A Project Document of the
Advanced Transportation Controller Joint Committee

APIVS CONOPS v02.04

Advanced Transportation Controller (ATC)
Application Programming Interface (API)
Validation Suite (APIVS) Concept of Operations (ConOps)

November 20, 2014

ConOps in support of: USDOT Work Order 14-0801, Tasks 7-8
For approval by: Members of the API Working Group
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 Validation Suite Project
Members of the ATC API Working Group
Consulting Team for the ATC API Reference Implementation Project
Prepared by: Ralph W. Boaz, Project Manager and Systems Engineer

© Copyright 2014 AASHTO/ITE/NEMA. All rights reserved.
## CHANGE HISTORY

<table>
<thead>
<tr>
<th>DATE</th>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/25/09</td>
<td>Initial Draft WGD Version 01.00.</td>
</tr>
<tr>
<td>04/15/10</td>
<td>Version 01.01 corrections per the API WG.</td>
</tr>
<tr>
<td>03/21/14</td>
<td>Version 02.00 Update for the API Reference Implementation Project.</td>
</tr>
<tr>
<td>04/18/14</td>
<td>Version 02.01 Update following a walkthrough of the ConOps by the API WG.</td>
</tr>
<tr>
<td>05/30/14</td>
<td>Version 02.02 Update following a review of the ConOps by the API WG.</td>
</tr>
<tr>
<td>07/16/14</td>
<td>Version 02.03 Updates based on SRS development.</td>
</tr>
<tr>
<td>11/20/14</td>
<td>Version 02.04 Updates for USDOT approved re-scope of the project which provides for a fixtureless test environment.</td>
</tr>
</tbody>
</table>
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.
1 SCOPE

This section provides the Identification, Document Overview and Software Overview.

1.1 Identification

This Concept of Operations (ConOps) applies to the Advanced Transportation Controller (ATC) Application Programming Interface (API) Validation Suite (APIVS) software.

1.2 Document Overview

The purpose of this ConOps is to communicate the user needs and expectations of the APIVS software and to serve as the foundation for developing a Software Requirements Specification (SRS, see IEEE Std 830-1998). The document organization is based on IEEE Std 1362-1998, IEEE Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document. This ConOps has been developed for:

a) The USDOT Intelligent Transportation Systems (ITS) Joint Program Office (JPO) who is sponsoring the work and requires the use of a formal software development process;

b) The consulting team contracted to develop the software described;

c) The consultants, manufacturers, and public transportation professionals who participate in the API Working Group (WG) who provide domain expertise, quality assurance, testing assistance and ultimately the maintenance of the software; and

d) The transportation industry as a whole that will depend upon this software to test implementations of the ATC 5401 Standard resident on ATC units.

1.3 Software Overview

This ConOps describes APIVS software to be 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). The goal of this project is to develop reliable software that is representative of the ATC 5401 Standard called the API Reference Implementation (APIRI) software and to provide it for free (at no cost to the user) in a manner that is usable and maintainable by the manufacturers, software vendors, consultants and agencies of the transportation industry. This ConOps is for the APIVS software that tests the APIRI software. The purpose of the APIVS software is to validate that the APIRI software developed meets the requirements and specifications in the ATC 5401 Standard. See Section 3 for background on the relationship of the APIRI software to the ATC Engine Board. See Section 5 for a description of the APIVS software.

2 REFERENCED DOCUMENTS

The documents referenced in this ConOps are listed below.

http://standards.ieee.org/index.html

http://standards.ieee.org/index.html

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 v01.01 Project Management Plan (PMP) for the Advanced Transportation Controller (ATC) Application Programming Interface (API) Reference Implementation Project. ATC Joint Committee, 14 January 2014.
http://www.ite.org/standards/index.asp

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


3 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 Standard and the ATC 5401 Application Programming Interface 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.
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 shown 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 1. Layered organization of ATC software.

4 JUSTIFICATION

As part of the APIRI project, the APIVS will be used to validate that the APIRI software developed meets the requirements and specifications in the ATC 5401 Standard. This is necessary to confirm that the work on the APIRI software is operationally ready. The APIVS software will be provided for free (at no cost to the user) in a similar manner to that of the APIRI software in order to provide agencies, integrators, software developers and testers with the ability to test ATC equipment for conformance to the ATC 5401 standard.

Some of the benefits provided by the APIVS include:

- The ability to have a common and impartial test suite for testing implementations of the API 5401 Standard;
- The ability to have manufacturers demonstrate conformance to the ATC 5401 Standard increasing confidence in and faster deployment of ATC units running API software;
- The ability for agencies to reference a software validation tool in their procurement specifications;
- The increased reliability of API implementations on ATC equipment;
- The increased portability and interoperability of application programs on ATC units; and
• The ability to enhance the APIVS in order to maintain consistency with enhancements to the ATC 5401 Standard and APIRI software.

