A Project Document of the
Advanced Transportation Controller Joint Committee

API VAL SUITE CONOPS v01.01

Application Programming Interface (API) Validation Suite Concept of Operations (ConOps)

April 15, 2010

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 ATC API Validation Suite Project
Members of the ATC API Working Group
Contractor Team for the ATC API Validation Suite Project
Tom Gauger, NFocal
Steve Kapp, NFocal

Prepared by: Ralph W. Boaz

© Copyright 2010 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>
</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 System Overview.

1.1 Identification

This Concept of Operations (ConOps) applies to Application Programming Interface (API) Validation Suite (VS) Version 1 (or simply API VS).

1.2 Document Overview

The purpose of this document is to communicate the user needs and expectations of the proposed API Validation Suite to the developers and to serve as a basis for developing a Software Requirements Specification.

1.3 System Overview

This document is a Concept of Operations (ConOps) for the test software to be developed as part of the Application Programming Interface (API) Validation Suite (VS) project under the United States Department of Transportation (USDOT) Work Order 14-0801, Tasks 7-8. The project is being performed under the direction of the Advanced Transportation Controller (ATC) Joint Committee (JC). The ATC JC is made up of representatives from three standards development organizations (SDOs): the American Association of State Highway and Transportation Officials (AASHTO), the Institute of Transportation Engineers (ITE) and the National Electrical Manufacturers Association (NEMA). The development effort will be carried out by the API Working Group (WG), a technical subcommittee of the ATC JC and a paid consultant team to support the API WG and SDO staff.

The software to be developed is intended to test implementations of the API Standard as part of the API Test Plan (see Section 2 Referenced Documents).

2 REFERENCED DOCUMENTS

The documents referenced in this ConOps are listed below.

"ATC API Standard v02.06a, Application Programming Interface (API) Standard for the Advanced Transportation Controller (ATC)," ATC JC, 31 May 2007. Available from the Institute of Transportation Engineers.

"ATC Controller Standard Revision v5.2b," ATC JC, 26 June 2006. Available from the Institute of Transportation Engineers.

"ATC Standard for the Type 2070 Controller v01.05," ATC JC, 29 March 2001. Available from the Institute of Transportation Engineers.


“NEMA Standards Publication TS 2-2003 v02.06 Traffic Controller Assemblies with NTCIP Requirements.” Available from the National Electrical Manufacturers Association.

Project Management Plan (PMP) for the Advanced Transportation Controller (ATC) Application Programming Interface (API) Validation Suite Project v01.05,” ATC JC, 14 May 2009. Available from the Institute of Electrical and Electronics Engineers.

"Test Plan for the Advanced Transportation Controller (ATC) Application Programming Interface (API) v01.04," ATC JC, 8 September 2009. Available from the Institute of Electrical and Electronics Engineers.

"www.ite.org/standards/atcapi/," Official API Website. Available through the Institute of Electrical and Electronics Engineers.

3 BACKGROUND

The ATC Controller Standard specifies a controller architecture where the computational components reside on a printed circuit board (PCB), called the “Engine Board,” with standardized connectors and pinout. It includes a central processing unit (CPU), a Linux operating system (O/S), memory, external and internal interfaces and other associated hardware necessary to create an embedded transportation computing platform. The Engine Board plugs into a “Host Module” that supplies power and physical connection to the I/O devices of the controller. While the interface to the Engine Board is completely specified, the Host Module may be of various shapes and sizes to accommodate controllers of various designs. Only minimum levels of performance are specified in the ATC Controller Standard. The CPU of an Engine Board may come from any manufacturer allowing future products to have higher performance processors and still be compliant to the standard.

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. The fixed resources "managed" through the API are the controller’s Front Panel, Field Input and Output (I/O) and Time-of-Day (TOD) Clock. 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 organization and 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 Controller Standard, Section 2.2.5, Annex A and Annex 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” refers to the software to be tested under this test plan. As shown in Figure 1, both users and application programs use the API to interface to ATC controller units.

![Layered organization of ATC software.](image)

Figure 1. Layered organization of ATC software.

4 JUSTIFICATION

Implementations of the API Standard are already being deployed on ATC Engine Boards. The API VS must be developed to provide agencies, integrators and second party software developers evidence of conformance to the API Standard.

Some of the benefits of the API VS include:

- Impartial testing common to the entire industry;
- Greater confidence and faster deployment of the ATC controllers with API software as manufacturers will have a level of provable compliance;
- A tool for end users to reference in their specifications;
- Increased reliability of API implementations;
- Increased portability and interoperability of application programs on ATC controllers;
- Provide a common source code for test software that can be expanded or enhanced when a new version of the API Standard is enhanced.

