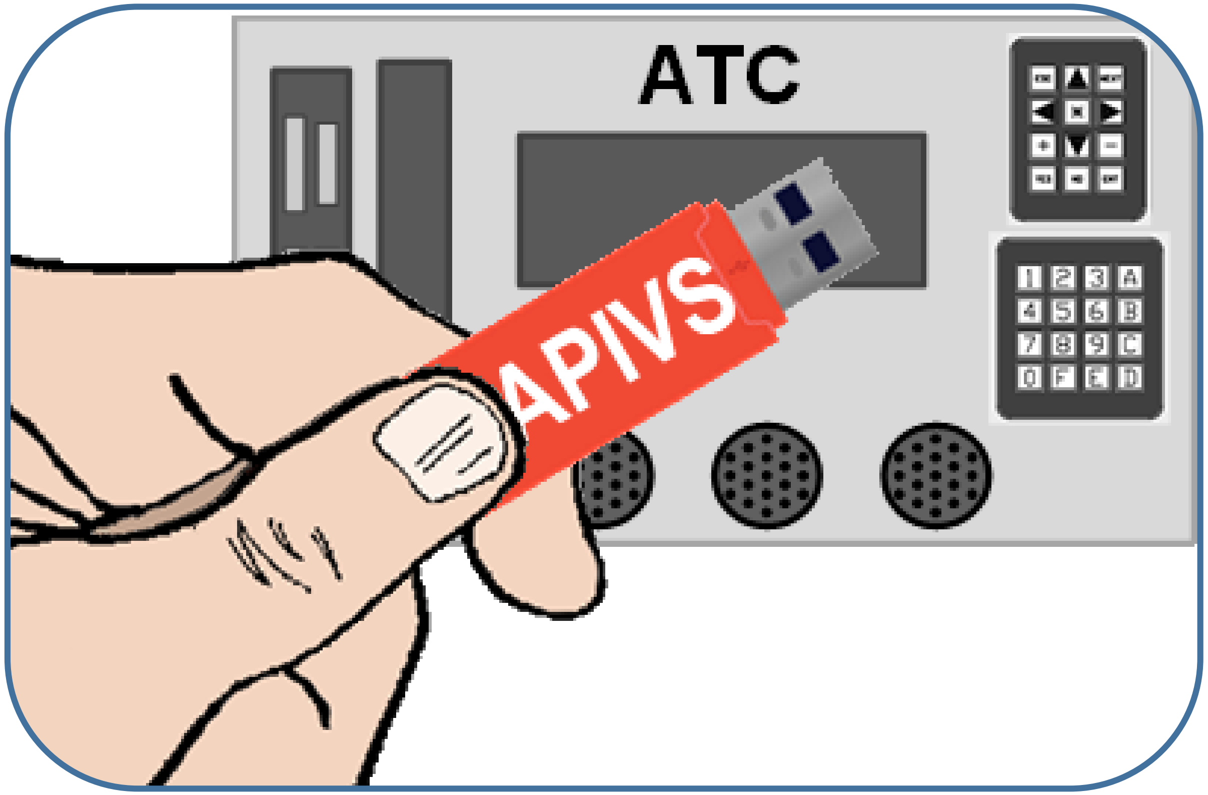
- National ITS Architecture

Added to T3 Archive

- Learn from the Experts: Open Data Policy Guidelines for Transit - Maximizing Real Time and Schedule Data-Legalities, Evolutions, Customer Perspectives, Challenges, and Economic Opportunities - Part II
Presented on August 7, 2014
- Saving Lives and Keeping Traffic Moving: Quantifying the Outcomes of Traffic Incident Management (TIM) Programs
Presented on July 31, 2014

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**

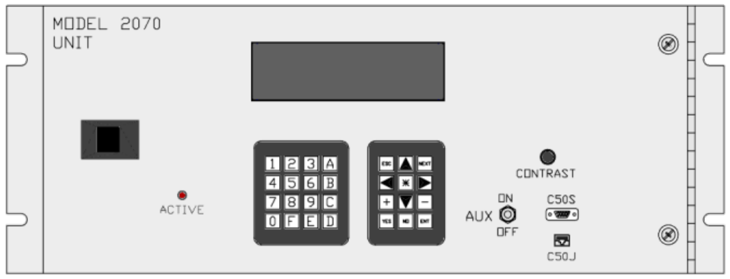
API Software Testing in the Context of ATC Unit Testing

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**

API Software Testing in the Context of ATC Unit Testing

Traditional controller units typically perform a single application



{

Traditional Applications

- Data Collection Application

OR

- Traffic Signal Application

OR

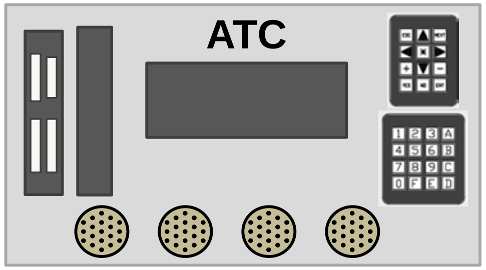
- Ramp Meter Application

Graphic: Ralph W. Boaz

API Software Testing in the Context of ATC Unit Testing

ATC units can perform numerous applications simultaneously

When using ATC API Software



ATC
ACTIVE
AUX
ON
OFF

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

API Software Testing in the Context of ATC Unit Testing

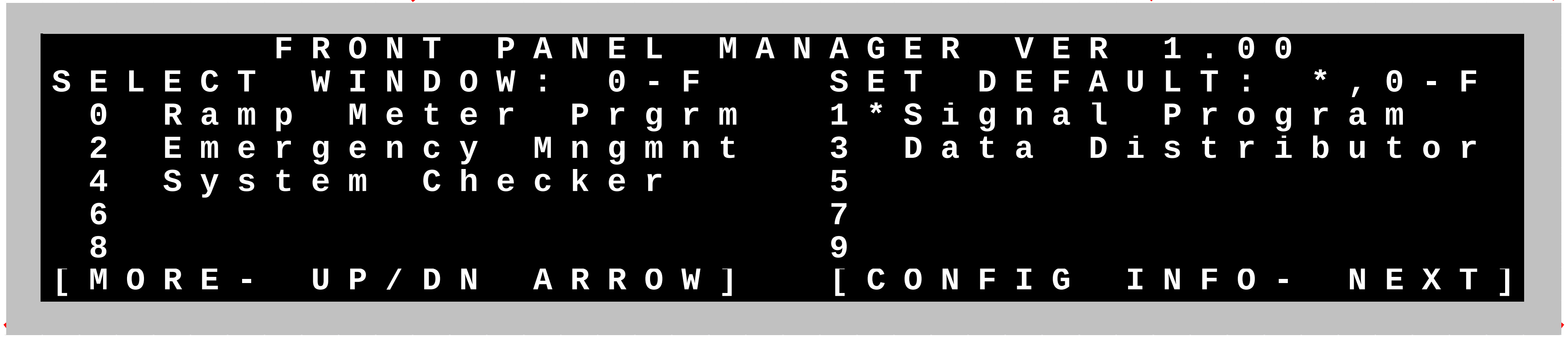
Elements of API Software

- 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**

Graphic: Thinkstock

API Software Testing in the Context of ATC Unit Testing

Example of the Front Panel Manager Window



```
      FRONT PANEL MANAGER VER 1.00
SELECT WINDOW: 0-F    SET DEFAULT: *,0-F
 0  Ramp Meter Prgrm   1*Signal Program
 2  Emergency Mngmnt   3 Data Distributor
 4  System Checker     5
 6                     7
 8                     9
[MORE- UP/DN ARROW]   [CONFIG INFO- NEXT]
```

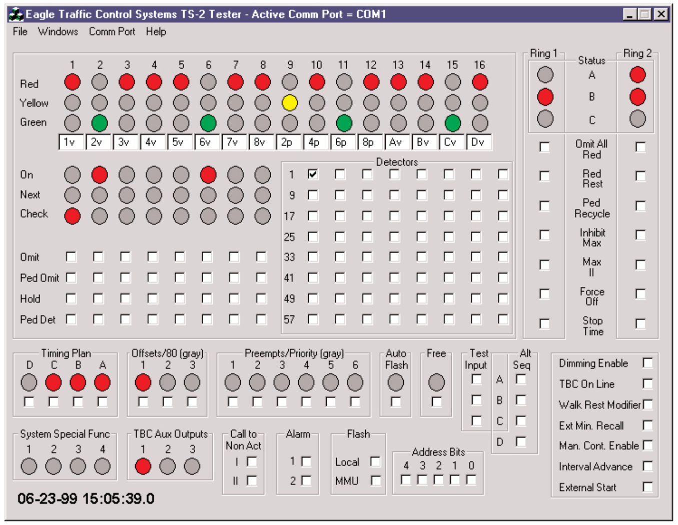
Graphic: Ralph W. Boaz

API Software Testing in the Context of ATC Unit Testing

Unit Testing

Traditional controller unit testing:

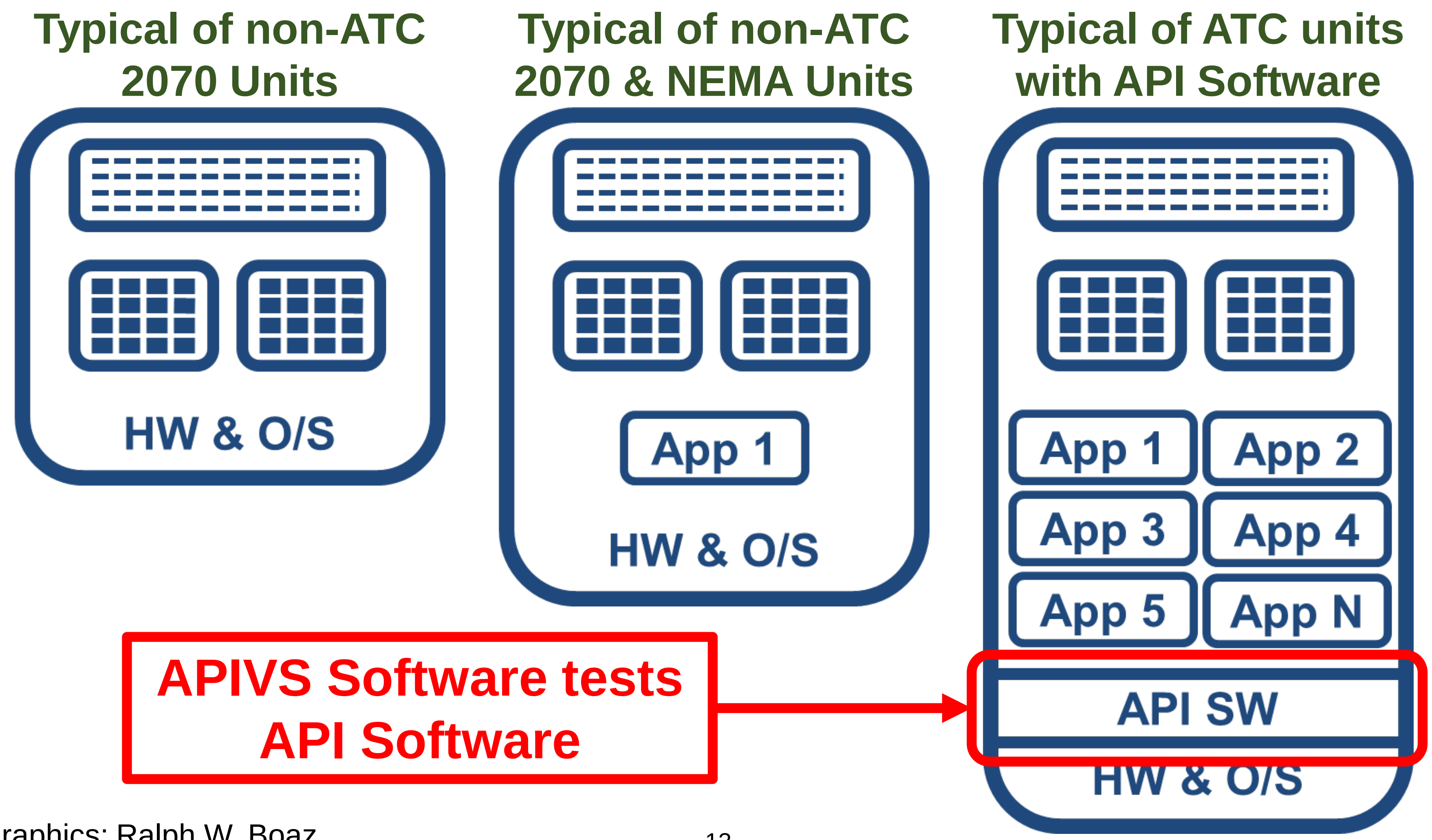
- **Tests** the **controller hardware** (may include the operating system)
- **Tests** the **application program** running on the controller



