***A Project Document of the***

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

ATC APIRI TCS v01.04

Test Case Specifications (TCS) for the Advanced Transportation Controller (ATC) Application Programming Interface Reference Implementation (APIRI)

**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** |
| 11/8/15 | Initial Draft TCS v01.00 |
| 12/1/15 | TCS v01.02 |
| 2/22/16 | TCS v01.03 (TRR) |
| 7/14/16 | TCS 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](#_Toc456255137)

[2 TEST CASE SPECIFICATIONS 6](#_Toc456255138)

**[2.1](#_Toc456255139)** [Common Elements Required by All Test Case Specifications 6](#_Toc456255139)

**[2.2](#_Toc456255140)** [Filename Conventions 7](#_Toc456255140)

**[2.3](#_Toc456255141)** [Test Case Specification 1 – FPUI Text UI Virtual Displays 8](#_Toc456255141)

**[2.4](#_Toc456255142)** [Test Case Specification 2 – FPUI Front Panel Manager 11](#_Toc456255142)

**[2.5](#_Toc456255143)** [Test Case Specification 3 – FPUI Character Set and Screen Attributes 15](#_Toc456255143)

**[2.6](#_Toc456255144)** [Test Case Specification 4 – FPUI Reading and Writing Data 18](#_Toc456255144)

**[2.7](#_Toc456255145)** [Test Case Specification 5 – FPUI Special Characters 20](#_Toc456255145)

**[2.9](#_Toc456255146)** [Test Case Specification 6 – FPUI Key Mapping 21](#_Toc456255146)

**[2.10](#_Toc456255147)** [Test Case Specification 7 – FPUI Asynchronous Notification and Focus 23](#_Toc456255147)

**[2.11](#_Toc456255148)** [Test Case Specification 8 – FPUI Raw Data Handling 24](#_Toc456255148)

**[2.12](#_Toc456255149)** [Test Case Specification 9 – API Version Information (All Libraries) 25](#_Toc456255149)

**[2.13](#_Toc456255150)** [Test Case Specification 10 – General FIO Operations 26](#_Toc456255150)

**[2.14](#_Toc456255151)** [Test Case Specification 11 – FIO Inputs and Outputs 29](#_Toc456255151)

**[2.15](#_Toc456255152)** [Test Case Specification 12 – FIO Channel Mapping 31](#_Toc456255152)

**[2.16](#_Toc456255153)** [Test Case Specification 13 – FIO Filtered Inputs and Transition Buffering 33](#_Toc456255153)

**[2.17](#_Toc456255154)** [Test Case Specification 14 – FIO Frame Frequency 35](#_Toc456255154)

**[2.18](#_Toc456255155)** [Test Case Specification 15 – FIO Failed State and Fault Monitoring 37](#_Toc456255155)

**[2.19](#_Toc456255156)** [Test Case Specification 16 – FIO Watchdog Outputs 39](#_Toc456255156)

**[2.20](#_Toc456255157)** [Test Case Specification 17 – FIO Device Status 41](#_Toc456255157)

**[2.21](#_Toc456255158)** [Test Case Specification 18 – FIO Health Monitor 43](#_Toc456255158)

**[2.22](#_Toc456255159)** [Test Case Specification 19 – FIO CMU Configuration 45](#_Toc456255159)

**[2.23](#_Toc456255160)** [Test Case Specification 20 – FIO Module Status 46](#_Toc456255160)

**[2.24](#_Toc456255161)** [Test Case Specification 21 – FIO Asynchronous Notification 47](#_Toc456255161)

**[2.25](#_Toc456255162)** [Test Case Specification 22 – FIO Dark Channel Mapping 48](#_Toc456255162)

**[2.26](#_Toc456255163)** [Test Case Specification 23 – TOD Time Handling Functions 49](#_Toc456255163)

**[2.27](#_Toc456255164)** [Test Case Specification 24 – Front Panel Manager and ATC Configuration Menu 50](#_Toc456255164)

**[2.28](#_Toc456255165)** [Test Case Specification 25 – System Configuration Utilities 54](#_Toc456255165)

**[2.29](#_Toc456255166)** [Test Case Specification 26 – Intrinsic API Requirements 59](#_Toc456255166)

**[2.30](#_Toc456255167)** [Test Case Specification 27 – FIO Serial Ports and Status Counters 61](#_Toc456255167)

**[2.31](#_Toc456255168)** [Test Case Specification 28 – FPUI Display Presence and Size 63](#_Toc456255168)

**[2.32](#_Toc456255169)** [Test Case Specification 29 – FPUI Bell Activation and Application Termination 65](#_Toc456255169)

**[2.33](#_Toc456255170)** [Test Case Specification 30 – FPUI Display Graphics 66](#_Toc456255170)

**[2.34](#_Toc456255171)** [Test Case Specification 31 – FPUI Display Focus 67](#_Toc456255171)

**[2.35](#_Toc456255172)** [Test Case Specification 32 – System Configuration Menu Display 69](#_Toc456255172)

# INTRODUCTION

This document, *Test Case Specifications (TCS) for the Advanced Transportation Controller (ATC) Application Programming Interface Reference Implementation (APIRI)*, provides the specific test cases necessary to test the required features of the API Reference Implementation.

These test cases have 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).

# TEST CASE SPECIFICATIONS

This section contains the individual Test Case Specifications referred to in the document *Test Plan for the*

*Advanced Transportation Controller (ATC) Application Programming Interface Reference Implementation (APIRI).*

## Common Elements Required by All Test Case Specifications

All of the test cases included in this document utilize a single hardware and software platform which is common to all tests. The compliance output files produced by all test cases also have a consistent format which allows pass/fail status to be easily ascertained.

To reduce the size of this document and to ease future maintenance, these common elements are described in the following subsections. Test cases which utilize these common elements will not contain identical subsections in their individual descriptions. If a test case deviates from any of these common elements the appropriate detail will appear in an appropriately-named subsection within that test case.

#### Hardware

All test cases utilize the hardware environment as described in the APIRI Test Plan, specifically:

* an ATC Controller with a primary USB port capable of running startup scripts and a minimum 8x40 character LCD display and associated keyboards
* a Personal Computer (PC) with 1GB available hard drive storage and an available USB port
* a 1GB USB Flash Drive, formatted with a suitable FAT file system

#### Software

All test cases require the porting of APIRI and APIVS software packages to the Engine Board platform used by the ATC Controller under test. 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 the APIRI Test Plan.

Prior to the execution of any test, the runtime APIVS package must be copied into the root directory of the USB Flash Drive. This package contains the executable VSE program and all configuration, script and data files necessary to execute all test cases using the associated test procedure(s).

#### Pass/Fail Criteria

Unless otherwise indicated, each test case execution produces an output compliance report (file) in XML format. This file contains an element (RunResult) toward the end of the file which indicates the overall completion status (PASS/FAIL) of the test.

<RunResult date="2015-10-01 12:33 PM EST" status="PASS" />

-OR-

<RunResult date="2015-10-01 12:33 PM EST" status="FAIL" />

If the test fails and the test execution was performed with detailed logging enabled, the file can be examined in more detail to determine the exact cause of the failure.

As this output XML file is textual, it can be viewed with a simple text editor such as Notepad. For a more structured view of the XML content in the file, there are many XML file viewing applications available, such as XML Notepad 2007, which can be downloaded via the Internet and installed on the test PC. There are also web sites available, such as <http://xmlgrid.net>, which provide customizable XML graphical views

If a test case does not produce a compliance report, there will be additional pass/fail criteria designated in that specific test case.

## Filename Conventions

A naming convention has been established for all files associated with individual APIRI test cases:

***Cnnnn\_xxxyyy.zzz***

where,

***C*** indicates a file associated with an APIVS test case

***nnnn*** is the test case number (from the Test Case Specification Identifier)

***xxx*** is an identifier for the specific file content:

**in** APIVSXML test script (XML format)

**log** conformance report (XML format)

**key** keystroke to Front Panel input file (VSE flat file format)

**vd** Virtual Display compare file (VSE flat file format)

**cmd** SDLC command message file (VSE flat file format)

**rsp** SDLC response message file (VSE flat file format)

***yyy*** is an (optional) numeric identifier, generally for VSE flat files only

***zzz*** is the standard file type (txt or xml)

VSE configuration files, which can be shared between test cases, following the naming convention ***VS\_config\_nnn.txt***, where nnn is a number from 1-999 indicating the specific configuration file to be used.

## Test Case Specification 1 – FPUI Text UI Virtual Displays

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.2010.

### Objective

The objective of this Test Case is to test the virtual application display handling capabilities of the API. A full complement of 16 applications with virtual displays will be created and display contents verified.

### Test Items

This test case tests the following requirements:

APIR3.1.1[1] The API shall provide a text-based user interface capability to allow application programs running concurrently on an ATC controller unit to share the controller’s Front Panel display.

APIR3.1.1[2] The API shall provide up to 16 virtual display screens (referred to as “windows”) that can be used by application programs as their user interface display.

APIR3.1.1[3] The display size of the windows shall be equal to the physical display size (lines x characters) of the controller’s Front Panel display (if one exists).

APIR3.1.1[6] Only one window shall be displayed at a time on the Front Panel display.

APIR3.1.1[7] When a window is displayed, the API shall display the character representation of the window on the Front Panel display (if one exists).

APIR3.1.1[8] The application program associated with the window displayed shall receive the characters input from the Front Panel input device (Ex. keyboard or keypad).

APIR3.1.1.2[1] The API shall provide a function to return the dimensions of a window in terms of number of lines and number of columns.

APIR3.1.1.2[2] The API shall provide a function to open a window and register a name for display on the Front Panel Manager Window.

APIR3.1.1.2[3] An application program shall be able to open multiple windows providing the windows resources are available.

APIR3.1.1.2[6] The API shall provide a function to close a window and release the resource for other application programs.

APIR3.1.1.2[22] The API shall provide a function to clear a window that operates on a window whether it is in or out of focus.

APIR3.1.1.2[23] The API shall provide a function to refresh a window that operates on a window whether it is in or out of focus.

APIR3.1.1.2[26] The API shall allow application programs to illuminate or extinguish the backlight of the ATC controller’s display if the command is received through a window that is in focus.

