***A Project Document of the***

***ATC Application Programming Interface (API) Working Group***

ATC APIVS TEST PLAN v01.04

Test Plan for the

Advanced Transportation Controller (ATC) Application Programming Interface Validation Suite (APIVS)

**July 14, 2016**

**In support of:** USDOT Contract # DTFH61-11-D-00052, Task Order # T-13-003

**For use by:** Siva Narla, Chief Engineer and ITS Standards Manager

Institute of Transportation Engineers

George Chen and Douglas Tarico, Co-Chairs

ATC API Working Group

Ralph W. Boaz, Project Manager and Systems Engineer

ATC API Reference Implementation Project

Members of the ATC API Working Group

Consulting Team for the ATC API RI Project

**Prepared by:** James Kinnard, Test Engineer

Adaptive Solutions, Inc.

Copyright 2015-2016 AASHTO/ITE/NEMA. All rights reserved.

**CHANGE HISTORY**

|  |  |
| --- | --- |
| **DATE** | **NOTE** |
| 10/7/15 | Initial Draft Test Plan and TDS v01.00 |
| 11/8/15 | Test Plan and TDS v01.01 |
| 12/1/15 | Test Plan and TDS v01.02 |
| 2/22/16 | Test Plan and TDS v01.03 (TRR) |
| 7/14/16 | Test Plan and TDS v01.04 (TRR2) |
|  |  |

**NOTICE**

**Joint NEMA, AASHTO and ITE Copyright and**

**Intelligent Transportation Systems (ITS) Working Group**

These materials are delivered "AS IS" without any warranties as to their use or performance.

AASHTO/ITE/NEMA AND THEIR SUPPLIERS DO NOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THESE MATERIALS. AASHTO/ITE/NEMA AND THEIR SUPPLIERS MAKE NO WARRANTIES, EXPRESSED OR IMPLIED, AS TO NON-INFRINGEMENT OF THIRD PARTY RIGHTS, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AASHTO, ITE, NEMA, OR THEIR SUPPLIERS BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY CLAIM OR FOR ANY CONSEQUENTIAL, INCIDENTAL, OR SPECIAL DAMAGES, INCLUDING ANY LOST PROFITS OR LOST SAVINGS ARISING FROM YOUR REPRODUCTION OR USE OF THESE MATERIALS, EVEN IF AN AASHTO, ITE, OR NEMA REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Some states or jurisdictions do not allow the exclusion or limitation of incidental, consequential, or special damages, or exclusion of implied warranties, so the above limitations may not apply to you.

Use of these materials does not constitute an endorsement or affiliation by or between AASHTO, ITE, or NEMA and you, your company, or your products and services.

If you are not willing to accept the foregoing restrictions, you should immediately return these materials.

ATC is a trademark of NEMA/AASHTO/ITE.

**CONTENTS**

[1 INTRODUCTION 5](#_Toc456255041)

**[1.1](#_Toc456255042)** [Purpose 5](#_Toc456255042)

**[1.2](#_Toc456255043)** [Background 5](#_Toc456255043)

**[1.3](#_Toc456255044)** [Definitions, Acronyms and Abbreviations 7](#_Toc456255044)

**[1.4](#_Toc456255045)** [References 10](#_Toc456255045)

[2 TEST ITEMS 11](#_Toc456255046)

[3 FEATURES TO BE TESTED 11](#_Toc456255047)

[4 FEATURES NOT TO BE TESTED 12](#_Toc456255048)

[5 APPROACH 12](#_Toc456255049)

[6 ITEM PASS/FAIL CRITERIA 13](#_Toc456255050)

[7 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS 13](#_Toc456255051)

[8 TEST DELIVERABLES 13](#_Toc456255052)

[9 TESTING TASKS 13](#_Toc456255053)

[10 ENVIRONMENTAL NEEDS 14](#_Toc456255054)

[11 RESPONSIBILITIES 14](#_Toc456255055)

[12 STAFFING AND TRAINING NEEDS 14](#_Toc456255056)

[13 SCHEDULE 14](#_Toc456255057)

[14 RISKS AND CONTINGENCIES 15](#_Toc456255058)

[15 APPROVALS 15](#_Toc456255059)

[16 APPENDICES 16](#_Toc456255060)

**[16.1](#_Toc456255061)** [APIVS Software Requirements to Validation Description Matrix 16](#_Toc456255061)

**[16.2](#_Toc456255062)** [APIVS Test Design Specifications 26](#_Toc456255062)

# INTRODUCTION

This Test Plan is for the test software referred to as the Advanced Transportation Controller (ATC) Application Programming Interface (API) Validation Suite (APIVS). It has been developed as part of the “Reference Implementation of ATC 5401 Application Programming Interface (API) Standard Version 2” project funded by the USDOT Contract Number DTFH61-11-D-00052, Work Order T-13003 (referred to as the APIRI project).