Graphics: Econolite (U), Siemens (LL), McCain (LR)

API Software Testing in the Context of ATC Unit Testing

API Validation Suite (APIVS) Software tests API Software



Typical of non-ATC 2070 Units
HW & O/S
Typical of non-ATC 2070 & NEMA Units
App 1
HW & O/S
Typical of ATC units with API Software
App 1
App 2
App 3
App 4
App 5
App N
API SW
HW & O/S
APIVS Software tests 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 **<u>used for testing</u>**
- Testing involves **initiating a test** and **comparing the result** to a known correct result
- Must be **repeatable**

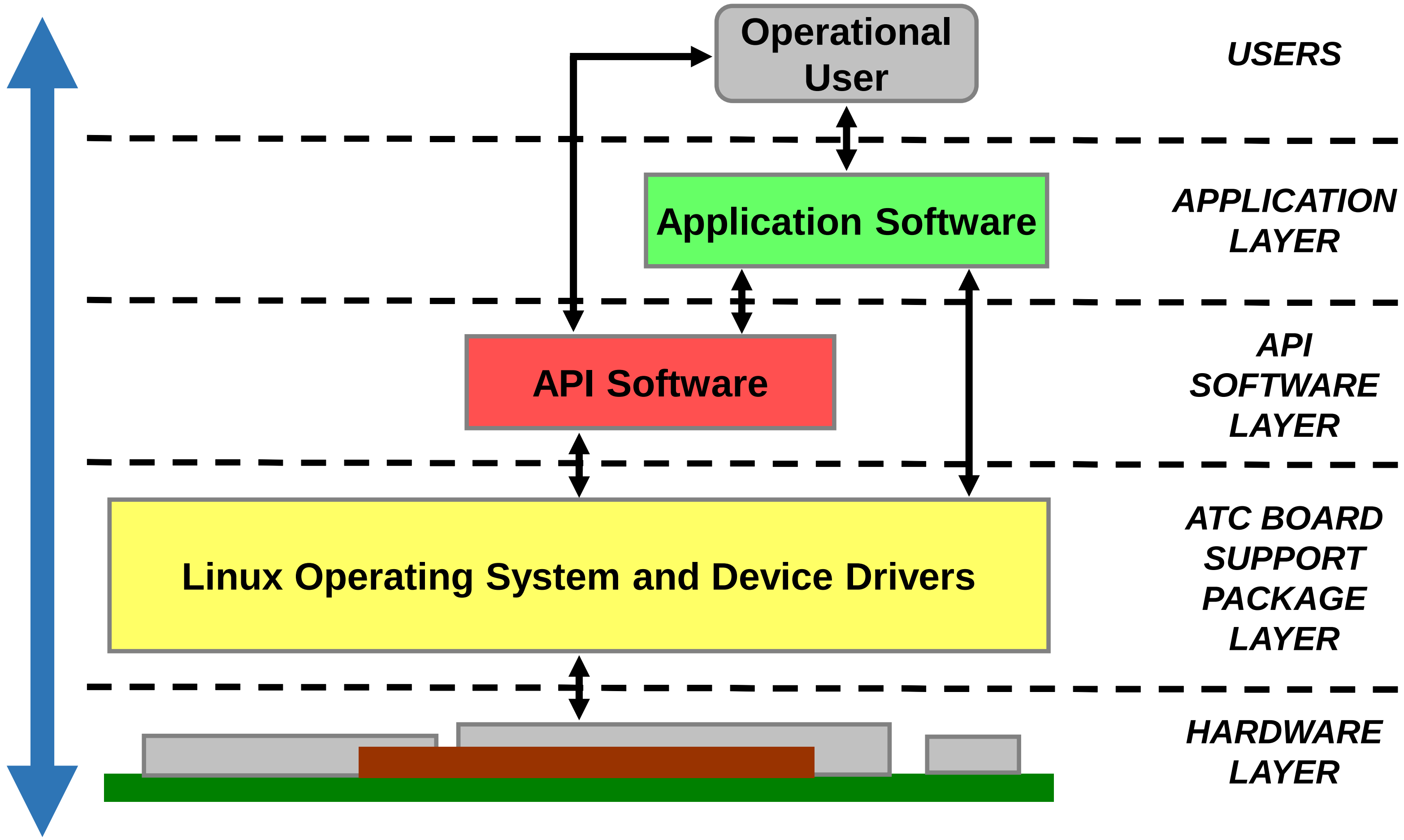
Architecture of the APIVS Software

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

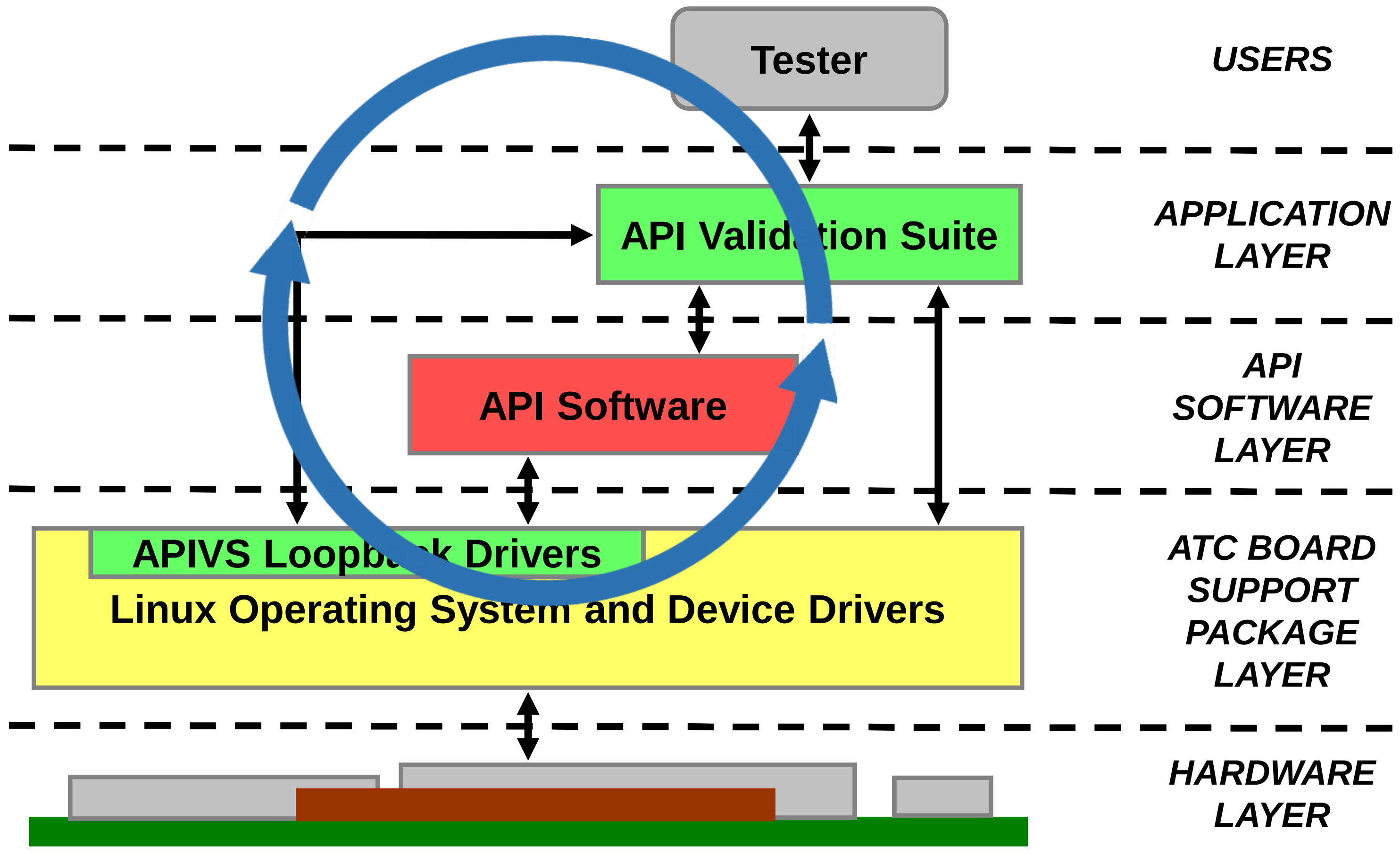
Recall the layered ATC software architecture



