T308: Acceptance Testing for Advanced Transportation Controller Application Programming Interface Software

Table of Contents

1. Module Description

Module T308: Acceptance Test for Advanced Transportation Controller Application Programming Interface Software provides training for the testing API Software. The API Software is to conform to the ATC 5402 v02 Standard.

The following are recommended prerequisites in the ATC curriculum for students taking this course.

- A207b: Building an ITS infrastructure Based on the ATC 5201 Standard Part 2 of 2
- A208: Using the ATC 5401 API Standard to Leverage ITS Infrastructure
- A307a: Understanding User Needs for Advanced Transportation Controller (ATC) Based on ATC 5201 Standard v06
- A307b: Understanding Requirements for Advanced Transportation Controller (ATC) Based on ATC 5201 Standard v06
- T307: Applying Your Test Plan to the Advanced Traffic Controller based on ATC 5201 Standard v06

The following are recommended prerequisites in the testing curriculum for students taking this course.

- T101: Introduction to ITS Standards Testing
- T201: How to Write a Test Plan
- T202: Overview of Test Design Specifications, Test Case Specifications, and Test Procedures
- T203: Part 1 of 2: How to Develop Test Cases for an ITS Standards-Based Test Plan, Part 1 of 2
- T203: Part 2 of 2: How to Develop Test Cases for an ITS Standards-Based Test Plan, Part 2 of 2
- T204: Part 1 of 2: How to Develop a Test Procedure for ITS Standards-based Test Plan
- T204: Part 2 of 2: How to Develop a Test Procedure for ITS Standards-based Test Plan

2. Introduction/Purpose

The Advanced Transportation Controller (ATC) family of standards provides an open architecture hardware and software platform that can support a wide variety of Intelligent Transportation Systems (ITS) field applications. These standards are characterized by their modularity, support of multiple and concurrent application programs, and design to facilitate the adoption of new technologies.

The ATC 5401 Application Programming Interface (API) Standard v02 defines API Software that provides both user and programmatic interfaces for transportation controller units that conform to the ATC 5201 Transportation Controller Standard (ATC units). The API Software allows application programs to be written so that they may run on any ATC unit regardless of the manufacturer. It also allows multiple application programs to be compatible on a single ATC unit by sharing the fixed resources of the controller.

As part of the API Reference Implementation (APIRI) Project, the API Validation Suite (APIVS) software was developed that may be used to test API Software resident on an ATC unit. The APIVS Software is maintained in an open source software (OSS) environment and is available to anyone. Test documentation based on IEEE 829-2008 that leverages that APIVS Software is also provided.

Module T308 focuses on testing the API Software based on the ATC 5401 Standard. At the conclusion of this module, participants will be able to: understand the purpose of the APIVS Software; use test documentation to specify API Software acceptance testing; use the APIVS Software to perform API Software testing; and interpret and report the results of testing.

3. Samples/Examples

From the ATC APIRI Test Plan v01.05, Section 3, Features to be Tested, Table 1.