### Input Specifications

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

File Description

**C2010\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_key0.txt** keystroke file (Key ‘0’)

**Cxxxx\_key1.txt** keystroke file (Key ‘1’)

**Cxxxx\_key2.txt** keystroke file (Key ‘2’)

**Cxxxx\_key3.txt** keystroke file (Key ‘3’)

**Cxxxx\_key4.txt** keystroke file (Key ‘4’)

**Cxxxx\_key5.txt** keystroke file (Key ‘5’)

**Cxxxx\_key6.txt** keystroke file (Key ‘6’)

**Cxxxx\_key7.txt** keystroke file (Key ‘7’)

**Cxxxx\_key8.txt** keystroke file (Key ‘8’)

**Cxxxx\_key9.txt** keystroke file (Key ‘9’)

**Cxxxx\_keyA.txt** keystroke file (Key ‘A’)

**Cxxxx\_keyB.txt** keystroke file (Key ‘B’)

**Cxxxx\_keyC.txt** keystroke file (Key ‘C’)

**Cxxxx\_keyD.txt** keystroke file (Key ‘D’)

**Cxxxx\_keyE.txt** keystroke file (Key ‘E’)

**Cxxxx\_keyF.txt** keystroke file (Key ‘F’)

**Cxxxx\_keyStar.txt** keystroke file (Key ‘\*’)

**Cxxxx\_keyESC.txt** keystroke file (Key ‘<Esc>’)

**C2010\_vd00.txt** Virtual Display compare file (App ‘0’)

**C2010\_vd01.txt** Virtual Display compare file (App ‘1’)

**C2010\_vd02.txt** Virtual Display compare file (App ‘2’)

**C2010\_vd03.txt** Virtual Display compare file (App ‘3’)

**C2010\_vd04.txt** Virtual Display compare file (App ‘4’)

**C2010\_vd05.txt** Virtual Display compare file (App ‘5’)

**C2010\_vd06.txt** Virtual Display compare file (App ‘6’)

**C2010\_vd07.txt** Virtual Display compare file (App ‘7’)

**C2010\_vd08.txt** Virtual Display compare file (App ‘8’)

**C2010\_vd09.txt** Virtual Display compare file (App ‘9’)

**C2010\_vd10.txt** Virtual Display compare file (App ‘A’)

**C2010\_vd11.txt** Virtual Display compare file (App ‘B’)

**C2010\_vd12.txt** Virtual Display compare file (App ‘C’)

**C2010\_vd13.txt** Virtual Display compare file (App ‘D’)

**C2010\_vd14.txt** Virtual Display compare file (App ‘E’)

**C2010\_vd15.txt** Virtual Display compare file (App ‘F’)

**C2010\_vd\_fpm\_default.txt** Virtual Display compare file (default attributes)

**C2010\_vd\_bl\_on.txt** Virtual Display compare file (backlight ON)

**C2010\_vd\_bl\_off.txt** Virtual Display compare file (backlight OFF)

**2010.txt**

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C2010\_log.xml** conformance report (XML format)

### Environmental Needs

#### Hardware

This test case utilizes the standard APIVS test hardware configuration as detailed in Section 2.1.

#### Software

This test case utilizes the standard APIVS test software as detailed in Section 2.1.

#### Other

None.

### Special Procedural Requirements

None.

### Intercase Dependencies

None.

### Additional Pass/Fail Criteria

None.

## Test Case Specification 2 – FPUI Front Panel Manager

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.2020.

### Objective

The objective of this Test Case is to test the navigational capabilities of the Front Panel Manager (FPM) portion of the API. A full complement of 16 applications with virtual displays will be created and navigated using the FPM.

### Test Items

This test case tests the following requirements:

APIR3.1.1.1[1] The API shall provide a window selection screen called the Front Panel Manager Window from which Operational Users may select a window to have focus.

APIR3.1.1.1[2] Application names associated with each window shall be listed.

APIR3.1.1.1[3] The application names shall be limited to 16 characters.

APIR3.1.1.1[4] If there is no application program associated with a window, the window number shall be listed with a blank application name.

APIR3.1.1.1[5] The default Front Panel Manager Window size shall be 8 lines x 40 characters with the format as shown in Figure 7.

APIR3.1.1.1[6] If the Operational User has not set the default window, the Front Panel Manager Window shall be the default window.

APIR3.1.1.1[7] The default window shall be settable by the Operational User from the Front Panel Manager Window by pressing {\*,[0-F],<ENT>}.

APIR3.1.1.1[8] The Operational User shall be capable of setting the default window to the Front Panel Manager Window by pressing {\*,<ENT>} from the Front Panel Manager Window.

APIR3.1.1.1[9] The default window shall be designated by a star “\*” character next to the window number.

APIR3.1.1.1[10] The Operational User shall be able to put the Front Panel Manager Window in focus by pressing {\*\*,<ESC>} from the keypad on the controller’s Front Panel regardless of the application program in operation.

APIR3.1.1.1[11] The Operational User shall be able to enter {\*\*} by pressing an asterisk (\*) twice within a 1.0 second time period.

APIR3.1.1.1[12] If the {\*\*} sequence is not completed within the 1.0 second time period or if the {\*\*} sequence is not followed by <ESC> character within a 1.0 second time period, then the characters shall be interpreted as individual “\*” characters.

APIR3.1.1.1[13] The Operational User shall have the capability to put a window in focus that is assigned to an application program by pressing {[0-F]} from the Front Panel Manager Window.

APIR3.1.1.1[14] The only possible window selections for focus from the Front Panel Manager Window shall be itself, the ATC Configuration Window, or a window assigned to an application program.

APIR3.1.1.1[15] If the Front Panel Manager Window is the default window, no asterisk shall be displayed next to any application name in the Front Panel Manager Window.

APIR3.1.1.1[19] The Operational User shall be able to scroll up and down the names of the windows in the Front Panel Manager Window one line at a time using the up and down arrow keys of the controller keypad.

APIR3.1.1.1[20] The Operational User shall have the capability to put a window in focus that is assigned to an application program by pressing {\*\*,[0-F]} from the keypad on the controller’s Front Panel regardless of the application program in operation.

### Input Specifications

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

File Description

**C2020\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_key0.txt** keystroke file (Key ‘0’)

**Cxxxx\_key1.txt** keystroke file (Key ‘1’)

**Cxxxx\_key2.txt** keystroke file (Key ‘2’)

**Cxxxx\_key3.txt** keystroke file (Key ‘3’)

**Cxxxx\_key4.txt** keystroke file (Key ‘4’)

**Cxxxx\_key5.txt** keystroke file (Key ‘5’)

**Cxxxx\_key6.txt** keystroke file (Key ‘6’)

**Cxxxx\_key7.txt** keystroke file (Key ‘7’)

**Cxxxx\_key8.txt** keystroke file (Key ‘8’)

**Cxxxx\_key9.txt** keystroke file (Key ‘9’)

**Cxxxx\_keyA.txt** keystroke file (Key ‘A’)

**Cxxxx\_keyB.txt** keystroke file (Key ‘B’)

**Cxxxx\_keyC.txt** keystroke file (Key ‘C’)

**Cxxxx\_keyD.txt** keystroke file (Key ‘D’)

**Cxxxx\_keyE.txt** keystroke file (Key ‘E’)

**Cxxxx\_keyF.txt** keystroke file (Key ‘F’)

**Cxxxx\_keyStar.txt** keystroke file (Key ‘\*’)

**Cxxxx\_keyESC.txt** keystroke file (Key ‘<Esc>’)

**Cxxxx\_keyDN.txt** keystroke file (Key ‘<Down>’)

**C2020\_vd\_fpm\_default\_app0.txt** Virtual Display compare file (default display App ‘0’)

**C2020\_vd\_fpm\_clr.txt** Virtual Display compare file (clear display)

**C2020\_vd\_fpm\_1.txt** Virtual Display compare file (display 1)

**C2020\_vd\_fpm\_2.txt** Virtual Display compare file (display 2)

**C2020\_vd\_00.txt** Virtual Display compare file (App ‘0’)

**C2020\_vd\_01.txt** Virtual Display compare file (App ‘1’)

**C2020\_vd\_02.txt** Virtual Display compare file (App ‘2’)

**C2020\_vd\_03.txt** Virtual Display compare file (App ‘3’)

**C2020\_vd\_04.txt** Virtual Display compare file (App ‘4’)

**C2020\_vd\_05.txt** Virtual Display compare file (App ‘5’)

**C2020\_vd\_06.txt** Virtual Display compare file (App ‘6’)

**C2020\_vd\_07.txt** Virtual Display compare file (App ‘7’)

**C2020\_vd\_08.txt** Virtual Display compare file (App ‘8’)

**C2020\_vd\_09.txt** Virtual Display compare file (App ‘9’)

**C2020\_vd\_10.txt** Virtual Display compare file (App ‘A’)

**C2020\_vd\_11.txt** Virtual Display compare file (App ‘B’)

**C2020\_vd\_12.txt** Virtual Display compare file (App ‘C’)

**C2020\_vd\_13.txt** Virtual Display compare file (App ‘D’)

**C2020\_vd\_14.txt** Virtual Display compare file (App ‘E’)

**C2020\_vd\_15.txt** Virtual Display compare file (App ‘F’)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C2020\_log.xml** conformance report (XML format)

## Test Case Specification 3 – FPUI Character Set and Screen Attributes

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.2030.

### Objective

The objective of this Test Case is to validate the display character set and screen attribute handling capabilities of the API. This Test Case also tests the API’s AUX switch interface functions.

### Test Items

This test case tests the following requirements:

APIR3.1.1[9] The API shall support the display character set as defined in the ATC Controller Standard, Section 7.1.4 (ATC 5201 has this in Section 6.1.4).

APIR3.1.1[10] Screen attributes described by the ATC Controller Standard, Section 7.1.4 (ATC 5201 has this in Section 6.1.4), shall be maintained for each window independently.

APIR3.1.1[11] Each window shall have separate input and output buffers unique from other windows.

APIR3.1.1[12] The screen attributes of the Front Panel Manager Window shall be set to the values of the controller unit at power up as described in ATC Controller Standard, Section 7.1.4 (ATC 5201 has this in Section 6.1.4).

APIR3.1.1.2[7] The API shall provide a function or set of functions to set the attributes of a Front Panel display as described in the ATC Controller Standard, Section 7.1.4 (ATC 5201 has this in Section 6.1.4).

APIR3.1.1.2[8] The API shall provide a function or set of functions to return the attributes of a Front Panel display as described in the ATC Controller Standard, Section 7.1.4 (ATC 5201 has this in Section 6.1.4).

APIR3.1.1.2[19] If a window was registered with access to the Aux Switch, the API shall provide a function to return its status.

APIR3.1.1.2[27] Display configuration and inquiry command codes (escape sequences) specified in the ATC Controller Standard, Section 7.1.4 (ATC 5201 has this in Section 6.1.4), shall be supported as separate functions in the API.

### Input Specifications

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

File Description

**C2030\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_key0.txt** keystroke file (Key ‘0’)

**Cxxxx\_key1.txt** keystroke file (Key ‘1’)

**Cxxxx\_keyStar.txt** keystroke file (Key ‘\*’)

**Cxxxx\_keyESC.txt** keystroke file (Key ‘<Esc>’)

**C2030\_allchar.txt** parameter load file (displayable ASCII characters)

**C2030\_vd\_default.txt** Virtual Display compare file (default)

**C2030\_vd\_all.txt** Virtual Display compare file (all characters)

**C2030\_vd\_reset.txt** Virtual Display compare file (reset)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C2030\_log.xml** conformance report (XML format)

## Test Case Specification 4 – FPUI Reading and Writing Data

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.2040.

### Objective

The objective of this Test Case is to test the operation of the API functions used to write display data to and read keypresses from the Front Panel. The various write functions will be called to display data at various locations followed by a verification of the display content. Keypresses will be simulated and verified using each of the available read functions.

### Test Items

This test case tests the following requirements:

APIR3.1.1.2[9] The API shall provide a function that is used to determine if there is data in the input buffer of a window.

APIR3.1.1.2[10] The API shall provide a function to read a queued character or key code from the input buffer of a window.