Graphic: Ralph W. Boaz

Architecture of the APIVS Software

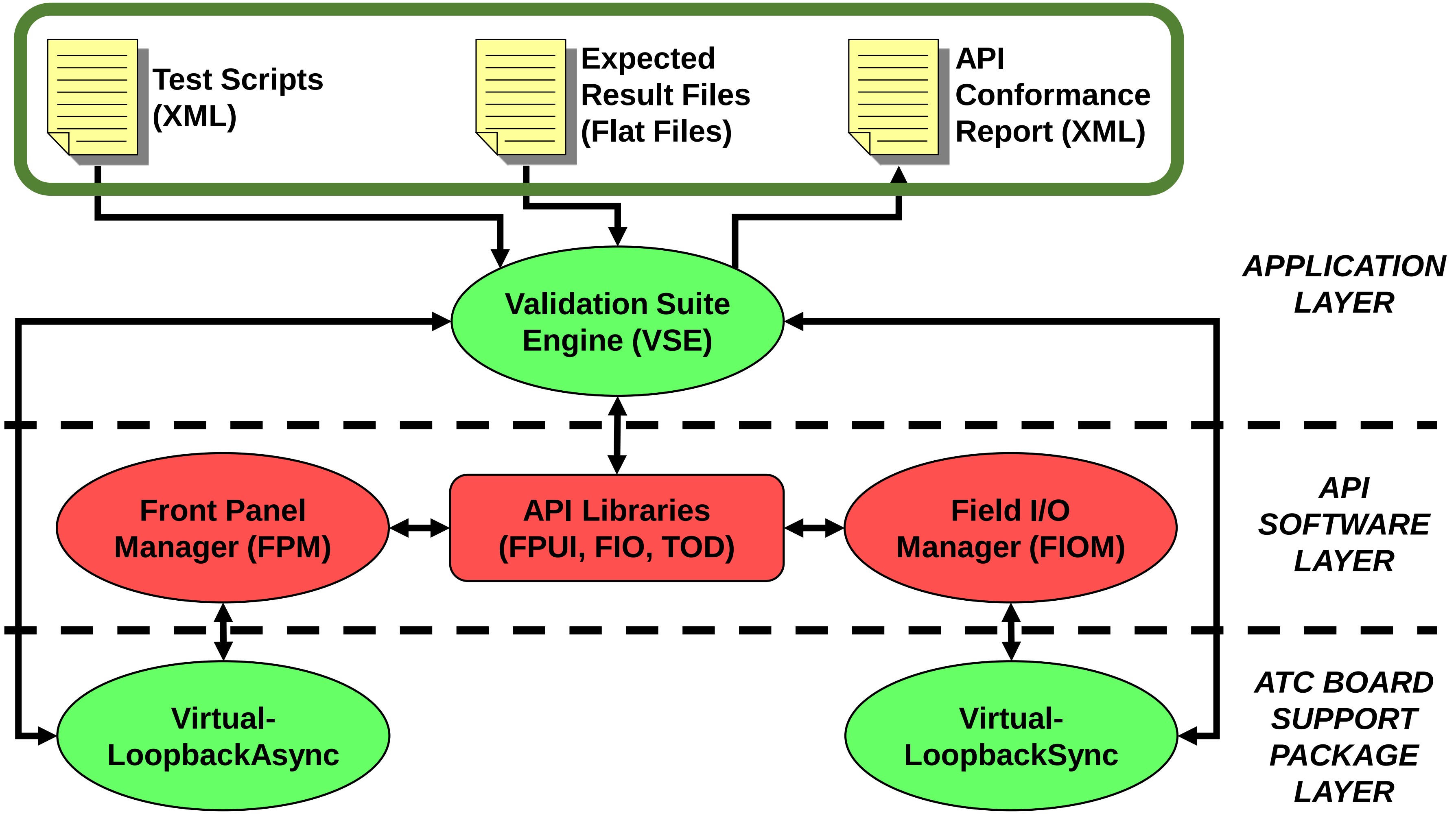
Modified architecture for the APIVS Software



Graphics: Ralph W. Boaz

Architecture of the APIVS Software

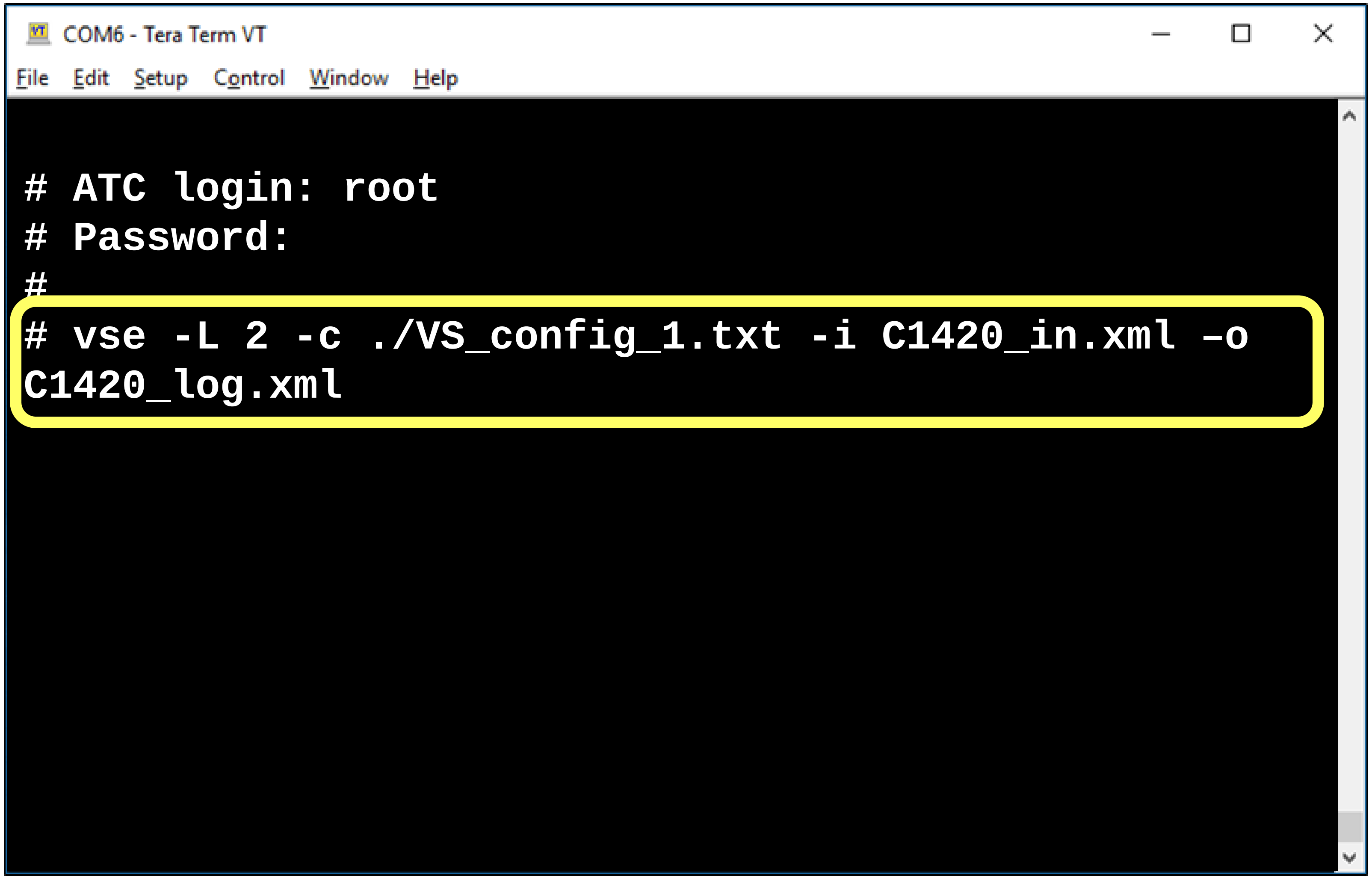
Detailed architecture of the APIVS Software


Test Scripts (XML)
Expected Result Files (Flat Files)
API Conformance Report (XML)
Validation Suite Engine (VSE)
APPLICATION LAYER
Front Panel Manager (FPM)
API Libraries (FPUI, FIO, TOD)
Field I/O Manager (FIOM)
API SOFTWARE LAYER
Virtual-LoopbackAsync
Virtual-LoopbackSync
ATC BOARD SUPPORT PACKAGE LAYER

Graphic: Ralph W. Boaz

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

vse -L [1-3] [-c configuration-file] [-i APIVSXML-file] [-o output-file] [-n test_suite_name] [-R count] [-H] [-C]

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

vse -L [1-3] [-c configuration-file] [-i APIVSXML-file] [-o output-file] [-n test_suite_name] [-R count] [-H] [-C]

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

vse -L [1-3] [-c configuration-file] [-i APIVSXML-file] [-o output-file] [-n test_suite_name] [-R count] [-H] [-C]

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

vse -L [1-3] [-c configuration-file] [-i APIVSXML-file] [-o output-file] [-n test_suite_name] [-R count] [-H] [-C]

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

vse -L [1-3] [-c configuration-file] [-i APIVSXML-file] [-o output-file] [-n test_suite_name] [-R count] [-H] [-C]

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.)

vse -L [1-3] [-c configuration-file] [-i APIVSXML-file] [-o output-file] [-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.)

vse -L [1-3] [-c configuration-file] [-i APIVSXML-file] [-o output-file] [-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.)

vse -L [1-3] [-c configuration-file] [-i APIVSXML-file] [-o output-file] [-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.)

vse -L [1-3] [-c configuration-file] [-i APIVSXML-file] [-o output-file] [-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.

ACTIVITY

U.S. Department of Transportation
Office of the Assistant Secretary for Research and Technology

Question

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

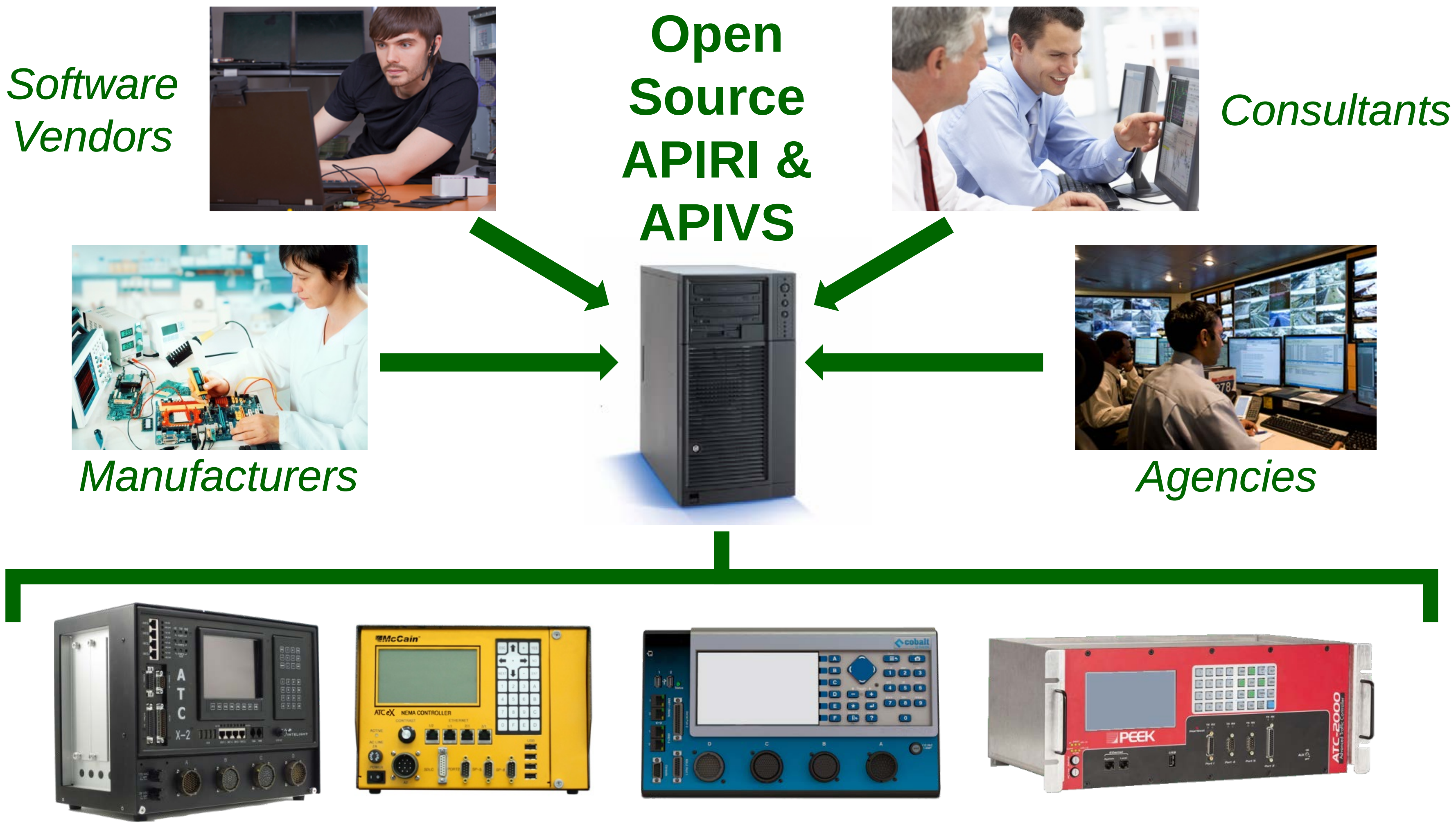
Explain the **purpose of** the **API Validation Suite** (APIVS) **Software**

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



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

API Reference Implementation (APIRI) Project

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**

Organization and Content of the APIRI Test Documentation

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

Test Document	Purpose
Test Plan	**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.
Test Design Specification (TDS)	**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.
Test Case Specification (TCS)	**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.
Test Procedure Specification (TPS)	**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.