Not developing this capability will result in:

- API library vendors may make unsubstantiated claims of conformance.
- There will be less confidence in the deployment of the API Standard.
- There will be lost time and money to the industry.
- There will be slower deployment.
- Some of the goals of the ATC program will not be achieved.
5 CONCEPTS FOR THE PROPOSED SYSTEM

This section provides the concepts for the API VS.

5.1 Background, Objectives and Scope

The API Validation Suite is software that tests API Software residing on an ATC Engine Board. Although exhaustive testing is impractical, successful testing provides an agreed upon level of confidence that the API Software under test is conformant to the API Standard. The API VS should carry out testing in accordance to the API Test Plan.

5.2 Operational Policies and Constraints

There are no specific operational policies or constraints for this ConOps. Specific needs on the construction of the API VS are listed as user needs in the subsections of Section 5.3.

5.3 Description of the Proposed System

The test environment is shown in Figure 2. It consists of a test fixture, an ATC Engine Board and a personal computer (PC). Cross over cables are used to connect the input ports to the output ports of the Engine Board. The test fixture has appropriate power and connections to host the Engine Board, it converts the logical levels of the Engine Board's serial I/O ports to those appropriate for the external serial connections specified by the ATC Standard, and it provides convenient access to the serial ports for connection to a PC and crossover connections to be used in testing. This environment allows test software to be developed that runs on the Engine Board and tests the API Software. 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 the API Test Plan. One or more vendors supply this type of fixture.

Since Engine Boards may have been implemented using a variety of processors, test software identified in this test plan will need to be compiled in a manner appropriate for the implementation of the Engine Board.

![Figure 2. API test environment uses a test fixture with crossover cables and a personal computer.](image-url)
Figure 3. API test fixture hosts an API Engine Board and provides convenient access to the Engine Board’s serial ports.

The layered software environment for the API VS is similar to the layered organization of the ATC software. The API VS takes the place of the Application Software in Figure 1. The API VS exercises the API Software and records results.

Figure 4. The layered software environment for the API Validation Suite.

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

The user needs the API VS source code and documentation to be available to anyone. Users will want to be able to use it in their organizations for testing and they may not want to use an executable version provided by a manufacturer. In this case, they will need to compile and load it themselves for the particular controller under test.

5.3.2 Unrestricted Use

The user needs the API VS to have a clearly defined license model which allows for unrestricted use. After obtaining the API VS, users should not be required to ask permission to use it.

5.3.3 Consistent with the API Test Plan

The user needs the API VS to be operational in the test environment as indicated in Section 5.3. There may be other methods of performing the same capability are multiple test configurations that are identified in the Test Plan.

5.3.4 C Programming Language

The user needs the API VS to be written using the C programming language as described by “ISO/IEC 9899:1999” commonly referred to as the C99 Standard. The API Standard specifies C for its function definitions. C is also the most commonly used programming language for embedded system (see Section 2 Referenced Documents).

5.3.5 Quality

The user needs the API VS to be written using the GNU Coding Standards (see Section 2 Referenced Documents). The software needs to be written in a consistent fashion so that it can be maintained by others.

5.3.6 Pass / Fail Indications

The user needs the API VS to provide a pass/fail indication of conformance of the API Software to the API Standard. Users must have a repeatable method to validate that they have conforming operational API Software.

5.3.7 Logs and Traces

The user needs the API VS to provide a log or trace of the tests performed and the results of each test and/or step. Users may need additional information to diagnose anomalies in the API Software.

5.3.8 Completeness

The user needs the API VS to test the API Software for completeness with respect to the functions specified within the API Standard. This will validate that each API function is present and that its arguments conform to the API Standard.

5.3.9 Correctness

The user needs the API VS to test the API Software for correctness of operation. Each API function is to be a part of at least one integrated test that validates that the behavior of the API function conforms to the API Standard.
5.3.10 Robustness

The user needs the API VS to test the robustness of the API Software. This is to verify that the API Software supports multiple and simultaneous applications.

5.3.11 Front Panel Manager Window

The user needs the API VS to test the Front Panel Manager Window of the API Software. This portion of the API Standard is a user interface, not a functional one.

5.3.12 Configuration Window

The user needs the API VS to test the Configuration Window of the API Software. This portion of the API Standard is a user interface, not a functional one.

5.4 Modes of Operation

There are no special modes of operation for this ConOps.