APIR3.1.1.2[11] The API shall provide a function to write a character to the current cursor position of a window.

APIR3.1.1.2[12] The API shall provide a function to write a character to a window at a position defined by column and line number.

APIR3.1.1.2[13] The API shall provide a function to write a string to a window at the current cursor position.

APIR3.1.1.2[14] The API shall provide a function to write a string to a window at a starting position defined by column number and line number.

APIR3.1.1.2[15] The API shall provide a function to write a buffer of characters to a window at the current cursor position.

APIR3.1.1.2[16] The API shall provide a function to write a buffer of characters to a window at a starting position defined by column number and line number.

APIR3.1.1.2[17] The API shall provide a function to set the cursor position of a window defined by column and line number.

APIR3.1.1.2[18] The API shall provide a function to return the cursor position of the window defined by column and line number.

### Input Specifications

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

File Description

**C2040\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_key0.txt** keystroke file (Key ‘0’)

**Cxxxx\_key1.txt** keystroke file (Key ‘1’)

**Cxxxx\_keyESC.txt** keystroke file (Key ‘<Esc>’)

**C2040\_vd\_1.txt** Virtual Display compare file (display 1)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C2040\_log.xml** conformance report (XML format)

## Test Case Specification 5 – FPUI Special Characters

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.2050.

### Objective

The objective of this Test Case is to validate the API functions which support the composition and display of special characters on the LCD display.

### Test Items

This test case tests the following requirements:

APIR3.1.1.2[20] The API shall provide a function to compose special characters as described by the ATC Controller Standard, Section 7.1.4 (ATC 5201 has this in Section 6.1.4).

APIR3.1.1.2[21] The API shall support the display of a composed character in the same manner as any other valid character.

### Input Specifications

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

File Description

**C2050\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_key0.txt** keystroke file (Key ‘0’)

**C2050\_sc1.txt** parameter load file (special character 1)

**C2050\_sc2.txt** parameter load file (special character 2)

**C2050\_sc3.txt** parameter load file (special character 3)

**C2050\_sc4.txt** parameter load file (special character 4)

**C2050\_sc5.txt** parameter load file (special character 5)

**C2050\_sc6.txt** parameter load file (special character 6)

**C2050\_sc7.txt** parameter load file (special character 7)

**C2050\_sc8.txt** parameter load file (special character 8)

**C2050\_vd\_sc.txt** Virtual Display compare file (special characters)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C2050\_log.xml** conformance report (XML format)

## Test Case Specification 6 – FPUI Key Mapping

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.2070.

### Objective

The objective of this Test Case is to validate the API functions which support the handling and mapping of keypad keypresses.

### Test Items

This test case tests the following requirements:

APIR3.1.1.2[28] Application programs shall be able to interpret all ATC controller keys as individual key codes.

APIR3.1.1.2[29] The escape sequences representing keys that do not have standard ASCII character codes on an ATC controller shall be mapped to specific character codes in the API as shown in Table 1.

### Input Specifications

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

File Description

**C2070\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_key0.txt** keystroke file (Key ‘0’)

**Cxxxx\_key1.txt** keystroke file (Key ‘1’)

**Cxxxx\_key2.txt** keystroke file (Key ‘2’)

**Cxxxx\_key3.txt** keystroke file (Key ‘3’)

**Cxxxx\_key4.txt** keystroke file (Key ‘4’)

**Cxxxx\_key5.txt** keystroke file (Key ‘5’)

**Cxxxx\_key6.txt** keystroke file (Key ‘6’)

**Cxxxx\_key7.txt** keystroke file (Key ‘7’)

**Cxxxx\_key8.txt** keystroke file (Key ‘8’)

**Cxxxx\_key9.txt** keystroke file (Key ‘9’)

**Cxxxx\_keyA.txt** keystroke file (Key ‘A’)

**Cxxxx\_keyB.txt** keystroke file (Key ‘B’)

**Cxxxx\_keyC.txt** keystroke file (Key ‘C’)

**Cxxxx\_keyD.txt** keystroke file (Key ‘D’)

**Cxxxx\_keyE.txt** keystroke file (Key ‘E’)

**Cxxxx\_keyF.txt** keystroke file (Key ‘F’)

**Cxxxx\_keyUP.txt** keystroke file (Key ‘Up’)

**Cxxxx\_keyDN.txt** keystroke file (Key ‘<Dn>’)

**Cxxxx\_keyRT.txt** keystroke file (Key ‘<Rt>’)

**Cxxxx\_keyLT.txt** keystroke file (Key ‘<Lt>’)

**Cxxxx\_keyNEXT.txt** keystroke file (Key ‘<Next>’)

**Cxxxx\_keyYES.txt** keystroke file (Key ‘<Yes>’)

**Cxxxx\_keyNO.txt** keystroke file (Key ‘<No>’)

**Cxxxx\_keyStar.txt** keystroke file (Key ‘\*’)

**Cxxxx\_keyPlus.txt** keystroke file (Key ‘<+>’)

**Cxxxx\_keyMinus.txt** keystroke file (Key ‘<->’)

**Cxxxx\_keyENT.txt** keystroke file (Key ‘<Ent>’)

**Cxxxx\_keyESC.txt** keystroke file (Key ‘<Esc>’)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C2070\_log.xml** conformance report (XML format)

## Test Case Specification 7 – FPUI Asynchronous Notification and Focus

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.2080.

### Objective

The objective of this Test Case is to validate the API functions which support the handling of window focus, including asynchronous notification of same.

### Test Items

This test case tests the following requirements:

APIR3.1.1.2[32] The API shall provide an asynchronous notification to alert programs when their associated windows go in and out of focus.

APIR3.1.1.2[33] The API shall provide a function which application programs may use to determine if their window is in focus.

### Input Specifications

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

File Description

**C2080\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_key0.txt** keystroke file (Key ‘0’)

**Cxxxx\_key1.txt** keystroke file (Key ‘1’)

**Cxxxx\_keyStar.txt** keystroke file (Key ‘\*’)

**Cxxxx\_keyESC.txt** keystroke file (Key ‘<Esc>’)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C2080\_log.xml** conformance report (XML format)

## Test Case Specification 8 – FPUI Raw Data Handling

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.2090.

### Objective

The objective of this Test Case is to validate the API functions which support the sending of raw display data to the LCD display and the receiving of raw data from the keypads.

### Test Items

This test case tests the following requirements:

APIR3.1.1.2[45] The API shall provide a function to send raw output data to the display.

APIR3.1.1.2[46] If the application window is in focus, the data shall be sent to the display port without interpretation or buffering by the API.

APIR3.1.1.2[47] If the application window is not in focus, the API shall discard the data.

APIR3.1.1.2[48] The API shall provide a function to read raw input data from the display (this does not include the Aux Switch which is handled separately; see Item “c”).

APIR3.1.1.2[49] This function shall return raw data from the input buffer without the key code interpretation described in item “y”.

### Input Specifications

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

File Description

**C2090\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_key0.txt** keystroke file (Key ‘0’)

**Cxxxx\_keyYES.txt** keystroke file (Key ‘<Yes>’)

**C2090\_vd\_blank.txt** Virtual Display compare file (blank display)

**C2090\_vd\_raw.txt** Virtual Display compare file (display raw data)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C2090\_log.xml** conformance report (XML format)

## Test Case Specification 9 – API Version Information (All Libraries)

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.2100.

### Objective

The objective of this Test Case is to validate the API functions which retrieve the version numbers of the three distinct API libraries (FPUI, FIO and TOD).

### Test Items

This test case tests the following requirements:

APIR3.2[1] The API shall provide a method to determine the version number(s) of the API.

### Input Specifications

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

File Description

**C2100\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_null.txt** parameter load file (null data)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C2100\_log.xml** conformance report (XML format)

## Test Case Specification 10 – General FIO Operations

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3010.

### Objective

The objective of this Test Case is to validate the API functions which support the general channel and device configuration for Field I/O devices. Three individual and unique configurations of FIO devices will be registered, configured, enabled for communications and tested.

### Test Items

This test case tests the following requirements:

APIR3.1.2[4] The API shall not open any serial communications port or initiate communications to any Field I/O Device unless explicitly commanded to do so by an application program.

APIR3.1.2[5] The API shall support all cabinet architectures and associated Field I/O Device types as listed in the ATC Controller Standard Section 8.

APIR3.1.2[6] The API shall support the Field I/O Device types shown in Table 2.

APIR3.1.2[8] The API shall support communication to multiple Field I/O Devices on a single communications port provided the Field I/O Devices have compatible physical communication attributes.

APIR3.1.2[9] The API shall support a maximum of one Field I/O Device of each type per communications port except in the case of BIUs and SIUs.

APIR3.1.2[10] The API shall support up to 8 Detector BIU and 8 Terminal & Facilities BIU Field I/O Devices per communications port.

APIR3.1.2[11] The API shall support up to 5 Input SIU, 2 14-Pack Output SIU and 4 6-Pack Output SIU Field I/O Devices per communications port.

APIR3.1.2[12] The API shall only support valid Output SIU combinations as defined in the ITS Cabinet Standard, Section 4.7.

APIR3.1.2[13] The API shall identify specific Field I/O Devices using the API Field I/O Device Names in Table 2.

APIR3.1.2[14] The API shall provide a method for application programs to register and deregister with the API for access to the API Field I/O services.

APIR3.1.2[15] The process of application program registration shall not cause the API to perform any communications with the Field I/O Device.

APIR3.1.2[16] When an application program deregisters for access to Field I/O services, the API shall deregister (as defined in Item “e”) all Field I/O devices registered by that application program.

APIR3.1.2[17] The API shall provide a method to allow application programs to register and deregister for access to specific Field I/O Devices by specifying the communications port, device type, and where applicable, the Field I/O Device number.

APIR3.1.2[18] Once a device has been registered on a communications port, the API shall permit the registration of additional compatible Field I/O Devices on the same communications port and prohibit the registration of incompatible Field I/O Devices on the same communications port.

APIR3.1.2[19] The Field I/O Device registration process shall not cause the API to perform any device communications.

APIR3.1.2[20] When an application program deregisters for access to a Field I/O Device, the API shall disable (as defined in Item “g”) the Field I/O Device, relinquish all output points for that device and set all application program settable states to their default values.

APIR3.1.2[21] The API shall provide a method for application programs to query for the presence of a Field I/O Device using the communications port, device type, and where applicable, the Field I/O Device number.

APIR3.1.2[22] If the API does not have the communications port open at the time of the query and it is necessary for the API to open the communications port to determine the Field I/O Device, the API shall close the communications port after the query is completed.

APIR3.1.2[23] If the API has the communications port open at the time of the query and the communications attributes for the Field I/O Device used in the query are not compatible with the current settings on the communications port, the API shall assume that the Field I/O Device is not present.

APIR3.1.2[24] If the API has the communications port open at the time of the query and API is already successfully completing scheduled communications to the Field I/O Device, the API shall indicate that the Field I/O Device is present without sending any additional frames to the device.

APIR3.1.2[25] The API shall provide a method which allows an application program to enable and disable communications to a Field I/O Device for which the application program has registered.

APIR3.1.2[26] When the communications enable method is called, the API shall initiate scheduled communications between the API and the specified Field I/O Device if not already active.