Organization and Content of the APIRI Test Documentation

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

Test Document	Purpose
Test Plan	**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.
Test Design Specification (TDS)	**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.
Test Case Specification (TCS)	**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.
Test Procedure Specification (TPS)	**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.

Organization and Content of the APIRI Test Documentation

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

Test Document	Purpose
Test Plan	**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.
Test Design Specification (TDS)	**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.
Test Case Specification (TCS)	**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.
Test Procedure Specification (TPS)	**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.

Organization and Content of the APIRI Test Documentation

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

Test Document	Purpose
Test Plan	**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.
Test Design Specification (TDS)	**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.
Test Case Specification (TCS)	**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.
Test Procedure Specification (TPS)	**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.

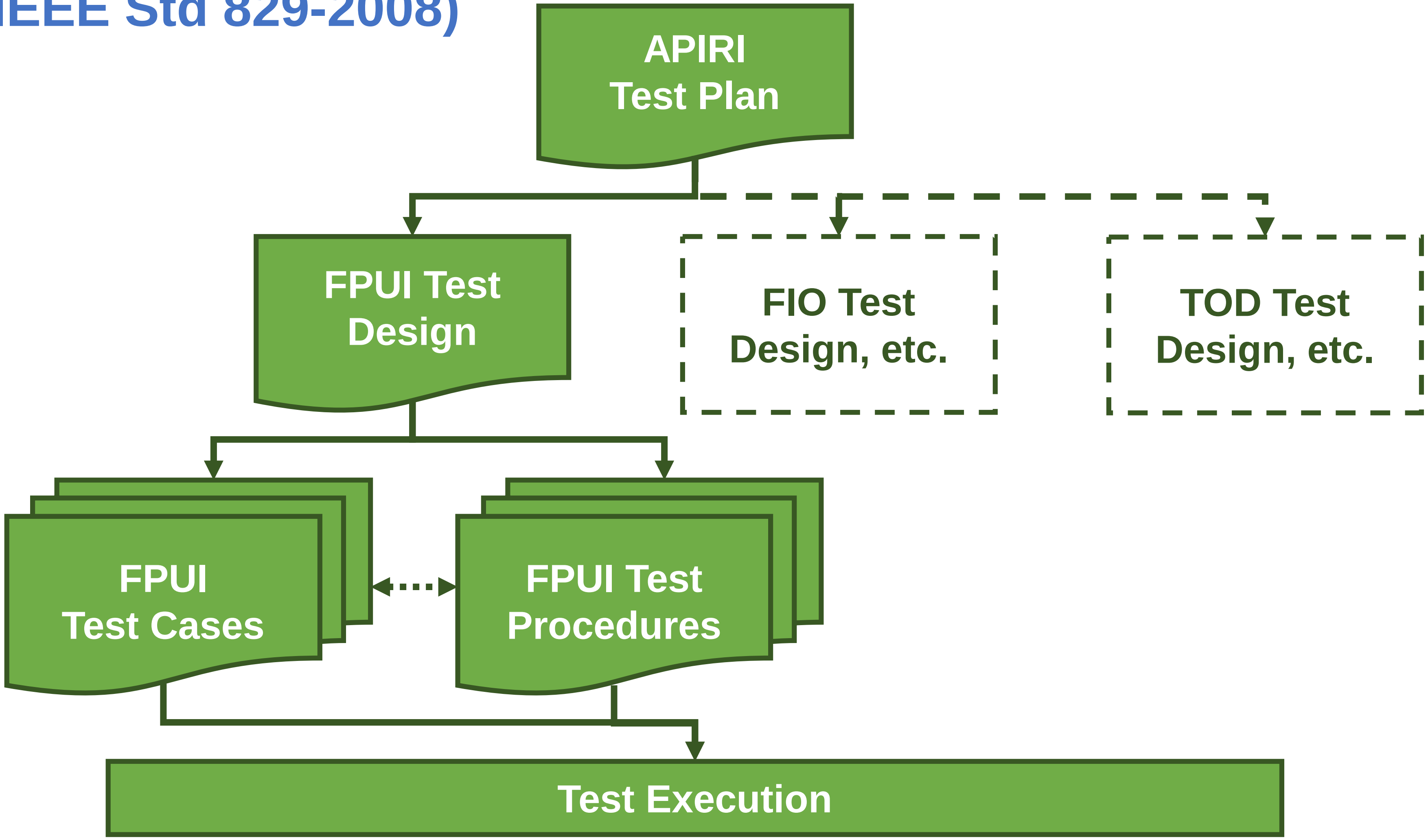
Organization and Content of the APIRI Test Documentation

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

Test Document	Purpose
Test Plan	**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.
Test Design Specification (TDS)	**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.
Test Case Specification (TCS)	**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.
Test Procedure Specification (TPS)	**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.

Organization and Content of the APIRI Test Documentation

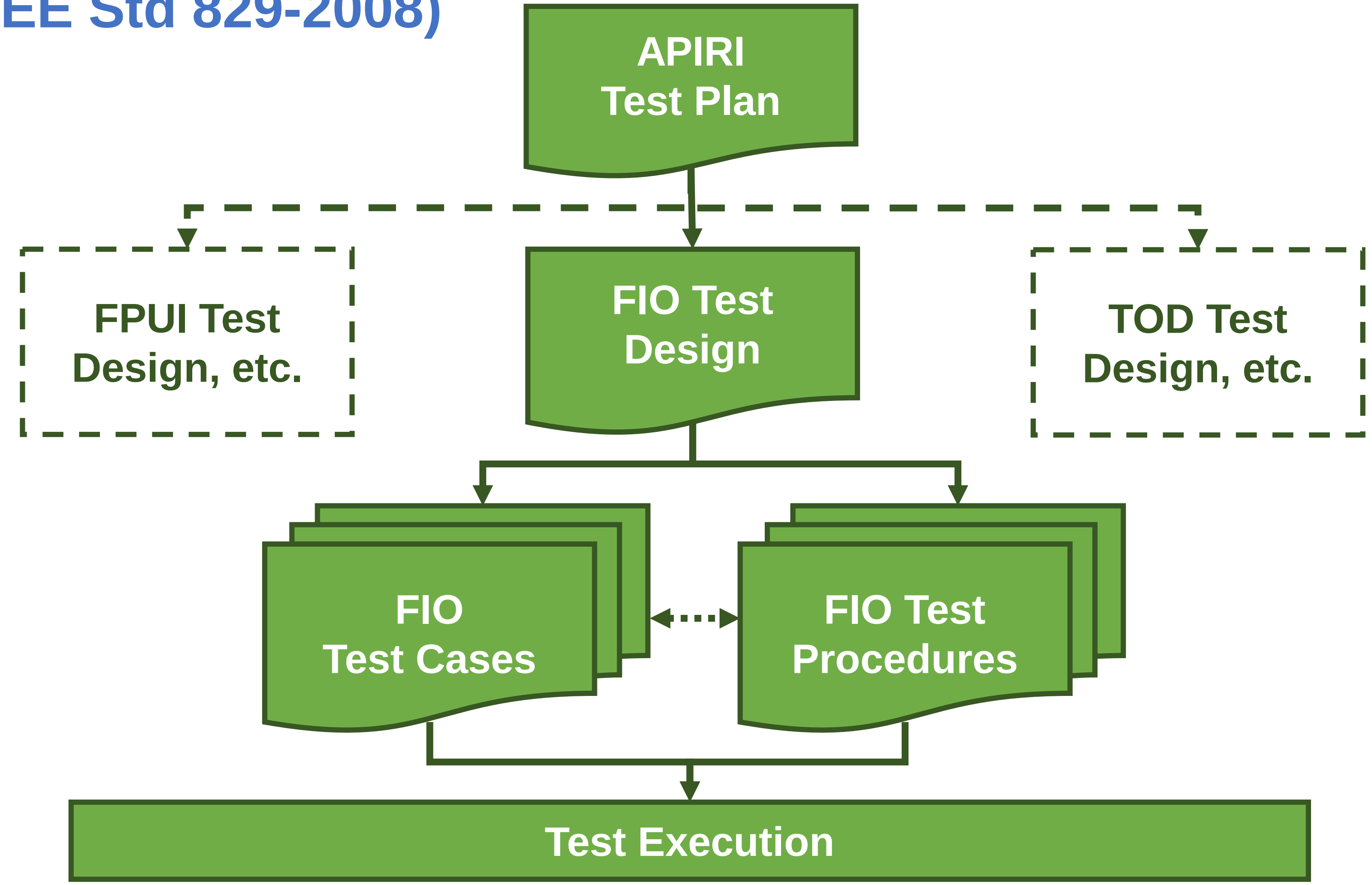
APIRI Documents Used to Specify Testing (IEEE Std 829-2008)