## Purpose

The APIVS software will be used to test and validate that the APIRI software being developed for the APIRI project properly conforms to the ATC 5401 API Standard This document defines a test plan to test the implementation of the APIVS software and provides an agreed upon level of confidence in that software. It identifies the items being tested, the features to be tested, the overall approach of the testing activities, the testing tasks to be performed, the qualifications of the personnel required for each task, and the risks associated with this plan. Since this plan is not specific to a particular project, it does not include a particular test schedule.

## Background

The Advanced Transportation Controller (ATC) standards program has been developed to meet the current and future needs for transportation field equipment. The goals of the program are to provide for transportation field equipment that is open architecture, modular, multi-process, multi-application, can grow with technology and can be used to upgrade existing transportation field cabinet systems (TFCSs). At the heart of this program are the ATC 5201 Advanced Transportation Controller (ATC) Standard and the ATC 5401 Application Programming Interface (API) Standard.

ATC 5201 specifies a controller architecture where the computational components reside on a single (5” x 4”) printed circuit board (PCB), called the “Engine Board,” with standardized connectors and pinout. It is made up of a central processing unit (CPU), a Linux operating system (O/S) and device drivers, memory, external and internal interfaces, and other associated hardware necessary to create an embedded transportation computing platform. ATC 5401 defines both user interface facilities and C programming language interfaces for ATC units that are not provided through ATC 5201 or the standard Linux O/S. The user interface facilities of ATC 5401 include a windowing system that allows operational users to interact with concurrently operating application programs (which in turn have their own user interfaces) and system-wide configuration management utilities. The C programming language interfaces of ATC 5401 provide C language function definitions that allow software developers to create application programs that share resources of the ATC unit including the front panel, field input/output (I/O)devices and real-time clock. When used with the Linux O/S and device drivers of the Engine Board, ATC 5401 provides for a software environment that allows application programs to be portable (runs on any ATC manufacturer’s equipment), compatible (will run concurrently with other application programs), and interchangeable (assuming they perform the same function) on a single ATC unit.

The ATC Application Programming Interface (API) Standard defines a software interface which resides on the Engine Board. This interface allows application programs to be written so that they may run on any ATC controller unit regardless of the manufacturer. It also defines a software environment that allows multiple application programs to be interoperable on a single controller unit by sharing the fixed resources of the controller. Software developed in compliance with the API Standard is known as the API Software.

Using the ATC Controller and API Standards together enables future advances in processing power to be applied to deployed ATC controllers while retaining the ability to operate the software applications of the existing transportation system. The API Standard provides for application software portability at the source code level. The application software source code may need to be recompiled to operate on different Engine Boards. This provides design freedom for the Engine Board manufacturers and allows Engine Boards to evolve and incorporate new technologies over time.

Figure 1 illustrates the layered architecture of the ATC software. The “Linux O/S and Device Drivers” reflects a specification of the Linux operating system defined in the ATC Board Support Package (BSP) (see ATC 5201 Standard, Appendix A and Appendix B). This includes functions for things typical in any computer system such as file I/O, serial I/O, interprocess communication and process scheduling. It also includes the specification of the device drivers necessary for the Linux O/S to operate on the ATC hardware. “API S/W” refers to software defined by the ATC 5401 Standard. Within the context of the APIRI project, the APIRI software being developed is the API software in the picture. As shown in Figure 1, user developers, operational users and application programs use the API software to interface to ATC units.



