A Project Document of the
Advanced Transportation Controller Joint Committee

API VAL SUITE SRS v01.01

Application Programming Interface (API) Validation Suite Software Requirements Specification (SRS)

May 21, 2010

SRS 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>05/13/10</td>
<td>Initial Draft WGD Version 01.00.</td>
</tr>
<tr>
<td>05/21/10</td>
<td>Version 01.01 includes changes from API WG Telecon 05/21/10.</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.
CONTENTS

1 INTRODUCTION .............................................................................................................................. 5
1.1 Purpose.................................................................................................................................... 5
1.2 Scope ....................................................................................................................................... 5
1.3 Definitions, Acronyms and Abbreviations ........................................................................... 6
1.4 References ............................................................................................................................. 6
1.5 Overview ............................................................................................................................... 7

2 OVERALL DESCRIPTION .............................................................................................................. 7
2.1 Product Perspective ............................................................................................................ 7
2.2 Product Features ............................................................................................................... 11
2.3 User Characteristics .......................................................................................................... 12
2.4 Constraints ........................................................................................................................ 13
2.5 Assumptions and Dependencies ...................................................................................... 13

3 SPECIFIC REQUIREMENTS ........................................................................................................ 13
3.1 No Cost ............................................................................................................................. 13
3.2 Uses an ITE Approved License Model.............................................................................. 13
3.3 Operational and Consistent with the API Test Plan .......................................................... 13
3.4 GNU Coding Standards .................................................................................................... 13
3.5 Conformance / Nonconformance Indications .................................................................... 13
3.6 Detailed Logs and Traces ................................................................................................. 13
3.7 Summary Result................................................................................................................ 14
3.8 Output Options .................................................................................................................. 14
3.9 XML Output Files .............................................................................................................. 14
3.10 Automated Use of APIVSXML Defined Test Cases ......................................................... 14
3.11 Function Present ............................................................................................................... 14
3.12 Conforming Arguments ..................................................................................................... 14
3.13 Integrated Tests ................................................................................................................ 14
3.14 Front Panel Manager Operation ....................................................................................... 14
3.15 ATC Configuration Window Operation .......................................................................... 14
3.16 Field I/O Manager Operation ............................................................................................ 15
3.17 Multiple and Simultaneous Applications ........................................................................ 15
3.18 C Programming Language ............................................................................................. 15
3.19 Incorporate New Functions ............................................................................................... 15

APPENDICES ............................................................................................................................................. 16
A.1 User Needs to Requirements Traceability Matrix ............................................................. 16
1 INTRODUCTION

This document is a Software Requirements Specification (SRS) for 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.

This section provides the Identification, Document Overview and System Overview. This section provides an introduction for this document. It includes subsections on Purpose; Scope; Definitions, Acronyms, and Abbreviations; References; and Overview.

1.1 Purpose

This SRS establishes the requirements for software that will be used to test and validate implementations of the API Standard Version 1 (see Section 1.4 References) referred to as API Software. It has been prepared by the API WG to establish a common understanding of the validation software requirements for:

a) The API WG who prepares this document using a consensus process;
b) The contractors who will develop the validation software;
c) The local, state and federal transportation agencies who specify and deploy ATC equipment;
d) The software developers, consultants and manufacturers who provide implementations of the API Standard and create application programs that utilize API Software; and
e) The public who benefit from the application programs that run on ATC equipment and who directly or indirectly pays for these products.

1.2 Scope

The validation software produced in response to this Software Requirements Specification (SRS) is referred to as the API Validation Suite (API Val Suite or APIVS) Version 1. The APIVS is software that tests API Software residing on an ATC Engine Board (see Section 2.1 Product Perspective). 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 APIVS should carry out testing in accordance to the API Test Plan (see Section 1.4 References).

Since implementations of the API Standard are already being deployed on ATC Engine Boards, the APIVS 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 APIVS 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 compliancy;
- A tool for API Software users to reference in their specifications;
- Increased reliability of API implementations;
- Increased portability and interoperability of application programs on ATC controllers; and
- Provide a common source code for test software that can be expanded or enhanced when a new version of the API Standard is produced.
1.3 Definitions, 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>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>API Software</td>
<td>Software developed which is intended to adhere to the API Standard.</td>
</tr>
<tr>
<td>APIVS</td>
<td>Application Programming Interface Validation Suite</td>
</tr>
<tr>
<td>APIVSXML</td>
<td>APIVS Extensible Markup Language</td>
</tr>
<tr>
<td>ATC</td>
<td>Advanced Transportation Controller</td>
</tr>
<tr>
<td>BSP</td>
<td>board support package</td>
</tr>
<tr>
<td>ConOps</td>
<td>Concept of Operations</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>I/O</td>
<td>input/output</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>NEMA</td>
<td>National Electrical Manufacturers Association</td>
</tr>
<tr>
<td>NTCIP</td>
<td>National Transportation Communications for ITS Protocol</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>USA</td>
<td>United States of America</td>
</tr>
<tr>
<td>USDOT</td>
<td>United States Department of Transportation</td>
</tr>
<tr>
<td>WG</td>
<td>Working Group</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>

