

Space Telecommunications Radio System (STRS) Compliance Testing

Louis M. Handler
Glenn Research Center, Cleveland, Ohio

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

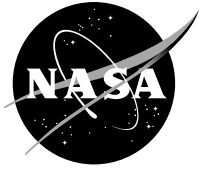
- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.
- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.

- **CONFERENCE PUBLICATION.** Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.
- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at <http://www.sti.nasa.gov>
- E-mail your question via the Internet to help@sti.nasa.gov
- Fax your question to the NASA STI Help Desk at 443-757-5803
- Telephone the NASA STI Help Desk at 443-757-5802
- Write to:
NASA Center for AeroSpace Information (CASI)
7115 Standard Drive
Hanover, MD 21076-1320



Space Telecommunications Radio System (STRS) Compliance Testing

Louis M. Handler
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

Trade names and trademarks are used in this report for identification only. Their usage does not constitute an official endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076-1320

National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Available electronically at <http://www.sti.nasa.gov>

Space Telecommunications Radio System (STRS) Compliance Testing

PREPARED BY:

Software Lead
Louis M. Handler

Date

APPROVED BY:

Systems Lead
Richard C. Reinhart

Date

Project Manager
Karen L. Tuttle

Date

Change Record

Revision	Effective Date	Description of Change
1.02	May 31, 2011	Baseline

Space Telecommunications Radio System (STRS) Compliance Testing		
Date: May 31, 2011	Document No.: STRS-ATP-00001	Page 3 of 46

Intentionally left blank

Contents

1.0	INTRODUCTION	6
1.1	Identification	6
1.2	System Overview	8
1.3	Document Overview	8
2.0	APPLICABLE DOCUMENTS	9
2.1	Reference Documents	9
3.0	TEST CONFIGURATIONS	9
3.1	Hardware Configuration	9
3.2	Software Configuration.....	10
3.3	Other Pretest Preparations.....	11
4.0	TEST DESCRIPTIONS	11
4.1	Requirements Being Tested	12
4.2	Test Instructions.....	12
4.3	STRS Compliance Testing.....	12
4.3.1	Prerequisite Conditions, Assumptions, and Constraints	13
4.3.2	Test Procedure for STRS Application Automated Testing.....	13
4.3.3	Test Procedure for STRS Infrastructure Automated Testing.....	16
4.3.4	Test Procedure for STRS Infrastructure Testing Using WFCCN.....	18
4.3.5	Test Procedure for STRS Configuration File Testing.....	21
4.3.6	Test Procedure for STRS Application Manual Code Testing.....	23
4.3.7	Test Procedure for STRS Infrastructure Manual Code Testing.....	25
APPENDIX A	—Glossary and Acronyms	27
A.1	Definitions	27
A.2	Acronyms.....	28
APPENDIX B	—Traceability to SWE-114 of NPR 7150.2A	29
APPENDIX C	—Application Compliance Testing Tables	30
APPENDIX D	—Infrastructure Compliance Testing Tables	32
APPENDIX E	—Infrastructure Compliance Testing by WFCCN Tables.....	33
APPENDIX F	—Compliance Testing by Requirement.....	36
APPENDIX G	—Document Compliance Testing Guidelines.....	43

Table of Figures

<i>Figure 1 - Operating Environment Compliance</i>	6
<i>Figure 2 - Application Compliance</i>	7
<i>Figure 3 - SDR-3000 and Development Platform</i>	10
<i>Figure 4 - General Automated Procedure with Manual Double-Check</i>	13

List of Tables

<i>Table 1 - Table of Documents</i>	9
<i>Table 2 - Hardware Used</i>	10
<i>Table 3 - Software Used</i>	11
<i>Table 4 - STRS Application Test Automated Procedure</i>	14
<i>Table 5 - STRS Infrastructure Test Automated Procedure</i>	16
<i>Table 6 - STRS WFCCN Test Automated Procedure</i>	18
<i>Table 7 - STRS Configuration Files Test Procedure</i>	21
<i>Table 8 - STRS Application Test Manual Procedure</i>	23
<i>Table 9 - STRS Infrastructure Test Manual Procedure</i>	25
<i>Table 10 - Glossary</i>	27
<i>Table 11 - Acronyms</i>	28
<i>Table 12 - Traceability to SWE-114 of 7150.2A</i>	29
<i>Table 13 - Compliance Script Execution</i>	30
<i>Table 14 - Compliance Script Web Page Output</i>	31
<i>Table 15 - OE Compliance Script Execution</i>	32
<i>Table 16 - OE Compliance Script Web Page Output</i>	32
<i>Table 17 - WFCCN Configurable Data</i>	33
<i>Table 18 - WFCCN tests in APP_RunTest</i>	34
<i>Table 19 - Requirements Testing</i>	36

1.0 INTRODUCTION

The intended audience is anyone who wants to check whether their STRS application or STRS infrastructure meets the STRS Architecture Standard. This includes platform providers, application developers, application integrators, and NASA. As of May, 2011, only NASA Glenn Research Center personnel have performed the STRS Compliance Testing.

1.1 Identification

The document describes the procedures for testing a software defined radio (SDR) implementation for STRS compliance. STRS compliance is concerned with how well the delivered artifacts conform to the STRS Architecture Standard. Broadly, STRS compliance may be categorized as either STRS application compliance or STRS infrastructure (OE) compliance. Within those categories, STRS compliance may be categorized as static or dynamic. Static STRS compliance is whether the code, configuration files, and documentation conform to the STRS Architecture Standard. Dynamic STRS compliance is whether the components execute properly together to become a functioning STRS radio. This document is mostly concerned with static STRS compliance by inspecting code, configuration files, and documents.

STRS OE compliance is depicted in Figure 1:

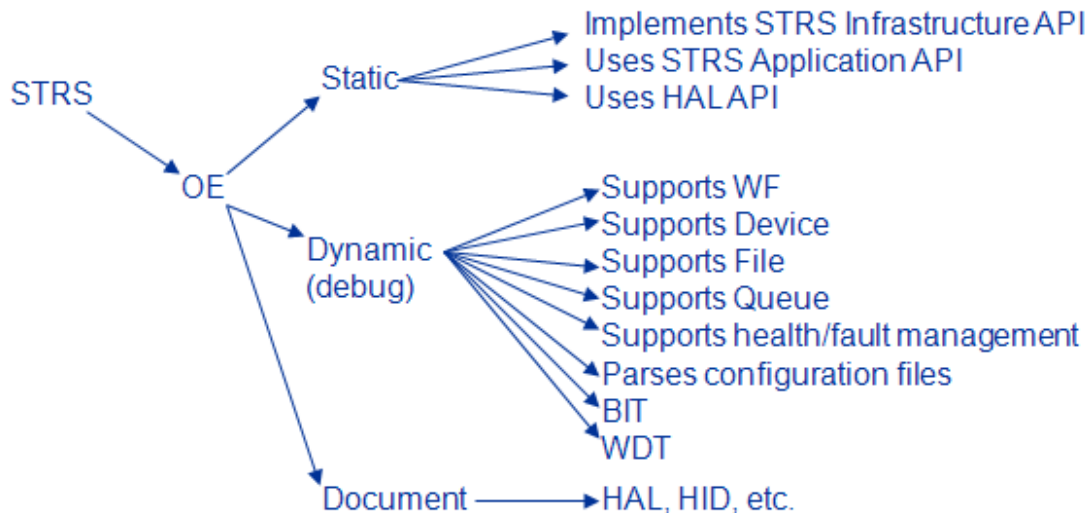


Figure 1 - Operating Environment Compliance

For the STRS OE, compliance includes:

- a. Implements required STRS infrastructure-provided APIs using standard “C” language interface
- b. Provides necessary header files for application developers
- c. Provides necessary run-time infrastructure to support STRS infrastructure-provided and STRS application-provided APIs
- d. Provides POSIX 1003.13 PSE51 conformant OS or a POSIX abstraction layer. Very small platforms can provide the minimum subset of PSE51 required to support mission waveforms (with a waiver)
- e. Verify configuration files, describing the platform resources, in XML, using the corresponding schema
- f. Test vendor supplied XML transformation tools with sample file
- g. Confirm documentation
 - i. Configuration files
 - ii. Flight computer
 - iii. Hardware Abstraction Layer (HAL) API

- iv. Hardware Interface Description (HID)
- v. Firmware interface to platform-specific wrapper
- vi. User's Guide
- vii. Reference Manual
- viii. Test plan
- h. Coding standards

For the STRS OE, compliance does not include:

- a. Memory footprint
- b. OE performance (Operations/second, interrupt response, etc.)
- c. Size, weight, and power (SWaP)
- d. Shake and bake, radiation tolerance
- e. NASA flight software/hardware requirements

STRS application compliance is depicted in Figure 2:

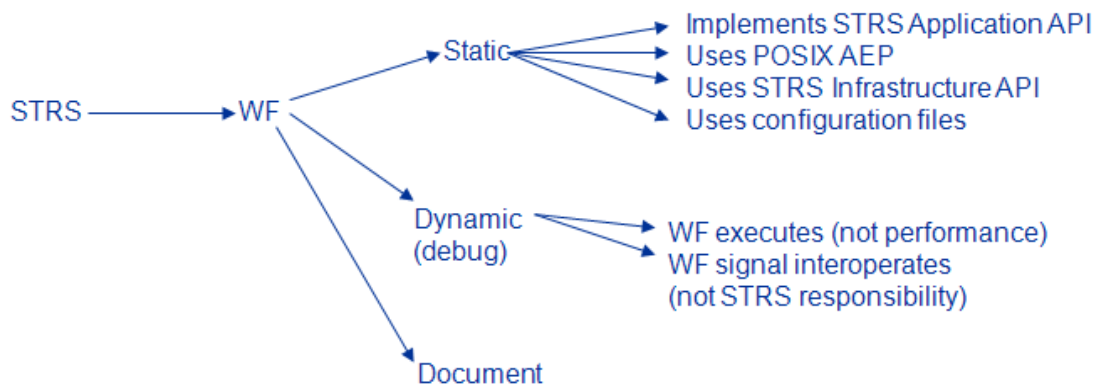


Figure 2 - Application Compliance

For an STRS application, compliance includes:

- a. Implements required STRS application-provided API per STRS Architecture Standard
- b. Verify that the only external interfaces called by the STRS application are the STRS infrastructure-provided APIs and allowed POSIX PSE51 APIs
- c. Verify that STRS application does not call restricted functions documented in section 8.5.1 of the STRS Architecture Standard
- d. Verify dynamic behavior of STRS application
 - i. Application responds properly to STRS application-provided API
 - ii. Application exhibits proper state transition behavior
- e. Verify that the STRS application uses the STRS predefined data
 - i. typedefs
 - ii. constants
 - iii. structs
- f. Verify configuration files, describing the STRS application, in XML, using the corresponding schema
- g. Test vendor supplied XML transformation tools
- h. Verify FPGA wrapper provided (if platform has FPGA)
- i. Verify documentation
 - i. Design
 - ii. Models
 - iii. Test plan
 - iv. User's Guide
 - v. FPGA Wrapper
- j. Model-based design

k. Coding standards

For an STRS application, compliance does not include:

- a. Memory footprint
- b. Over the air behavior of waveform
- c. Interoperability
 - i. Performance (BPS, BER, etc)
 - ii. Functional Requirements to meet missions objectives; other than being compliant with STRS architecture
- d. NASA flight software requirements

1.2 System Overview

Compliance testing determines to what degree the tested STRS application or STRS infrastructure in a software defined radio (SDR) implementation meets the STRS Architecture Standard. The purpose of having a standard architecture is to allow different companies to work together or separately, at the same or different times, to create a software defined radio. STRS requirements may be verified by inspecting documents, code, configuration files, and other artifacts, as well as observation. A component is compliant when a subset of the features in the specification is implemented in accordance with the architecture specification. A component is conformant when all the features in the architecture specification are implemented in accordance with the specification.