Figure . Layered organization of ATC software

In order to perform a consistent software validation of an API software implementation, the software under test must be isolated (to the extent possible) from other software or systems that may unpredictably influence its operation. The Engine Board based architecture specified in the ATC 5201 standard is ideal for this purpose by isolating the computational components and the software environment of the controller unit from other components of the controller unit.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| AASHTO | American Association of State Highway and Transportation Officials |
| API | Application Programming Interface |
| API Managers | API software that manages an ATC resource for use by concurrently running application programs. |
| API Utilities | API software not included in the API Managers that is used for configuration purposes. |
| APIRI Project | Entire project managed by this PMP including software, hardware and documentation. |
| APIRI Software | API Reference Implementation Software |
| APIVS Software | API Validation Suite Software |
| APIVSXML | APIVS Extensible Markup Language (XML) as defined by the *API Validation Suite APIVSXML Specification* (see Section 1.5 References). This version of XML includes elements for use with the APIVS software. APIVSXML is used to create test case scripts that are both human-readable and machine-readable. APIVSXML and XML are used synonymously within this document. |
| Application Program | Any program designed to perform a specific function directly for the user or, in some cases, for another application program. Examples of application programs include word processors, database programs, Web browsers and traffic control programs. Application programs use the services of a computer's O/S and other supporting programs such as an application programming interface. |
| API | Application Programmer Interface |
| ATC | Advanced Transportation Controller |
| ATC Device Drivers | Low-level software not included in a typical Linux distribution that is necessary for ATC-specific devices to operate in a Linux O/S environment. |
| ATP | Authorization to Proceed |
| Board Support Package | Software usually provided by processor board manufacturers which provides a consistent software interface for the unique architecture of the board. In the case of the ATC, the Board Support Package also includes the O/S |
| BSP | See Board Support Package |
| ConOps | Concept of Operations |
| CO | Contracting Officer |
| COR | Contract Officer’s Representative |
| COTM | Contract Officer’s Task Manager |
| CPU | Central Processing Unit. A programmable logic device that performs the instruction, logic and mathematical processing in a computer. |
| Device Driver | A software routine that links a peripheral device to the operating system. It acts like a translator between a device and the application programs that use it. |
| FHWA | Federal Highway Administration |
| FIO | Field Input and Output |
| FIOMAN | Field I/O Manager |
| FIOMSG | Field I/O Message Scheduler |
| FPMW | Front Panel Manager Window |
| FPUI | Front Panel User Interface |
| H/W | Hardware |
| I/O | Input/Output |
| IEC | International Electrotechnical Commission |
| IEEE | Institute of Electrical and Electronics Engineers |
| ISO | International Organization for Standardization |
| ITE | Institute of Transportation Engineers |
| ITS | Intelligent Transportation Systems |
| JC | Joint Committee |
| JPO | Joint Program Office |
| Linux | Low-level software that is freely available in the Linux community for use with common hardware components operating in a standard fashion. |
| Linux Kernel | The Unix-like operating system kernel that was begun by Linus Torvalds in 1991. The Linux Kernel provides general O/S functionality. This includes functions for things typical in any computer system such as file I/O, serial I/O, interprocess communication and process scheduling. It also includes Linux utility functions necessary to run programs such as shell scripts and console commands. It is generally available as open source (free to the public). The Linux Kernel referenced in this standard is defined in the ATC Controller Standard Section 2.2.5, Annex A and Annex B. |
| Loopback Driver | A virtual device driver that loops back the output ports to a device to the input ports from a device without actually going to through the physical device. |
| N/A | Not Applicable |
| Operational User | A technician or transportation engineer who uses the controller to perform its operational tasks. |
| O/S | Operating System |
| OSS | Open Source Software |
| PCB | Printed Circuit Board |
| PMP | Project Management Plan |
| POP | Period of Performance |
| PRL | Protocol Requirements List |
| RI | Reference Implementation |
| RITA | Research and Innovative Technology Administration |
| RTC | Real-Time Clock |
| RTM | Requirements Traceability Matrix |
| SDD | Software Design Descriptions |
| SDO | Standards Development Organization |
| SE | Systems Engineer |
| SEP | Systems Engineering Process |
| SEMP | Systems Engineering Management Plan |
| SOW | Statement of Work |
| SPDD | Serial Port Device Driver |
| SRS | Software Requirements Specification |
| SSH | Secure Shell. An encrypted network protocol for initiating text-based shell sessions. |
| S/W | Software |
| TBD | To Be Determined |
| TCS | Test Case Specification |
| TOD | Time of Day |
| TOPR | Task Order Proposal Request |
| TX | Transmission |
| US | United States |
| USDOT | United States Department of Transportation |
| User Developer | A software developer that designs and develops programs for controllers. |
| VD | Virtual Display: the virtual front-panel display data maintained by the VSE during a test run. |
| VSE | Validation Suite Engine: the main executable program of the APIVS software. |
| Walkthrough | A step-by-step presentation by the author of a document in order to gather information and to establish a common understanding of its content. |
| WBS | Work Breakdown Structure |
| WG | Working Group |
| XML | Extensible Markup Language. Used synonymously with APIVSXML within this document. |

## References

Institute of Electrical and Electronics Engineers, IEEE Std 829-1998, IEEE Standard for Software Test Documentation. IEEE, 1998.

[http://standards.ieee.org/index.html](http://standards.ieee.org/index.html%20)

Institute of Transportation Engineers, API Reference Implementation Project Open Source Software (OSS) Concept Paper. Institute of Transportation Engineers, 12 June 2014.

<http://www.ite.org/standards/index.asp>

Institute of Transportation Engineers, API Validation Suite APIVSXML Specification v02.00. ATC Joint Committee, 31 December 2010.

<http://www.ite.org/standards/index.asp>

Institute of Transportation Engineers, Advanced Transportation Controller (ATC) Application Programming Interface (API) Validation Suite (APIVS) Concept of Operations (ConOps) v02.04. ATC Joint Committee, 20 November 2014.

<http://www.ite.org/standards/index.asp>

Institute of Transportation Engineers, Advanced Transportation Controller (ATC) Application Programming Interface (API) Validation Suite (APIVS) Software Requirements Specification (SRS) v02.03. ATC Joint Committee, 20 November 2014.