APIR3.1.2[27] When the disable communications method is called, the API shall cease scheduled communications between the API and the specified Field I/O Device if the device is no longer enabled by any application program.

### Input Specifications

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

File Description

**C3010\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_rsp128.txt** FIO Response Message data (Frame 128)

**Cxxxx\_rsp129.txt** FIO Response Message data (Frame 129)

**Cxxxx\_rsp131.txt** FIO Response Message data (Frame 131)

**Cxxxx\_rsp138.txt** FIO Response Message data (Frame 138)

**Cxxxx\_rsp148.txt** FIO Response Message data (Frame 148)

**Cxxxx\_rsp149.txt** FIO Response Message data (Frame 149)

**Cxxxx\_rsp150.txt** FIO Response Message data (Frame 150)

**Cxxxx\_rsp151.txt** FIO Response Message data (Frame 151)

**Cxxxx\_rsp177.txt** FIO Response Message data (Frame 177)

**Cxxxx\_rsp181a.txt** FIO Response Message data (Frame 181)

**Cxxxx\_rsp181b.txt** FIO Response Message data (Frame 181)

**Cxxxx\_rsp183.txt** FIO Response Message data (Frame 183)

**Cxxxx\_rsp195.txt** FIO Response Message data (Frame 195)

**Cxxxx\_cmd49.txt** FIO Command Message data (Frame 49)

**Cxxxx\_cmd49a.txt** FIO Command Message data (Frame 49)

**Cxxxx\_cmd49b.txt** FIO Command Message data (Frame 49)

**Cxxxx\_cmd53.txt** FIO Command Message data (Frame 53)

**Cxxxx\_cmd55a.txt** FIO Command Message data (Frame 55)

**Cxxxx\_cmd55b.txt** FIO Command Message data (Frame 55)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3010\_log.xml** conformance report (XML format)

## Test Case Specification 11 – FIO Inputs and Outputs

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3020.

### Objective

The objective of this Test Case is to validate the API functions which support the management of input and output points on Field I/O devices. Multiple applications will be instantiated during the test in order to confirm the sharing of input and output data for reading and the exclusive nature of output points registered for writing (setting).

### Test Items

This test case tests the following requirements:

APIR3.1.2[28] When a Field I/O Device is disabled, any output points which have been reserved by that application program shall be set to Off.

APIR3.1.2[29] The API shall provide a method for application programs to read the states of the input and output points on registered Field I/O Devices, including both filtered and non-filtered states for the input points (depending on which input frames are scheduled).

APIR3.1.2[30] If multiple application programs have registered for the same Field I/O Device, the API shall provide shared read access to the input and output point states for all application programs which have registered that device.

APIR3.1.2[31] When the state of an output point is read, the API shall return the current state of that output point within the API.

APIR3.1.2[32] The API shall provide a method for application programs to reserve/relinquish exclusive “write access” to individual output points of a Field I/O Device.

APIR3.1.2[33] If an application program reserves a point that has already been reserved by that application program, it shall not be considered an error.

APIR3.1.2[34] If an application program relinquishes a point that is already in the relinquished state for that application program, it shall not be considered an error.

APIR3.1.2[35] If a point in a group of points cannot be reserved, the reservation attempt shall fail for all of them.

APIR3.1.2[36] The API shall allow only one application program to reserve write access to any individual output point.

APIR3.1.2[37] The API shall allow multiple application programs to reserve different output points on a single Field I/O Device.

APIR3.1.2[38] Exclusive reservation of an output point for write access by one application program shall not preclude other application programs from reading the state of the output point.

APIR3.1.2[39] The API shall provide error codes so that the application program can determine if the reservation action was successful or if there was a conflict with another application program.

APIR3.1.2[40] The API shall make output point reservations on a “first come first served basis.”

APIR3.1.2[41] An application program shall be able to set the state of an output point if it has registered the associated Field I/O Device and reserved exclusive write access to the output point.

APIR3.1.2[43] The API shall provide a method for application programs to query the reservation status of output points on registered Field I/O Devices.

### Input Specifications

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

File Description

**C3020\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_rsp177.txt** FIO Response Message data (Frame 177)

**Cxxxx\_rsp180a.txt** FIO Response Message data (Frame 180)

**Cxxxx\_rsp181a.txt** FIO Response Message data (Frame 181)

**Cxxxx\_rsp183.txt** FIO Response Message data (Frame 183)

**C3020\_cmd55a.txt** FIO Command Message data (Frame 55)

**C3020\_cmd55b.txt** FIO Command Message data (Frame 55)

**C3020\_outputs\_null.txt** parameter load file (null outputs)

**C3020\_outputs\_a.txt** parameter load file (outputs)

**C3020\_outputs\_b.txt** parameter load file (outputs)

**C3020\_outputs\_c.txt** parameter load file (outputs)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3020\_log.xml** conformance report (XML format)

## Test Case Specification 12 – FIO Channel Mapping

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3030.

### Objective

The objective of this Test Case is to validate the API functions which support the management of output channels on MMU and CMU Field I/O devices. Multiple applications will be instantiated during the test in order to confirm the sharing of devices and the exclusive nature of output channels registered for exclusive access.

### Test Items

This test case tests the following requirements:

APIR3.1.2[44] The API shall provide a method for application programs to map/unmap reserved output points to reserved channels and colors on a registered FIOMMU or FIOCMU device.

APIR3.1.2[45] The API shall use this mapping to set the contents of FIOMMU Frame 0 and FIOCMU Frames 61 and 67.

APIR3.1.2[46] Any channel and color not mapped to an output point shall be set to Off.

APIR3.1.2[47] The API shall provide a method for application programs to reserve/relinquish exclusive control of individual monitored channels on the FIOMMU or FIOCMU device.

APIR3.1.2[48] If an application program reserves a channel that has already been reserved by that application program, it shall not be considered an error.

APIR3.1.2[49] If an application program relinquishes a channel that is already in the relinquished state for that application program, it shall not be considered an error.

APIR3.1.2[50] If a channel in a group of channels cannot be reserved, the reservation attempt shall fail for all of them.

APIR3.1.2[51] The API shall allow multiple applications to reserve different channels on a single FIOMMU or FIOCMU device.

APIR3.1.2[52] The API shall provide error codes so that the application program can determine if the reservation action was successful or if there was a conflict with another application.

APIR3.1.2[53] The API shall make channel reservations on a “first come first served basis.”

APIR3.1.2[54] The API shall provide a method for applications to query the reservation status of channels on registered FIOMMU or FIOCMU devices.

APIR3.1.2[55] Relinquishing a reserved output point or channel shall clear the associated assignments.

### Input Specifications

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

File Description

**C3030\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_rsp177.txt** FIO Response Message data (Frame 177)

**Cxxxx\_rsp183.txt** FIO Response Message data (Frame 183)

**Cxxxx\_rsp195.txt** FIO Response Message data (Frame 195)

**C3020\_cmd55b.txt** FIO Command Message data (Frame 55)

**C3030\_cmd67a.txt** FIO Command Message data (Frame 67)

**C3020\_outputs\_null.txt** parameter load file (null outputs)

**C3020\_outputs\_a.txt** parameter load file (outputs)

**C3020\_outputs\_c.txt** parameter load file (outputs)

**C3030\_channels\_null.txt** parameter load file (null channels)

**C3030\_channels\_a.txt** parameter load file (channels)

**C3030\_channels\_b.txt** parameter load file (channels)

**C3030\_channels\_c.txt** parameter load file (channels)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3030\_log.xml** conformance report (XML format)

## Test Case Specification 13 – FIO Filtered Inputs and Transition Buffering

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3040.

### Objective

The objective of this Test Case is to validate the API functions which support the filtering of input data and the management of input transition monitoring on Field I/O devices. Multiple applications will be instantiated during the test in order to confirm the sharing of input transition entries between applications.

### Test Items

This test case tests the following requirements:

APIR3.1.2[56] The API shall provide functions which allow application programs to set and get the leading and trailing edge filter values on a per input basis for all Field I/O Devices that support configurable filtered inputs.

APIR3.1.2[57] If multiple application programs set the filter values of an input, the shortest filter values shall be used.

APIR3.1.2[58] The API shall provide a return code containing the status and the value used for the set filter operation.

APIR3.1.2[59] The default leading and trailing edge filter values shall be 5 consecutive samples.

APIR3.1.2[60] The API shall have the ability to collect and buffer the transition buffer information for each registered Field I/O Device used for input.

APIR3.1.2[61] When the API reads the transition buffer of a Field I/O Device, it shall read the entire transition buffer.

APIR3.1.2[62] The API shall buffer the transition data on a per application program basis with the capability of storing 1024 transition entries in a FIFO fashion.

APIR3.1.2[63] The API shall provide a function which allows application programs to enable or disable transition monitoring of selected input points.

APIR3.1.2[64] By default, transition monitoring for all input points shall be disabled.

APIR3.1.2[65] If an application program enables an input point for transition monitoring and that input point is already in the enabled state, it shall not be considered an error.

APIR3.1.2[66] If an application program disables an input point for transition monitoring and that input point is already in the disabled state, it shall not be considered an error.

APIR3.1.2[67] The API shall provide functions that allow application programs to access the API transition buffer information asynchronously (i.e. read the transition entries from the API buffer independent of any Field I/O Device communications).

APIR3.1.2[68] When an application program reads a transition entry from an API transition buffer, that transition entry shall be cleared for that application program only, without affecting the API transition buffers for other application programs.

APIR3.1.2[69] If the transition buffer in the Field I/O Device overruns before information can be copied to the API transition buffer information, the API shall indicate that a device overrun condition has occurred in the transition buffer for that Field I/O Device.

APIR3.1.2[70] If the transition buffer of the API overruns before the information is retrieved by the application program, the API shall indicate that an API overrun condition has occurred.

### Input Specifications

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

File Description

**C3040\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_rsp177.txt** FIO Response Message data (Frame 177)

**Cxxxx\_rsp180a.txt** FIO Response Message data (Frame 180)

**Cxxxx\_rsp181a.txt** FIO Response Message data (Frame 181)

**C3040\_rsp182a.txt** FIO Response Message data (Frame 182)

**C3040\_rsp182b.txt** FIO Response Message data (Frame 182)

**C3040\_rsp182c.txt** FIO Response Message data (Frame 182)

**C3040\_rsp182d.txt** FIO Response Message data (Frame 182)

**C3040\_rsp182e.txt** FIO Response Message data (Frame 182)

**Cxxxx\_rsp183.txt** FIO Response Message data (Frame 183)

**C3040\_inputs\_null.txt** parameter load file (null inputs)

**C3040\_inputs\_a.txt** parameter load file (inputs)

**C3040\_inputs\_b.txt** parameter load file (inputs)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3040\_log.xml** conformance report (XML format)

## Test Case Specification 14 – FIO Frame Frequency

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3050.

### Objective

The objective of this Test Case is to validate the API functions which allow the user to specify the frequency with which individual Command Frames are issued to their respective Field I/O devices.