Graphic: Ralph W. Boaz

Organization and Content of the APIRI Test Documentation

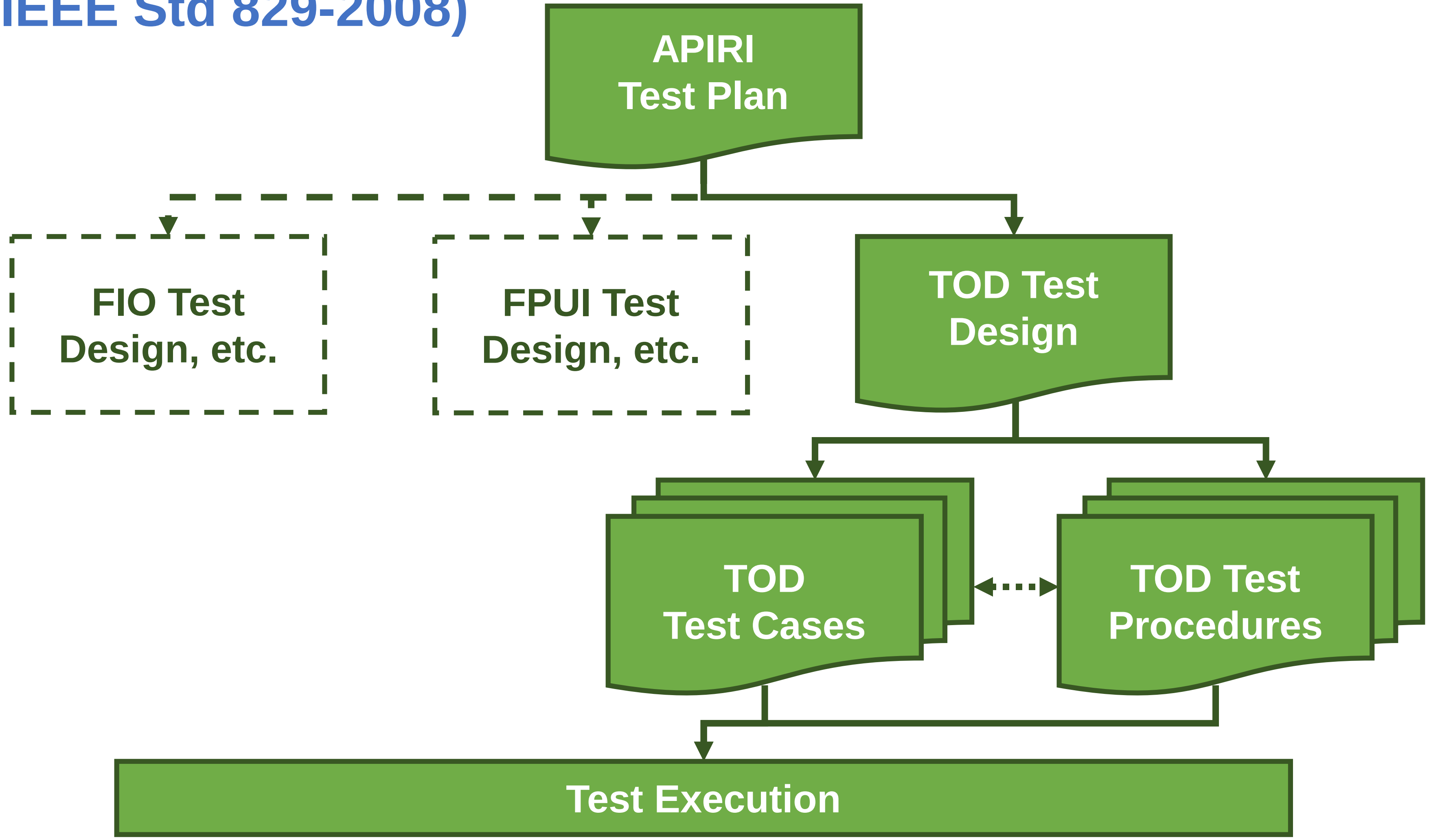
APIRI Documents Used to Specify Testing (IEEE Std 829-2008)



Graphic: Ralph W. Boaz

Organization and Content of the APIRI Test Documentation

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



Graphic: Ralph W. Boaz

Organization and Content of the APIRI Test Documentation

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

Organization and Content of the APIRI Test Documentation

Features to Be Tested

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

Organization and Content of the APIRI Test Documentation

APIRI Test Plan Outline (cont.)

Organization and Content of the APIRI Test Documentation

FPUI Library Requirements to Validation Description Matrix

Req ID	Req Description	ATC API Function	APIRI SDD Design Narrative	Test Cases	Test Procedures
…	…	…	…	…	…
APIR3.1.1.2[10]	The API shall provide a function to read a queued character or key code from the input buffer of a window.	fpui_read_char	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).	APIRI.TCS.2040	APIRI.TPS.1001
APIR3.1.1.2[11]	The API shall provide a function to write a character to the current cursor position of a window.	fpui_write_char	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).	APIRI.TCS.2040	APIRI.TPS.1001
…	…	…	…	…	…

Organization and Content of the APIRI Test Documentation

APIRI Test Design Outlines

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…

Organization and Content of the APIRI Test Documentation

APIRI Test Design Outlines (cont.)

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.

Organization and Content of the APIRI Test Documentation

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.

APIRI Test Case Outlines (cont.)

2.6.4 Input Specifications

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

File	Description
C2040_in.xml	APIVSXML test script (XML format)
Cxxxx_key0.txt	keystroke file (Key ‘0’)
Cxxxx_key1.txt	keystroke file (Key ‘1’)
Cxxxx_keyESC.txt	keystroke file (Key ‘<Esc>’)
C2040_vd_1.txt	Virtual Display compare file (display 1)
VS_config_1.txt	VSE configuration file (for VSE command line)

2.6.5 Output Specifications

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

File	Description
C2040_log.xml	Conformance report (XML format)

Organization and Content of the APIRI Test Documentation

APIRI Test Procedure Outlines

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…

Organization and Content of the APIRI Test Documentation

APIRI Test Procedure Outlines (cont.)

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

Test Document	Purpose
APIRI Test Scripts	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.
Flat Files	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.
Validation Suite Engine (VSE) Configuration File	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.
Linux Shell Scripts	Allows the testers to **run successive executions** of the VSE **without typing** them in a line at a time.
Output Files	**Output** from the VSE **in XML format**, which allows various tools to be used for analyzing test results.

Files Needed for Executing the Test Cases

Test Document	Purpose
APIRI Test Scripts	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.
Flat Files	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.
Validation Suite Engine (VSE) Configuration File	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.
Linux Shell Scripts	Allows the testers to **run successive executions** of the VSE **without typing** them in a line at a time.
Output Files	**Output** from the VSE **in XML format**, which allows various tools to be used for analyzing test results.

Files Needed for Executing the Test Cases

Test Document	Purpose
APIRI Test Scripts	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.
Flat Files	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.
Validation Suite Engine (VSE) Configuration File	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.
Linux Shell Scripts	Allows the testers to **run successive executions** of the VSE **without typing** them in a line at a time.
Output Files	**Output** from the VSE **in XML format**, which allows various tools to be used for analyzing test results.

Files Needed for Executing the Test Cases

Test Document	Purpose
APIRI Test Scripts	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.
Flat Files	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.
Validation Suite Engine (VSE) Configuration File	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.
Linux Shell Scripts	Allows the testers to **run successive executions** of the VSE **without typing** them in a line at a time.
Output Files	**Output** from the VSE **in XML format**, which allows various tools to be used for analyzing test results.

Files Needed for Executing the Test Cases

Test Document	Purpose
APIRI Test Scripts	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.
Flat Files	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.
Validation Suite Engine (VSE) Configuration File	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.
Linux Shell Scripts	Allows the testers to **run successive executions** of the VSE **without typing** them in a line at a time.
Output Files	**Output** from the VSE **in XML format**, which allows various tools to be used for analyzing test results.

Files Needed for Executing the Test Cases

Test Document	Purpose
APIRI Test Scripts	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.
Flat Files	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.
Validation Suite Engine (VSE) Configuration File	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.
Linux Shell Scripts	Allows the testers to **run successive executions** of the VSE **without typing** them in a line at a time.
Output Files	**Output** from the VSE **in XML format**, which allows various tools to be used for analyzing test results.

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
```

Files Needed for Executing the Test Cases

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                = ./
ScreenHeight               = 8
ScreenWidth                = 40
FPUICompareFilePath        = ./
FPUIInputFilePath          = ./
…
```

ACTIVITY

U.S. Department of Transportation
Office of the Assistant Secretary for Research and Technology

Question

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**

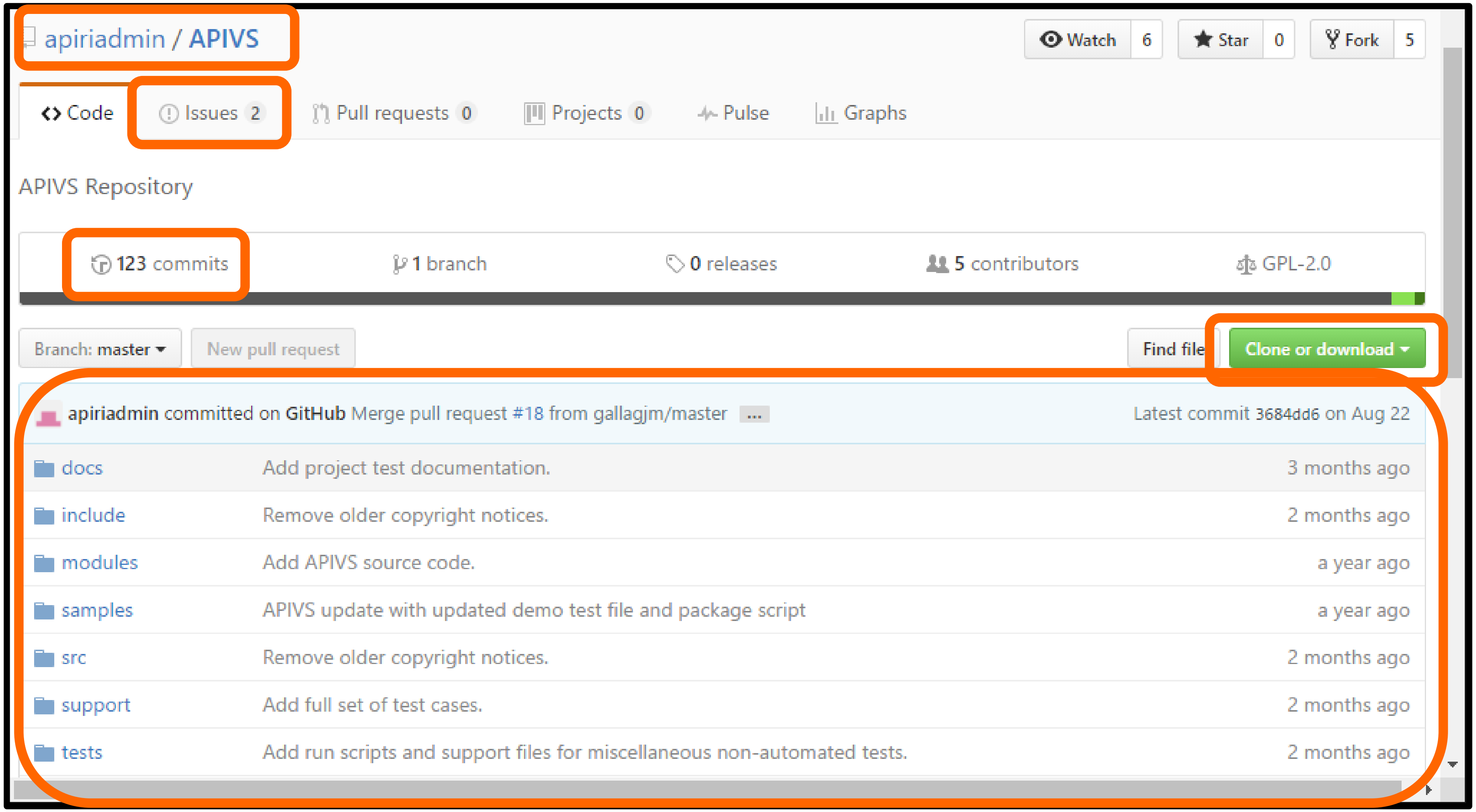
Use the **APIVS Software to test** the **API Software**