There are four methods used to test for STRS compliance described in this document: automated inspection, manual inspection, observation, and execution of an STRS application that tests the standard STRS capabilities. Although most of the automated tests may be performed manually, validation of the STRS OE, an STRS application, or STRS configuration file is facilitated by automated tools.

- Automated inspection uses a variety of tools to look for the required artifacts and lists those that are problematic.
- Manual inspection is used to augment the automated inspection methods especially when the other methods cannot be done.
- Observation is performed by the project verification and validation team.
- The insertion of an STRS application created by GRC (WFCCN) into the radio and execution of that waveform application to test the capabilities of the radio can be used to exercise all the required STRS application-provided methods beginning with “APP_” and STRS infrastructure-provided methods beginning with “STRS_”.

1.3 Document Overview

The document contains an Introduction, Applicable Documents, Test Configurations, and Test Descriptions. There is a table of all the requirements being tested, general test instructions, and test instructions associated with 4 groups of semi-automated tests and 2 groups of manual tests. The groups are separated by Infrastructure (OE) depicted in Figure 1 and Application (WF) depicted in Figure 2, as well as the available automated procedures.

In the event of a conflict between the requirements in this document and the requirements in the STRS Architecture Standard, the requirements in the STRS Architecture Standard shall take precedence over this document.

The artifacts to be tested and the results of testing are often proprietary to NASA and the supplier of the artifacts. The results of testing are used to inform the supplier of any deficiencies as well as providing lessons learned to NASA.

2.0 APPLICABLE DOCUMENTS

2.1 Reference Documents

Table 1 lists the number and title of all documents referenced in this specification.

Table 1 - Table of Documents

Document number	Document title
*	Altova XPLSpy® Professional Edition Online Manual http://manual.altova.com/XMLSpy/spyprofessional/
GLPR 7150.1	Glenn Research Center (GRC) Software Engineering Requirements
NPR 7150.2A	NASA Software Engineering Requirements
GRC-TPLT-STPr	Software Test Description Template
NASA/TM-2010-216809	Space Telecommunications Radio System (STRS) Architecture Standard, STRS_AR_0002, Revision 1.02.1. See http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20110002806_2011001718.pdf
NASA/TM-2008-215445	Space Telecommunications Radio System (STRS) Definitions and Acronyms. See http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20090005977_2009004914.pdf
STRS-OE-00002	Space Telecommunications Radio System (STRS) Reference Implementation User's Guide
*	Tornado development environment documentation in TVL
*	VxWorks operating system documentation in TVL
DOC-12067-ZD-00	VxWorks Programmers Guide 5.3.1 at http://www-cdfonline.fnal.gov/daq/commercial/vxworks_guide.pdf

3.0 TEST CONFIGURATIONS

3.1 Hardware Configuration

The SDR-3000 development PC in the Technology Verification Lab (TVL) was used for these tests because all the software (scripts, compiler, Subversion CM tool, GRC's STRS reference implementation, Cygwin, etc.) ran on it and the software to be tested was kept in Subversion CM tool accessible to the SDR-3000. The SDR-3000 uses an Ethernet connection to store and obtain artifacts to/from the Subversion CM server. The SDR-3000 and its development PC is shown in Figure 3. Other hardware platforms may work as well if the appropriate software is loaded.

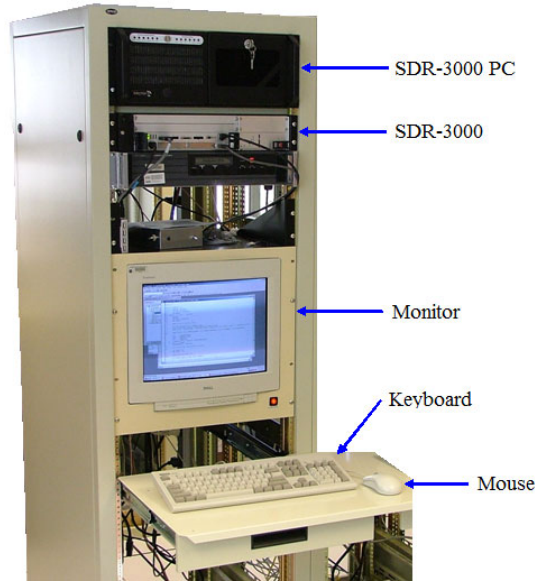


Figure 3 - SDR-3000 and Development Platform

Table 2 is used to record information about the hardware used for testing purposes.

Table 2 - Hardware Used

Name	Part	Model	Serial Number	Manufacturer	Revision Level	Calibration Date
SDR-3000				Spectrum Signal Processing	VxWorks 5.5.1	NA
SDR-3000 Development PC	800-00257		SS4903-00132	Spectrum Signal Processing	Win2000 SP4	NA

3.2 Software Configuration

This paragraph describes the procedures necessary to prepare the item(s) under test and any related software, including data, for the test. Reference may be made to published software manuals for these procedures. The following information is to be provided, as applicable:

- a) The specific software to be used in the test including version number and name
 1. STRS Application Automated Testing
 - a. Windows batch file: STRS/Compliance/ComplianceTool.bat
 - b. Bourne shell script: STRS/Compliance/ComplianceTool.sh
 - c. STRS application files in Subversion to be tested
 2. STRS Infrastructure Automated Testing
 - a. Windows batch file: STRS/Compliance/ComplianceToolOE.bat
 - b. Bourne shell script: STRS/Compliance/ComplianceToolOE.sh
 - c. STRS infrastructure files in Subversion to be tested

3. STRS Infrastructure Testing Using WFCCN
 - a. STRS application source file CommandAndComplianceApplication.cpp
 - b. STRS application header file CommandAndComplianceApplication.h
 - c. VxWorks cross-compiler ccppc or equivalent
 - d. STRS infrastructure files in Subversion to be tested
 4. STRS Configuration File Testing
 - a. XMLSpy or equivalent
 - b. STRS configuration files in Subversion to be tested
- b) The storage medium of the item(s) under test (e.g., magnetic tape or diskette)
Hard disk
- c) The storage medium of any related software (e.g., simulators, test drivers, or databases)
Hard disk

Table 3 is used to record information about the software used on the SDR-3000.

Table 3 - Software Used

Name of software	Version number
ComplianceTool.bat (Windows batch file)	SVN 1067 (12/4/2009)
ComplianceTool.sh (Bourne shell script)	SVN 1118 (1/27/2010)
ComplianceToolOE.bat (Windows batch file)	SVN 1069 (12/4/2009)
ComplianceToolOE.sh (Bourne shell script)	SVN 1624 (7/13/2010)
CommandAndComplianceApplication.cpp (WFCCN)	SVN 1099 (1/12/2010)
CommandAndComplianceApplication.h (WFCCN)	SVN 1070 (12/4/2009)
ccppc	gcc-2.96
Cygwin - GNU bash	2.05b.0(1)
Tortoise SVN	1.6.18415
XMLSpy	Altova XMLSpy Professional 2011 (COTS)

3.3 Other Pretest Preparations

Find and copy the STRS application (WF) and/or infrastructure (OE) items being tested to the computer on which the tests will be executed. For example, at GRC, enter the items to be tested into Subversion using the SDR-3000 or other PC in the TVL and extract from Subversion to the SDR-3000.

4.0 TEST DESCRIPTIONS

There are multiple test procedures used to test for STRS compliance. The requirements are all shown in Table 19 even though only certain tests have been automated and only certain tests can have detailed descriptions of how to perform an inspection.

4.1 Requirements Being Tested

The full range of tests is listed in Table 19. To aid the tester, there are four types of semi-automated tests. Two types of manual source code tests are also described.

1. For STRS Application Automated Testing, the requirements in Table 19 have the entry for **OE/App** as App and the entry for **Tested** as Script.
2. For STRS Infrastructure Automated Testing, the requirements in Table 19 have the entry for **OE/App** as OE and the entry for **Tested** as Script.
3. For STRS Infrastructure Testing Using WFCCN, the requirements in Table 19 have the entry for **OE/App** as OE and the entry for **Tested** as WFCCN.
4. For STRS Configuration File Testing, the requirements in Table 19 have the entry for **OE/App** as App and the entry for **Tested** as XMLSpy.
5. For STRS Manual Application Testing, the requirements in Table 19 have the entry for **OE/App** as App and the entry for **Tested** as Inspect.
6. For STRS Manual Infrastructure Testing, the requirements in Table 19 have the entry for **OE/App** as OE and the entry for **Tested** as Inspect.

4.2 Test Instructions

There are no general instructions for executing the test procedures. If there are any questions or problems that are not resolved by this document or the [referenced documents](#), email: STRS@lists.nasa.gov.

4.3 STRS Compliance Testing

STRS compliance testing is performed on software defined radio artifacts. Even when there is an automated procedure, it is necessary to check any warnings or errors manually to be sure that the error or warning does not indicate a false positive. The general procedure is shown in Figure 4.