Test ID	Document Name	Brief Description
APIRI.TDS.2001	APIRI Test Design Specification 1	Test All APIRI FPUI Required Features
APIRI.TDS.3001	APIRI Test Design Specification 2	Test All APIRI FIO Required Features
APIRI.TDS.4001	APIRI Test Design Specification 3	Test All APIRI TOD Required Features
APIRI.TCS.2010	APIRI Test Case Specification 1	FPUI Text UI Virtual Displays
APIRI.TCS.2020	APIRI Test Case Specification 2	FPUI Front Panel Manager
APIRI.TCS.2030	APIRI Test Case Specification 3	FPUI Character Set and Screen Attributes
APIRI.TCS.2040	APIRI Test Case Specification 4	FPUI Reading and Writing Data
APIRI.TCS.2050	APIRI Test Case Specification 5	FPUI Special Characters
APIRI.TCS.2070	APIRI Test Case Specification 6	FPUI Key Mapping
APIRI.TCS.2080	APIRI Test Case Specification 7	FPUI Asynchronous Notification and Focus
APIRI.TCS.2090	APIRI Test Case Specification 8	FPUI Raw Data Handling
APIRI.TCS.2100	APIRI Test Case Specification 9	API Version Information (All Libraries)
APIRI.TCS.3010	APIRI Test Case Specification 10	General FIO Operations
APIRI.TCS.3020	APIRI Test Case Specification 11	FIO Inputs and Outputs
APIRI.TCS.3030	APIRI Test Case Specification 12	FIO Channel Mapping
APIRI.TCS.3040	APIRI Test Case Specification 13	FIO Filtered Inputs and Transition Buffering
APIRI.TCS.3050	APIRI Test Case Specification 14	FIO Frame Frequency
APIRI.TCS.3060	APIRI Test Case Specification 15	FIO Failed State and Fault Monitoring
APIRI.TCS.3070	APIRI Test Case Specification 16	FIO Watchdog Outputs
APIRI.TCS.3080	APIRI Test Case Specification 17	FIO Device Status
APIRI.TCS.3090	APIRI Test Case Specification 18	FIO Health Monitor
APIRI.TCS.3100	APIRI Test Case Specification 19	FIO CMU Configuration
APIRI.TCS.3110	APIRI Test Case Specification 20	FIO Module Status
APIRI.TCS.3120	APIRI Test Case Specification 21	FIO Asynchronous Notification
APIRI.TCS.3130	APIRI Test Case Specification 22	FIO Dark Channel Mapping
APIRI.TCS.4010	APIRI Test Case Specification 23	TOD Time Handling Functions
APIRI.TCS.6010	APIRI Test Case Specification 24	FPM and ATC Configuration Menu
APIRI.TCS.6020	APIRI Test Case Specification 25	System Configuration Utilities
APIRI.TCS.6030	APIRI Test Case Specification 26	Intrinsic API Requirements

Test ID	Document Name	Brief Description
APIRI.TCS.6040	APIRI Test Case Specification 27	FIO Serial Ports and Status Counters
APIRI.TCS.7010	APIRI Test Case Specification 28	FPUI Display Presence and Size
APIRI.TCS.7020	APIRI Test Case Specification 29	FPUI Bell Activation and App Termination
APIRI.TCS.7030	APIRI Test Case Specification 30	Test FPUI Display Graphics
APIRI.TCS.7040	APIRI Test Case Specification 31	FPUI Display Focus
APIRI.TCS.7050	APIRI Test Case Specification 32	System Configuration Menu Display
APIRI.TPS.1001	APIRI Test Procedure Specification 1	Auto-Execute Selected APIRI Script(s)
APIRI.TPS.6010	APIRI Test Procedure Specification 2	FPM and ATC Configuration Menu
APIRI.TPS.6020	APIRI Test Procedure Specification 3	System Configuration Utilities
APIRI.TPS.6030	APIRI Test Procedure Specification 4	Intrinsic API Requirements
APIRI.TPS.6040	APIRI Test Procedure Specification 5	FIO Serial Ports and Status Counters
APIRI.TPS.7010	APIRI Test Procedure Specification 6	FPUI Display Presence and Size
APIRI.TPS.7020	APIRI Test Procedure Specification 7	FPUI Bell Activation and App Termination
APIRI.TPS.7040	APIRI Test Procedure Specification 9	FPUI Display Focus
APIRI.TPS.7050	APIRI Test Procedure Specification 10	System Configuration Menu Display