1.4 References

"API Validation Suite APIVSXML Specification, vXX.YY," ATC WG, Date TBD, Available from the Institute of Transportation Engineers.

"ATC API Standard v02.06a, Application Programming Interface (API) Standard for the Advanced Transportation Controller (ATC)," ATC JC, 31 May 2007. This document is also known as API Standard Version 1. 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.


1.5 Overview

This standard is made up of four sections, appendixes and an index. Section 1, Introduction, provides an overview of the entire document. Section 2, Overall Description, provides background information and identifies the user needs for the requirements defined in Section 3. Section 3, Specific Requirements, establishes the requirements that must be satisfied by the APIVS. Appendix A contain traceability matrix showing the relationship of user needs to the requirements.

2 OVERALL DESCRIPTION

This section describes the general factors that affect the APIVS and its requirements. It includes subsections on Product Perspective; Product Features; User Characteristics; Constraints; Assumptions and Dependencies; and Apportioning of Requirements. The section on Product Features identifies the user needs for the APIVS, specifically, from which justify and support the requirements defined in Section 3. Each user need identified is also included in a traceability matrix in Appendix A. The terms “function” or “functions” used in this section refer to software functions or software function calls. If a general functionality or capability is intended, an effort was made to use other terms.

2.1 Product Perspective

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 both a user and software interface which resides on the Engine Board. The user interface provides a window system which enables...
programs to run concurrently and share the same physical front panel display of the controller. The software interface provides functions that can be called by application programs to utilize the window system, share the field I/O ports of the controller and manage the real-time clock.

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.

Figure 1. Layered organization of ATC software.

The layered software environment for the APIVS shown in Figure 2 is similar to the layered organization of the ATC software. The APIVS takes the place of the Application Software in Figure 1. The APIVS exercises the API Software and records results.
Figure 2. The layered software environment for the API Validation Suite.

The test environment described by the API Test Plan is shown in Figure 3. 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, as shown in Figure 4, has appropriate power and connections to host the Engine Board. To facilitate testing, 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 these serial ports for 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 as the API Test Plan evolves. The test environment itself is considered commercial off-the-shelf and not a requirement of this SRS.

Figure 3. API test environment uses a test fixture with crossover cables and a personal computer.
Figure 4. API test fixture hosts an API Engine Board and provides convenient access to the Engine Board's serial ports.

2.1.1 System Interfaces

The test environment described in Section 2.1 and the APIVS is a self contained system. There are no other system interfaces necessary.

2.1.2 User Interfaces

A PC running a HyperTerminal program may serve as a console in the operation of some tests. The intent of the APIVS is to limit human interaction during the test. Any specific user needs for user interfaces are identified in Section 2.2.

2.1.3 Hardware Interfaces

The hardware interface for the Engine Board and the test fixture is like the interface between the Engine Board and Host Module defined in the ATC Controller Standard (see Section 1.4 References). In this case, the test fixture acts as the Host Module.

2.1.4 Software Interfaces

The APIVS will operate on the ATC Engine Board utilizing standard Linux interprocess communications as is determined necessary by the contractor. The APIVS will interface to the Engine Board via the API Software and the Linux environment defined in the ATC Controller Standard.

2.1.5 Communications Interfaces

Crossover serial cables are used to tie the input and output ports of the Engine Board together so that the APIVS can perform testing without human interaction. A serial or Ethernet connection may be used between the PC and the test fixture to load the APIVS, initiate tests, and extract results. This is already provided for through the Linux environment of the Engine Board and physical ports of the test fixture.

2.1.6 Memory

There are no limits on memory except that APIVS must be able to operate on a controller unit with memory limits described in ATC Controller Standard.
2.1.7 Operations

There are no additional operational needs.

2.1.8 Site Adaption Requirements

There are no site adaptation requirements.

2.2 Product Features

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.