<http://www.ite.org/standards/index.asp>

Institute of Transportation Engineers, ATC 5401 Application Programming Interface (API) Standard for the Advanced Transportation Controller (ATC) v02. ATC Joint Committee, 15 September 2013. <http://www.ite.org/standards/index.asp>

Institute of Transportation Engineers, ATC APIRI PMP v01.03 Project Management Plan (PMP) for the Advanced Transportation Controller (ATC) Application Programming Interface (API) Reference Implementation Project. ATC Joint Committee, 5 November 2014.

<http://www.ite.org/standards/index.asp>

Institute of Transportation Engineers, Intelligent Transportation System (ITS) Standard Specification for Roadside Cabinets v01.02.17b. ATC Joint Committee, 16 November 2006.

<http://www.ite.org/standards/index.asp>

Institute of Transportation Engineers, User Comment Draft ATC 5201 Advanced Transportation Controller (ATC) Standard Version 06.10. ATC Joint Committee, 30 July 2012.

<http://www.ite.org/standards/index.asp>

International Organization for Standardization, ISO/IEC 9899:2011 Programming Language C. ISO, 8 December 2011.

National Electrical Manufacturers Association, NEMA Standards Publication TS 2-2003 v02.06 Traffic Controller Assemblies with NTCIP Requirements. NEMA, 2003.

# TEST ITEMS

The APIVS software is made up of an executable application program - the Validation Suite Engine (VSE), a VSE configuration file and individual APIVSXML test script files corresponding to Test Case Specifications (TCSs). All of these items are tested by this test plan.

Requirements for the operation of this software can be found in the Advanced Transportation Controller (ATC) Application Programming Interface (API) Validation Suite (APIVS) Software Requirements Specification (SRS).

# FEATURES TO BE TESTED

This specific features and combination of features to be tested under this Test Plan are defined by the Test Design Specifications (TDS), Test Case Specifications (TCS) and Test Procedure Specifications (TPS) listed in Table 1.

**Table 1. APIVS Test Specifics**

|  |  |  |
| --- | --- | --- |
| **Test ID** | **Document Name** | **Brief Description** |
| APIVS.TDS.1001 | APIVS Test Design Specification 1 | Test All APIVS Required Features |
| APIVS.TCS.1110 | APIVS Test Case Specification 1 | FPUI Library C Function Present |
| APIVS.TCS.1120 | APIVS Test Case Specification 2 | FPUI Library C Function Conforming Arguments |
| APIVS.TCS.1130 | APIVS Test Case Specification 3 | FPUI Library C Function Error Checking  and Argument Boundary Checking |
| APIVS.TCS.1150 | APIVS Test Case Specification 4 | FPUI Library Composite Testing |
| APIVS.TCS.1160 | APIVS Test Case Specification 5 | Front Panel Manager Window Testing |
| APIVS.TCS.1310 | APIVS Test Case Specification 6 | FIO Library C Function Present |
| APIVS.TCS.1320 | APIVS Test Case Specification 7 | FIO Library C Function Conforming Arguments |
| APIVS.TCS.1330 | APIVS Test Case Specification 8 | FIO Library C Function Error Checking  and Argument Boundary Checking |
| APIVS.TCS.1350 | APIVS Test Case Specification 9 | FIO Library Composite Testing |
| APIVS.TCS.1410 | APIVS Test Case Specification 10 | TOD Library C Function Present |
| APIVS.TCS.1420 | APIVS Test Case Specification 11 | TOD Library C Function Conforming Arguments |
| APIVS.TCS.1430 | APIVS Test Case Specification 12 | TOD Library C Function Error Checking  and Argument Boundary Checking |
| APIVS.TCS.1450 | APIVS Test Case Specification 13 | TOD Library Composite Testing |
| APIVS.TCS.1510 | APIVS Test Case Specification 14 | Multiple and Concurrent Applications |
| APIVS.TCS.6010 | APIVS Test Case Specification 15 | APIVS Software Licensing Details |
| APIVS.TCS.6020 | APIVS Test Case Specification 16 | C Programming and Source Code Quality |
| APIVS.TCS.6030 | APIVS Test Case Specification 17 | XML Scripting, Execution and Logging |
| APIVS.TPS.1001 | APIVS Test Procedure Specification 1 | Auto-Execute Selected APIVS Script(s) |
| APIVS.TPS.6010 | APIVS Test Procedure Specification 2 | APIVS Software Licensing Details |
| APIVS.TPS.6020 | APIVS Test Procedure Specification 3 | C Programming and Source Code Quality |
| APIVS.TPS.6030 | APIVS Test Procedure Specification 4 | XML Scripting, Execution and Logging |

Appendix 16.1 contains a traceability matrix mapping the specific validation methods and related test items to the individual software requirements as found in the APIVS SRS.

# FEATURES NOT TO BE TESTED

As standards development and maintenance are continuous processes, there may be some features added to the API Standard after the development of this test plan. Any features added to versions of the API Standard that are not listed in the reference section of this document may not be tested as part of this test plan.