### Test Items

This test case tests the following requirements:

APIR3.1.2[71] The ATC Controller Standard, Section 8, specifies the frames for communication with Field I/O Devices for Model 332 Cabinets, NEMA TS 1 and TS 2 Type 2 Cabinets and ITS Cabinets. The API shall support a subset of these frames at the scheduled frame frequencies as shown in Table 3.

APIR3.1.2[72] The NEMA TS 2 Standard, Section 3.3, specifies the frames for communication with Field I/O Devices for NEMA TS 2 Type 1 Cabinets. The API shall support a subset of these frames at the scheduled frame frequencies as shown in Table 4.

APIR3.1.2[74] The API shall provide a method for application programs to set/get the scheduled frame frequencies for a registered Field I/O Device.

APIR3.1.2[75] The frame frequency used by the API shall be the highest frequency requested by all application programs registered for that Field I/O Device.

APIR3.1.2[76] The API shall provide a method to send a frame from either Table 3 or Table 4 one time (non-scheduled).

### Input Specifications

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

File Description

**C3050\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_rsp128.txt** FIO Response Message data (Frame 128)

**Cxxxx\_rsp129.txt** FIO Response Message data (Frame 129)

**Cxxxx\_rsp131.txt** FIO Response Message data (Frame 131)

**Cxxxx\_rsp138.txt** FIO Response Message data (Frame 138)

**Cxxxx\_rsp139.txt** FIO Response Message data (Frame 139)

**Cxxxx\_rsp140.txt** FIO Response Message data (Frame 140)

**Cxxxx\_rsp141.txt** FIO Response Message data (Frame 141)

**Cxxxx\_rsp148.txt** FIO Response Message data (Frame 148)

**Cxxxx\_rsp149.txt** FIO Response Message data (Frame 149)

**Cxxxx\_rsp150.txt** FIO Response Message data (Frame 150)

**Cxxxx\_rsp151.txt** FIO Response Message data (Frame 151)

**Cxxxx\_rsp177.txt** FIO Response Message data (Frame 177)

**Cxxxx\_rsp181a.txt** FIO Response Message data (Frame 181)

**Cxxxx\_rsp183.txt** FIO Response Message data (Frame 183)

**Cxxxx\_rsp188a.txt** FIO Response Message data (Frame 188)

**Cxxxx\_rsp195.txt** FIO Response Message data (Frame 195)

**Cxxxx\_cmd60.txt** FIO Command Message data (Frame 60)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3050\_log.xml** conformance report (XML format)

## Test Case Specification 15 – FIO Failed State and Fault Monitoring

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3060.

### Objective

The objective of this Test Case is to validate the API methods for setting and clearing various application-initiated “fault” conditions related to CMU devices (Failed State Action) and TS1 and TS2 FIO devices (Fault Monitor and Voltage Monitor).

### Test Items

This test case tests the following requirements:

APIR3.1.2[77] The API shall provide a method to set/get the Failed State Action of a FIOCMU Field I/O Device.

APIR3.1.2[78] The Failed State Action shall be settable to None (LFSA=0, NFSA=0), Latched (LFSA=1, NFSA=0), or Non Latched (LFSA=0, NFSA=1).

APIR3.1.2[79] The default Failed State Action shall be None.

APIR3.1.2[80] If any application program sets the state to Latched, the API shall set the Failed State Action to Latched.

APIR3.1.2[81] If no application program has set the Failed State Action to Latched, then if any application program sets the state to Non Latched, the API shall set the Failed State Action to Non Latched.

APIR3.1.2[82] If all application programs have a state of None, then the API shall set the Failed State Action to None.

APIR3.1.2[83] The API shall provide a method to set/get the state of the Fault Monitor output point of FIOTS1 and FIOTS2 Field I/O Devices.

APIR3.1.2[84] The API shall retain ownership of the Fault Monitor output point and not allow application programs to reserve this output point.

APIR3.1.2[85] If any application program sets the Fault Monitor state to Off, the API shall turn Off the Fault Monitor output point on that device.

APIR3.1.2[86] If all application programs have a Fault Monitor state of On for a FIOTS1 or FIOTS2 Device, then the API shall turn On the Fault Monitor output point on that device.

APIR3.1.2[87] The default state of the Fault Monitor output point shall be On.

APIR3.1.2[88] The API shall provide a method to set/get the state of the Voltage Monitor output point of a FIOTS1 Field I/O Device.

APIR3.1.2[89] The API shall retain ownership of the Voltage Monitor output point and not allow application programs to reserve this output point.

APIR3.1.2[90] If any application program sets the Voltage Monitor state to Off, the API shall turn Off the Voltage Monitor output point on that device.

APIR3.1.2[91] If all application programs have a Voltage Monitor state of On for a FIOTS1 Device, then the API shall turn On the Voltage Monitor output point on that device.

APIR3.1.2[92] The default state of the Voltage Monitor output point shall be On.

### Input Specifications

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

File Description

**C3060\_in.xml** APIVSXML test script (XML format)

**C3060\_outputs\_fm.txt** parameter load file (FM output)

**C3060\_outputs\_vm.txt** parameter load file (VM output)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3060\_log.xml** conformance report (XML format)

## Test Case Specification 16 – FIO Watchdog Outputs

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3070.

### Objective

The objective of this Test Case is to validate the API methods for managing Watchdog outputs on FIO devices, including ownership assignment and toggling responsibilities.

### Test Items

This test case tests the following requirements:

APIR3.1.2[93] The API shall provide a method which allows application programs to assign the output point used for the Watchdog output of any registered Field I/O Device.

APIR3.1.2[94] The API shall restrict the ability to assign the Watchdog output point to the first application program to call the assignment method.

APIR3.1.2[95] The API shall retain ownership of the Watchdog output point and not allow application programs to reserve that output point directly.

APIR3.1.2[96] The API shall provide a method for application programs to register for shared control of the Watchdog output point.

APIR3.1.2[97] The API shall provide a method for Watchdog registered application programs to “request” that the API toggle the state of the Watchdog output point.

APIR3.1.2[98] The API shall only toggle the Watchdog output point if all Watchdog registered application programs have made the toggle request (Watchdog Triggered Condition).

APIR3.1.2[99] Upon a Watchdog Triggered Condition, the API shall toggle the state of the Watchdog output point within the API.

APIR3.1.2[100] When the API updates the output states of the Field I/O Device (see Item “n”), the API shall clear all previous toggle requests and the Watchdog Triggered Condition so that a new Watchdog Triggered Condition can be generated. fio\_fiod\_wd\_heartbeat(3fio).

APIR3.1.2[101] The API shall not toggle the Watchdog output point more than once per update of the output states on the Field I/O Device.

### Input Specifications

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

File Description

**C3070\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_rsp177.txt** FIO Response Message data (Frame 177)

**Cxxxx\_rsp181a.txt** FIO Response Message data (Frame 181)

**Cxxxx\_rsp183.txt** FIO Response Message data (Frame 183)

**C3070\_outputs\_wd.txt** parameter load file (watchdog outputs)

**C3070\_cmd55a.txt** FIO Command Message data (Frame 55)

**C3070\_cmd55b.txt** FIO Command Message data (Frame 55)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3070\_log.xml** conformance report (XML format)

## Test Case Specification 17 – FIO Device Status

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3080.

### Objective

The objective of this test case is to test the operation of the API functions used to retrieve and reset device status information from a registered Field I/O Device. A device will be registered and allowed to communicate for a period of time, after which the status elements will be examined and then reset. .

### Test Items

This test case tests the following requirements:

APIR3.1.2[102] The API shall provide functions which allow application programs to obtain status information of a registered Field I/O Device.

APIR3.1.2[105] The API shall provide the following communication status information for each registered Field I/O Device:

i) Communications Enabled/Disabled;

ii) Cumulative successful response count for all frames to this device;

iii) Cumulative error count for all frames to this device; and

iv) Command frames sent to this device with the following information for each frame type: current scheduled frequency, cumulative successful response count, cumulative error count, numbers of errors in the last 10 frames, a response frame sequence number, frame size in bytes and the raw data from the most recent response frame.

APIR3.1.2[107] The API shall provide a method for application programs to reset the communications status counters to 0 (zero) for a registered Field I/O Device.

APIR3.1.2[108] A response frame shall only be considered successful if it is fully received within the time period defined by the “Handshaking” algorithm in Section 3.3.1.5.3 of the NEMA TS 2 Standard.

### Input Specifications

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

File Description

**C3080\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_rsp128.txt** FIO Response Message data (Frame 128)

**Cxxxx\_rsp129.txt** FIO Response Message data (Frame 129)

**Cxxxx\_rsp131.txt** FIO Response Message data (Frame 131)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3080\_log.xml** conformance report (XML format)

## Test Case Specification 18 – FIO Health Monitor

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3090.

### Objective

The objective of this Test Case is to validate the API Health Monitor Function, which can be used by registered Field I/O applications to disable all FIO outputs registered by that application should it become non-responsive for a time period longer than it’s Health Monitor Timeout.

### Test Items

This test case tests the following requirements:

APIR3.1.2[109] The API shall provide an API Health Monitor Function which registered application programs use to indicate to the API that they are operational.

APIR3.1.2[110] The API shall provide a method to set an API Health Monitor Timeout for each application program (each application program has its own unique API Health Monitor Timeout).

APIR3.1.2[111] This API Health Monitor Timeout shall indicate the maximum allowable time between calls to the API Health Monitor Function.

APIR3.1.2[112] The API Health Monitor Timeout shall be specified in tenths of a second.

APIR3.1.2[113] If the API Health Monitor Timeout expires for an application, the API shall disable (as defined previously in Item “g”) all Field I/O Devices registered by that application program.

APIR3.1.2[114] The API shall provide a method for an application program to disable the API Health Monitor feature for itself.

APIR3.1.2[115] The API shall provide a method for an application program to reset an API Health Monitor fault condition and allow the API to resume Field I/O Device communications.

APIR3.1.2[116] An application shall only be able to reset its own Health Monitor fault condition and not that of any other application program.

APIR3.1.2[117] If an application program resets the API Health Monitor fault condition, then any devices that were disabled due to that condition shall be re-enabled.

APIR3.1.2[118] If an application program attempts to enable a device (as defined in Item “g”) that has been disabled due to an API Health Monitor fault condition, then the enable operation shall return an error and the Field I/O Device remain disabled.

APIR3.1.2[119] A call to the API Health Monitor Function after a Health Monitor fault has occurred shall not reset the Health Monitor fault condition.

APIR3.1.2[120] The API Health Monitor Function shall return whether an API Health Monitor fault condition exists.

### Input Specifications

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

File Description

**C3090\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_rsp177.txt** FIO Response Message data (Frame 177)

**Cxxxx\_rsp181a.txt** FIO Response Message data (Frame 181)

**Cxxxx\_rsp183.txt** FIO Response Message data (Frame 183)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3090\_log.xml** conformance report (XML format)

## Test Case Specification 19 – FIO CMU Configuration

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3100.

### Objective

The objective of this Test Case is to validate the API method for retrieving Datakey configuration information from a CMU device.

### Test Items

This test case tests the following requirements:

APIR3.1.2[121] The API shall provide a method for an application program to send the Get CMU Configuration frame to a registered FIOCMU device.