5.5 User Classes and Other Involved Personnel

The "User Developer" is the only user class for the API VS. The "User Developer" is a software developer that designs and develops programs for controllers. These individuals typically have software, computer, or electrical engineering backgrounds. Once the testing procedures and practices within an organization have been established, it is conceivable that a User Developer could train others in their organization to exercise the testing. Such scenarios are beyond the scope of this ConOps.

5.6 Support Environment

Specific support environments for testing will vary based on the organization performing the tests. The API VS 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 API VS. 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 system on the users, developers, and support and maintenance organizations. This information is provided in order to allow preparations for the API VS by agencies, user and working groups, sponsoring organizations, and support and maintenance organizations.

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 API VS provides additional methods to validate conformance to the API Standard. Buyers may ask equipment providers to demonstrate their conformance to the API Standard using the API VS. 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 API VS to validate the API Software. If the buyer has a testing capability, they may choose to perform testing using the API VS 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 API VS will have minimal organizational impacts. If there are agencies that have testing capabilities, they may be adding the API VS 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 API VS Project Management Plan (see Section 2 References).

8 ANALYSIS OF THE PROPOSED SYSTEM

This section summarizes

8.1 Summary of Improvements

The API VS is a new system with the benefits listed under Section 4 Justification.

8.2 Disadvantages and Limitations

The extensive combinations of functions that are possible using the API Software make exhaustive testing impractical. This means that there may be combinations of features that may not be tested by the API VS.

8.3 Alternatives and Trade-Offs Considered

Alternative testing methodologies were considered. Figures 5 and 6 show tests environments used by traffic engineers to test operational software. The involve 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 would be slow to operate and require extensive human interaction.

There are similar methods in which the inputs and outputs of the ATC Engine Board can be crossed and the API VS could be used to test API Software. These are not discussed specifically as they would be a variation of the approach already taken.
9 NOTES

9.1 Acronyms and Abbreviations

<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>AC</td>
<td>alternating current</td>
</tr>
<tr>
<td>ATC</td>
<td>Advanced Transportation Controller</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>ConOps</td>
<td>Concept of Operations</td>
</tr>
<tr>
<td>DC</td>
<td>direct current</td>
</tr>
<tr>
<td>EMI</td>
<td>electromagnetic interference</td>
</tr>
<tr>
<td>FHWA</td>
<td>Federal Highway Administration</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>I/F</td>
<td>interface</td>
</tr>
<tr>
<td>I/O</td>
<td>input/output</td>
</tr>
<tr>
<td>IMSA</td>
<td>International Municipal Signal Association</td>
</tr>
<tr>
<td>IPC</td>
<td>Formerly, the Institute for Printed Circuits. This same institution was later called the Institute Interconnecting and Packaging Electronic Circuits. It is now referred to as IPC-Association Connecting Electronics Industries.</td>
</tr>
<tr>
<td>ITE</td>
<td>Institute of Transportation Engineers</td>
</tr>
<tr>
<td>ITS</td>
<td>Intelligent Transportation System</td>
</tr>
<tr>
<td>JPO</td>
<td>Joint Program Office</td>
</tr>
<tr>
<td>LED</td>
<td>light emitting diode</td>
</tr>
<tr>
<td>NEC</td>
<td>National Electrical Code</td>
</tr>
<tr>
<td>NEMA</td>
<td>National Electrical Manufacturers Association</td>
</tr>
<tr>
<td>NRTL</td>
<td>Nationally Recognized Testing Lab</td>
</tr>
<tr>
<td>NTCIP</td>
<td>National Transportation Communications for ITS Protocol</td>
</tr>
<tr>
<td>NFPA</td>
<td>National Fire Protection Association</td>
</tr>
<tr>
<td>RFI</td>
<td>radio frequency interference</td>
</tr>
<tr>
<td>SDO</td>
<td>Standard Development Organization</td>
</tr>
<tr>
<td>SEMP</td>
<td>System Engineering Management Plan</td>
</tr>
<tr>
<td>SEP</td>
<td>Systems Engineering Process</td>
</tr>
<tr>
<td>TFCS</td>
<td>transportation field cabinet system</td>
</tr>
<tr>
<td>UPS</td>
<td>Uninterruptible Power Source</td>
</tr>
<tr>
<td>USA</td>
<td>United States of America</td>
</tr>
<tr>
<td>USDOT</td>
<td>United States Department of Transportation</td>
</tr>
<tr>
<td>VAC</td>
<td>voltage alternating current</td>
</tr>
<tr>
<td>VDC</td>
<td>voltage direct current</td>
</tr>
<tr>
<td>WG</td>
<td>Working Group</td>
</tr>
</tbody>
</table>

10 **APPENDICES**

There are no appendices at this time.

11 **GLOSSARY**

TBD

§