# APPROACH

The APIVS software is made up of an executable application program, a configuration file and test script files based on the individual Test Case Specifications (TCSs). The main executable program is the Validation Suite Engine (VSE). The configuration file is used to set up the test environment for the tests defined by TCSs. The test script files are written in XML format which are parsed and interpreted by the VSE, invoking the API software tests, and validating results.

In order to automate this testing, a capability needs to exist which brings the outputs of the functioning API software back to the inputs so that proper operation can be verified. This “loopback” capability could be done with some limitations using physical loopback cables connected to the various external serial inputs and outputs of the ATC unit. The design of the APIVS software incorporates a virtual loopback capability where the loopback occurs internal to the ATC unit. This includes a set of asynchronous and synchronous virtual loopback drivers for the Linux kernel which replace like drivers on the ATC Engine Board for the purposes of testing. This allows the testing to be more comprehensive and facilitate the testing for the test personnel. The layered software environment for the APIVS software is similar to the layered organization of the ATC software (see Figure 2).



Figure . Layered organization of APIVS software

Specific refinements to this approach can be found in the Test Design Specifications in Appendix 16.2.

# ITEM PASS/FAIL CRITERIA

The test items as specified in Section 2 will be tested according to the TDSs, TCSs and TPSs identified in Table 1. If any of the tests for a test item fail, the test item is considered to have failed. If all of the tests for a test item pass, the test item is considered to have passed.

# SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

Suspension criteria and resumption requirements are subject to the organization exercising this test plan and its associated TDSs, TCSs and TPSs.

# TEST DELIVERABLES

The target audience for this test plan is ATC manufacturers who desire to implement the APIRI on their own ATC Controllers. Since Engine Boards may have been implemented using a variety of processors, test software identified in this test plan will need to be built (compiled) in a manner appropriate for the implementation of the Engine Board on the specific ATC Controller used for testing. Proper operation of the API Validation Suite (VSE executable) will need to be confirmed by the use of this test plan after the build process is complete and prior to the execution of the test plan for the APIRI.

Specific deliverables for individual tests (if any) are identified in the TDS, TCS and TPSs listed in Table 1.

While manufacturers are not required to produce any particular test deliverables as a result of utilizing this test plan, it is requested and encouraged that any relevant feedback from the execution of this plan be provided to the ATC Program Manager and the API WG. Any suggested changes or enhancements to this test plan and the API Standard are welcomed. A Test Summary Report as defined by IEEE Std 829-1998 should be completed and sent to the chairpersons of the API Working Group via the ATC Program Manager of ITE. Unless expressly required by a TDS, TCS or TPS, other test deliverables including Test Item Transmittal Reports, Test Logs and Test Incident Reports, and/or test equipment are subject to specification by the organization exercising this test plan (see outlines for these documents in IEEE Std 829-1998).

# TESTING TASKS

The tasks necessary to prepare for and to perform testing are as follows:

1. Port APIVS and APIRI software packages to ATC Controller platform;
2. Gather and prepare test equipment and test software;
3. Perform testing per TDS, TCS and TPS; and
4. Deliver Test Summary Report to ATC Program Manager (optional)

Task #1 requires the skills of a software engineer. For more information regarding obtaining the source code and building the APIRI and APIVS software components, please refer to the APIRI User Manual and APIVS User Manual references in Section 1.3.

Tasks #2 and #3 may be performed by a technician but some tests may require understanding of traffic operation. Task #4 will likely require some formatting and editing of any automated reports.

# ENVIRONMENTAL NEEDS

The test environment shown in Figure 3 is to be used to execute this test plan. It consists of an ATC Controller, a USB flash drive and a personal computer (PC). This environment allows the Validation Suite software to run on the Engine Board and to perform validation testing. Both the PC and the controller are required to have an available USB port. The ATC is required to have a minimum 8x40 character LCD display and associated keyboards.

The PC interface is necessary to load test software, configuration files and test scripts onto the USB flash drive prior to testing and to extract and examine conformance reports from the drive after testing. It is possible that the PC may also serve directly in the operation of some tests. Details of the operation of the test environment and tests can be found in the individual TDS, TCS and TPSs listed in Table 1.

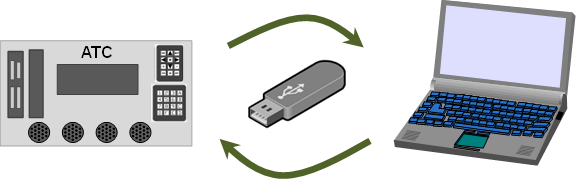


Figure . API Test Environment

# RESPONSIBILITIES

It is the responsibility of the API WG to maintain this test plan. It is the responsibility of the organization exercising this test plan to provide the test environment and test personnel to execute the software identified in the TDSs, TCSs and TPSs for the controller under test. It is requested that any organization exercising this test plan to notify the API WG via the ATC Program Manger of ITE of any corrections or suggested modifications to the API Standard, this test plan, TDSs, TCSs, TPSs and test software found during testing.