The runAPIVS.sh Linux Shell Script.

```
#!/bin/sh

# ATC 5401 API Reference Implementation Project
#
#     Filename: runAPIVS
#    File Type: Linux shell script
#    Test Case: many
#   Description: run VSE from USB at startup on specific test cases
#
# Date      Revision    Description
# 2/24/16   1.0         initial release

# start async loopback driver; add symbolic links
insmod /media/sda1/APIVS/bin/tty0tty.ko
ln -s /dev/tnt0 /dev/sp6_loopback_a
ln -s /dev/tnt1 /dev/sp6_loopback_b

# start the API Front Panel Manager (loopback mode)
modprobe front_panel
FrontPanelManager /dev/sp6_loopback_a 1>/dev/null 2>&1 &
MasterSelection 1>/dev/null 2>&1 &

# start sync loopback driver; fio driver; add symbolic links
rmmod fiodriver
insmod /media/sda1/APIVS/bin/virtual-loopback-sync.ko
insmod /media/sda1/APIVS/bin/fiodriver.ko apivs=1
ln -s /dev/vlsync0 /dev/sp3s_loopback_a
ln -s /dev/vlsync1 /dev/sp3s_loopback_b
ln -s /dev/vlsync2 /dev/sp5s_loopback_a
ln -s /dev/vlsync3 /dev/sp5s_loopback_b
ln -s /dev/vlsync4 /dev/sp8s_loopback_a
ln -s /dev/vlsync5 /dev/sp8s_loopback_b

# initialize test counters
pass_count=0
fail_count=0

# a couple of useful subroutines
clear_test_line() {
   printf "\x1B[3;1f" >/dev/sp6
   printf "                                        " >/dev/sp6
   printf "\x1B[3;1f" >/dev/sp6
}
print_test_result() {
   if [ "$?" == "0" ]; then
      printf "PASS" >/dev/sp6
      pass_count=$((pass_count + 1))
      else
      printf "FAIL" >/dev/sp6
      fail_count=$((fail_count + 1)); fi
   printf "\x1B[8;1f" >/dev/sp6
   printf "Test cases passed:%d failed:%d" "$pass_count" "$fail_count"
>/dev/sp6
#  keep backlight on
   printf "\033[<5h" >/dev/sp6
   sleep 1
}
reset_modules() {
#  start the API Front Panel Manager (loopback mode)
   killall MasterSelection
   killall FrontPanelManager
   rmmod front_panel
   modprobe front_panel
   FrontPanelManager /dev/sp6_loopback_a 1>/dev/null 2>&1 &
   MasterSelection 1>/dev/null 2>&1 &
#  start fio driver
   rmmod fiodriver
   insmod /media/sda1/APIVS/bin/fiodriver.ko apivs=1
}
misc_test_C6020() {
   #Check for uClibc load object identifier (APIVS[3.8])
   LD_TRACE_LOADED_OBJECTS=1 vse | grep -q "ld-uClibc"
   if [ $? != 0 ]
   then
        echo "$(date -u): vse is not compatible with uClibc"
>/tmp/C6020_log.txt
        retval=1
   else
        echo "$(date -u): vse is compatible with uClibc"
>/tmp/C6020_log.txt
        retval=0
   fi
   mv /tmp/C6020_log.txt ./
   sync
   return $retval
}
misc_test_C6030() {
   #Check for ELF format (APIR3.5.2[5])
   elfmagic="7f454c46"
   FILES="/usr/lib/libfpui.so
   /usr/lib/libfio.so
   /usr/lib/libtod.so"

   for f in $FILES
   do
   if [ $(od -An -N4 -tx4 <$f) == $elfmagic ]; then
      echo "$(date -u): $f is ELF format file" >>/tmp/C6030_log.txt
      retval=0
   else
      echo "$(date -u): $f is not ELF format file" >>/tmp/C6030_log.txt
      retval=1
   fi
   done
   mv /tmp/C6030_log.txt ./
   sync
   return $retval
}

# set the conformance level this run (1,2,3)
LEVEL=3

# delete old log files? (FALSE will append if exists)
DELETE_LOGS=TRUE

# how many times to run?
loop_count=1

# loop through all specified test cases
while [ $loop_count -ge 1 ]
do
   reset_modules
   if [ "$DELETE_LOGS" == TRUE ]; then rm C1110_log.xml; fi
   clear_test_line; printf "Testing APIVS.TCS.1110... " >/dev/sp6
   vse -L $LEVEL -c ./VS_config_1.txt -i C1110_in.xml -o C1110_log.xml
   print_test_result

   reset_modules
   if [ "$DELETE_LOGS" == TRUE ]; then rm C1120_log.xml; fi
   clear_test_line; printf "Testing APIVS.TCS.1120... " >/dev/sp6
   vse -L $LEVEL -c ./VS_config_1.txt -i C1120_in.xml -o C1120_log.xml
   print_test_result

   reset_modules
   if [ "$DELETE_LOGS" == TRUE ]; then rm C1130_log.xml; fi
   clear_test_line; printf "Testing APIVS.TCS.1130... " >/dev/sp6
   vse -L $LEVEL -c ./VS_config_1.txt -i C1130_in.xml -o C1130_log.xml
   print_test_result

   reset_modules
   if [ "$DELETE_LOGS" == TRUE ]; then rm C1150_log.xml; fi
   clear_test_line; printf "Testing APIVS.TCS.1150... " >/dev/sp6
   vse -L $LEVEL -c ./VS_config_1.txt -i C1150_in.xml -o C1150_log.xml
   print_test_result

   reset_modules
   if [ "$DELETE_LOGS" == TRUE ]; then rm C1160_log.xml; fi
```

```
clear_test_line; printf "Testing APIVS.TCS.1160... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C1160_in.xml -o C1160_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C1310_log.xml; fi
clear_test_line; printf "Testing APIVS.TCS.1310... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C1310_in.xml -o C1310_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C1320_log.xml; fi
clear_test_line; printf "Testing APIVS.TCS.1320... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C1320_in.xml -o C1320_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C1330_log.xml; fi
clear_test_line; printf "Testing APIVS.TCS.1330... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C1330_in.xml -o C1330_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C1350_log.xml; fi
clear_test_line; printf "Testing APIVS.TCS.1350... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C1350_in.xml -o C1350_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C1410_log.xml; fi
clear_test_line; printf "Testing APIVS.TCS.1410... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C1410_in.xml -o C1410_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C1420_log.xml; fi
clear_test_line; printf "Testing APIVS.TCS.1420... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C1420_in.xml -o C1420_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C1430_log.xml; fi
clear_test_line; printf "Testing APIVS.TCS.1430... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C1430_in.xml -o C1430_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C1450_log.xml; fi