### Input Specifications

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

File Description

**C3100\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_rsp193.txt** FIO Response Message data (Frame 193)

**Cxxxx\_cmd65.txt** FIO Command Message data (Frame 65)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3100\_log.xml** conformance report (XML format)

## Test Case Specification 20 – FIO Module Status

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3110.

### Objective

The objective of this Test Case is to validate the API methods for managing Module Status bits from FIO devices which use such status bits.

### Test Items

This test case tests the following requirements:

APIR3.1.2[122] The API shall reset all Module Status bits using the Request Module Status frame when a FIO332, FIOTS1, FIOTS2 or SIU device is first Enabled (as defined in Item “g”).

APIR3.1.2[123] Anytime a response to a Request Module Status frame has Module Status bits indicating hardware reset, comm loss, or watchdog reset, then the API shall clear those bits, reset the input point filter values (Item “k”) and reconfigure transition reporting (Item “l”).

### Input Specifications

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

File Description

**C3110\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_rsp177.txt** FIO Response Message data (Frame 177)

**Cxxxx\_rsp177a.txt** FIO Response Message data (Frame 177)

**Cxxxx\_rsp177b.txt** FIO Response Message data (Frame 177)

**Cxxxx\_rsp177c.txt** FIO Response Message data (Frame 177)

**Cxxxx\_cmd49a.txt** FIO Command Message data (Frame 49)

**Cxxxx\_cmd49b.txt** FIO Command Message data (Frame 49)

**Cxxxx\_cmd49c.txt** FIO Command Message data (Frame 49)

**Cxxxx\_cmd49d.txt** FIO Command Message data (Frame 49)

**Cxxxx\_cmd49e.txt** FIO Command Message data (Frame 49)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3110\_log.xml** conformance report (XML format)

## Test Case Specification 21 – FIO Asynchronous Notification

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3120.

### Objective

The objective of this Test Case is to validate the API asynchronous notification methods which are used by applications to signal the receipt of a response frame from a FIO device.

### Test Items

This test case tests the following requirements:

APIR3.1.2[124] The API shall provide a method to notify an application program when a command frame is acknowledged (response frame received by the API) or when an error occurs.

APIR3.1.2[125] The command frame shall be identified by the frame type and a registered Field I/O Device.

APIR3.1.2[126] The response frame notification shall include the Field I/O Device, response frame type, response frame sequence number, response frame size in bytes and an indication as to why the notification occurred (response received or error detected).

APIR3.1.2[127] The notification shall be able to be set for a one time occurrence or continuous occurrence.

### Input Specifications

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

File Description

**C3120\_in.xml** APIVSXML test script (XML format)

**Cxxxx\_rsp177.txt** FIO Response Message data (Frame 177)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3120\_log.xml** conformance report (XML format)

## Test Case Specification 22 – FIO Dark Channel Mapping

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.3130.

### Objective

The objective of this Test Case is to validate the API methods which permit the configuration of the Dark Channel Mask on a CMU device and the Load Switch Flash bit on an MMU device.

### Test Items

This test case tests the following requirements:

APIR3.1.2[128] The API shall provide a method to set and get the Dark Channel Map selection for a registered FIOCMU device.

APIR3.1.2[129] If multiple application programs attempt to set the Dark Channel Map selection, the API shall use the most recent selection.

APIR3.1.2[130] The default value of the Dark Channel Map Select bits shall be 0 (Mask #1).

APIR3.1.2[131] The API shall provide a method to set and get the state of the Load Switch Flash bit of a registered FIOMMU device.

APIR3.1.2[132] If multiple application programs attempt to set the state of the Load Switch Flash bit, the API shall use the most recent state.

APIR3.1.2[133] The default value of the Load Switch Flash bit shall be 0.

### Input Specifications

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

File Description

**C3130\_in.xml** APIVSXML test script (XML format)

**C3130\_cmd67b.txt** FIO Command Message data (Frame 67)

**C3130\_cmd67c.txt** FIO Command Message data (Frame 67)

**C3130\_cmd0a.txt** FIO Command Message data (Frame 0)

**C3130\_cmd0b.txt** FIO Command Message data (Frame 0)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C3130\_log.xml** conformance report (XML format)

## Test Case Specification 23 – TOD Time Handling Functions

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.4010.

### Objective

The objective of this Test Case is to validate the API’s time-handling functions by using them to asjust the controller’s clock and verifying the result.

### Test Items

This test case tests the following requirements:

APIR3.2.2[10] The API shall provide a function(s) to set and get the system time including the date, time, time zone, and DST information.

APIR3.2.2[11] The system time function(s) shall not require "root" permissions to operate..

### Input Specifications

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

File Description

**C4010\_in.xml** APIVSXML test script (XML format)

**VS\_config\_1.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C4010\_log.xml** conformance report (XML format)

## Test Case Specification 24 – Front Panel Manager and ATC Configuration Menu

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.6010.

### Objective

The objective of this Test Case is to validate certain functional and navigational API requirements for the ATC Front Panel Manager window and the ATC Configuration Menu.

### Test Items

This test case tests the following requirements:

APIR3.1.1[13] The API shall provide an additional window referred to as the "ATC Configuration Window" for use by system configuration utility programs.

APIR3.1.1.1[1] The API shall provide a window selection screen called the Front Panel Manager Window from which Operational Users may select a window to have focus.

APIR3.1.1.1[16] The Operational User shall be able to put the ATC Configuration Window in focus by pressing {<NEXT>} in the Front Panel Manager Window.

APIR3.1.1.1[17] The top two lines and bottom line of the Front Panel Manager Window shall be fixed as shown in Figure 7.

APIR3.1.1.1[18] The number of lines between the second line and bottom lines used for displaying window names shall vary according to the size of the ATC display.

APIR3.1.1.1[21] If the Operational User attempts to put a window in focus that that does not have an application window assigned either by pressing {[0-F]} while the Front Panel Manager Window is in focus or {\*\*,[0-F]} while any other window is in focus, the API shall activate the bell of the controller unit (see Section 7.1.4 of the ATC Controller Standard) (ATC 5201 has this in Section 6.1.4) and the {[0-F]} or {\*\*,[0-F]} shall be ignored.

APIR3.1.1.1[22] If the user presses any key that is undefined in the context of the Front Panel Manager Window, the API shall activate the bell of the controller unit and the key shall be ignored.

APIR3.2.1[1] The API shall provide Operational User the ability to view or configure system-wide parameters of the ATC controller unit using the ATC Configuration Window.

APIR3.2.1[2] The display size of the ATC Configuration Window shall be consistent with the windows of the Front Panel Manager (see Section 3.1.1).

APIR3.2.1[3] The API shall provide a menu ("Configuration Menu") of up to 16 items ("configuration items") which can be selected by the Operational User from the keypad of the Front Panel display.

APIR3.2.1[4] When the associated key of a configuration item is pressed, a configuration utility program ("configuration utility") shall be executed to view or configure the parameters associated with the configuration item.

APIR3.2.1[5] The configuration utility shall use the ATC Configuration Window as its user interface display and receive the characters input from the Front Panel input device.

APIR3.2.1[6] Only one configuration utility shall be operational at a time.

APIR3.2.1[7] The configuration items shall be limited to 16 characters.

APIR3.2.1[8] If there is no configuration item associated to a key, the configuration item shall be listed as a blank.

APIR3.2.1[9] There shall be five configuration items defined in the API standard as follows: "System Time," "Ethernet," "System Services," "Linux/API Info" and "Host EEPROM Info".

APIR3.2.1[11] The Configuration Menu shall have the format shown in Figure 9.

APIR3.2.1[12] The Configuration Menu shall be displayed in the ATC Configuration Window when there is no configuration utility in operation.

APIR3.2.1[13] When the API displays the Configuration Menu, the screen attributes of the ATC Configuration Window shall be set to the values of the controller unit at power up as described in ATC Controller Standard, Section 7.1.4 (ATC 5201 has this in Section 6.1.4).

APIR3.2.1[14] The Operational User shall be able to put the ATC Configuration Window in focus from any application program or the Front Panel Manager Window by pressing {\*\*,<NEXT>} (\*\* is as defined in Section 3.1.1.1 Item "e") from the keypad on the controller’s Front Panel.

APIR3.2.1[15] The Operational User shall have the capability to select a configuration utility by pressing the corresponding key [0-F] from the Configuration Menu.

APIR3.2.1[16] The top two lines and bottom line of the Configuration Menu shall be fixed as shown in Figure 9.

APIR3.2.1[17] The number of lines between the second line and bottom line used for displaying the configuration items shall vary according to the size of the ATC Front Panel display.

APIR3.2.1[18] The Operational User shall be able to scroll the Configuration Menu up and down one line at a time to view the configuration items using the up and down arrow keys of the controller keypad.

APIR3.2.1[19] The Operational User shall be able to put the Front Panel Manager Window in focus by pressing {<NEXT>} in the ATC Configuration Menu.

APIR3.2.1[20] The only possible selections from the ATC Configuration Menu shall be itself, the Front Panel Manager Window (see Section 3.1.1.1), a window selected by pressing {\*\*,[0-F]} or a configuration item.

APIR3.2.1[21] The API shall provide a mechanism for Operational Users to modify configuration fields as identified in the requirements of the configuration utilities (see Section 3.2.1.1 through Section 3.2.1.5).

APIR3.2.1[22] The user shall be able to move the cursor from field to field using the left and right arrow keys from the keypad of the controller's Front Panel.

APIR3.2.1[23] If a right arrow is pressed, the cursor shall jump to the next configurable field to the right or, if there is no configurable field to the right, it will jump to the first configurable field of the next downward line that contains a configurable field.

APIR3.2.1[24] If there are no configurable fields to the right or downward, then the right arrow key shall have no effect.

APIR3.2.1[25] If a left arrow is pressed, the cursor shall jump to the next configurable field to the left or, if there is no configurable field to the left, it will jump to the last configurable field of the next upward line that contains a configurable field.

APIR3.2.1[26] If there are no configurable fields to the left or upward, then the left arrow key shall have no effect.

APIR3.2.1[27] The user shall be able to apply the values of fields modified by the user by pressing the <ENT> key.

APIR3.2.1[28] When the <ENT> key is pressed, the configuration utility shall remain in focus.

APIR3.2.1[29] When the cursor is over the value of a configurable field, the user shall then be able to scroll continuously through all of the possible values of the field using the + and – keys.

APIR3.2.1[30] When the user finds the desired value, the user shall be able set the value by pressing the <YES> key, by pushing arrow (left, right, up or down) keys, or by pushing the <ENT> key.

APIR3.2.1[31] If the user presses the <NO> key while scrolling through values of a field, the field shall revert to the value prior to pushing the + and – key.

APIR3.2.1[32] In addition to the method described in (ii), when the cursor is over the value of a configurable field that is made up of one or more digits 0-9, the user shall be able to enter the value using the keys 0-9 from the keypad of the controller's Front Panel display.

APIR3.2.1[33] Numeric values as entered shall be displayed right justified within the field.

APIR3.2.1[34] If while entering digits, the user enters more digits than the field will hold, the left most digit shall be removed from the field.

APIR3.2.1[35] The user shall be able to set the value by pressing the <YES> key by pushing arrow keys (left, right, up or down), or by pushing the <ENT> key.