Not developing the APIVS will result in:
• The inability of manufacturers to demonstrate conformance to the ATC 5401 Standard in an impartial manner;
• Agencies not requiring API software in their procurement specifications and potentially purchasing controller units without the benefits of API software;
• A duplication of effort as each ATC equipment manufacturer must develop their own API software validation capabilities; and
• Some of the goals of the ATC program not being achieved.

5 CONCEPTS FOR THE PROPOSED SOFTWARE

This section provides the concepts and user needs for the APIVS.

5.1 Background, Objectives and Scope

The APIVS is test software that validates that API software meets the requirements and specifications in the ATC 5401 Standard. Successful testing provides an agreed upon level of confidence (identified by the user needs in Section 5.3) that the API software under test conforms to the ATC 5401 Standard.

5.2 Operational Policies and Constraints

The following operational policies and constraints have been identified:

a) Portions of the APIVS software will be resident on an ATC Engine Board with operational API software. These portions will need to be compatible with the Board Support Package as defined by the ATC 5201 Standard.

b) Since ATC Engine Boards may have been implemented using a variety of processors, the APIVS software that is to be resident on the Engine Board will need to be compiled, linked and loaded in a manner compatible with the processor on the Engine Board.

Specific needs on the construction of the APIVS are listed as user needs in Section 5.3 and its subsections.

5.3 Description of the Proposed 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. The APIVS test environment proposed is shown in Figure 2. It consists of an ATC unit and a personal computer (PC). The PC interface is necessary to load test software, initiate tests, and extract results. It is possible that the PC can also serve in the operation of some tests. Details of the operation of the test environment and tests are to be documented according to a test plan.
Figure 2. API test environment uses a personal computer connected to the console port of the controller.

The layered software environment for the APIVS software is similar to the layered organization of the ATC software (see Figure 3). The APIVS takes the place of the Application Software in Figure 1. The user is now a Tester which may be a User Developer, Test Engineer or Test Technician. The APIVS resident on the Engine Board exercises the API software and records results. Special device drivers are necessary for the APIVS to perform the testing without requiring the physical connectors on the controller unit.

Figure 3. The layered software environment for the APIVS.

The following subsections identify the user needs for the APIVS. Each user need is listed separately with a paragraph number. The rationale behind the need is included in italics. The APIVS requirements and design will be based on these needs.

5.3.1 Open Source Software (OSS) Environment

The user needs the APIVS source code, software and hardware design files, test documents, process documents and manuals to be available to anyone at no cost in an OSS environment. Users need access to these files in order to perform APIVS testing within their own organizations whether they perform the
work themselves or hire an outside consultant. APIVS software that is to operate on an Engine Board needs to be compiled, linked and loaded in a manner compatible with the processor on the Engine Board. The OSS should provide tools that are used for version control, bug tracking, mailing lists, and real-time chat.

5.3.2 Unrestricted Use

The user needs the APIVS to have a software license model that allows for unrestricted use of the software by the user. After obtaining the APIVS, users should not be required to ask permission to use it. Users should be able to modify it for their own purposes without restrictions.

5.3.3 Redistribution

The user needs the APIVS to have a software license model that requires entities that make enhancements to the APIVS software and redistribute it, to provide the source code for the enhanced version publicly at no cost. This provides for improvements to be shared across the industry and potentially added to the original source of the APIVS software.

5.3.4 Testing Environment

The user needs the APIVS to be consistent with the testing architecture described in Section 5.3. This provides for isolation of the API software from the front panel and field I/O elements of an ATC unit. This environment also requires no special cabling, no special test fixture and minimal effort on the part of the Tester.

5.3.5 C Programming Language

The user needs the APIVS software to be written using the C programming language as described by ISO/IEC 9899:2011 commonly referred to as the C99 Standard. The ATC 5401 Standard specifies C for its function definitions. C is also the most commonly used programming language for embedded systems (see Section 2 Referenced Documents). C should also be used on the PC platform except where a higher level scripting language may be more advantageous.

5.3.6 Source Code Quality

The user needs the APIVS to be written using the GNU Coding Standards (see Section 2 Referenced Documents). The software needs to be written in a clear and consistent fashion so that it can be understood by others. If a scripting language is employed in the APIVS, an appropriate style for that language should be utilized consistent with the concepts of the GNU Coding Standards.