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



apiriadmin / APIVS

APIVS Repository

apiriadmin committed on GitHub Merge pull request #18 from gallagjm/master … Latest commit 3684dd6 on Aug 22

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)

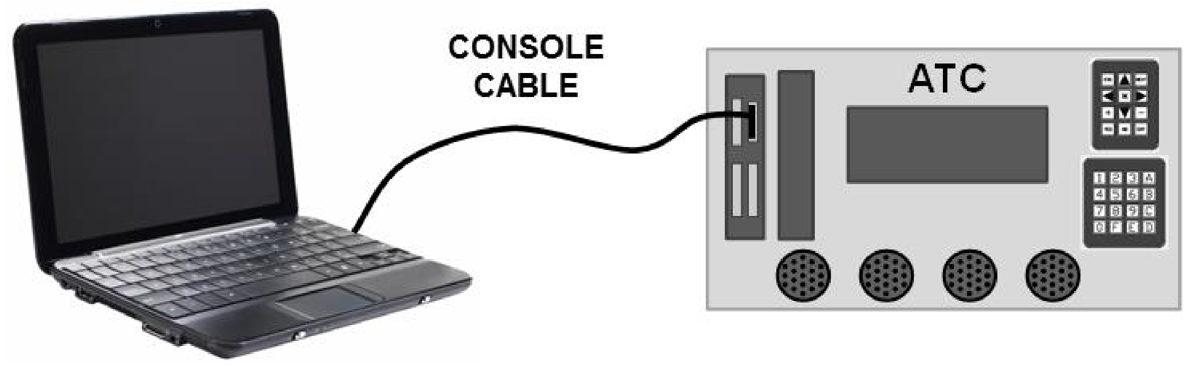
- If using **CLI Method** add
  - Serial or Ethernet cable to connect the PC to the ATC unit

- 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

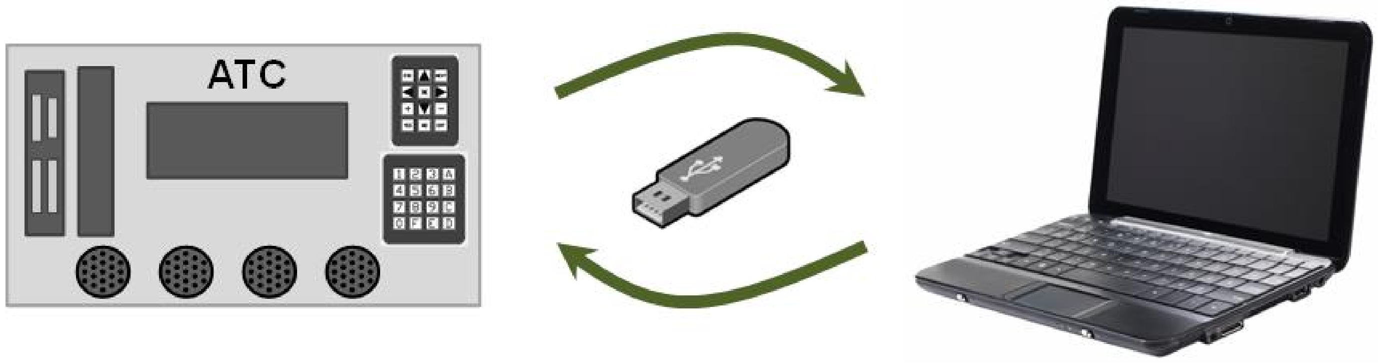


Graphics: Thinkstock (L) and Ralph W. Boaz (R)

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**

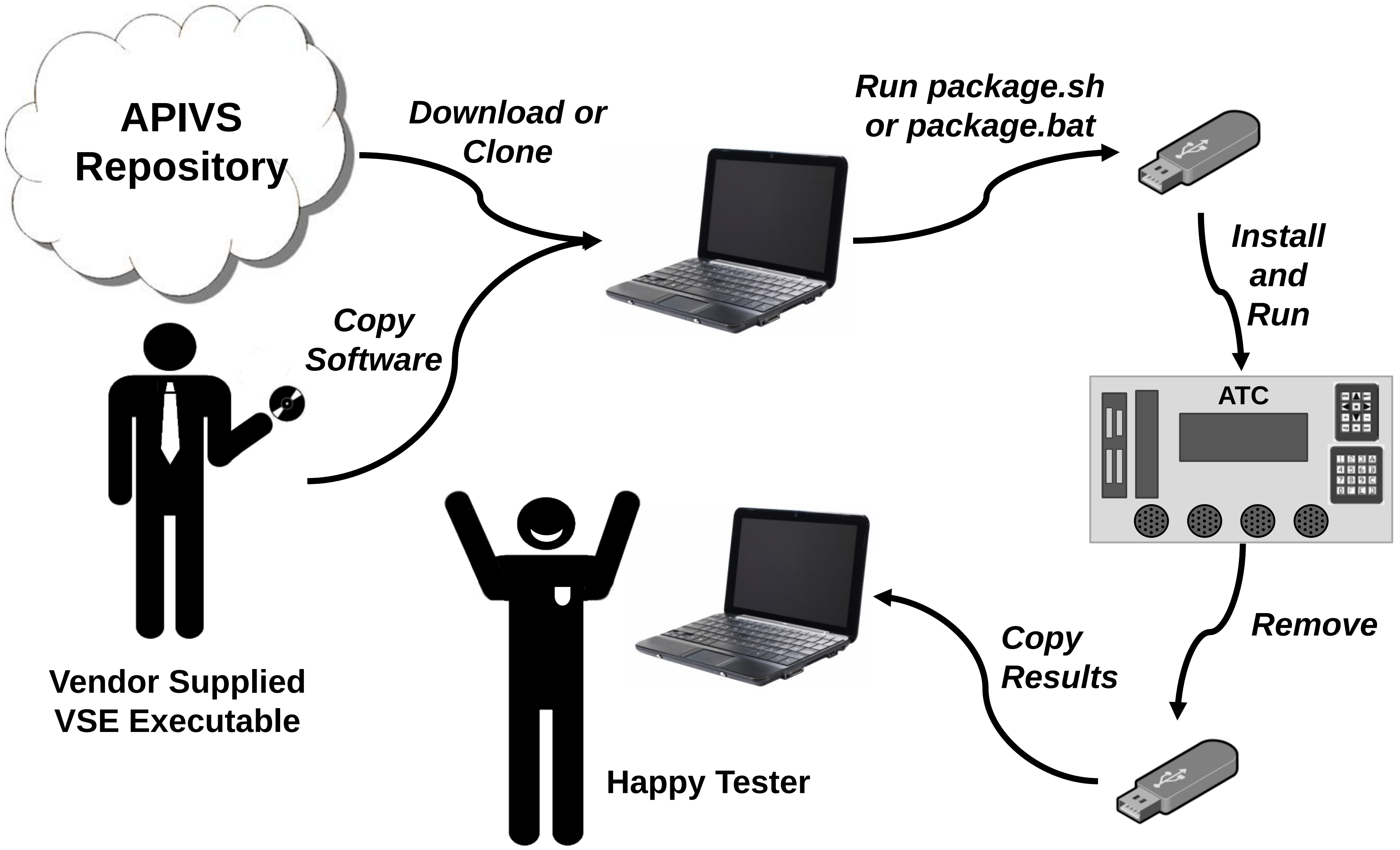


Graphics: Ralph W. Boaz (L) and Thinkstock (R)

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



Prepare the APIVS Software for Testing

Command-Line Interface (CLI) of the APIVS Software

vse -L [1-3] [-c configuration-file] [-i APIVSXML-file] [-o output-file] [-n test_suite_name] [-R count] [-H] [-C]

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
…
```

Editing the runAPIVS Linux Shell Script

```
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

…
```

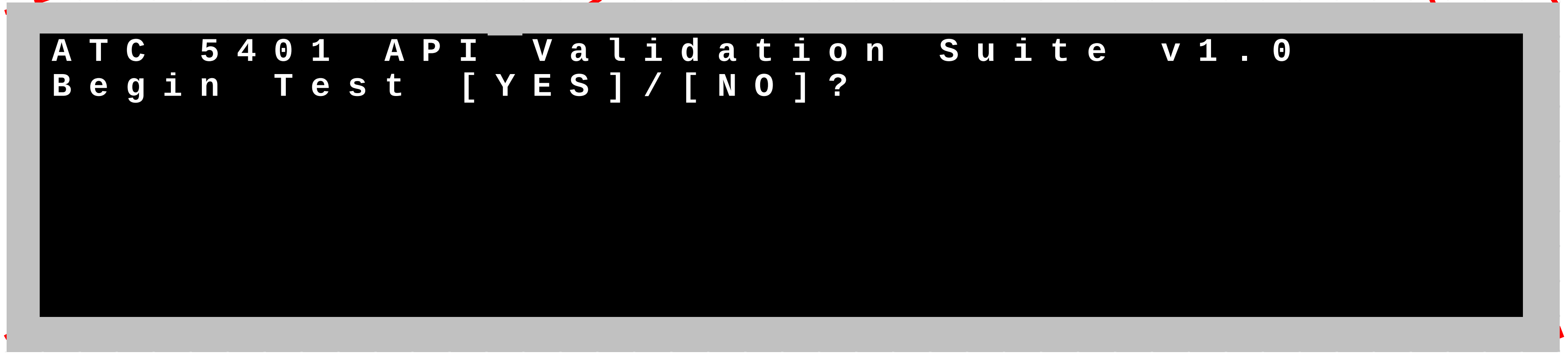
Editing the runAPIVS Linux Shell Script