clear_test_line; printf "Testing APIVS.TCS.1450... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C1450_in.xml -o C1450_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C2010_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.2010... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2010_in.xml -o C2010_log.xml
print_test_result
```

```
reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C2020_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.2020... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2020_in.xml -o C2020_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C2030_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.2030... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2030_in.xml -o C2030_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C2040_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.2040... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2040_in.xml -o C2040_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C2050_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.2050... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2050_in.xml -o C2050_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C2070_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.2070... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2070_in.xml -o C2070_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C2080_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.2080... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2080_in.xml -o C2080_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C2090_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.2090... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2090_in.xml -o C2090_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C2100_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.2100... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C2100_in.xml -o C2100_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C3010_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.3010... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C3010_in.xml -o C3010_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C3020_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.3020... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C3020_in.xml -o C3020_log.xml
```

```
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C3030_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.3030... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C3030_in.xml -o C3030_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C3040_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.3040... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C3040_in.xml -o C3040_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C3050_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.3050... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C3050_in.xml -o C3050_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C3060_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.3060... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C3060_in.xml -o C3060_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C3070_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.3070... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C3070_in.xml -o C3070_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C3080_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.3080... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C3080_in.xml -o C3080_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C3090_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.3090... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C3090_in.xml -o C3090_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C3100_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.3100... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C3100_in.xml -o C3100_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C3110_log.xml; fi
clear_test_line; printf "Testing APIRI.TCS.3110... " >/dev/sp6
vse -L $LEVEL -c ./VS_config_1.txt -i C3110_in.xml -o C3110_log.xml
print_test_result

reset_modules
if [ "$DELETE_LOGS" == TRUE ]; then rm C3120_log.xml; fi
```

```
   clear_test_line; printf "Testing APIRI.TCS.3120... " >/dev/sp6
   vse -L $LEVEL -c ./VS_config_1.txt -i C3120_in.xml -o C3120_log.xml
   print_test_result

   reset_modules
   if [ "$DELETE_LOGS" == TRUE ]; then rm C3130_log.xml; fi
   clear_test_line; printf "Testing APIRI.TCS.3130... " >/dev/sp6
   vse -L $LEVEL -c ./VS_config_1.txt -i C3130_in.xml -o C3130_log.xml
   print_test_result

   reset_modules
   if [ "$DELETE_LOGS" == TRUE ]; then rm C4010_log.xml; fi
   clear_test_line; printf "Testing APIRI.TCS.4010... " >/dev/sp6
   vse -L $LEVEL -c ./VS_config_1.txt -i C4010_in.xml -o C4010_log.xml
   print_test_result

   reset_modules
   if [ "$DELETE_LOGS" == TRUE ]; then rm C6010_log.xml; fi
   clear_test_line; printf "Testing APIRI.TCS.6010... " >/dev/sp6
   vse -L $LEVEL -c ./VS_config_1.txt -i C6010_in.xml -o C6010_log.xml
   print_test_result

   reset_modules
   if [ "$DELETE_LOGS" == TRUE ]; then rm C6020_log.txt; fi
   clear_test_line; printf "Testing APIRI.TCS.6020... " >/dev/sp6
   misc_test_C6020
   print_test_result

   reset_modules
   if [ "$DELETE_LOGS" == TRUE ]; then rm C6030_log.txt; fi
   clear_test_line; printf "Testing APIRI.TCS.6030... " >/dev/sp6
   misc_test_C6030
   print_test_result

   reset_modules
   if [ "$DELETE_LOGS" == TRUE ]; then rm C7030_log.xml; fi
   clear_test_line; printf "Testing APIRI.TCS.7030... " >/dev/sp6
   vse -L $LEVEL -c ./VS_config_1.txt -i C7030_in.xml -o C7030_log.xml
   print_test_result

   mv /tmp/*log.xml ./
   sync

   loop_count=$((loop_count - 1))
done
```