5.3.7 Extensible

The user needs the APIVS to be extensible. Extensible means that the APIVS is designed to facilitate expansion of the testing capabilities while keeping within the general architecture of the APIVS that is developed. It is anticipated that the user developers will add tests to the APIVS as the ATC 5401 Standard and the APIRI software evolve and improve.

5.3.8 Selectable Tests

The user needs the APIVS to be able to run all of its tests or user selectable subset of tests. Running all of the tests on an API software library may be time consuming. Users need an option to only run a subset of the APIVS’s available tests.
5.3.9 Continuous Loop

The user needs the APIVS to be able to run all or a user selected subset of tests in a continuous fashion. Users may find certain areas of an API software implementation problematic and wish to repeatedly run a set of tests. Users may also want to use this capability to run tests in an environmental chamber over a period time.

5.3.10 Pass / Fail Indications

The user needs the APIVS to provide a pass/fail indication of conformance of the API software to the ATC 5401 Standard. Users must have a simple repeatable method to test that they have conforming operational API software for the tests they select. The return values will be 0, indicating conformance, or -1, indicating nonconformance of the API Software to the ATC 5401 Standard. This would apply if a user is running all of the tests available or a subset of the tests. It is anticipated that most agencies will only require a simple pass / fail result of running all of the tests available.

5.3.11 Logging Option

The user needs the APIVS to provide an option to log the tests performed and the results of each test and step. Users may need additional information to diagnose anomalies in the API software. The log should include:

- The library, function and arguments on an API function call and the return values;
- If a function fails, guidance to the user on the cause of the failure;
- The test case being executed;
- Line # in the APIVS source code;
- Step in the test case; and
- Time stamps for each step in the test case.

5.3.12 API Front Panel User Interface (FPUI) Library Testing

5.3.12.1 API FPUI Library C Functions Completeness Testing

The user needs the APIVS to test the completeness of the FPUI C programming functions specified in Section 4.1 of the ATC 5401 Standard. This will validate that each API C function is present and that its arguments conform to the ATC 5401 Standard.

5.3.12.2 API FPUI Library C Functions Correctness Testing

The user needs the APIVS to test the correctness of the FPUI C programming functions specified in Section 4.1 of the ATC 5401 Standard. This testing will include boundary and error condition testing for the FPUI C functions. Each FPUI function will be included in at least one composite test with other functions to test under typical operating conditions for the function.

5.3.12.3 API Front Panel Manager Software Testing

The user needs the APIVS to test the correctness of the Front Panel Manager software specified in Section 3.1.1 of the ATC 5401 Standard. It is anticipated that some form of emulation of the Front Panel will be required. Each requirement will be tested in at least one composite test under typical operating conditions for the requirement.

5.3.12.4 API Utility Software Testing
The user needs the APIVS to test the correctness of the API Utility software specified in Section 3.2 of the ATC 5401 Standard. It is anticipated that some form of emulation of the ATC Configuration Window will be required. Each requirement will be tested in at least one composite test under typical operating conditions for the requirement.

5.3.13 API Field I/O (FIO) Library Testing

5.3.13.1 API FIO Library C Functions Completeness Testing

The user needs the APIVS to test the completeness of the FIO C programming functions specified in Section 4.2 of the ATC 5401 Standard. This will validate that each API C function is present and that its arguments conform to the ATC 5401 Standard.

5.3.13.2 API FIO Library C Functions Correctness Testing

The user needs the APIVS to test the correctness of the FIO C programming functions specified in Section 4.2 of the ATC 5401 Standard. This testing will include boundary and error condition testing for the FIO C functions. Each FIO function will be included in at least one composite test with other functions to test under typical operating conditions for the function.

5.3.13.3 API FIO Manager Software Testing

The user needs the APIVS to test the correctness of the Field I/O Manager software specified in Section 3.1.2 of the ATC 5401 Standard. It is anticipated that some form of emulation of the Field I/O devices will be required. Each requirement will be tested in at least one composite test under typical operating conditions for the requirement.

5.3.14 API Time of Day (TOD) Library Testing

5.3.14.1 API TOD Library C Functions Completeness Testing

The user needs the APIVS to test the completeness of the TOD C programming functions specified in Section 4.3 of the ATC 5401 Standard. This will validate that each API C function is present and that its arguments conform to the ATC 5401 Standard.

5.3.14.2 API TOD Library C Functions Correctness Testing

The user needs the APIVS to test the correctness of the TOD C programming functions specified in Section 4.3 of the ATC 5401 Standard. This testing will include boundary and error condition testing for the TOD C functions. Each TOD function will be included in at least one composite test with other functions to test under typical operating conditions for the function.