```
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
…
```

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

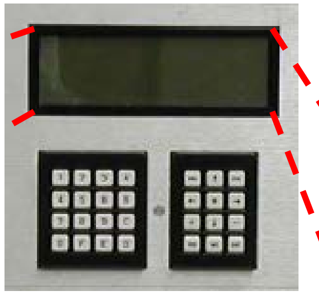


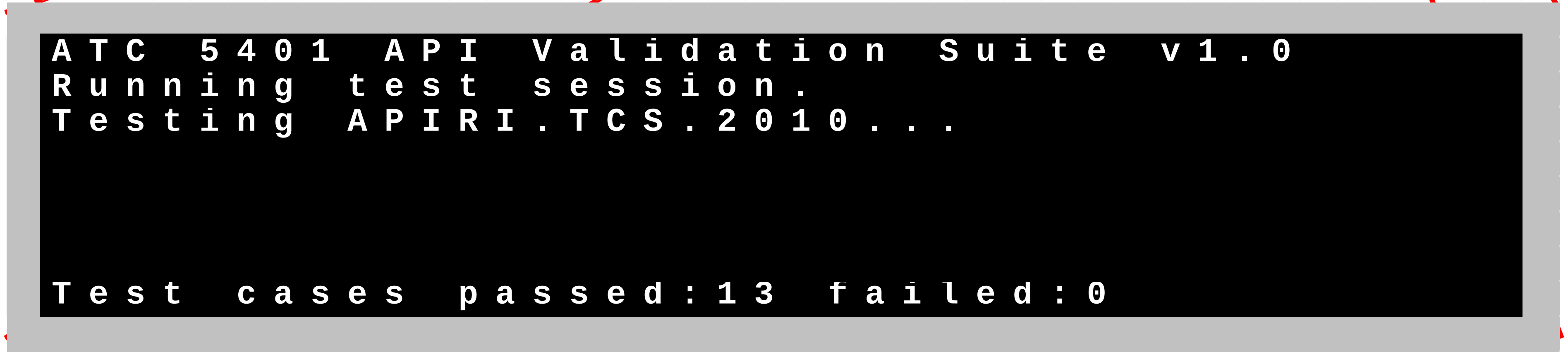
```
ATC 5401 API Validation Suite v1.0
Begin Test [YES]/[NO]?
```

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



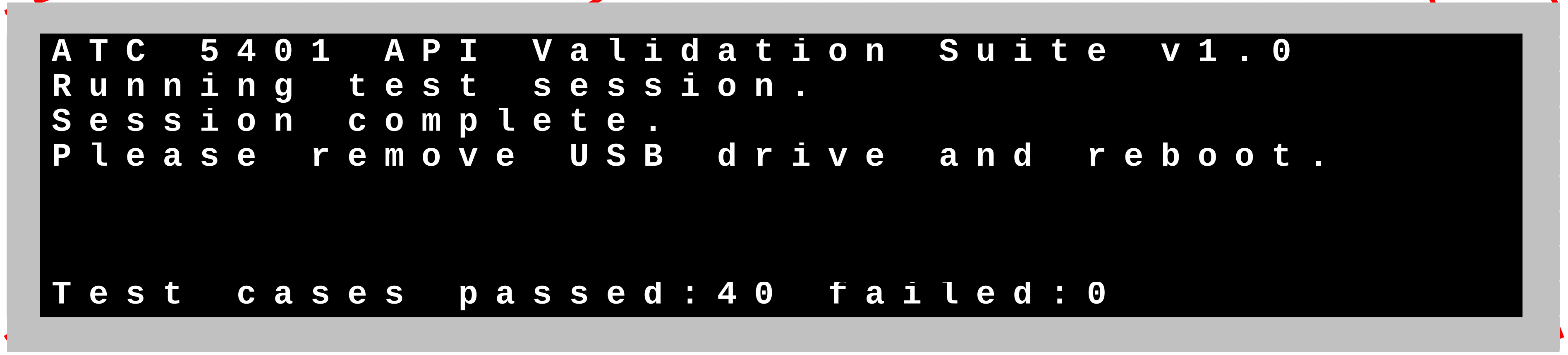


```
ATC 5401 API Validation Suite v1.0
Running test session.
Testing APIRI.TCS.2010...




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

STANDARDS ITS TRAINING

U.S. Department of Transportation
Office of the Assistant Secretary for
Research and Technology

Question

What is <u>not</u> 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

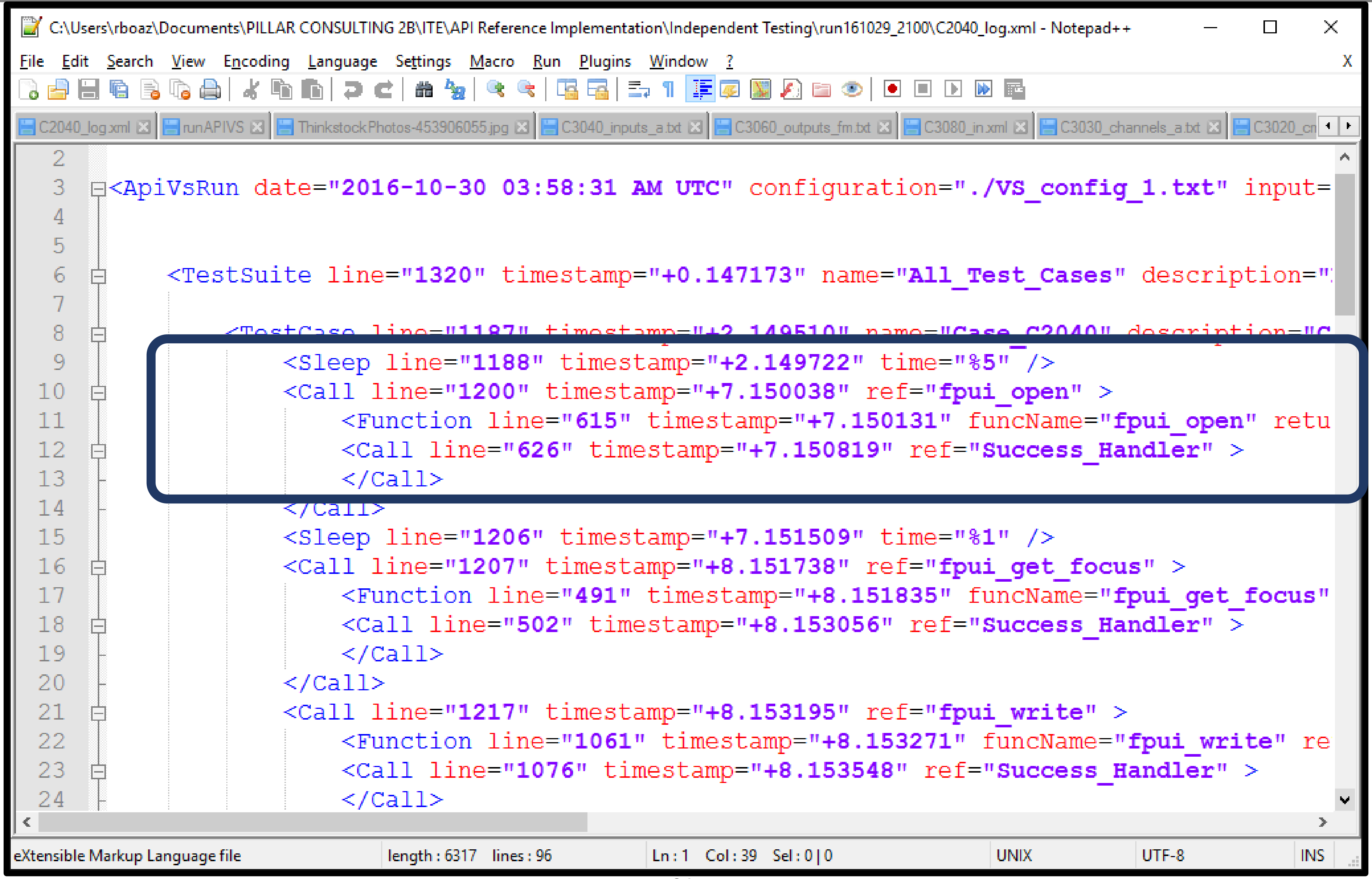
Learning Objective 4

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)
    - **General purpose editing tool** for software-related files
    - Color coding and **formatting of XML text files**
  - **XML Differences** (www.corefiling.com/opensource/xmldiff.html)
    - Online **comparison** of XML files
  - **XmlGrid** (http://xmlgrid.net)
    - Online editor **displays** in formatted XML text or **in grids** (tables)
  - **XML Viewer** (www.codebeautify.org/xmlviewer)
    - Online editor displays in formatted XML text or in **tree view**

Analyze Results Using Off-the-Shelf Tools

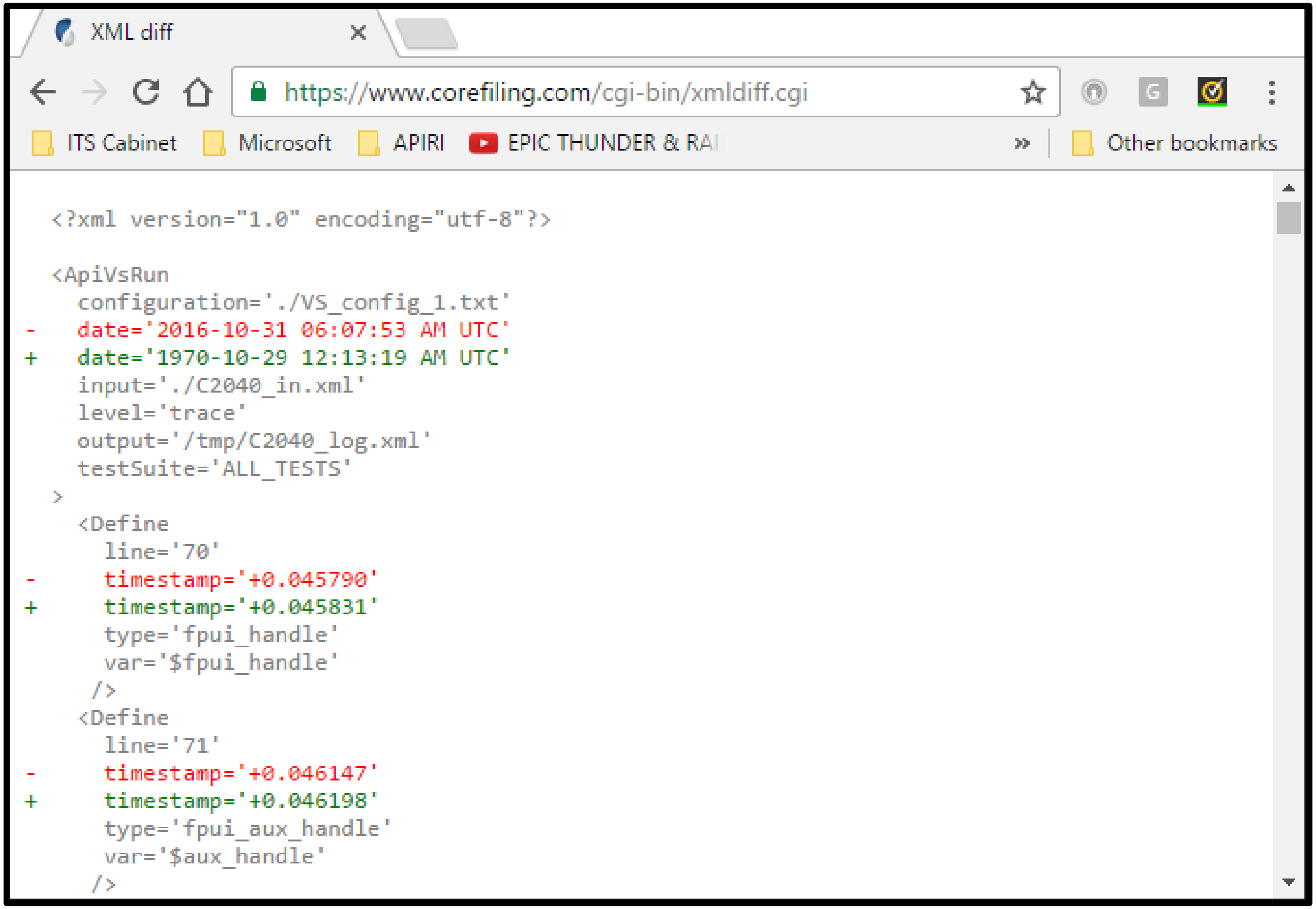

C:\Users\rboaz\Documents\PILLAR CONSULTING 2B\ITE\API Reference Implementation\Independent Testing\run161029_2100\C2040_log.xml - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
C2040_log.xml
runAPIVS
ThinkstockPhotos-453906055.jpg
C3040_inputs_a.txt
C3060_outputs_fm.txt
C3080_in.xml
C3030_channels_a.txt
C3020_cn

```
<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="

        <TestCase line="1187" timestamp="+2.149510" name="Case_C2040" description="C
            <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" retu
                <Call line="626" timestamp="+7.150819" ref="Success_Handler" >
                </Call>
            </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" >
                </Call>
            </Call>
            <Call line="1217" timestamp="+8.153195" ref="fpui_write" >
                <Function line="1061" timestamp="+8.153271" funcName="fpui_write" re
                <Call line="1076" timestamp="+8.153548" ref="Success_Handler" >
                </Call>
```