# STAFFING AND TRAINING NEEDS

Staffing and training needs are subject to the organization exercising this test plan.

# SCHEDULE

There is no fixed schedule for this test plan. It is designed to be applied whenever an organization determines it is necessary.

# RISKS AND CONTINGENCIES

Exhaustive testing of the APIVS Software is impractical. This means that there may be combinations of features that are not tested by this test plan. If anomalies are found in an implementation of API software that this test plan did not cover, the API WG should be contacted through the ATC Program Manager so that an appropriate test can be added to the next version of this test plan.

# APPROVALS

This test plan has been developed by the ATC API WG to facilitate the deployment of the API Standard. The official approvals required are still to be determined.

# APPENDICES

## APIVS Software Requirements to Validation Description Matrix

The following table shows the relationship between the individual requirements of the APIVS Software Requirements Specification (SRS) and the specific Test Plans and Procedures used to validate the APIVS requirement(s).

| **Category** | **APIVS SRS Req ID** | **APIVS SRS Requirement Name** | **APIVS SDD Design Narrative** | **Test Cases** | **Test Procedures** |
| --- | --- | --- | --- | --- | --- |
| Open Source Software (OSS) Environment | 3.1 | No Cost | All APIVS software shall be free of licensing fees and available for free download. | APIVS.TCS.6010 | APIVS.TPS.6010 |
| 3.2 | Open Source | All APIVS software source code shall be available under open source licensing terms. | APIVS.TCS.6010 | APIVS.TPS.6010 |
| 3.3 | ITE Approved License | All APIVS software source code shall be available under open source licensing terms approved by ITE. | APIVS.TCS.6010 | APIVS.TPS.6010 |
| Unrestricted Use | 3.4 | Unrestrictive Use by Users | All APIVS software shall be available under open source licensing terms which allow unrestricted use. | APIVS.TCS.6010 | APIVS.TPS.6010 |
| Redistribution | 3.5 | Redistribution of Modified Source Code | All APIVS software source code shall be available under open source licensing terms (Gnu Public License) which require modifications and derived works to be distributed under the same terms. | APIVS.TCS.6010 | APIVS.TPS.6010 |
| 3.6 | No Cost Redistributed APIVS Source Code | All APIVS software source code shall be available under the terms of the Gnu Public License (GPL). | APIVS.TCS.6010 | APIVS.TPS.6010 |
| Testing Environment | 3.7 | Testing Environment | The APIVS software package is designed to run on an *ATC 5201 Standard* conforming controller. See Section 2.1. | APIVS.TCS.6030 | APIVS.TPS.6030 |
| C Programming Language | 3.8 | C Programming Language | All APIVS software programs are written in the C Programming Language and are designed to be compatible with the uClibc C library. See Section 2.3. | APIVS.TCS.6020 | APIVS.TPS.6020 |
| Source Code Quality | 3.9 | C Source Code Quality | All APIVS source code is written to follow the styling practices of the Linux kernel and GNU standard library source code. | APIVS.TCS.6020 | APIVS.TPS.6020 |
| Extensible | 3.10 | XML Scripting Language | The APIVS software allows validation tests to be configured using XML files. Details of the APIVSXML language are found in *API Validation Suite APIVSXML Specification* (see Section 1.5, References). | APIVS.TCS.6030 | APIVS.TPS.6030 |
| 3.11 | Interpreted Test Scripts | XML TCS files may be established and interpreted by the APIVS software at run-time without requiring recompilation, as described in Section 2.2. | APIVS.TCS.6030 | APIVS.TPS.6030 |
| Selectable Tests | 3.12 | Run All Tests | The VSE executable may be configured to run through all tests supplied in the distribution by use of the command line switch enabling this option, as described in Section 2.2. | APIVS.TCS.6030 | APIVS.TPS.6030 |
| 3.13 | Run Selected Tests | The VSE executable may be configured to run one or more individual tests from the full set of tests supplied in the distribution, as described in Section 2.2. | APIVS.TCS.6030 | APIVS.TPS.6030 |
| Continuous Loop | 3.14 | Continuous Loop | The VSE executable may be configured to run each test load once or continuously, as described in Section 2.2. | APIVS.TCS.6030 | APIVS.TPS.6030 |
| Pass / Fail Indications | 3.15 | Conformance Indication | The VSE executable returns a value of 0 to indicate a pass or conformance condition for the test run as stated in Section 2.2. | APIVS.TCS.6030 | APIVS.TPS.6030 |
| 3.16 | Nonconformance Indications | The VSE executable returns a value of -1 to indicate a failure or non-conformance condition for the test run as stated in Section 2.2. | APIVS.TCS.6030 | APIVS.TPS.6030 |
| Logging Option | 3.17 | Detailed Log | When configured for verbose output, the APIVS software returns a detailed conformance report. An example of the detailed form of log file when opened with an XML viewer is shown in Appendix B. | APIVS.TCS.6030 | APIVS.TPS.6030 |