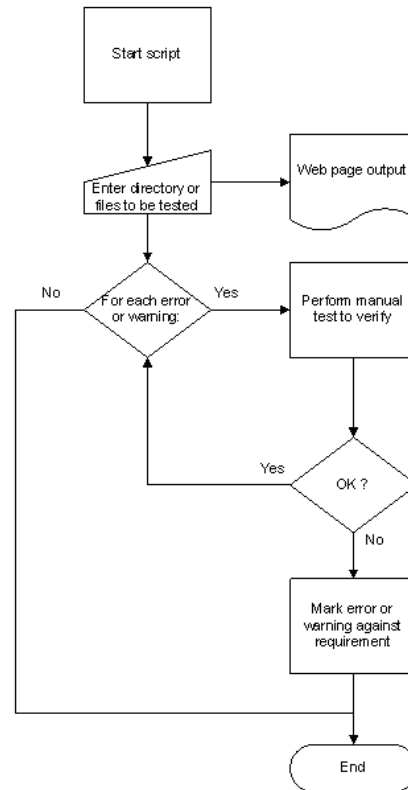


Figure 4 - General Automated Procedure with Manual Double-Check

4.3.1 Prerequisite Conditions, Assumptions, and Constraints

The prerequisite conditions and assumptions are minimal:

- a) For testing of code and configurations files, the PC containing those artifacts must be available and turned on. For GRC's testing, the STRS-3000 development PC must be available and turned on.
- b) The test shell scripts are limited to testing method signatures, prototypes, constants, and typedefs that appear entirely on a single line. Lines of code with errors or warnings are then tested with a manual procedure, directly against the requirement.
- c) Use a browser on the reviewer's PC to access documents in eRoom or CMTS.
- d) The artifacts to be tested and results of testing are to be considered proprietary to NASA and the company submitting the artifacts.
- e) It is assumed that everything works as described. For unusual situations, refer to the documentation in Table 1 - Table of Documents.
- f) Waivers or exceptions are the project responsibility, not STRS; however, they must be documented and submitted to the STRS repository.

4.3.2 Test Procedure for STRS Application Automated Testing

A UNIX Bourne shell script named ComplianceTool.sh was written to test for the various STRS application-provided method signatures beginning with "APP_" and STRS infrastructure-provided method signatures beginning with "STRS_" as well as to search for deprecated methods, disallowed POSIX methods, and non-portable QuicComm methods. The script also tests for "non-standard APP methods", "extra APP methods", and "extra STRS methods".

The compliance tool looks for the required artifacts in the source code and lists those that are problematic. The number of required STRS application-provided methods beginning with “APP_” may vary depending on the application. If the STRS application is a source of data supplied to the infrastructure, the standard requires the STRS application to have an APP_Read method. If the STRS application is a sink that receives data from the infrastructure, the standard requires the STRS application to have an APP_Write method. The STRS Architecture Standard defines 43 distinct STRS infrastructure-provided methods beginning with “STRS_”. The maximum number of required STRS application-provided methods beginning with “APP_” that an STRS application can have is 11 and the minimum number is 9. The STRS application-provided methods beginning with “APP_” are described in the STRS Architecture Standard.

The following terminology is used in the statistics. “Full signature” means a method declaration or definition as described in the STRS Architecture Standard including the return type, name, and definition of the arguments. “Non-standard APP methods” represent those STRS application-provided methods beginning with “APP_” that do not contain a full signature as described in the STRS Architecture Standard. “Extra APP methods” represent those methods that begin with “APP_” but are not defined in the STRS Architecture Standard. Although APP_SetBT is included in the reference implementation, APP_SetBT is not standard and is included as an extra method. “Distinct STRS methods out of 43” report how many of the Standard’s 43 methods occur. “Extra STRS methods” represent those methods that begin with “STRS_” but are not defined in the STRS Architecture Standard.

The compliance tool was written for the Bourne shell (sh) and may be executed by any superset of sh such as the Bourne-again shell (bash), which is available with Cygwin on the SDR-3000 (or OE1). Cygwin is a Linux-like environment for Windows. For more information on Cygwin, see <http://www.cygwin.com/>. There is also a MS Windows batch file that may be used to execute ComplianceToolOE.sh by either double clicking on ComplianceTool.bat or entering the file location in a DOS command window. ComplianceTool.bat is in the same directory as ComplianceTool.sh.

The step-by-step procedure for performing these tests is shown in Table 4. A test operator should fill in the blank columns and additional information following the table to show compliance.

Table 4 - STRS Application Test Automated Procedure

Step	Requirement ID	Test Operator Action	Expected Result	Actual Result	Pass or Fail
1		Find/obtain the compliance testing script(s) and put into a common directory. The files are: <u>ComplianceTool.sh</u> , <u>ComplianceTool.bat</u> , and <u>removeComments.awk</u> ,which may be obtained in the TVL from the Subversion configuration management tool at directory: <u>ComplianceToolScript/</u> or the files may be found in the CoNNeCT eRoom at: <u>CoNNeCT > Principal Investigator > STRS > Compliance ></u> Record the directory where the compliance testing scripts are stored.			
2		Find the STRS application source files to be tested. Record the directory where the files are stored.			
3		Change directory to that containing the compliance tools.			
4		Start test by typing sh ComplianceTool.sh in a UNIX (Cygwin, bash) command window or double-clicking: ComplianceTool. The script will display progress as it executes.	Compliance tool prompts for source directory or file. The prompt string is “Enter directory or source file to test for STRS compliance (or Q, C, or ?):”.		

Space Telecommunications Radio System (STRS) Compliance Testing

5		Enter source directory or file to test. The file or directory may be absolute or relative to the directory containing the ComplianceTool scripts. Summary of numbers of files and methods found and errors is displayed. Enter an upper or lower case Q to exit the script. Enter an upper or lower case C to search for directories containing files with extensions h, H, c, C, cpp, and CPP; however, directories containing /components/, /corba/, /public_tools/, and /sca/ are eliminated. The C entry is not usually used because it takes a long time and still doesn't necessarily look in the right places.	Output file web page named Complianceyyyymoddhmnss.html is created for each execution of the script where yyyymoddhmnss is the date and time when the script is invoked. For example, see Table 13.		
6		Display output file in a browser.	For example, see Table 14.		
7	STRS-10, STRS-20, STRS-23, STRS-26, STRS-29, STRS-30, STRS-31, STRS-32, STRS-33, STRS-34, STRS-35, STRS-36, STRS-37, STRS-38, STRS-39, STRS-91	Errors are in red and warnings are in blue on the web page output. Check errors and warnings manually against requirements to eliminate false positives. Record errors, potential problems, and discrepancies. Note the associated requirement, if appropriate.			
8		To be sure that there are no additional problems, a manual code inspection should be performed based on the additional data following the summary information. The additional output displays the file name, line number, and actual line where the problem or potential problem occurred. These should be examined and the errors corrected to attain compliance.			

Verification (Pass/fail): _____

Comments: _____

Test operator: _____ Date: _____

Product assurance: _____ Date: _____

The Test Operator Action should provide enough detail to enable successful repetition of the test.

1. To halt or interrupt the test procedure, press Control-C
2. If the shell script does not execute properly, the test operator may obtain additional data reflecting the execution of the shell script by adding the “xv” options to the script invocation; i.e., from

```
sh .\ComplianceTool.sh
```

to

```
sh -xv .\ComplianceTool.sh
```

4.3.3 Test Procedure for STRS Infrastructure Automated Testing

A UNIX Bourne shell script was written to test for the various STRS required files, typedefs, constants, and structs required to be provided in the STRS infrastructure. The required typedefs, constants, and structs are described in the STRS Architecture Standard. The shell script creates a web page named ComplianceOExxxxmoddhmnss.html for each execution of the script where xxxxmoddhmnss is the date and time when the script is invoked. The shell script checks whether the typedefs, constants, and structs are defined in STRS.h or in a #include file referenced by STRS.h contained in the same directory.

The OE compliance tool was written for the Bourne shell (sh) and may be executed by any superset of sh such as the Bourne-again shell (bash), which is available with Cygwin on the SDR-3000 in the TVL. Cygwin is a Linux-like environment for Windows. For more information on Cygwin, see <http://www.cygwin.com/>. There is also a MS Windows batch file that may be used to execute ComplianceToolOE.sh by either double clicking on ComplianceToolOE.bat or entering the file location in a DOS command window.

The step-by-step procedure for performing these tests is shown in Table 5. A test operator should fill in the blank columns and additional information following the table to show compliance.

Table 5 - STRS Infrastructure Test Automated Procedure

Step	Requirement ID	Test Operator Action	Expected Result	Actual Result	Pass or Fail
1		Find/obtain the OE compliance testing scripts. The files are <u>ComplianceToolOE.sh</u> , <u>ComplianceToolOE.sh</u> , and <u>removeComments.awk</u> , which may be obtained in the TVL from the Subversion configuration management tool at directory: <u>ComplianceToolScript/</u> or the files may be found in the CoNNeCT eRoom at: <u>CoNNeCT > Principal Investigator > STRS > Compliance ></u> Record the directory where the OE compliance testing scripts are stored.			
2		Find the STRS infrastructure source files to be tested including STRS.h. Record the directory where the files are stored.			
3		Change directory to that containing the compliance tools.			
4		Start test by typing <u>sh ComplianceToolOE.sh</u> in a UNIX (Cygwin, bash) command window or double-clicking:	Compliance tool prompts for source directory or file. The prompt string is “Enter directory or STRS.h source file to test for		

Space Telecommunications Radio System (STRS) Compliance Testing

Date: May 31, 2011

Document No.: STRS-ATP-00001

Page 17 of 46

		ComplianceToolOE. The script will display progress as it executes.	STRS OE source compliance (or Q):”		
5		Enter source directory containing STRS.h or path to file STRS.h. The directory may be absolute or relative to the directory containing the ComplianceToolOE scripts. Names of files processed are displayed as well as a summary of errors. Enter an upper or lower case Q to exit the script. Enter an upper or lower case C to search for directories containing files STRS.h; however, directories containing /components/, /corba/, /public_tools/, and /sca/ are eliminated. The C entry is not usually used because it takes a long time and doesn't necessarily look in the right places.	Output file web page named ComplianceOeyyyymoddhmns s.html is created for each execution of the script where yyymoddhmns is the date and time when the script is invoked. For example, see Table 15.		
6		Display the output file in a browser.	For example, see Table 16.		
7	STRS-17, STRS-89	Errors are in red on the web page output. Check errors and warnings manually against requirements to eliminate false positives. Record errors, potential problems, and discrepancies. Note the associated requirement, if appropriate.			
8		To be sure that there are no additional problems, a manual code inspection should be performed based on the additional data following the summary information. The additional output displays the file name, line number, and actual line where the problem or potential problem occurred. Details are displayed showing each individual missing item. The STRS infrastructure-provided method prototypes missing from the OE are also displayed. These should be examined and the errors corrected to attain compliance.			
9		The files containing the prototypes for each STRS infrastructure-provided method are displayed so that the proper #include statements may be added to STRS applications as necessary. See section 4.3.4 below.			

Verification (Pass/fail): _____

Comments: _____

Test operator: _____ Date: _____

Product assurance: _____ Date: _____

The Test Operator Action should provide enough detail to enable successful repetition of the test.