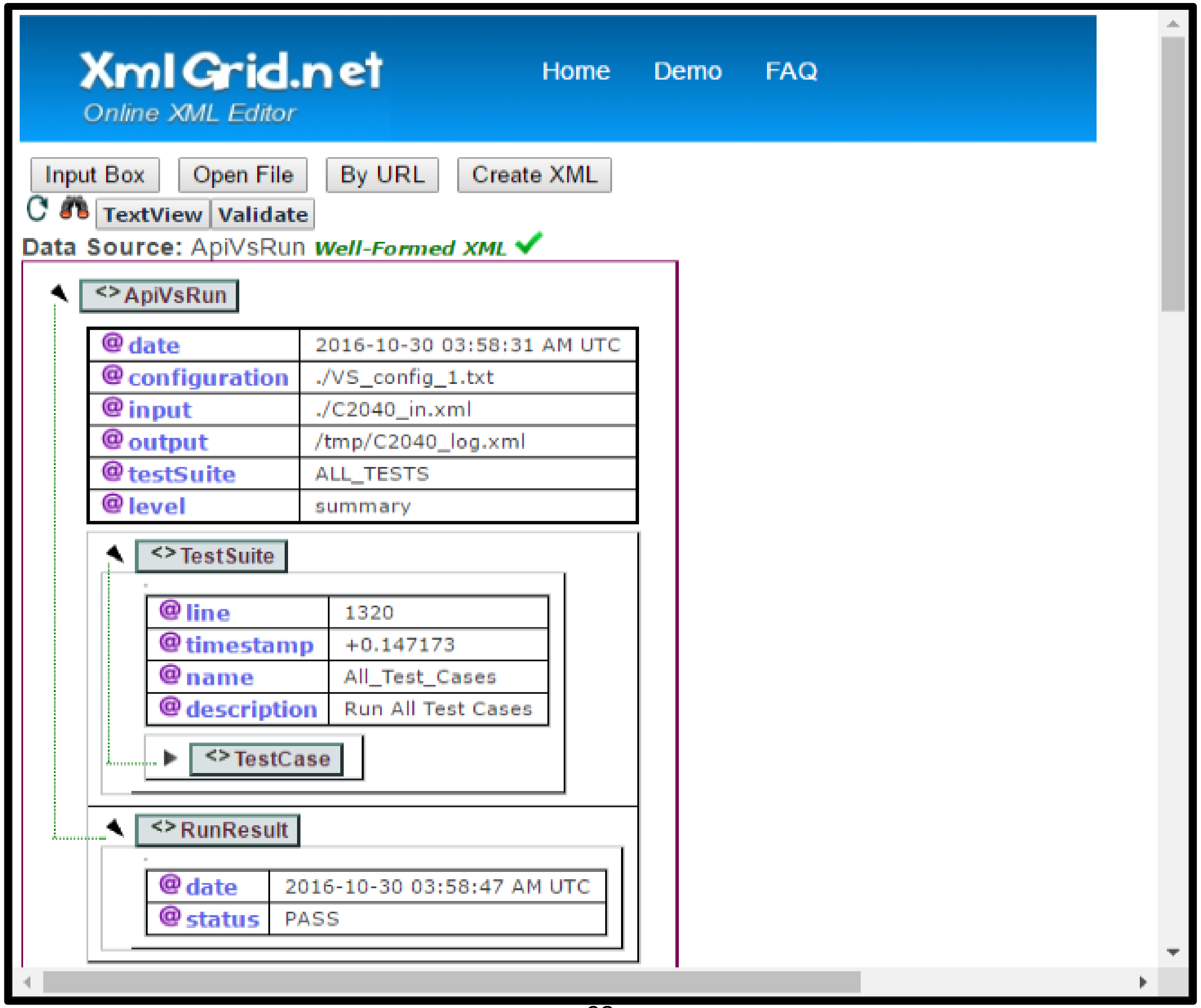
eXtensible Markup Language file   length : 6317   lines : 96   Ln : 1   Col : 39   Sel : 0 | 0   UNIX   UTF-8   INS

Analyze Results Using Off-the-Shelf Tools



```
  <?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



Data Source: ApiVsRun *Well-Formed XML* ✓

<> ApiVsRun

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

<> TestSuite

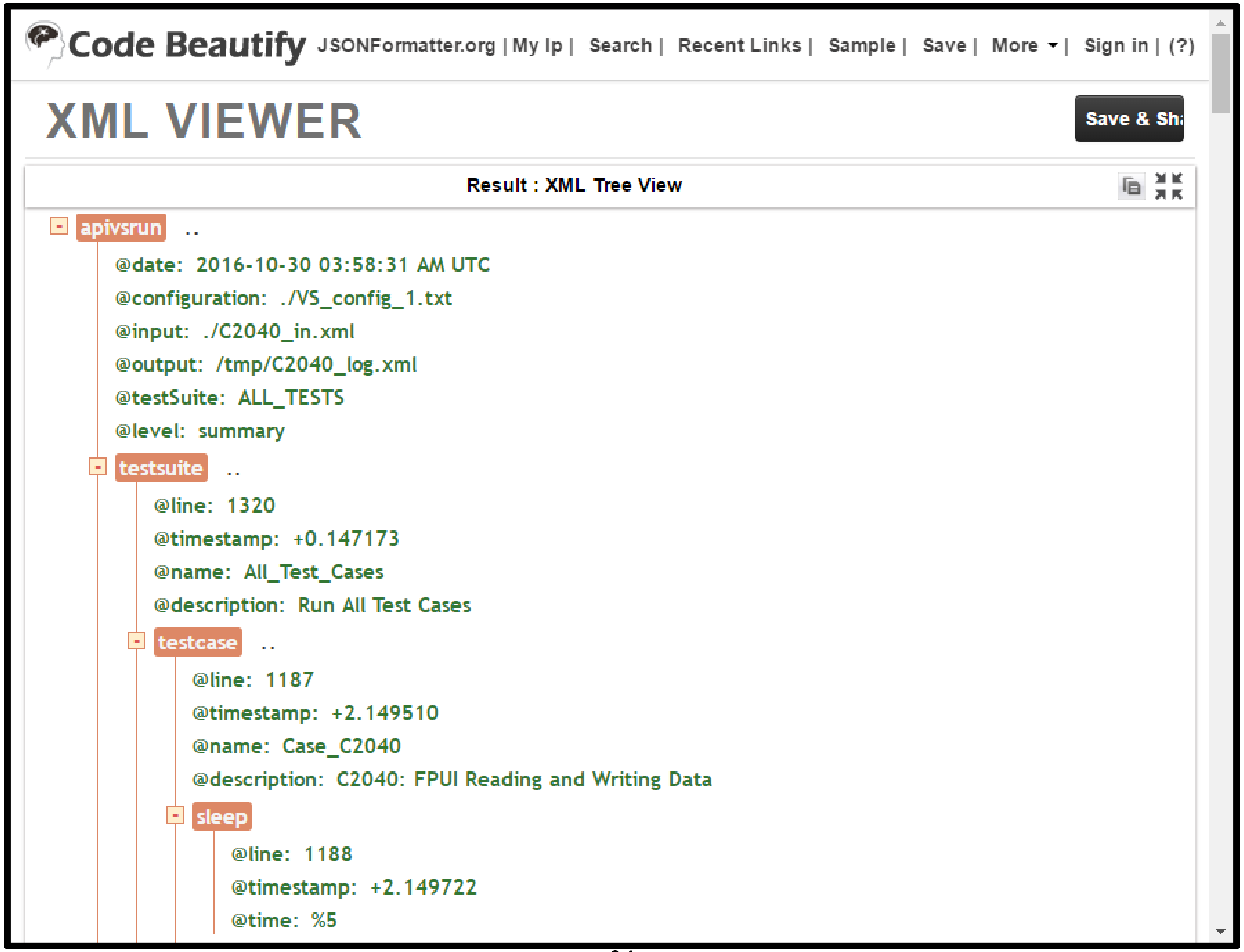
@line	1320
@timestamp	+0.147173
@name	All_Test_Cases
@description	Run All Test Cases

<> TestCase

<> RunResult

@date	2016-10-30 03:58:47 AM UTC
@status	PASS

Analyze Results Using Off-the-Shelf Tools


Code Beautify
JSONFormatter.org | My Ip | Search | Recent Links | Sample | Save | More | Sign in | (?)
XML VIEWER
Save & Sh
Result : XML Tree View

```
apivsrun ..
    @date: 2016-10-30 03:58:31 AM UTC
    @configuration: ./VS_config_1.txt
    @input: ./C2040_in.xml
    @output: /tmp/C2040_log.xml
    @testSuite: ALL_TESTS
    @level: summary
  testsuite ..
      @line: 1320
      @timestamp: +0.147173
      @name: All_Test_Cases
      @description: Run All Test Cases
    testcase ..
        @line: 1187
        @timestamp: +2.149510
        @name: Case_C2040
        @description: C2040: FPUI Reading and Writing Data
      sleep
          @line: 1188
          @timestamp: +2.149722
          @time: %5
```

Create Test Reports Using the APIVS

Conformance report options

Graphic: Thinkstock

- 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

Create Test Reports for the API Software

IEEE 829-2008 Level Test Report Outline

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

Graphic: Thinkstock

Combine All of the Out Log Files into the Detailed Test Results

2.2 Detailed Test Results

…

- <APIVSRun date="2016-10-30 04:48:07 AM UTC" configuration="./VS_config_1.txt" input="./C1310_in.xml" output="/tmp/C1310_log.xml" testSuite="ALL_TESTS" level="conformance" >

  <RunResult date="2016-10-30 04:48:17 AM UTC" status="PASS" />

- < APIVSRun date="2016-10-30 04:48:18 AM UTC" configuration="./VS_config_1.txt" input="./C1320_in.xml" output="/tmp/C1320_log.xml" testSuite="ALL_TESTS" level="conformance" >

  <RunResult date="2016-10-30 04:48:29 AM UTC" status="PASS" />

- < APIVSRun date="2016-10-30 04:48:30 AM UTC" configuration="./VS_config_1.txt" input="./C1330_in.xml" output="/tmp/C1330_log.xml" testSuite="ALL_TESTS" level="conformance" >

  <RunResult date="2016-10-30 04:48:41 AM UTC" status="PASS" />

…

ACTIVITY

U.S. Department of Transportation
Office of the Assistant Secretary for Research and Technology

Question

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!