APIR3.2.1[36] If the user presses the <NO> key while modifying a field, the field shall revert to the value prior to pushing a numerical key.

APIR3.2.1[37] If the user edits a field and enters a value that is out of range, the API shall activate the bell of the controller unit (see Section 7.1.4 (ATC 5201 has this in Section 6.1.4) of the ATC Controller Standard) and the field shall revert to the value prior to editing.

APIR3.2.1[38] If the user enters a keyboard character that cannot be interpreted under the context of the ATC Configuration Window or active configuration utility, the API shall activate the bell of the controller unit and the invalid key shall be ignored.

APIR3.2.1[39] The Operational User shall quit a configuration utility by pressing {\*\*,<NEXT>}.

APIR3.2.1[40] If the configuration utility terminates in 2 seconds or less, the API shall display the Configuration Menu.

APIR3.2.1[41] If a configuration utility does not terminate within the 2 second period, the user shall be provided an option to terminate the configuration utility as shown in Figure 10.

APIR3.2.1[42] If the user presses <YES>, the API shall terminate the configuration utility using the Linux SIGTERM signal then display the Configuration Menu.

APIR3.2.1[43] If the user presses <NO>, the configuration utility shall be redisplayed.

APIR3.2.1[44] When a configuration utility terminates, changes made to configurable fields that were not applied using the <ENT> key prior to the termination shall be discarded.

### Input Specifications

This test case requires no input.

### Output Specifications

This test case produces no output.

### Environmental Needs

#### Hardware

This test case utilizes the standard APIVS test hardware configuration as detailed in Section 2.1.

#### Software

This test case utilizes the standard APIVS test software as detailed in Section 2.1.

#### Other

None.

### Special Procedural Requirements

None.

### Intercase Dependencies

None.

### Additional Pass/Fail Criteria

All requirements in Section 2.27.3 must be satisfied for this test case to pass.

## Test Case Specification 25 – System Configuration Utilities

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.6020.

### Objective

The objective of this Test Case is to validate certain functional and navigational API requirements for the

System Configuration Utilities.

### Test Items

This test case tests the following requirements:

APIR3.2.2[1] The API shall provide a method for the Operational User to view and set the system time through the System Time Utility.

APIR3.2.2[2] The System Time Configuration Utility shall have the format as shown in Figure 12.

APIR3.2.2[3] The top line and bottom line of the System Time Configuration Utility shall be fixed as shown in Figure 12.

APIR3.2.2[4] The number of lines between the top line and bottom line used for displaying the Ethernet configuration information shall vary according to the size of the ATC Front Panel display.

APIR3.2.2[5] The Operational User shall be able to scroll the System Time Configuration Utility up and down one line at a time using the up and down arrow keys of the controller unit's keypad.

APIR3.2.2[6] The Operational User shall be able to modify the following fields on Line 7 of Figure 12 using the method described in Section 3.2.1 Item "h":

1. MM/DD/YYYY is the date where MM is the two digit month with values ranging 01-12, DD is the two digit day with values ranging 01-31, and YYYY is the two digit year with values ranging 0000-9999.
2. hh:mm:ss is the time based on a 24 hour clock (00:00:00 is 12:00 midnight) where hh is the two digit hour with values ranging 00-23, mm is the two digit minute with values ranging 00-59 and ss is the two digit second with values ranging 00-59.
3. shh:mm is the time zone represented as an offset from Coordinated Universal Time (UTC) where s is the sign with values + or -, hh is the two digit number of hours with values 00-12, and mm is the two digit number of minutes with values ranging 00-59.
4. enadis is a field with values "Enable" or "Disabl" indicating whether Daylight Savings Time (DST) is enabled or disabled for the location of the controller unit.

APIR3.2.2[7] Line 3 of Figure 12 shall represent the current date and time values of the controller unit and shall be updated once a second by the System Time Configuration Utility.

APIR3.2.2[8] The fields of Line 3 shall be as follows:

1. All of the fields described in Item "d" of this section.
2. status is a field with values "Active" or "Inactv" indicating whether or not the controller unit is currently applying DST.

APIR3.2.2[9] The default values of all fields shall be those values at the time the Operational User invokes the System Time Configuration Utility.

APIR3.2.3[1] The API shall provide a method for the Operational User to view and configure the Ethernet parameters of the ATC controller unit through the Ethernet Configuration Utility.

APIR3.2.3[2] The Ethernet Configuration Utility shall have the format as shown in Figure 13.

APIR3.2.3[3] The top line and bottom line of the Ethernet Configuration Utility shall be fixed as shown in Figure 13.

APIR3.2.3[4] The number of lines between the top line and bottom line used for displaying the Ethernet configuration information shall vary according to the size of the ATC Front Panel display.

APIR3.2.3[5] The Operational User shall be able to scroll the Ethernet Configuration Utility up and down one line at a time using the up and down arrow keys of the controller unit's keypad.

APIR3.2.3[6] The Operational User shall be able to modify the following fields using the method described in Section 3.2.1 Item "h" :

1. yon is a field that can be "Yes" or "No" indicating whether a port is enabled.
2. ### is a numerical field 0-255 used to set the Ethernet addresses (IPv4 addresses) associated with the port or network.
3. hostname is an alphanumeric field up to 255 characters long indicating the Host Name of the controller unit. If hostname extends beyond the edge of the display, it will wrap to the next line. Each character of the field may be letters a-z, letters A-Z, the digits 0-9, the period (.) and the hyphen (-). In addition to the valid alphanumeric characters, the user may enter an asterisk (\*) to clear the existing character and all characters to the right of the cursor position.

APIR3.2.3[7] The field identified as numofpackets shall be an unsigned integer value up to 10 digits long indicating the number of packets Sent and Received, Good and Bad over the Ethernet port as shown in Figure 13.

APIR3.2.3[8] numofpackets shall be updated by the Ethernet Configuration Utility one time per second.

APIR3.2.3[9] The Operational User shall not be able to modify these fields directly.

APIR3.2.3[10] The default values of all fields shall be those values at the time the Operational User invokes the Ethernet Configuration Utility.

APIR3.2.4[1] The API shall provide a method for the Operational User to enable and disable services through the System Services Configuration Utility.

APIR3.2.4[2] The Systems Services Configuration Utility shall have the format as shown in Figure 14.

APIR3.2.4[3] The top line and bottom line of the Systems Services Configuration Utility shall be fixed as shown in Figure 14.

APIR3.2.4[4] The number of lines between the top line and bottom line used for displaying the system services information shall vary according to the size of the ATC Front Panel display.

APIR3.2.4[5] The Operational User shall be able to scroll the Systems Services Configuration Utility up and down one line at a time using the up and down arrow keys of the controller unit's keypad.

APIR3.2.4[6] The Systems Services Configuration Utility shall display all of the services available on the controller with up to 50 services listed.

APIR3.2.4[7] The services listed in the Systems Services Configuration Utility shall be limited to 22 characters.

APIR3.2.4[8] The current status of the services listed shall be displayed in the "STATUS" column as shown in Figure 14.

APIR3.2.4[9] The field enaordis shall be "Enabled" or "Disabled" based on the status of the service.

APIR3.2.4[10] The Operational User shall not be able to modify this field directly.

APIR3.2.4[11] The Operational User shall be able to modify the enaordis field in the "CHANGE" column as shown in Figure 14 using the method described in Section 3.2.1 Item "h".

APIR3.2.4[12] The possible values shall be "Enabled" or "Disabled".

APIR3.2.5[1] The API shall provide a method for the Operational User to view Linux system and API library information through the Linux/API Information Utility as shown in Figure 15.

APIR3.2.5[2] The top line and bottom line of the Linux/API Information Utility shall be fixed as shown in Figure 15.

APIR3.2.5[3] The number of lines between the top line and bottom line used for displaying the system services information shall vary according to the size of the ATC Front Panel display.

APIR3.2.5[4] The Operational User shall be able to scroll the Linux/API Information Utility up and down one line at a time using the up and down arrow keys of the controller unit's keypad.

APIR3.2.5[5] The Linux/API Information Utility shall display the Linux system information available from the Linux "uname()" function (see also ATC Controller Standard, Section 2.2.5) as follows: "Kernel Name," "Network Node Name," "Kernel Release," "Machine Hardware Name," "Processor Type," "Hardware Platform," and "Operating System" as shown in Figure 15.

APIR3.2.5[6] The text field shall represent the character strings returned from the uname() function.

APIR3.2.5[7] The Linux/API Information Utility shall display the version information for API library and drivers including the manufacturer's name, the manufacturer's version number of the software, and the version of the API Standard to which the software is conformant.

APIR3.2.5[8] The API library and driver lines shall be repeated for each API library and driver installed on the controller unit.

APIR3.2.5[9] The API shall provide function(s) which return the version information of the API software in use on the controller unit.

APIR3.2.5[10] If the value of a field is too long to fit on the remainder of a line, the Linux/API Information Utility shall provide additional lines to accommodate the field in a wrapping fashion and the remaining Linux/API information shall start on a new line as shown in Figure 16.

APIR3.2.6[1] The API shall provide a method for the Operational User to view the contents of the ATC Host Module EEPROM information (as defined by ATC Controller Standard, Annex B) through the Host EEPROM Information Utility as shown in Figures 17 and 18.

APIR3.2.6[2] The field values shown are generally consistent with the "Default Configuration" as described in the ATC Controller Standard, Annex B. Actual field values will vary. The "User Data" field shall not be included in the Host EEPROM Information Utility.

APIR3.2.6[3] The top line and bottom line of the Host EEPROM Information Utility shall be fixed as shown in Figures 17 and 18.

APIR3.2.6[4] The number of lines between the top line and bottom line used for displaying the system services information shall vary according to the size of the ATC Front Panel display.

APIR3.2.6[5] The Operational User shall be able to scroll the Host EEPROM Information Utility up and down one line at a time using the up and down arrow keys of the controller unit's keypad..

APIR3.2.6[6] The values the "Ethernet Switch/Router Mac Addresses," "Host Board Serial Ports Used" and the "Agency Reserved" fields shall be displayed as hexadecimal pairs as represented by HH in Figure 18.

APIR3.2.6[7] If the value of a field is too long to fit on the remainder of a line, the Host EEPROM Information Utility shall provide additional lines to accommodate the field in a wrapping fashion and the remaining Linux/API information shall start on a new line (similar operation to that of Linux/API Information Utility in Figure 16).

### Input Specifications

This test case requires no input.

### Output Specifications

This test case produces no output.

### Environmental Needs

#### Hardware

This test case utilizes the standard APIVS test hardware configuration as detailed in Section 2.1.

#### Software

This test case utilizes the standard APIVS test software as detailed in Section 2.1.

#### Other

None.

### Special Procedural Requirements

None.

### Intercase Dependencies

None.

### Additional Pass/Fail Criteria

All requirements in Section 2.28.3 must be satisfied for this test case to pass.

## Test Case Specification 26 – Intrinsic API Requirements

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.6030.

### Objective

This Test Case is a placeholder for a number of API ‘requirements’ which are all intrinsic to the API Standard as well as the API Reference Implementation. As such, there is no specific testing to be performed for this test case per se, but rather just an acknowledgement that these ‘requirements’ are all inherent qualities of the API as it was designed.