1. To halt or interrupt the test procedure, press Control-C.
2. If the shell script does not execute properly, the test operator may obtain additional data reflecting the execution of the shell script by adding the “xv” options to the script invocation; i.e., from

```
sh .\ComplianceToolOE.sh
```

to

```
sh -xv .\ComplianceToolOE.sh
```

4.3.4 Test Procedure for STRS Infrastructure Testing Using WFCCN

Because of the complexity allowed in C/C++, a good way to verify that an STRS application and the STRS infrastructure work together properly is by compiling, linking, and executing. To that end, GRC has developed an STRS application that implements all the required STRS application-provided (APP_) methods and uses all the STRS infrastructure-provided (STRS_) methods. This command and control application, WFCCN, is inserted into the radio and executed to test the compliance of the STRS radio infrastructure to the STRS Architecture Standard both statically in the porting process as well as dynamically in its execution. The WFCCN porting process including compilation and linking should perform most of the STRS OE static checks as well as STRS application static checks. Linking WFCCN demonstrates the existence of the necessary run-time infrastructure to support STRS infrastructure-provided and STRS application-provided APIs.

The WFCCN execution performs many of the dynamic checks. The dynamic tests may be performed individually by APP_RunTest except for the one that tests STRS_RunTest with a target of WFCCN, which would cause an infinite loop. Any of the tests may be performed by APP_Start by specifying the appropriate value of START_TESTS shown in Table 17. APP_Start may call each APP_RunTest except for the one that tests STRS_Start with a target of WFCCN, which would cause an infinite loop. The dynamic tests executed by APP_RunTest are shown in Table 18. When the value of the variable named START_TESTS is “YES”, APP_Start performs all the tests shown in Table 18 in test ID order. Some of the tests with complex dependencies may return errors without generating a non-conformance. The test ID values shown in Table 18 assume that STRS_TEST_STATUS is zero and that STRS_TEST_USER_BASE is one, as defined in GRC’s reference implementation. If that is not the case, zero would be replaced by the value of STRS_TEST_STATUS, one would be replaced by the value of STRS_TEST_USER_BASE, two would be replaced by the value of STRS_TEST_USER_BASE+1, etc. Note that there is one method missing from this list because it is not tested independently. STRS_FileGetStreamPointer is tested when STRS_FileOpen is tested. STRS_FileGetStreamPointer can only be tested when the file is open.

The step-by-step procedure for performing these tests is shown in Table 6. A test operator should fill in the blank columns and additional information following the table to show compliance.

Table 6 - STRS WFCCN Test Automated Procedure

Step	Requirement ID	Test Operator Action	Expected Result	Actual Result	Pass or Fail
1		Find/obtain the OE compliance testing command and control application, WFCCN. The files are: <u>CommandAndComplianceApplication.cpp</u> and <u>CommandAndComplianceApplication.h</u> which may be obtained in the TVL from the Subversion configuration management tool at directory <u>WFCCN/</u> or the files may be found in the CoNNeCT eRoom at: <u>CoNNeCT > Principal Investigator > STRS > Compliance ></u> . Record the directory where the files are stored.	WFCCN\ <u>CommandAndComplianceAp</u> <u>plication.cpp</u> and WFCCN\ <u>CommandAndComplianceAp</u> <u>plication.h</u>		

Space Telecommunications Radio System (STRS) Compliance Testing

Date: May 31, 2011

Document No.: STRS-ATP-00001

Page 19 of 46

2	STRS-89	Find STRS infrastructure source files to test including STRS.h. Record the directory where the files are stored.	vendorOE\STRS.h		
3		Create a directory in GRC's reference implementation for the compilation of WFCCN for the particular OE being tested in the same directory as CommandAndComplianceApplication.cpp and CommandAndComplianceApplication.h.	WFCCN\myOE\		
4		Determine file names containing STRS infrastructure prototypes. Record the filenames and directory(s). This is facilitated by the last step of the previous section (section 4.3.3, step 9).	vendorOE\prototypeName.h		
5		Create a file named STRS_APIs.h containing a #include statement for each STRS infrastructure prototype file. This is facilitated by the last step of the previous section (section 4.3.3, step 9).	WFCCN\myOE\STRS_APIs.h		
6		Edit the makefile or set up the IDE to compile CommandAndComplianceApplication.cpp, using CommandAndComplianceApplication.h; the prototypes, constants, structs, and typedefs of the infrastructure to be tested; and the prototypes of step 5.			
7	STRS-19, STRS-40, STRS-41, STRS-42, STRS-43, STRS-44, STRS-45, STRS-46, STRS-47, STRS-48, STRS-49, STRS-50, STRS-51, STRS-52, STRS-53, STRS-54, STRS-55, STRS-56, STRS-57, STRS-58, STRS-59, STRS-61, STRS-62, STRS-63, STRS-64, STRS-65, STRS-66, STRS-67, STRS-68, STRS-69,	Compile CommandAndComplianceApplication.cpp and analyze compilation outputs manually. Output errors and warnings indicate discrepancies between WFCCN and the infrastructure to be tested. Record errors, potential problems, and discrepancies. Note the associated requirement, if appropriate.			

Space Telecommunications Radio System (STRS) Compliance Testing

Date: May 31, 2011

Document No.: STRS-ATP-00001

Page 20 of 46

	STRS-70, STRS-71, STRS-72, STRS-73, STRS-74, STRS-75, STRS-76, STRS-78, STRS-79, STRS-80, STRS-81, STRS-83, STRS-84, STRS-85, STRS-86, STRS-87, STRS-88, STRS-89, STRS-95				
8		Determine whether WFCCN can be configured and controlled to perform dynamic testing. If WFCCN can be executed in the infrastructure to be tested, continue; otherwise, stop here.	Usually WFCCN cannot be used to perform dynamic testing and the process stops here.		NA
9	STRS-99, STRS-100	Create WFCCN configuration file in XML as described in documentation for the OE to be tested. Determine values for all items in Table 17 with START_TESTS set to YES.	WFCCN\myOE\WFCCN.xml		
10	STRS-104	Transform to deployed form as described in documentation for OE to be tested.	WFCCN\myOE\WFCCN.cfg		
11		If needed by the OE, perform any additional modifications of CommandAndComplianceApplication.cpp, CommandAndComplianceApplication.h, and recompile. Need to recompile, for example, when WFCCN must be compiled as part of OS.			
12		Start OE and WFCCN. Record errors, potential problems, and discrepancies. Note the associated requirement, if appropriate.			

Verification (Pass/fail): _____

Comments: _____

Test operator: _____ Date: _____

Product assurance: _____ Date: _____

The Test Operator Action should provide enough detail to enable successful repetition of the test.

1. To halt or interrupt the test procedure in step 9, press Control-C.
2. If WFCCN does not execute properly, the test operator may obtain additional data reflecting the execution of WFCCN using another IDE (e.g. Tornado) window.
3. STRS_APIs.h is the only file that should be changed (in step 2) to reflect the specific names of the prototype files in the vendor's OE.
4. CommandAndComplianceApplication.cpp and CommandAndComplianceApplication.h may be changed (in step 8) if the infrastructure being tested dynamically has special start-up methods that must be implemented or certain restrictions pertaining to output and logging in the infrastructure being tested.

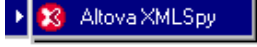
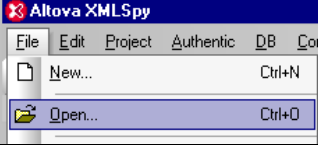
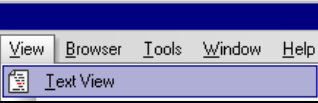


4.3.5 Test Procedure for STRS Configuration File Testing

An area of both inspection and semi-automated testing is the STRS configuration file in XML and its transformation. Although other products are available, GRC uses the COTS product XMLSpy to verify that the XML schema matches the XML data. This test procedure is described for XMLSpy since that is what GRC used.

The step-by-step procedure for performing these tests is shown in Table 7. A test operator should fill in the blank columns and additional information following the table to show compliance.

Table 7 - STRS Configuration Files Test Procedure

Step	Requirement ID	Test Operator Action	Expected Result	Actual Result	Pass or Fail
1	STRS-100	Find the STRS platform integrator's pre-deployed application configuration file in XML.			
2	STRS-101	Verify that the pre-deployed application configuration file found in step 1 contains the following application attributes and default values: <ul style="list-style-type: none"> • Identification • Unique STRS handle name for the application • Class name (if applicable) • State after processing the configuration file • Required resources: memory in bytes or number of gates or logic elements • Configuration parameters containing the STRS handle, names of files, devices, queues, waveforms and services needed by the STRS application • Values and constraints for all operationally configurable parameters • Filename(s) of loadable images for resources 			
3	STRS-102	Find the STRS platform provider's XML schema to validate the format and data for pre-deployed STRS application configuration files, including the order of the tags, the number of occurrences of each tag, and the values or attributes.			
4		Verify that the pre-deployed application configuration file found in step 1 contains a tag for an XMLSchema-instance. Verify that the name of the file matches the one found in step 4. An example for GRC's referenced implementation is: <STRS			

		<pre>xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="STRS.xsd"></pre>			
5		<p>Start up Altova XMLSpy.</p> 	XMLSpy window appears		
6		<p>Using file menu, open the pre-deployed configuration files in XML and corresponding schema definition files (XSD) using XMLSpy.</p> 	XMLSpy window displays files (tabbed).		
7		<p>Using view menu, change to text view, if not already there.</p> 	XMLSpy window displays text.		
8		<p>Check well-formedness by clicking on icon or pressing F7.</p> 	XMLSpy message window displays check mark in yellow circle ☺ and text: File X is well-formed.		
9		<p>Perform validation by clicking on icon or pressing F8.</p> 	XMLSpy message window displays check mark in green circle ✔ and text: File X is valid.		
10	STRS-98	Find the STRS platform provider's documentation of the necessary platform information to develop a pre-deployed application configuration file in XML (including a sample file).			
11	STRS-99	Find the STRS application developer's documentation of the necessary application information to develop a pre-deployed application configuration file in XML.			
12	STRS-103	<p>Use the STRS platform provider's tools to transform pre-deployed application configuration file in XML into a deployed application configuration file.</p> <p>Check the pre-deployed application configuration file found in step 1 for an xml-stylesheet to define a transformation.</p>			

13	STRS-104	Find the STRS platform integrator's deployed STRS application configuration file for the STRS infrastructure.			
----	----------	---	--	--	--

Verification (Pass/fail): _____

Comments: _____

Test operator: _____ Date: _____

Product assurance: _____ Date: _____

4.3.6 Test Procedure for STRS Application Manual Code Testing

The automated testing described above leaves many areas untested. Manual code compliance includes inspection of application artifacts to verify:

- a. The usage of infrastructure-provided interfaces; i.e., that the only external interfaces called by the STRS application are the STRS infrastructure-provided APIs and allowed POSIX PSE51 APIs (STRS-10, STRS-91).
- b. That the application implements the appropriate functionality and exhibits proper state transition behavior.
- c. That the application has the appropriate C++ class hierarchy, if written in C++.
- d. That the application software artifacts have been submitted to the STRS application repository (STRS-12).
- e. FPGA wrapper is provided if the platform has an FPGA (STRS-14).