5.3.15 Multiple and Concurrent Applications

The user needs the APIVS to test that multiple application programs, running concurrently, can operate on an ATC unit. The API software should exercise the Front Panel Manager Window, the Front Panel Manager functions, the Field I/O Manager functions and the Time of Day functions simultaneously.

5.4 Modes of Operation

There are no special modes of operation for this ConOps.
5.5 User Classes and Other Involved Personnel

The "Tester" is the only user class for the APIVS. A Tester must be an individual with enough computing experience to follow test procedures, connect cables, remove and install an ATC Engine Board, run software on a PC, and keep records of results. Typically the Tester is a software developer, a test engineer or test technician.

5.6 Support Environment

Specific support environments for testing will vary based on the organization performing the tests. The APIVS itself is to be supported by the API WG. This is in terms of maintaining the integrity of the source code, making corrections and enhancements, and assisting users with use of the APIVS. Users should be able to get assistance through the API WG Chairs, the ATC Program Manager, or the API Website (see Section 2 Referenced Documents).

6 OPERATIONAL SCENARIOS

There are no special operational scenarios for this ConOps.

7 SUMMARY OF IMPACTS

This section provides the operational impacts of the software on the users, developers, and support and maintenance organizations. This information is provided in order to allow preparations for the APIVS by agencies, user and working groups, sponsoring organizations, and support and maintenance organizations. The impacts listed are organizational in nature versus the operation of the APIVS software itself.

7.1 Operational Impacts

One of the challenging issues in the use of ITS standards is proving conformance. Providers of equipment may claim adherence to a standard but the buyer may not have a method to validate that it is conformant. Some agencies refer to agencies outside of their jurisdiction which have testing labs to determine conformance. The APIVS provides additional methods to validate conformance to the ATC 5401 Standard. Buyers may ask equipment providers to demonstrate their conformance to the ATC 5401 Standard using the APIVS. They may ask the suppliers to demonstrate it using the suppliers own test equipment. The buyer may enlist an independent testing lab to use the APIVS to validate the API software. If the buyer has a testing capability, they may choose to perform testing using the APIVS themselves. Agencies should consider their alternatives prior to distributing a Request for Proposal or solicitation for bids on equipment that contains API software.

7.2 Organizational Impacts

The APIVS will have minimal organizational impacts. If there are agencies that have testing capabilities, they may be adding the APIVS to their testing procedures and responsibilities assigned accordingly.

7.3 Impacts During Development

There are no impacts during development except for those already identified for the API Working Group and its contractors as established in the APIRI Project Management Plan (see Section 2 References).

8 ANALYSIS OF THE PROPOSED SOFTWARE
This section summarizes improvements, states disadvantages and limitations, and alternatives and trade-offs considered.

8.1 Summary of Improvements

The APIVS is new software with the benefits listed in Section 4 and features described in Section 5.3 of this ConOps.

8.2 Disadvantages and Limitations

Most agencies who test or qualify equipment as part of their procurement process actually qualify an application program (i.e. signal control program) on a dedicated transportation controller. This kind of testing is usually performed by traffic operations personnel, not Testers as described in Section 5.5. Those agencies that do not have a Tester will need to hire a qualified individual or contract out the testing of the APIVS.

8.3 Alternatives and Trade-Offs Considered

Alternative testing methodologies were considered for the ConOps. Figures 4 and 5 show tests environments used by traffic engineers to test operational software. They utilize suitcase testers, conflict monitors and cabinet bus simulators. While such testing environments could be used to test API software, these methods were not selected as they: a) do not provide the isolation of the API software from the input and output components of the controller hardware or b) it is difficult to provide the level of automation desired for testing.

Figure 4. Test environment for the FIO library for TS 1, TS 2 Type 2 and Model 332 cabinets.
Figure 5. Test environment for the FIO library for TS 2 Type 1 and ITS cabinets.

Figure 6 shows a test environment which employs an external test fixture. Crossover cables are used to connect the input ports to the output ports of the Engine Board. While this approach does offer more automation over the previous two approaches, it requires the testing agency to bear the initial cost of buying or building a test fixture and cables. It also requires the user to extract the Engine Board out of the controller unit to perform the tests.