| 3.18 | Summary Result | The format of the conformance report, as described in Section 3.4.4, allows interactive viewing of test results at increasing levels of detail. An example of the summary form of the log file when opened with an XML viewer is shown in Appendix A. | APIVS.TCS.6030 | APIVS.TPS.6030 |
| 3.19 | Output Options | The format of the conformance report, as described in Section 3.4.4, allows interactive viewing of test results at increasing levels of detail. Examples of different output options are shown in Appendix A and B | APIVS.TCS.6030 | APIVS.TPS.6030 |
| 3.20 | XML Output Files | The XML format of the conformance report, as described in Section 3.4.4, allows interactive viewing of test results at increasing levels of detail. An example of this format is shown in Appendix B. | APIVS.TCS.6030 | APIVS.TPS.6030 |
| API FPUI Library C Functions Completeness Testing | 3.21.1 | FPUI Library C Function Present | A TCS file “FPUI\_every\_func.xml” is provided in the APIVS software distribution which includes calls to all FPUI library functions and indicates any absent functions in the conformance report. | APIVS.TCS.1110 | APIVS.TPS.1001 |
| 3.21.2 | FPUI Library C Function Conforming Arguments | A TCS file “FPUI\_every\_func.xml” is provided in the APIVS software distribution which includes calls to all FPUI library functions and indicates any function argument or parameter type non-conformance in the conformance report. Function parameter checking is performed by the InterpretXML entity as described in Section 3.4.2. | APIVS.TCS.1120 | APIVS.TPS.1001 |
| API FPUI Library C Functions Correctness Testing | 3.21.3 | FPUI Library C Function Error Checking | The VSE program includes the validation test interpreter entity InterpretXML, described in Section 3.4.2, which validates the return value of each FPUI library function as it is called. The extent of error code checking depends on the TCS making invalid function calls and comparing the resultant returned codes against expected values. | APIVS.TCS.1130 | APIVS.TPS.1001 |
| 3.21.4 | FPUI Library C Function Argument Boundary Checking | The VSE program includes the validation test interpreter InterpretXML, described in Section 3.4.2, which validates the argument type and bounds of each FPUI library function as it is called. The extent of function parameter range checking depends on the TCS making invalid function calls and comparing the resultant returned codes against expected values. | APIVS.TCS.1130 | APIVS.TPS.1001 |
| 3.21.5 | FPUI Library Composite Testing | A TCS file “FPUI\_Scenario1.xml” is provided in the APIVS software distribution and described in Appendix C, which calls multiple FPUI library functions in a typical use case scenario, checking each function call for correctness and validating the resultant controller behavior against expected results. | APIVS.TCS.1150 | APIVS.TPS.1001 |
| API Front Panel Manager Software Testing | 3.21.6 | Front Panel Manager Window Testing | A TCS file “FPUI\_Scenario1.xml” is provided in the APIVS software distribution and described in Appendix C, which calls multiple FPUI library functions in a scenario which validates typical use of the Front Panel Manager Window specified in Section 3.1.1.1 of *ATC 5401 Standard*, checking each function call for correctness and validating the resultant controller behavior against expected results. | APIVS.TCS.1160 | APIVS.TPS.1001 |
| API Utility Software Testing | 3.21.7 | ATC Configuration Window Testing | A TCS file “FPUI\_Scenario2.xml” is provided in the APIVS software distribution and described in Appendix C, which calls multiple FPUI library functions in a scenario which validates typical use of the ATC Configuration Window specified in Section 3.2.1 of *ATC 5401 Standard*, checking each function call for correctness and validating the resultant controller behavior against expected results. | APIRI Test Case  APIRI.TCS.6020  (see APIRI TCS document) | APIRI Test Procedure  APIRI.TPS.6020  (see APIRI TPS document) |
| 3.21.8 | Configuration Utility Testing | A TCS file “FPUI\_Scenario2.xml” is provided in the APIVS software distribution and described in Appendix C, which calls multiple FPUI library functions in a scenario which validates typical use of the ATC Configuration Window system, as specified in Section 3.2.2 of *ATC 5401 Standard*, for editing a configuration item, checking each function call for correctness and validating the resultant controller behavior against expected results. | APIRI Test Case  APIRI.TCS.6030  (see APIRI TCS document) | APIRI Test Procedure  APIRI.TPS.6030  (see APIRI TPS document) |
| API FIO Library C Functions Completeness Testing | 3.22.1 | FIO Library C Function Present | A TCS file “FIO\_Every\_Func.xml” is provided in the APIVS software distribution which includes calls to all FIO library functions and indicates any absent functions in the conformance report. | APIVS.TCS.1310 | APIVS.TPS.1001 |
| 3.22.2 | FIO Library C Function Conforming Arguments | A TCS file “FIO\_every\_func.xml” is provided in the APIVS software distribution which includes calls to all FIO library functions and indicates any function argument or parameter type non-conformance in the conformance report. Function parameter checking is performed by the InterpretXML entity as described in Section 3.4.2. | APIVS.TCS.1320 | APIVS.TPS.1001 |