The step-by-step procedure for performing these tests is shown in Table 8. A test operator should fill in the blank columns and additional information following the table to show compliance.

Table 8 - STRS Application Test Manual Procedure

Step	Requirement ID	Test Operator Action	Expected Result	Actual Result	Pass or Fail
1	STRS-10, STRS-91	Verify that the only external interfaces called by the STRS application are the STRS infrastructure-provided APIs and allowed POSIX PSE51 APIs.			
2	STRS-12	Find application development artifacts submitted to the NASA STRS Repository. Find appropriate license agreements. Verify that the application development artifacts include the following: <ul style="list-style-type: none"> • High level system or component software model • Documentation of application firmware external interfaces (e.g. signal names and descriptions, signal polarity and format, timing constraints of signals) • Documentation of STRS application behavior • Application function sources 			

Space Telecommunications Radio System (STRS) Compliance Testing

Date: May 31, 2011

Document No.: STRS-ATP-00001

Page 24 of 46

		<p>(e.g. C, C++, header files, VHDL, Verilog)</p> <ul style="list-style-type: none"> • Application libraries, if applicable (e.g. EDIF, DLL) • Documentation of application development environment and tool suite • Test plan and results documentation • Identification of Flight Software Development Standards used 			
3	STRS-14	Verify that FPGA wrapper is provided (if platform has FPGA)			
4	STRS-16	Determine whether the STRS <i>Application-provided Application Control API</i> is implemented using C or C++.			
5	STRS-22	If the <i>STRS Application-provided Application Control API</i> is implemented in C++, verify that the STRS application class is derived from the <i>STRS_ApplicationControl</i> base class.			
6	STRS-25	If the <i>STRS Application-provided Application Control API</i> is implemented in C++ AND the STRS application provides the <i>APP_Write</i> method, verify that the STRS application class is derived from the <i>STRS_Sink</i> base class.			
7	STRS-28	If the <i>STRS Application-provided Application Control API</i> is implemented in C++ AND the STRS application provides the <i>APP_Read</i> method, verify that the STRS application class is derived from the <i>STRS_Source</i> base class.			
8	STRS-60	Verify that the STRS applications use the STRS infrastructure <i>Device Control</i> methods to control the STRS Devices.			
9	STRS-77	Verify that the STRS applications use the <i>STRS Infrastructure Messaging</i> methods to send messages between applications and/or the infrastructure with a single target handle ID.			
10	STRS-82	Verify that any portion of the STRS Applications on the GPP needing time control uses the <i>STRS Infrastructure Time Control</i> methods to access the hardware and software timers.			

11	STRS-97	Verify that any STRS application uses the <i>STRS_Log</i> and <i>STRS_Write</i> methods to send STRS telemetry set information to the external system.			
----	---------	--	--	--	--

Verification (Pass/fail): _____

Comments: _____

Test operator: _____ Date: _____

Product assurance: _____ Date: _____

4.3.7 Test Procedure for STRS Infrastructure Manual Code Testing

The automated testing described above leaves many areas of the infrastructure untested. Additional compliance testing for the STRS OE may include:

- a. Provides POSIX 1003.13 PSE51 conformant OS or a POSIX abstraction layer. Very small platforms can provide the minimum subset of PSE51 required to support mission waveforms (with a waiver) (STRS-90)
- b. Verify that the STRS predefined data for typedefs, constants, and structs is provided in header file STRS.h (STRS-89)
- c. Provides necessary header files for application developers (STRS-20, STRS-21, STRS-24, STRS-27)

The step-by-step procedure for performing these tests is shown in Table 9. A test operator should fill in the blank columns and additional information following the table to show compliance.

Table 9 - STRS Infrastructure Test Manual Procedure

Step	Requirement ID	Test Operator Action	Expected Result	Actual Result	Pass or Fail
1	STRS-18	Verify that the STRS Operating Environment supports C or C++ language interfaces for the <i>STRS Application-provided Application Control API</i> at compile-time.			
2	STRS-21	Verify that the STRS platform provider supplied an “STRS_ApplicationControl.h” that contains the method prototypes and, for C++, the class definition for the base class STRS_ApplicationControl.			
3	STRS-24	Verify that the STRS platform provider supplied an “STRS_Sink.h” that contains the method prototypes and, for C++, the class definition for the base class STRS_Sink.			
4	STRS-27	Verify that the STRS platform provider supplied an “STRS_Source.h” that contains the method prototypes and, for C++, the class definition for the base class STRS_Source.			

Space Telecommunications Radio System (STRS) Compliance Testing

Date: May 31, 2011

Document No.: STRS-ATP-00001

Page 26 of 46

5	STRS-90	Verify that the STRS Operating Environment supplies the interfaces described in POSIX IEEE Standard 1003.13-2003 profile PSE51.			
6	STRS-96	Verify that the STRS infrastructure uses the <i>STRS_Query</i> method to service external system requests for information from an STRS application.			

Verification (Pass/fail): _____

Comments: _____

Test operator: _____ Date: _____

Product assurance: _____ Date: _____

APPENDIX A—Glossary and Acronyms

This section should include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of terms and definitions needed to understand this document.

A.1 Definitions

The glossary in Table 10 contains an alphabetized list of definitions for special terms used in the document; that is, the terms are used in a sense that differs from or is more specific than the common usage for such terms. STRS-specific terms are defined in NASA/TM-2008-215445.

Table 10 - Glossary

Term	Definition
Test case	Same as a test procedure.
Test procedure	A set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. Also referred to as test scripts.
Test suite	A collection of test procedures.

A.2 Acronyms

Table 11 contains the definitions for the abbreviations and acronyms used in this document.

Table 11 - Acronyms

Acronym	Definition
API	Application Programmers Interface
BER	Bit Error Rate
BIT	Built-In Test
BPS	Bits Per Second
FPGA	Field-Programmable Gate Array
HAL	Hardware Abstraction Layer
HID	Hardware Interface Description
ID	Identifier
OE	Operating Environment = OS & STRS Infrastructure
OS	Operating System
POSIX	Portable Operating System Interface for Unix
SDR	Software Defined Radio
STRS	Space Telecommunications Radio System
TVL	Technology Verification Lab at GRC
WDT	Watchdog Timer
WF	Waveform = STRS application
WFCCN	Command and Control Application
XML	Extensible Markup Language
XSD	XML Schema Definition

APPENDIX B—Traceability to SWE–114 of NPR 7150.2A*Table 12 - Traceability to SWE-114 of 7150.2A*

Document Section(s)	SWE–114 Requirement
	The STRS Compliance Test Procedures shall contain: [SWE-114]
3.0 Test Configuration	a. Test preparations, including hardware and software.
4.0 Test Descriptions	b. Test descriptions, including:
4.3 STRS Compliance Testing	1. Test identifier.
4.1 Requirements Being Tested, Table 19, Table 4, Table 5, Table 6, Table 7, Table 8, Table 9	2. System or CSCI requirements addressed by the test case.
4.3.1 Prerequisite Conditions, Assumptions, and Constraints	3. Prerequisite conditions.
Table 4, Table 5, Table 6, Table 7, Table 8, Table 9	4. Test input.
4.2 Test Instructions, Table 4, Table 5, Table 6, Table 7, Table 8, Table 9	5. Instructions for conducting procedure.
Table 4, Table 5, Table 6, Table 7, Table 8, Table 9	6. Expected test results, including assumptions and constraints.
Table 4, Table 5, Table 6, Table 7, Table 8, Table 9	7. Criteria for evaluating results.
Table 19	c. Requirements traceability.
3.0 Test Configuration	d. Identification of test configuration.

APPENDIX C—Application Compliance Testing Tables

Table 13 is an example of the execution of the application compliance test using ComplianceTool.sh as described in section 4.3.2 for the preliminary GRC/GSFC waveform in directory WFgsfc. Summary information is displayed during execution of the script to indicate its progress and show the number of errors.

Table 13 - Compliance Script Execution

```

Enter directory or source file to test for STRS compliance (or Q, C, or ?): WFgsfc
Entered: WFgsfc
Process directory WFgsfc
Found    3 files to test.
APP_:    19
STRS_:   131
2 POSIX, 0 deprecated, and 0 QuicComm method errors.
0 APP methods missing and 9 APP methods found correctly and 0 extra APP methods.
1 non-standard APP method definitions.
    131 STRS methods and 0 extra STRS methods.
Need to address 2 errors.
POSIX strtok not allowed (consider strtok_r).
...
...
2 POSIX, 0 deprecated, and 0 QuicComm method errors.
0 APP methods missing and 9 APP methods found correctly and 0 extra APP methods.
1 non-standard APP method definitions.
131 STRS methods and 0 extra STRS methods.
Need to address 2 errors.

```

Table 14 is an example of the output of the application compliance test using ComplianceTool.sh as described in section 4.3.2 saved as a web page so that color and formatting can be shown. From either type of output, one can see that there were 2 POSIX method calls that are not allowed and there were 0 deprecated method calls used for a total of 2 definite problems to address. The extra or non-standard methods used are probably not a problem but rather an artifact of the simple method of testing that is being used. In the example output below, the missing STRS include file was STRS.h which was in a #include within a #include file. This is a potential portability issue but not currently a non-conformance. The web page looks like:

Table 14 - Compliance Script Web Page Output

Directory WFgsfc/src

- Process directory WFgsfc/src

Occurrences	Item Examined
2	POSIX methods not allowed
0	Deprecated methods
0	QuicComm methods
0	Missing STRS_TEST_STATUS from APP_RunTest
9	APP required methods found out of 9
0	APP required methods missing
1	Non-standard APP method definitions or invocations
0	Extra APP methods
131	STRS methods
10	Distinct STRS methods out of 43
0	Extra STRS methods
1	Missing STRS include files
2	Need to address 2 errors

APPENDIX D—Infrastructure Compliance Testing Tables

Table 15 is an example of the execution of the infrastructure compliance test using ComplianceToolOE.sh as described in section 4.3.3 for GRC's reference implementation. Summary information is displayed to show the number of errors and the progress of the execution of the script.

Table 15 - OE Compliance Script Execution

```

Enter directory or STRS.h source file to test for STRS OE source compliance
(or Q): STRS_ReferenceImplementation/OE
Entered: STRS_ReferenceImplementation/OE
Process directory STRS_ReferenceImplementation/OE
Found    4 files to test.
Process: STRS.h
Skip: stdlib.h
Process: Property.h
Process: Properties.h

0 typedefs missing and 24 typedefs found correctly.
0 constants missing and 26 constants found correctly.
0 structs missing and 2 structs found correctly.
0 files missing and 4 files found correctly out of    4.
Need to address 0 errors.