4. Reference to Other Standards

Institute of Electrical and Electronics Engineers, *IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications*. IEEE, 1998.
http://standards.ieee.org/index.html

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

Institute of Transportation Engineers, *User Comment Draft ATC 5201 Advanced Transportation Controller (ATC) Standard Version 06.10*. ATC Joint Committee, 30 July 2012. http://www.ite.org/standards/index.asp

National Electrical Manufacturers Association, *NEMA Standards Publication TS 2-2003 v02.06 Traffic Controller Assemblies with NTCIP Requirements*. NEMA, 2003. https://www.nema.org/Standards/ComplimentaryDocuments/Contents%20and%20Scope%20TS%202-2003%20(R2008).pdf

5. Glossary

To include additional **descriptions/acronyms** used primarily in the module.

Term	Definition
AASHTO	American Association of State Highway and Transportation Officials.
API	Application Programming Interface.
API Utilities	API software that is used for setting system-wide purposes on an ATC controller unit.
APIRI Project	Entire project managed by *ATC APIRI PMP v01.01 Project Management Plan (PMP) for the Advanced Transportation Controller (ATC) Application Programming Interface (API) Reference Implementation Project* including software, hardware and documentation.
APIRI Software	API Reference Implementation (software). API software developed as part of the ATC APIRI Project.
APIVS Software	API Validation Suite Software.
Application Program	Any program designed to perform a specific function directly for the user or, in some cases, for another application program. Examples of application programs include word processors, database programs, Web browsers and traffic control programs. Application programs use the services of a computer's O/S and other supporting programs such as an application programming interface.
ATC	Advanced Transportation Controller.
ATC Device Drivers	Low-level software not included in a typical Linux distribution that is necessary for ATC-specific devices to operate in a Linux O/S environment.
Board Support Package	Software usually provided by processor board manufacturers that provide a consistent software interface for the unique architecture of the board. In the case of the ATC, the Board Support Package also includes the O/S.
BOM	Bill of Materials. A list of the raw materials, sub-assemblies, intermediate assemblies, sub-components, parts and the quantities of each needed to manufacture an end product.
BSP	See Board Support Package.
ConOps	Concept of Operations.