2.2.1 Accessibility

The user needs the APIVS 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.

2.2.2 Defined License Model

The user needs the APIVS to have a clearly defined license model which has limited or no restrictions on the use of the APIVS. *After obtaining the APIVS, users should not be required to ask permission to use it.*

2.2.3 Consistent with the API Test Plan

The user needs the APIVS to be operational in the test environment as indicated in Section 2.1. *The APIVS needs to operate in a manner that supports the API Test Plan even though there may be other conceivable test environments, approaches and methods.*

2.2.4 Quality

The user needs the APIVS to be written using consistent naming conventions software constructs and effective use of inline comments. *The software needs to be written in a consistent fashion so that it can be maintained by the members of the API WG and those users with software development capabilities.*

2.2.5 Conformance Indications

The user needs the APIVS to provide a clear and concise 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.*

2.2.6 Detailed Test Results

The user needs the APIVS 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.*
2.2.7 Completeness

The user needs the APIVS to test the API Software for completeness with respect to the API Standard. *The user needs all of the components of the API Software tested. The user needs to know that the standard is adequately covered.*

2.2.8 Correctness

The user needs the APIVS to test the API Software for correctness of operation. *The user needs to know that the API Software operates properly.*

2.2.9 Robustness

The user needs the APIVS to test the robustness of the API Software. *The user must have confidence that the API Software supports multiple and simultaneous applications.*

2.2.10 Portable

The user needs the APIVS to be portable to any ATC Engine Board. *Since Engine Boards may have been implemented using a variety of processors, the APIVS needs to be designed and developed in a manner that will facilitate it being ported to ATC compliant Engine Boards from any manufacturer. Users that are software developers will need to be able to compile and build executable versions of the APIVS for various Engine Boards. Users of the APIVS that are not software developers will need to request that the provider of the API Software also provide a compatible executable version of the APIVS for the engine board used in the test.*

2.2.11 Extensible

The user needs the APIVS to be extensible so that tests can be added to the APIVS without the need to perform code changes to the APIVS software. *It is expected that a group of tests will be developed initially. Users need the APIVS to facilitate the addition of new tests in the future.*

2.3 User Characteristics

The typical user of APIVS is a test technician with the following characteristics:

1) Is proficient with PCs running Microsoft Windows operating systems;
2) Understands the organization and use of the Windows file system;
3) Is experienced with using the communications services Telnet and FTP;
4) Is adept at handling electronic equipment, circuit boards and cabling;
5) Is able to follow detailed procedures; and
6) Is capable of careful record keeping and can organize test results.

A second type of user of the APIVS is a software developer with the following characteristics in addition to those of the test technician:

1) Generally has the skills and experience to design, construct, test, and maintain the software systems;
2) Understands processes and procedures to compile, link and load software for embedded systems;
3) Is proficient at writing software in the C programming language; and
4) Is experienced with the Linux Operating System.
2.4 Constraints

The APIVS must be operable on the Test Environment as described in Section 2.1 Product Perspective.

2.5 Assumptions and Dependencies

There are no additional assumptions or dependencies than those implicit in Sections 2.1 through 2.4.

3 SPECIFIC REQUIREMENTS

This section defines the requirements for the APIVS. Each requirement is listed with a separate paragraph number and included in a traceability matrix in Appendix A. The terms “function” or “functions” used in this section refer to software functions or software function calls. If a general functionality or capability is intended, an effort was made to use other terms.

3.1 No Cost

The APIVS shall not have any content or component that inhibits the no cost distribution of its source code and documentation.

3.2 Uses an ITE Approved License Model

The APIVS shall follow a license model that is approved by ITE and shall make no other restrictions in its use.

3.3 Operational and Consistent with the API Test Plan

The APIVS shall be operational within the test environment and testing approach described in Section 2.1 and consistent with the API Test Plan.

3.4 GNU Coding Standards

The APIVS code shall be written in adherence to the GNU Coding Standards (see Section 1.4 References).

3.5 Conformance / Nonconformance Indications

The return values of the APIVS shall be 0, indicating conformance, or -1, indicating nonconformance of the API Software to the API Standard.

3.6 Detailed Logs and Traces

The APIVS shall provide a log or trace of the tests performed including:

- 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 Specification (TCS) being executed;
- Line # in the APIVS source code;
- Step in the TCS; and
- Time stamps for each step in the TCS.
3.7 Summary Result

The APIVS shall provide a summary of the result of the testing including TCSs performed and their pass/fail result.