```

Table 16 is an example of the output of the application compliance test using ComplianceToolOE.sh as described in section 4.3.3. The results are saved as a web page so that color and formatting can be shown. From either type of output, one can see that there are no errors left in the NASA GRC STRS reference implementation. The web page looks like:

Table 16 - OE Compliance Script Web Page Output

Directory STRS_ReferenceImplementation/OE

- Process directory STRS_ReferenceImplementation/OE

Occurrences	Item Examined
0	typedefs missing
24	typedefs found correctly
0	constants missing
26	constants found correctly
0	structs missing
2	structs found correctly
0	files missing
4	files found correctly

The web page also displays a list of #include statements needed for the STRS infrastructure-provided methods. This list may be used to create STRS_APIs.h in section 4.3.4 (WFCCN).

APPENDIX E—Infrastructure Compliance Testing by WFCCN Tables

Table 17 shows the properties to be configured by WFCCN used to test for dynamic compliance as described in section 4.3.4.

Table 17 - WFCCN Configurable Data

Item	Name	Description
1	ABORT_NAME	Value is handle name of application to abort. This should not be WFCCN.
2	ACCESS	Value for file open may be: READ, WRITE, BOTH, or APPEND.
3	DEVICE_LOAD	Value is file name to be loaded into Device. This is usually a bit file produced as the result of VHDL processing.
4	DEVICE_NAME	Value is handle name of Device.
5	FILE_NAME	Value is file or directory name.
6	FILE_RENAME	Value is file name to remove or rename to.
7	FILE_TYPE	Value for file open may be: BINARY or TEXT.
8	HANDLE_NAME	Value is a handle name to look up for STRS_HandleRequest.
9	IO_DATA	Value is data for testing STRS_Log, STRS_Write, and STRS_Read.
10	MSG	Value is data for testing for APP_Read.
11	PRIORITY	Value of priority when creating a queue may be: LOW, MEDIUM, or HIGH.
12	QUEUE_NAME	Value is name of queue to create.
13	QUEUE_TARGET	Value is name of subscriber queue.
14	QUEUE_TYPE	Value of queue type may be: SIMPLE or PUBSUB.
15	READ_NAME	Value is handle name of target for STRS_Read.
16	RELEASE_NAME	Value is handle name of application to release.
17	START_TESTS	Value is YES to start all APP_RunTests in APP_Start, NO to wait and start individual tests when requested, or an individual test ID number as shown in Table 18.
18	TEST_ID	Value is the appropriate test ID number for testing STRS_RunTest or STRS_GroundTest.
19	TIMER_DELTA	Value is timer offset from base used by STRS_SetTime.
20	TIMER_KIND	Value is kind of timer used for testing STRS_GetTime and STRS_SetTime.
21	TIMER_NAME	Value is handle name of timer.
22	TIMER_REF	Value is kind of reference timer to synchronize to.
23	TIMER_TGT	Value is kind of timer to synchronize.
24	USE	Value is handle name of target.
25	WRITE_NAME	Value is handle name of target for STRS_Write.

Table 18 associates a test ID with a test of each infrastructure-provided method and describes the data that is configured to run the test. These tests are usually done multiple times; once en mass by configuring START_TESTS=YES and then individually as needed.

Table 18 - WFCCN tests in APP_RunTest

Test ID	Test API or other	Description
0	STRS_TEST_STATUS	Obtains state information.
1	STRS_IsOK	Verifies for each type of error that the return value is valid.
2	STRS_GetErrorQueue	Verifies for each matching constant error and error queue that the handle ID of the error queue equals STRS_GetErrorQueue(error).
3	STRS_InstantiateApp	Configuration file is value(FILE_NAME).
4	STRS_Configure	Target to configure is value(USE) with properties given for APP_RunTest.
5	STRS_Query	Target to query is value(USE) and property names are those given for APP_RunTest, if there are any defined; otherwise, no property names are given but the STRS_Properties structure contains room for many to be queried.
6	STRS_RunTest	Target to test is value(USE) with test ID specified as value(TEST_ID) and properties given for APP_RunTest.
7	STRS_GroundTest	Target to test is value(USE) with test ID specified as value(TEST_ID) and properties given for APP_RunTest.
8	STRS_Initialize	Target to initialize is value(USE).
9	STRS_Start	Target to start is value(USE).
10	STRS_DeviceLoad	Target Device to load is value(DEVICE_NAME) with file specified as value(DEVICE_LOAD).
11	STRS_DeviceOpen	Target Device to open is value(DEVICE_NAME).
12	STRS_DeviceReset	Target Device to reset is value(DEVICE_NAME).
13	STRS_DeviceStart	Target Device to start is value(DEVICE_NAME).
14	STRS_FileGetFreeSpace	Target file system (if needed) is value(FILE_NAME).
15	STRS_FileOpen	Target file is value(FILE_NAME), file access is value(Access), and file type is value(FILE_TYPE).
16	STRS_QueueCreate	Name of queue to create is value(Queue_Name), type of queue is value(Queue_Type), and priority of queue is value(Priority).
17	STRS_Register	Target to register as a publisher is value(Queue_Name). Target to register as a subscriber is value(Queue_Target).
18	STRS_Log	Target to write log to is value(USE) and data is value(IO_DATA).
19	STRS_Write	Target to write is value(WRITE_NAME) and data is value(IO_DATA).
20	STRS_Read	Target to read is value(READ_NAME) and size of data is determined from value(IO_DATA).
21	STRS_Unregister	Target to unregister is value(Queue_Name) and subscriber is value(Queue_Target).
22	STRS_QueueDelete	Target to delete is value(Queue_Name).
23	STRS_FileClose	Target file to close is value(FILE_NAME).

Space Telecommunications Radio System (STRS) Compliance Testing

Date: May 31, 2011

Document No.: STRS-ATP-00001

Page 35 of 46

Test ID	Test API or other	Description
24	STRS_FileGetSize	Target file is value(FILE_NAME).
25	STRS_FileRename	File to rename is value(FILE_NAME) and file name to rename it to is value(FILE_RENAME).
26	STRS_FileRemove	Target file is value(FILE_RENAME).
27	STRS_DeviceStop	Target Device to stop is value(DEVICE_NAME).
28	STRS_DeviceFlush	Target Device to flush is value(DEVICE_NAME).
29	STRS_DeviceClose	Target Device to close is value(DEVICE_NAME).
30	STRS_SetISR	Target to set ISR is value(DEVICE_NAME).
31	STRS_DeviceUnload	Target Device to unload is value(DEVICE_NAME).
32	STRS_Stop	Target to stop is value(USE).
33	STRS_HandleRequest	Obtains the handle ID for value(HANDLE_NAME).
34	STRS_ReleaseObject	Target to release is value(RELEASE_NAME).
35	STRS_AbortApp	Target to abort is value(ABORT_NAME).
36	STRS_GetNanoseconds	Verifies a constant number of nanoseconds set in an STRS_TimeWarp item.
37	STRS_GetSeconds	Verifies a constant number of seconds set in an STRS_TimeWarp item.
38	STRS_GetTimeWarp	Verifies constants set in an STRS_TimeWarp item.
39	STRS_GetTime	Target to get time from is value(TIMER_NAME) and kind of timer is value(TIMER_KIND).
40	STRS_SetTime	Target to set time is value(TIMER_NAME), kind of timer is value(TIMER_KIND), and timer offset in seconds is value(TIMER_DELTA).
41	STRS_Synch	Target to synchronize is value(TIMER_NAME), kind of timer to synchronize is value(TIMER_TGT) and kind of timer to use as reference is value(TIMER_REF).
42	Predefined data	This tests whether the constants, typedefs, and structs may be used consistently.

APPENDIX F—COMPLIANCE TESTING BY REQUIREMENT

The following Table 19 shows which requirements are tested by STRS using manual inspection, observation, scripts, or porting WFCCN as described in previous sections. Table 19 shows whether the requirement applies to the OE, the platform, or the STRS application. The “OK?” column is added to use the table as a checklist. The table numbers that appear within Table 19 are references to the tables in the STRS Architecture Standard.

Table 19 - Requirements Testing

Requirements	Description	OE/App	Tested	OK?
STRS-1	An STRS platform shall have a known state after completion of the power-up process.	OE	Observe	
STRS-2	The STRS Operating Environment shall provide access to platform module’s diagnostic information via the STRS APIs.	OE	Observe	
STRS-3	Self diagnostic and fault detection data of a module shall be accessible to the STRS Operating Environment for collection.	OE	Observe	
STRS-4	The STRS platform provider shall describe in the HID document, the behavior and capability of each major functional device or resource available for use by waveforms, services, or other applications (e.g. FPGA, GPP, DSP, memory), noting any operational limitations.	Platform	Inspect document	
STRS-5	The STRS platform provider shall describe in the HID document, the reconfigurability behavior and capability of each reconfigurable component.	Platform	Inspect document	
STRS-6	The STRS platform provider shall describe in the HID document, the behavior and performance of the RF modular component(s).	Platform	Inspect document	
STRS-7	The STRS platform provider shall describe in the HID document, the interfaces that are provided to and from each modular component of the radio platform.	Platform	Inspect document	
STRS-8	The STRS platform provider shall describe in the HID document, the control, telemetry, and data mechanisms of each modular component (i.e. how to program or control each modular component of the platform, and how to use or access each device or software component, noting any proprietary aspects).	Platform	Inspect document	
STRS-9	The STRS platform provider shall describe in the HID document, the behavior and performance of any power supply or power converter modular component(s).	Platform	Inspect document	
STRS-10	An STRS application shall use the infrastructure STRS API and POSIX API for access to platform resources.	App	Script & Inspect	
STRS-11	The STRS infrastructure shall use the STRS Platform HAL APIs to communicate with application components on the platform specialized hardware via the physical interface defined by the platform provider.	OE	No	