Term	Definition
CPU	Central Processing Unit. A programmable logic device that performs the instruction, logic and mathematical processing in a computer.
Device Driver	A software routine that links a peripheral device to the operating system. It acts like a translator between a device and the application programs that use it.
FIO	Field Input and Output.
FPUI	Front Panel User Interface.
H/W	Hardware.
I/O	Input/Output.
IEC	International Electrotechnical Commission.
IEEE	Institute of Electrical and Electronics Engineers.
ISO	International Organization for Standardization.
ITE	Institute of Transportation Engineers.
ITS	Intelligent Transportation Systems.
JC	Joint Committee.
JPO	Joint Program Office.
Linux	Low-level software that is freely available in the Linux community for use with common hardware components operating in a standard fashion.
Linux Kernel	The Unix-like operating system kernel that was begun by Linus Torvalds in 1991. The Linux Kernel provides general O/S functionality. This includes functions for things typical in any computer system such as file I/O, serial I/O, interprocess communication and process scheduling. It also includes Linux utility functions necessary to run programs such as shell scripts and console commands. It is generally available as open source (free to the public). The Linux Kernel referenced in this standard is defined in the ATC Controller Standard Section 4.3.5, Appendix A and Appendix B.
Loopback Driver	A virtual device driver that loops back the output ports to a device to the input ports from a device without actually going through the physical device.
Mechanical Drawing	A drawing to scale of a machine, machine component, or device from which dimensions can be taken for manufacturing.
N/A	Not Applicable.
O/S	Operating System.
Operational User	A technician or transportation engineer who uses the controller to perform its operational tasks.
PCB	Printed Circuit Board.
PMP	Project Management Plan.
RI	Reference Implementation.
RTC	Real-Time Clock.

Term	Definition
S/W	Software.
Schematic Diagram	A diagram that shows, by means of graphic symbols, the electrical connections and functions of a specific circuit arrangement.
SDD	Software Design Document or Software Design Description.
SDO	Standards Development Organization.
SE	Systems Engineer.
Software Validation	The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.
SOW	Statement of Work.
SRS	Software Requirements Specification.
TBD	To Be Determined.
Tester	A user developer, test engineer or test technician capable of operating the API Validation Suite described by this document.
TFCS	Transportation Field Cabinet System.
TOD	Time of Day.
TOPR	Task Order Proposal Request.
US	United States.
USDOT	United States Department of Transportation.
User Developer	A software developer that designs and develops programs for controllers.
Walkthrough	A step-by-step presentation by the author of a document in order to gather information and to establish a common understanding of its content.
WG	Working Group.
XML	Extensible Markup Language.

6. References

Application Programming Interface (API) Reference Implementation Project
Website http://www.ite.org/standards/atcapi/referenceimplementation.asp

API Reference Implementation (APIRI) Repository
https://github.com/apiriadmin/APIRI

API Validation Suite (APIVS) Repository
https://github.com/apiriadmin/APIVS

Notepad++. General purpose editing tool for software related files Color coding and formatting of XML text files
http://notepad-plus-plus.org

XML Differences. Online comparison of XML files.
www.corefiling.com/opensource/xmldiff.html

XmlGrid. Online editor displays in formatted XML text or in grids (tables).

XML Viewer. Online editor displays in formatted XML text or in tree view.
www.codebeautify.org/xmlviewer

7. Study Questions

To include the quiz/poll questions and answer choices as presented in the PowerPoint slide to allow students to either follow along with the recording or refer to the quiz at a later date in the supplement.

1. What controller software is NOT traditionally tested by agencies?
   a) Data Collection Software
   b) Signal Control Software
   c) Application Programming Interface Software
   d) Ramp Meter Software

2. What document is used to specify the inputs and outputs for a particular test of the API Software?
   a) Test Design
   b) Test Procedure Specification
   c) Test Plan
   d) Test Case Specification

3. What is not an appropriate reason to edit the runAPIVS shell script?
   a) Turn off all test output
   b) Change the number of iterations on a test
   c) Change the conformance report logging
   d) Select a subset of the existing test cases

4. True or False: It’s a good idea to always log as much information as possible on all tests.
   a) True
   b) False

8. Icon Guide

The following icons are used throughout the module to visually indicate the corresponding learning concept listed out below, and/or to highlight a specific point in the training material.

1) **Background information:** General knowledge that is available elsewhere and is outside the module being presented. This will be used primarily in the beginning of slide set when reviewing information readers are expected to already know.

2) **Tools/Applications:** An industry-specific item a person would use to accomplish a specific task, and applying that tool to fit your need.

3) **Remember:** Used when referencing something already discussed in the module that is necessary to recount.

4) **Refer to Student Supplement:** Items or information that are further explained/detailed in the Student Supplement.

SUPPLEMENT

5) **Example:** Can be real-world (case study), hypothetical, a sample of a table, etc.

6) **Checklist:** Use to indicate a process that is being laid out sequentially.