3.8 Output Options

The APIVS shall provide the user with the ability to specify output options as follows:
1) Conformance/nonconformance Indication only;
2) Conformance/nonconformance indication and summary result;
3) Conformance/nonconformance indication, summary result and all logs and traces.

3.9 XML Output Files

The APIVS shall output any summary, log or traces into a file in an XML (Extensible Markup Language) format.

3.10 Automated Use of APIVSXML Defined Test Cases

The APIVS shall read Test Case Specifications (TCSs) written in APIVSXML (see Section 1.4 References) and perform the testing steps identified therein.

3.11 Function Present

The APIVS shall validate that each API function defined in the API Standard Section 4 Application Programming Interface and its subsections is present in the API Software. [Guidance: This requirement could possibly be met through the use of the APIVSXML test cases.]

3.12 Conforming Arguments

The APIVS shall validate that all arguments of each API function conform to the API Standard. [Guidance: This requirement could possibly be met through the use of the APIVSXML test cases.]

3.13 Integrated Tests

Each API function shall be a part of at least one integrated test that validates that the behavior of the API function conforms to the API Standard. [Guidance: This could possibly be met through the use of the APIVSXML test cases.]

3.14 Front Panel Manager Operation

The APIVS shall validate that the API Software meets the requirements of the window system defined in the API Standard Section 3.1.1 Front Panel Manager Requirements and its subsections. [Guidance: This will require emulation of the ATC front panel display which is an enhanced version of a VT100 terminal.]

3.15 ATC Configuration Window Operation

The APIVS shall validate that the API Software meets the requirements of the ATC Configuration Window defined in the API Standard Section 3.2.1 ATC Configuration Window Requirements.
3.16 Field I/O Manager Operation

The APIVS shall validate that the API Software meets the requirements of the field I/O system defined in the API Standard Section 3.1.2 Field I/O Manager Requirements and its subsections.

3.17 Multiple and Simultaneous Applications

The API VS shall validate that 16 application programs, running concurrently, can exercise the window system, the Front Panel Manager functions and the Field I/O Manager functions simultaneously. [Guidance: This requirement could be met with multiple subprocesses or threads of the same test application program.]

3.18 C Programming Language

The APIVS shall be written using the C programming language as described by “ISO/IEC 9899:1999” commonly referred to as the C99 Standard.

3.19 Incorporate New Functions

The APIVS shall have constructs and documentation with specific directions on how new API functions can be added in the future.
APPENDICES

A.1 User Needs to Requirements Traceability Matrix

<table>
<thead>
<tr>
<th>User Need ID</th>
<th>User Need Description</th>
<th>Req ID</th>
<th>Requirement Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.1</td>
<td>Accessibility</td>
<td>3.1</td>
<td>No Cost</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Defined License Model</td>
<td>3.2</td>
<td>Uses an ITE Approved License Model</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Consistent with the API Test Plan</td>
<td>3.3</td>
<td>Operational and Consistent with the API Test Plan</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Quality</td>
<td>3.4</td>
<td>GNU Coding Standards</td>
</tr>
<tr>
<td>2.2.5</td>
<td>Conformance Indications</td>
<td>3.5</td>
<td>Conformance / Nonconformance Indications</td>
</tr>
<tr>
<td>2.2.6</td>
<td>Detailed Test Results</td>
<td>3.6</td>
<td>Detailed Logs and Traces</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.7</td>
<td>Summary Result</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.8</td>
<td>Output Options</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.9</td>
<td>XML Output Files</td>
</tr>
<tr>
<td>2.2.7</td>
<td>Completeness</td>
<td>3.11</td>
<td>Function Present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.12</td>
<td>Conforming Arguments</td>
</tr>
<tr>
<td>2.2.8</td>
<td>Correctness</td>
<td>3.13</td>
<td>Integrated Tests</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.14</td>
<td>Front Panel Manager Operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.15</td>
<td>ATC Configuration Window Operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.16</td>
<td>Field I/O Manager Operation</td>
</tr>
<tr>
<td>2.2.9</td>
<td>Robustness</td>
<td>3.17</td>
<td>Multiple and Simultaneous Applications</td>
</tr>
<tr>
<td>2.2.10</td>
<td>Portable</td>
<td>3.18</td>
<td>C Programming Language</td>
</tr>
<tr>
<td>2.2.11</td>
<td>Extensible</td>
<td>3.10</td>
<td>Automated Use of APIVSXML Defined Test Cases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.19</td>
<td>Incorporate New Functions</td>
</tr>
</tbody>
</table>