### Test Items

This test case tests the following requirements:

APIR3.4[1] The API shall operate on an ATC controller unit under the hardware limitations defined in the ATC Controller Standard.

APIR3.4[2] The API function calls shall be specified using the C programming language as described by “ISO/IEC 9899:1999” commonly referred to as the C99 Standard.

APIR3.5.2[1] The operational look and feel of user interfaces developed for the API shall have consistent window titling conventions, scrolling methods, menu styles and selection methods.

APIR3.5.2[2] If API functions have a similar operation to existing Linux functions, they shall have a similar name and argument style to those functions to the extent possible without causing compilation issues.

APIR3.5.2[3] The API function names shall be lower case.

APIR3.5.2[4] API functions shall use the Linux “errno” error notification mechanism if an error indication is expected for a function.

APIR3.5.2[5] The API shall be loadable as an ELF (Executable and Linking Format) library.

APIR3.6[1] The API software shall only reference operating system commands and features that are available in the Linux environment defined in the ATC Board Support Package (see ATC Controller Standard, Section 2.2.5, Annex A and Annex B).

### Input Specifications

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

File Description

**C6030\_log.txt** ELF library validation file (text format)

### Output Specifications

This test case produces no output.

### Environmental Needs

#### Hardware

This test case utilizes the standard APIVS test hardware configuration as detailed in Section 2.1.

#### Software

This test case utilizes the standard APIVS test software as detailed in Section 2.1.

#### Other

None.

### Special Procedural Requirements

None.

### Intercase Dependencies

None.

### Additional Pass/Fail Criteria

TBD.

## Test Case Specification 27 – FIO Serial Ports and Status Counters

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.6040.

### Objective

The objective of this Test Case is to validate a number of software-related API requirements which do not have ‘exposure’ outside the internal workings of the API and so must be confirmed by an inspection of the relevant source code files.

### Test Items

This test case tests the following requirements:

APIR3.1.2[1] The API shall assume it has exclusive access to the serial communications ports of the ATC Engine Board that are designated for Field I/O Devices.

APIR3.1.2[2] The supported Field I/O serial communications ports shall be SP3, SP5 and SP8.

APIR3.1.2[3] The supported communication modes on those ports shall be 153.6 Kbps and 614.4 Kbps SDLC.

APIR3.1.2[7] The API shall assume that BIU and MMU Field I/O Devices operate at 153.6 Kbps and all other Field I/O Device types operate at 614.4 Kbps.

APIR3.1.2[42] To set the state of an output point and control dimming, the API shall use separate arrays for control of the Load Switch + and Load Switch – (see Section 3.3.1.4.1.5 of the TS 2 Standard).

APIR3.1.2[73] The timing for the command/response cycle of the frames shall be defined by the “Handshaking” algorithm in Section 3.3.1.5.3 of the NEMA TS 2 Standard.

APIR3.1.2[103] All counters contained in the Field I/O Device status information shall be four byte unsigned values each with a maximum value of 4,294,967,295.

APIR3.1.2[104] The counters shall be frozen when they reach the maximum value to prevent rollover.

APIR3.1.2[106] The response frame sequence number shall be a four byte unsigned value and rollover after the maximum value.

### Input Specifications

This test case requires no input.

### Output Specifications

This test case produces no output.

### Environmental Needs

#### Hardware

This test case utilizes the standard APIVS test hardware configuration as detailed in Section 2.1.

#### Software

This test case utilizes the standard APIVS test software as detailed in Section 2.1.

#### Other

None.

### Special Procedural Requirements

None.

### Intercase Dependencies

None.

### Additional Pass/Fail Criteria

All requirements in Section 2.30.3 must be satisfied for this test case to pass.

## Test Case Specification 28 – FPUI Display Presence and Size

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.7010.

### Objective

The objective of this Test Case is to validate certain API requirements related to the window display size and the detection of the presense (or absence) of a connected display panel.

### Test Items

This test case tests the following requirements:

APIR3.1.1[4] The display size of the windows shall have a minimum size of 4 lines x 40 characters and a maximum size of 24 lines x 80 characters.

*Guidance: the ATC Controller Standard requires a minimum display size of 8 lines x 40 characters.*

APIR3.1.1[5] If no physical display exists, the API shall operate as if it has a display with a size of 8 lines x 40 characters.

APIR3.1.1.2[38] The API shall provide a mechanism to allow application programs to detect the presence or absence of a Front Panel.

APIR3.1.1.2[39] The API shall recognize the presence or absence of a Front Panel in 5 seconds.

APIR3.1.1.2[40] The API shall provide an asynchronous notification to alert application programs of a change in the presence or absence of a Front Panel.

APIR3.1.1.2[41] The API shall provide an asynchronous notification to alert all application programs when their associated windows change size.

### Input Specifications

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

File Description

**C7010\_in.xml** APIVSXML test script (XML format)

**VS\_config\_2.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C7010\_log.xml** conformance report (XML format)

### Environmental Needs

#### Hardware

This test case utilizes the standard APIVS test hardware configuration as detailed in Section 2.1, except that it also requires a display panel connection with a modified serial data communications path that can be interrupted on-demand by the use of custom cabling with an integrated mechanical switch (exact details TBD). An example of such a controller would be a Model 2070 with a custom ribbon cable between the motherboard and Front Panel assembly.

#### Software

This test case utilizes the standard APIVS test software as detailed in Section 2.1.

### Additional Pass/Fail Criteria

All requirements in Section 2.31.3 must be satisfied for this test case to pass, although an exception may need to be be granted for requirement APIR3.1.1.2[41], as the hardware necessary to test this requirement (display window size changes) would not appear to be currently available.

## Test Case Specification 29 – FPUI Bell Activation and Application Termination

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.7020.

### Objective

The objective of this Test Case is to test the API’s bell activation requirements. It also verifies that an FPUI API connection is closed when an application terminates without explicitly closing the connection.

### Test Items

This test case tests the following requirements:

APIR3.1.1.2[24] The bell of the controller’s Front Panel shall be activated only if a bell character, ^G (hex value 07), is sent from an application program which has a window that has focus.

APIR3.1.1.2[25] If a bell character is sent from an application program that does not have a window that has focus, the bell character shall be ignored by the API.

APIR3.1.2[134] When an application program exits or terminates for any reason, the API shall deregister the application program from the API (as defined in Item “d”).

### Input Specifications

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

File Description

**C7020\_in.xml** APIVSXML test script (XML format)

**VS\_config\_2.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C7020\_log.xml** conformance report (XML format)

### Environmental Needs

#### Hardware

This test case utilizes the standard APIVS test hardware configuration as detailed in Section 2.1.

#### Software

This test case utilizes the standard APIVS test software as detailed in Section 2.1.

### Additional Pass/Fail Criteria

All requirements in Section 2.32.3 must be satisfied for this test case to pass.

## Test Case Specification 30 – FPUI Display Graphics

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.7030.

### Objective

The objective of this Test Case is to test the API’s handling of graphical display commands.

### Test Items

This test case tests the following requirements:

APIR3.1.1.2[30] The ATC Controller Standard, Section 7.1.4 (ATC 5201 has this in Section 6.1.4), describes a graphics interface to the Front Panel’s display. The API shall support the operation of the graphics commands on a window only if that window is in focus.

APIR3.1.1.2[31] If application programs use graphics on a window, the API shall not redisplay these graphics when a window is refreshed or goes out/in focus.

### Input Specifications

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

File Description

**C7030\_in.xml** APIVSXML test script (XML format)

**C7030\_vd\_blank.txt** Virtual Display compare file (blank display)

**C7030\_vd\_raw\_graphic.txt** Virtual Display compare file (raw graphic)

**C7030\_enable\_graphic\_underlay.txt** Virtual Display compare file (graphics enabled)

**VS\_config\_2.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

This test case produces no output:

### Environmental Needs

#### Hardware

This test case utilizes the standard APIVS test hardware configuration as detailed in Section 2.1.

#### Software

This test case utilizes the standard APIVS test software as detailed in Section 2.1.

### Additional Pass/Fail Criteria

All requirements in Section 2.33.3 must be satisfied for this test case to pass.

## Test Case Specification 31 – FPUI Display Focus

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.7040.

### Objective

The objective of this Test Case is to test the API requirements regarding an FPUI application’s request for display focus.

### Test Items

This test case tests the following requirements:

APIR3.1.1.2[34] The API shall provide a method to allow application programs to indicate that a window desires focus from the Operational User.

APIR3.1.1.2[35] This method shall cause the Front Panel backlight to flash and the window name in the Front Panel Manager Window to blink.

APIR3.1.1.2[36] The window name blinking shall cease once the indicated window receives focus.

APIR3.1.1.2[37] The backlight flashing shall cease when all windows requesting focus have been given focus.

### Input Specifications

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

File Description

**C7040\_in.xml** APIVSXML test script (XML format)

**VS\_config\_2.txt** VSE configuration file (specified on VSE command line)

### Output Specifications

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

File Description

**C7040\_log.xml** conformance report (XML format)

### Environmental Needs

#### Hardware

This test case utilizes the standard APIVS test hardware configuration as detailed in Section 2.1.

#### Software

This test case utilizes the standard APIVS test software as detailed in Section 2.1.

### Additional Pass/Fail Criteria

All requirements in Section 2.34.3 must be satisfied for this test case to pass.

## Test Case Specification 32 – System Configuration Menu Display

### Test Case Specification Identifier

The identifier for this Test Case Specification is APIRI.TCS.7050.

### Objective

The objective of this Test Case is to validate the use of the configuration text file used by the System Configuration Utility to create the System Configuration Menu.

### Test Items

This test case tests the following requirements:

APIR3.2.1[10] The Configuration Menu shall be extensible by software developers with up to twelve additional configuration items that are not defined in the API Standard.

APIR3.2.1[45] The API shall provide a configuration file called "ATCConfigurationMenu.txt" to be used by the API to form the Configuration Menu.

APIR3.2.1[46] The format of the file shall be comma delimited with each line representing a configuration item for the Configuration Menu as follows:

configitemname, executablepathname

configitemname, executablepathname

configitemname, executablepathname

etc.

where "configitemname" represents the text displayed in the Configuration Menu and "executablepathname" is the pathname to the executable file of the appropriate configuration utility.

APIR3.2.1[47] The lines shall be processed by the API in order until the end of file is reached or the Configuration Menu is full.

APIR3.2.1[48] The API shall provide a function to allow a configuration utility to open the ATC Configuration Window and reserve the resource.

APIR3.2.1[49] The API shall provide a function to close the ATC Configuration Window and release the resource.

### Input Specifications

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

File Description

ATCConfigurationMenu.txt file used to configure System Configuration Menu

SCTestApp test application for SC menu configuration

SConfig.c ‘C’ source file for SC menu manager

### Output Specifications

This test case produces no output.

### Environmental Needs

#### Hardware

This test case utilizes the standard APIVS test hardware configuration as detailed in Section 2.1.

#### Software

This test case utilizes the standard APIVS test software as detailed in Section 2.1.

### Additional Pass/Fail Criteria

All requirements in Section 2.35.3 must be satisfied for this test case to pass.