Space Telecommunications Radio System (STRS) Compliance Testing

Date: May 31, 2011

Document No.: STRS-ATP-00001

Page 37 of 46

Requirements	Description	OE/App	Tested	OK?
STRS-12	<p>Application development artifacts shall be submitted to the NASA STRS Repository. The use will be subject to the appropriate license agreements. The application development artifacts shall include, as a minimum, the following:</p> <ul style="list-style-type: none"> • High level system or component software model • Documentation of application firmware external interfaces (e.g. signal names and descriptions, signal polarity and format, timing constraints of signals) • Documentation of STRS application behavior • Application function sources (e.g. C, C++, header files, VHDL, Verilog) • Application libraries, if applicable (e.g. EDIF, DLL) • Documentation of application development environment and tool suite • Test plan and results documentation • Identification of Flight Software Development Standards used 	App	Inspect	
STRS-13	<p>If the STRS application has a component resident in an SPM (e.g. FPGA firmware), then it shall accept configuration and control commands from the STRS Operating Environment.</p>	App	Observe	
STRS-14	<p>The STRS SPM developer shall provide a platform specific wrapper for each user-programmable FPGA on the SPM, which performs, as a minimum, the following functions</p> <ul style="list-style-type: none"> • Provides an interface for command and data from the GPM to the waveform application • Provides the platform specific pinout for the application developer. This may be a complete abstraction of the actual FPGA pinouts with only waveform application signal names provided. 	Platform	Inspect document & code	
STRS-15	<p>The STRS SPM developer shall provide documentation on the firmware interfaces of the platform specific wrapper for each user-programmable FPGA on the SPM, which describe, as a minimum, the following</p> <ul style="list-style-type: none"> • Signal names and descriptions • Signal polarity and format • Signal timing constraints of all signals • Clock generation and synchronization methods • Signal registering methods • Identification of development tool set used 	Platform	Inspect document	
STRS-16	<p>The STRS <i>Application-provided Application Control API</i> shall be implemented using C or C++.</p>	App	Inspect	
STRS-17	<p>The STRS infrastructure shall use the <i>STRS Application-provided Application Control API</i> to control STRS applications.</p>	OE	Script	

Space Telecommunications Radio System (STRS) Compliance Testing

Date: May 31, 2011

Document No.: STRS-ATP-00001

Page 38 of 46

Requirements	Description	OE/App	Tested	OK?
STRS-18	The STRS Operating Environment shall support C or C++ language interfaces for the <i>STRS Application-provided Application Control API</i> at compile-time.	OE	Inspect	
STRS-19	The STRS Operating Environment shall support C or C++ language interfaces for the <i>STRS Application-provided Application Control API</i> at run-time.	OE	WFCCN	
STRS-20	Each STRS application shall contain: <i>#include "STRS_ApplicationControl.h"</i>	App	Script	
STRS-21	The STRS platform provider shall provide an "STRS_ApplicationControl.h" that contains the method prototypes and, for C++, the class definition for the base class STRS_ApplicationControl.	OE	Inspect	
STRS-22	If the <i>STRS Application-provided Application Control API</i> is implemented in C++, the STRS application class shall be derived from the <i>STRS_ApplicationControl</i> base class.	App	Inspect	
STRS-23	If the STRS application provides the <i>APP_Write</i> method, the STRS application shall contain <i>#include "STRS_Sink.h"</i>	App	Script	
STRS-24	The STRS platform provider shall provide an "STRS_Sink.h" that contains the method prototypes and, for C++, the class definition for the base class STRS_Sink.	OE	Inspect	
STRS-25	If the <i>STRS Application-provided Application Control API</i> is implemented in C++ AND the STRS application provides the <i>APP_Write</i> method, the STRS application class shall be derived from the <i>STRS_Sink</i> base class.	App	Inspect	
STRS-26	If the STRS application provides the <i>APP_Read</i> method, the STRS application shall contain <i>#include "STRS_Source.h"</i>	App	Script	
STRS-27	The STRS platform provider shall provide an "STRS_Source.h" that contains the method prototypes and, for C++, the class definition for the base class STRS_Source.	OE	Inspect	
STRS-28	If the <i>STRS Application-provided Application Control API</i> is implemented in C++ AND the STRS application provides the <i>APP_Read</i> method, the STRS application class shall be derived from the <i>STRS_Source</i> base class.	App	Inspect	
STRS-29	Each STRS application shall contain a callable <i>APP_Configure</i> method as described in Table 8-3.	App	Script	
STRS-30	Each STRS application shall contain a callable <i>APP_GroundTest</i> method as described in Table 8-4.	App	Script	
STRS-31	Each STRS application shall contain a callable <i>APP_Initialize</i> method as described in Table 8-5.	App	Script	
STRS-32	Each STRS application shall contain a callable <i>APP_Instance</i> method as described in Table 8-6.	App	Script	

Space Telecommunications Radio System (STRS) Compliance Testing

Date: May 31, 2011

Document No.: STRS-ATP-00001

Page 39 of 46

Requirements	Description	OE/App	Tested	OK?
STRS-33	Each STRS application shall contain a callable <i>APP_Query</i> method as described in Table 8-7.	App	Script	
STRS-34	If the STRS application provides data to the infrastructure, then the STRS application shall contain a callable <i>APP_Read</i> method as described in Table 8-8.	App	Script	
STRS-35	Each STRS application shall contain a callable <i>APP_ReleaseObject</i> method as described in Table 8-9.	App	Script	
STRS-36	Each STRS application shall contain a callable <i>APP_RunTest</i> method as described in Table 8-10.	App	Script	
STRS-37	Each STRS application shall contain a callable <i>APP_Start</i> method as described in Table 8-11.	App	Script	
STRS-38	Each STRS application shall contain a callable <i>APP_Stop</i> method as described in Table 8-12.	App	Script	
STRS-39	If the STRS application receives data from the infrastructure, then the STRS application shall contain a callable <i>APP_Write</i> method as described in Table 8-13.	App	Script	
STRS-40	The STRS infrastructure shall contain a callable <i>STRS_Configure</i> method as described in Table 8-14.	OE	WFCCN	
STRS-41	The STRS infrastructure shall contain a callable <i>STRS_GroundTest</i> method as described in Table 8-15.	OE	WFCCN	
STRS-42	The STRS infrastructure shall contain a callable <i>STRS_Initialize</i> method as described in Table 8-16.	OE	WFCCN	
STRS-43	The STRS infrastructure shall contain a callable <i>STRS_Query</i> method as described in Table 8-17.	OE	WFCCN	
STRS-44	The STRS infrastructure shall contain a callable <i>STRS_ReleaseObject</i> method as described in Table 8-18.	OE	WFCCN	
STRS-45	The STRS infrastructure shall contain a callable <i>STRS_RunTest</i> method as described in Table 8-19.	OE	WFCCN	
STRS-46	The STRS infrastructure shall contain a callable <i>STRS_Start</i> method as described in Table 8-20.	OE	WFCCN	
STRS-47	The STRS infrastructure shall contain a callable <i>STRS_Stop</i> method as described in Table 8-21.	OE	WFCCN	
STRS-48	The STRS infrastructure shall contain a callable <i>STRS_AbortApp</i> method as described in Table 8-22.	OE	WFCCN	
STRS-49	The STRS infrastructure shall contain a callable <i>STRS_GetErrorQueue</i> method as described in Table 8-23.	OE	WFCCN	
STRS-50	The STRS infrastructure shall contain a callable <i>STRS_HandleRequest</i> method as described in Table 8-24.	OE	WFCCN	
STRS-51	The STRS infrastructure shall contain a callable <i>STRS_InstantiateApp</i> method as described in Table 8-25.	OE	WFCCN	
STRS-52	The STRS infrastructure shall contain a callable <i>STRS_IsOK</i> method as described in Table 8-26.	OE	WFCCN	
STRS-53	The STRS infrastructure shall contain a callable <i>STRS_Log</i> method as described in Table 8-27.	OE	WFCCN	

Space Telecommunications Radio System (STRS) Compliance Testing

Date: May 31, 2011

Document No.: STRS-ATP-00001

Page 40 of 46

Requirements	Description	OE/App	Tested	OK?
STRS-54	When an STRS application has a non-fatal error, the STRS application shall use the <i>STRS_Log method</i> (Table 8-27) with a target handle ID of constant <i>STRS_ERROR_QUEUE</i> .	App	WFCCN	
STRS-55	When an STRS application has a fatal error, the STRS application shall use the <i>STRS_Log method</i> (Table 8-27) with a target handle ID of constant <i>STRS_FATAL_QUEUE</i> .	App	WFCCN	
STRS-56	When an STRS application has a warning condition, the STRS application shall use the <i>STRS_Log method</i> (Table 8-27) with a target handle ID of constant <i>STRS_WARNING_QUEUE</i> .	App	WFCCN	
STRS-57	When an STRS application needs to send telemetry, the STRS application shall use the <i>STRS_Log method</i> (Table 8-27) with a target handle ID of constant <i>STRS_TELEMETRY_QUEUE</i> .	App	WFCCN	
STRS-58	The STRS infrastructure shall contain a callable <i>STRS_Write</i> method as described in Table 8-28.	OE	WFCCN	
STRS-59	The STRS infrastructure shall contain a callable <i>STRS_Read</i> method as described in Table 8-29.	OE	WFCCN	
STRS-60	The STRS applications shall use the STRS infrastructure <i>Device Control</i> methods to control the STRS Devices.	OE	Inspect	
STRS-61	The STRS infrastructure shall contain a callable <i>STRS_DeviceClose</i> method as described in Table 8-30.	OE	WFCCN	
STRS-62	The STRS infrastructure shall contain a callable <i>STRS_DeviceFlush</i> method as described in Table 8-31.	OE	WFCCN	
STRS-63	The STRS infrastructure shall contain a callable <i>STRS_DeviceLoad</i> method as described in Table 8-32.	OE	WFCCN	
STRS-64	The STRS infrastructure shall contain a callable <i>STRS_DeviceOpen</i> method as described in Table 8-33.	OE	WFCCN	
STRS-65	The STRS infrastructure shall contain a callable <i>STRS_DeviceReset</i> method as described in Table 8-34.	OE	WFCCN	
STRS-66	The STRS infrastructure shall contain a callable <i>STRS_DeviceStart</i> method as described in Table 8-35.	OE	WFCCN	
STRS-67	The STRS infrastructure shall contain a callable <i>STRS_DeviceStop</i> method as described in Table 8-36.	OE	WFCCN	
STRS-68	The STRS infrastructure shall contain a callable <i>STRS_DeviceUnload</i> method as described in Table 8-37.	OE	WFCCN	
STRS-69	The STRS infrastructure shall contain a callable <i>STRS_SetISR</i> method as described in Table 8-38.	OE	WFCCN	
STRS-70	The STRS infrastructure shall contain a callable <i>STRS_FileClose</i> method as described in Table 8-39.	OE	WFCCN	
STRS-71	The STRS infrastructure shall contain a callable <i>STRS_FileGetFreeSpace</i> method as described in Table 8-40.	OE	WFCCN	
STRS-72	The STRS infrastructure shall contain a callable <i>STRS_FileGetSize</i> method as described in Table 8-41.	OE	WFCCN	
STRS-73	The STRS infrastructure shall contain a callable <i>STRS_FileGetStreamPointer</i> method as described in Table 8-42.	OE	WFCCN	
STRS-74	The STRS infrastructure shall contain a callable <i>STRS_FileOpen</i> method as described in Table 8-43.	OE	WFCCN	
STRS-75	The STRS infrastructure shall contain a callable	OE	WFCCN	