| API FIO Library C Functions Correctness Testing | 3.22.3 | FIO Library C Function Error Checking | The VSE program includes the validation test interpreter entity InterpretXML, described in Section 3.4.2, which validates the return value of each FIO library function as it is called. The extent of error code checking depends on the TCS making invalid function calls and comparing the resultant returned codes against expected values. | APIVS.TCS.1330 | APIVS.TPS.1001 |
| 3.22.4 | FIO Library C Function Argument Boundary Checking | The VSE program includes the validation test interpreter entity InterpretXML, described in Section 3.4.2, which validates the argument type and bounds of each FIO library function as it is called. The extent of function parameter range checking depends on the TCS making invalid function calls and comparing the resultant returned codes against expected values. | APIVS.TCS.1330 | APIVS.TPS.1001 |
| 3.22.5 | FIO Library Composite Testing | A TCS file “FIO\_Scenario1.xml” is provided in the APIVS software distribution and described in Appendix C, which calls multiple FIO library functions in a typical use case scenario, checking each function call for correctness and validating the resultant controller behavior against expected results. | APIVS.TCS.1350 | APIVS.TPS.1001 |
| API FIO Manager Software Testing | 3.22.6 | Field I/O Manager Testing | Several TCS files are provided in the APIVS software distribution and described in Appendix C, which establish typical Field I/O Manager scenarios for cabinet I/O device control and communication, and which validate the resultant controller behavior against expected results. | APIRI Test Cases  APIRI.TCS.3001-3999  (see APIRI TCS document) | APIRI.TPS.1001 |
| API TOD Library C Functions Completeness Testing | 3.23.1 | TOD Library C Function Present | A TCS file “TOD\_Every\_Func.xml” is provided in the APIVS software distribution which includes calls to all TOD library functions and indicates any absent functions in the conformance report. | APIVS.TCS.1410 | APIVS.TPS.1001 |
| 3.23.2 | TOD Library C Function Conforming Arguments | A TCS file “TOD\_every\_func.xml” is provided in the APIVS software distribution which includes calls to all TOD library functions and indicates any function argument or parameter type non-conformance in the conformance report. | APIVS.TCS.1420 | APIVS.TPS.1001 |
| API TOD Library C Functions Correctness Testing | 3.23.3 | TOD Library C Function Error Checking | The VSE program includes the validation test interpreter entity InterpretXML, described in Section 3.4.2, which validates the return value of each TOD library function as it is called. The extent of error code checking depends on the TCS making invalid function calls and comparing the resultant returned codes against expected values. | APIVS.TCS.1430 | APIVS.TPS.1001 |
| 3.23.4 | TOD Library C Function Argument Boundary Checking | The VSE program includes the validation test interpreter entity InterpretXML, described in Section 3.4.2, which validates the argument type and bounds of each TOD library function as it is called. The extent of function parameter range checking depends on the TCS making invalid function calls and comparing the resultant returned codes against expected values. | APIVS.TCS.1430 | APIVS.TPS.1001 |
| 3.23.5 | TOD Library Composite Testing | A TCS file “TOD\_Scenario1.xml” is provided in the APIVS software distribution and described in Appendix C, which calls multiple TOD library functions in a typical use case scenario, checking each function call for correctness and validating the resultant controller behavior against expected results. | APIVS.TCS.1450 | APIVS.TPS.1001 |
| Multiple and Concurrent Applications | 3.24 | Multiple and Concurrent Applications | The APIVS software supports the ability for multiple instances of the VSE program to run concurrently, as described in Section 2.2, in order to test multiple client support of the API libraries. | APIVS.TCS.1510 | APIVS.TCS.1510 |

## APIVS Test Design Specifications

### Test Design Specification Description 1 - Test All APIVS Required Features

#### Test Design Specification Identifier

The identifier for this Test Design Specification is APIVS.TDS.1001.

#### Features To Be Tested

This Test Design Specification will test all features of the API Validation Suite which are subject to testing for validation (i.e., which have a numbered Test Case of the form APIVS.TCS.nnnn).

Specific features to be tested (referenced to the API VS SRS) and assigned Test Cases to be used can be found in Appendix 16.1.

#### 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 APIVS.TPS.1001 unless otherwise noted in an individual test case specification.

#### Test Identification

All test items and their identifiers to be used by this Test Design Specification can be found in Section 3, Table 1.

#### Feature Pass/Fail Criteria

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