Figure 6. Test environment that uses a test fixture with crossover cables and a personal computer.
## 9 NOTES

### 9.1 Definitions and Acronyms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AASHTO</td>
<td>American Association of State Highway and Transportation Officials</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>API Utilities</td>
<td>API software that is used for setting system-wide purposes on an ATC controller unit.</td>
</tr>
<tr>
<td>APIRI</td>
<td>API Reference Implementation (software). API software developed as part of the ATC APIRI Project.</td>
</tr>
<tr>
<td>APIRI Project</td>
<td>Entire project managed by ATC APIRI PMP v01.01 Project Management Plan (PMP) for the Advanced Transportation Controller (ATC) Application Programming Interface (API) Reference Implementation Project including software, hardware and documentation.</td>
</tr>
<tr>
<td>APIVS</td>
<td>API Validation Suite</td>
</tr>
<tr>
<td>Application Program</td>
<td>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.</td>
</tr>
<tr>
<td>ATC</td>
<td>Advanced Transportation Controller</td>
</tr>
<tr>
<td>ATC Device Drivers</td>
<td>Low-level software not included in a typical Linux distributions that is necessary for ATC-specific devices to operate in a Linux O/S environment.</td>
</tr>
<tr>
<td>BOM</td>
<td>Bill of Materials. A list of the raw materials, sub-assemblies, intermediate assemblies, sub-components, parts and the quantities of each needed to manufacture an end product.</td>
</tr>
<tr>
<td>Board Support Package</td>
<td>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</td>
</tr>
<tr>
<td>BSP</td>
<td>See Board Support Package</td>
</tr>
<tr>
<td>ConOps</td>
<td>Concept of Operations</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit. A programmable logic device that performs the instruction, logic and mathematical processing in a computer.</td>
</tr>
<tr>
<td>Device Driver</td>
<td>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.</td>
</tr>
<tr>
<td>FIO</td>
<td>Field Input and Output</td>
</tr>
<tr>
<td>FPUI</td>
<td>Front Panel User Interface</td>
</tr>
<tr>
<td>H/W</td>
<td>Hardware</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>IEC</td>
<td>International Electrotechnical Commission</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>ITE</td>
<td>Institute of Transportation Engineers</td>
</tr>
<tr>
<td>ITS</td>
<td>Intelligent Transportation Systems</td>
</tr>
<tr>
<td>JC</td>
<td>Joint Committee</td>
</tr>
<tr>
<td>JPO</td>
<td>Joint Program Office</td>
</tr>
<tr>
<td>Linux</td>
<td>Low-level software that is freely available in the Linux community for use with common hardware components operating in a standard fashion.</td>
</tr>
<tr>
<td>Linux Kernel</td>
<td>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 4.3.5, Appendix A and Appendix B.</td>
</tr>
<tr>
<td>Loopback Driver</td>
<td>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.</td>
</tr>
<tr>
<td>Mechanical Drawing</td>
<td>A drawing to scale of a machine, machine component, or device from which dimensions can be taken for manufacturing.</td>
</tr>
<tr>
<td>N/A</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Operational User</td>
<td>A technician or transportation engineer who uses the controller to perform its operational tasks.</td>
</tr>
<tr>
<td>O/S</td>
<td>Operating System</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>PMP</td>
<td>Project Management Plan</td>
</tr>
<tr>
<td>RI</td>
<td>Reference Implementation</td>
</tr>
<tr>
<td>RTC</td>
<td>Real-Time Clock</td>
</tr>
<tr>
<td>SDO</td>
<td>Standards Development Organization</td>
</tr>
<tr>
<td>Schematic Diagram</td>
<td>A diagram which shows, by means of graphic symbols, the electrical connections and functions of a specific circuit arrangement.</td>
</tr>
<tr>
<td>SE</td>
<td>Systems Engineer</td>
</tr>
<tr>
<td>Software Validation</td>
<td>The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.</td>
</tr>
<tr>
<td>SOW</td>
<td>Statement of Work</td>
</tr>
<tr>
<td>SRS</td>
<td>Software Requirements Specification</td>
</tr>
<tr>
<td>S/W</td>
<td>Software</td>
</tr>
<tr>
<td>TBD</td>
<td>To Be Determined</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Tester</td>
<td>A user developer, test engineer or test technician capable of operating the API Validation Suite described by this document.</td>
</tr>
<tr>
<td>TOD</td>
<td>Time of Day</td>
</tr>
<tr>
<td>TOPR</td>
<td>Task Order Proposal Request</td>
</tr>
<tr>
<td>US</td>
<td>United States</td>
</tr>
<tr>
<td>USDOT</td>
<td>United States Department of Transportation</td>
</tr>
<tr>
<td>User Developer</td>
<td>A software developer that designs and develops programs for controllers.</td>
</tr>
<tr>
<td>Walkthrough</td>
<td>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.</td>
</tr>
<tr>
<td>WG</td>
<td>Working Group</td>
</tr>
</tbody>
</table>

10 APPENDICES

There are no appendices at this time.

§