Space Telecommunications Radio System (STRS) Compliance Testing

Date: May 31, 2011

Document No.: STRS-ATP-00001

Page 41 of 46

Requirements	Description	OE/App	Tested	OK?
	<i>STRS_FileRemove</i> method as described in Table 8-44.			
STRS-76	The STRS infrastructure shall contain a callable <i>STRS_FileRename</i> method as described in Table 8-45.	OE	WFCCN	
STRS-77	The STRS applications shall use the <i>STRS Infrastructure Messaging</i> methods to send messages between applications and/or the infrastructure with a single target handle ID.	App	Inspect	
STRS-78	The STRS infrastructure shall contain a callable <i>STRS_QueueCreate</i> method as described in Table 8-46.	OE	WFCCN	
STRS-79	The STRS infrastructure shall contain a callable <i>STRS_QueueDelete</i> method as described in Table 8-47.	OE	WFCCN	
STRS-80	The STRS infrastructure shall contain a callable <i>STRS_Register</i> method as described in Table 8-48.	OE	WFCCN	
STRS-81	The STRS infrastructure shall contain a callable <i>STRS_Unregister</i> method as described in Table 8-49.	OE	WFCCN	
STRS-82	Any portion of the STRS Applications on the GPP needing time control shall use the <i>STRS Infrastructure Time Control</i> methods to access the hardware and software timers.	App	Inspect	
STRS-83	The STRS infrastructure shall contain a callable <i>STRS_GetNanoseconds</i> method as described in Table 8-50.	OE	WFCCN	
STRS-84	The STRS infrastructure shall contain a callable <i>STRS_GetSeconds</i> method as described in Table 8-51.	OE	WFCCN	
STRS-85	The STRS infrastructure shall contain a callable <i>STRS_GetTime</i> method as described in Table 8-52.	OE	WFCCN	
STRS-86	The STRS infrastructure shall contain a callable <i>STRS_GetTimewarp</i> method as described in Table 8-53.	OE	WFCCN	
STRS-87	The STRS infrastructure shall contain a callable <i>STRS_SetTime</i> method as described in Table 8-54.	OE	WFCCN	
STRS-88	The STRS infrastructure shall contain a callable <i>STRS_Synch</i> method as described in Table 8-55.	OE	WFCCN	
STRS-89	The STRS platform provider shall provide an STRS.h file containing the STRS predefined data shown in Table 8-56.	OE	OE script & WFCCN	
STRS-90	The STRS Operating Environment shall provide the interfaces described in POSIX IEEE Standard 1003.13-2003 profile PSE51.	OE	Inspect	
STRS-91	STRS Applications shall use POSIX methods except for the unsafe functions listed in Table 8-57.	App	Script	
STRS-92	The STRS platform provider shall provide the STRS platform HAL documentation. The HAL documentation shall include, but not be limited to, the following <ul style="list-style-type: none"> • For each method/function, its calling sequence, return values, an explanation of its functionality, any preconditions for using the method/function, and the postconditions after using the method/function. • Information required to address the underlying hardware, including interrupt input and output, memory mapping, and other configuration data necessary to operate in the STRS platform environment. 	Platform	Inspect document	
STRS-93	The STRS infrastructure shall use the HAL APIs to communicate with the specialized hardware via the physical interface defined by the platform provider.	OE	No	
STRS-94	An STRS platform shall accept, validate, and respond to external commands.	OE	Observe	

Space Telecommunications Radio System (STRS) Compliance Testing

Date: May 31, 2011

Document No.: STRS-ATP-00001

Page 42 of 46

Requirements	Description	OE/App	Tested	OK?
STRS-95	An STRS platform shall execute external application control commands using the standardized STRS APIs.	OE	WFCCN	
STRS-96	The STRS infrastructure shall use the <i>STRS_Query</i> method to service external system requests for information from an STRS application.	OE	Inspect	
STRS-97	An STRS application shall use the <i>STRS_Log</i> and <i>STRS_Write</i> methods to send STRS telemetry set information to the external system.	App	Inspect	
STRS-98	The STRS platform provider shall document the necessary platform information (including a sample file) to develop a pre-deployed application configuration file in XML.	OE	Inspect document & sample file	
STRS-99	The STRS application developer shall document the necessary application information to develop a pre-deployed application configuration file in XML.	App	Inspect document	
STRS-100	The STRS platform integrator shall provide a pre-deployed application configuration file in XML.	OE	Inspect & check using XMLSpy	
STRS-101	The pre-deployed STRS application configuration file shall identify, as a minimum, the following application attributes and default values <ul style="list-style-type: none"> • Identification • Unique STRS handle name for the application • Class name (if applicable) • State after processing the configuration file • Required resources • Memory in bytes • Number of gates or logic elements • Configuration parameters containing the STRS handle, names of files, devices, queues, waveforms and services needed by the STRS application • Values and constraints for all operationally configurable parameters • Filename(s) of loadable images for resources 	App	Inspect	
STRS-102	The STRS platform provider shall provide an XML schema to validate the format and data for pre-deployed STRS application configuration files, including the order of the tags, the number of occurrences of each tag, and the values or attributes.	OE	Inspect & check using XMLSpy	
STRS-103	The STRS platform provider shall provide the tools and documentation to transform pre-deployed application configuration file in XML into a deployed application configuration file.	OE	Inspect document & tools	
STRS-104	The STRS platform integrator shall provide deployed STRS application configuration file for the STRS infrastructure to place the STRS application in the specified state.	OE	Inspect	

APPENDIX G—Document Compliance Testing Guidelines

The document review process is initiated by lead(s) being assigned. The lead(s) will find the deliverable documents to be reviewed for STRS compliance and assign the documents to reviewers. If there is submission by more than one company, it is recommended that reviewers look at similar documents for two companies to be able to compare and contrast. If there are more documents than reviewers, the lead(s) may assign multiple short documents to a reviewer or request additional reviewers. The lead(s) will create a spreadsheet or database in eRoom or something equivalent for reviewers to enter their comments. The lead(s) will coordinate the review process specifying deadlines, sending out reminders and answering questions.

Then the reviewers will review the documents and enter their comments into the spreadsheet or database. Some comments may be STRS changes only and nothing fed back to company, or company comments only and no STRS change. The reviewers should keep track of both compliances and non-compliances to be sure that all requirements are addressed.

Once the reviewers have finished, the lead(s) will review the comments for clarity and completeness. Comments pertaining to STRS Architecture only are reviewed by the STRS team for inclusion into the STRS Architecture Standard with the resolution passed back to the lead(s) for inclusion into the spreadsheet or data base. The lead(s) will then prepare company feedback.

Here is specific guidance to document reviewers for STRS compliance:

- a. Look for meeting the requirements in the STRS Architecture Standard. Reviewers need to record when a document satisfies the requirements for that document and not just the variances. The purpose is to be able to see if they missed anything.
- b. Look for common practices that might be standardized.
- c. Look for misunderstandings.
- d. For the FPGA wrappers, see what commonalities help future waveform developers and what is platform specific.
- e. Determine if we should have a common document format.
- f. Look for items that contribute to waveform (firmware) portability that may become part of the standard.
- g. For STRS, look at their description of their implementation. Did they interpret the APIs as intended? Do we need more in our API descriptions, etc? Are there other aspects that should become part of the STRS standard?
- h. For HIDs, see what types of resources are made available to waveform developers. Look for common formatting, etc. Determine if we should have a common document format.

From STRS Architecture Standard version 1.02:

1. (STRS-4) The STRS platform developer shall describe in the HID document, the behavior and capability of each major functional device or resource available for use by waveforms, services, or other applications (e.g. FPGA, GPP, DSP, memory), noting any operational limitations.

Although not in the requirements, some things to look for are:

- a. Identification
 - i. Manufacturer,
 - ii. Model number,
 - iii. Part number and any revision levels (if applicable).
 - iv. Device type
- b. Performance capabilities:
 - i. Microprocessor clock speed(s) or MIPS
 - ii. Data I/O rate maximum in bits per second
 - iii. Memory size(s), type(s), and speed(s)
 - iv. Reconfigurable capacity
2. (STRS-5) The STRS platform developer shall describe in the HID document, the reconfigurability behavior and capability of each reconfigurable component.

3. (STRS-6) The STRS platform developer shall describe in the HID document, the behavior and performance of the RF modular component(s).
Although not in the requirements, some things to look for are:
- a. Receiver information:
 - i. Input impedance
 - ii. Center frequency
 - iii. Bandwidth(s)
 - iv. IF frequency(s)
 - v. IF input/output level(s)
 - vi. Signal to Noise Ratio (SNR), in dB
 - vii. Dynamic Range
 - viii. Receiver sensitivity
 - ix. Third Order Intermodulation Intercept Point (IP3)
 - x. Overall receiver Noise Figure, in dB
 - xi. AGC operational parameters
 - xii. Carrier frequency accuracy
 - xiii. Tuning frequency resolution
 - xiv. Selectivity in dBc
 - b. Transmitter information:
 - i. Output impedance
 - ii. Carrier center frequency
 - iii. Bandwidth(s)
 - iv. Operational frequency bandwidth
 - v. Intermediate frequencies
 - vi. IF input/output levels
 - vii. Local oscillator phase noise
 - viii. Signal to Noise Ratio in dB
 - ix. Output signal flatness over operating frequencies
 - x. Temperature vs. Power output
 - xi. 1dB compression point
 - xii. Tuning frequency resolution
 - xiii. Carrier frequency accuracy
 - xiv. Maximum reverse power at output connector
 - xv. Voltage standing wave ratio (VSWR) measured across operating frequency
4. (STRS-7) The STRS platform provider shall describe in the HID document, the interfaces that are provided to and from each modular component of the radio platform.
Although not in the requirements, some things to look for are:
- a. Electrical connection
 - i. Name
 - ii. Data type (serial/parallel, digital/analog)
 - iii. Bus width
 - iv. Timing diagram
 - b. Hardware
 - i. Describe conduction cooling paths, or specify air-flow capabilities, depending on the intended operational environment
 - ii. Total heat dissipation limits, and dissipation limits for individual module slots, as appropriate
 - iii. Thermal cooling requirements
 - iv. Operational and storage environmental constraints (temperature, humidity, etc.)

- v. Mechanical information required to build a module for the platform. This includes all dimensions, mass, clearances, mounting method, and connector locations.
- vi. Vibration loading limitations
- c. Table 20 provides typical interface characteristics:

Table 20 – STRS Module Interface Characterization

STRS Module Interface Characterization Table	
Parameter	Description / Comments
Name	Interface Name (data, control, DC power, RF, security, etc)
Interface type	Point to point, point-multipoint, multipoint, serial, bus, other
Implementation level	Component, module, board, chassis, remote node
Reference documents / Standards	Applicable documents for interface standards or description of custom interfaces
Note / Constraints	Variances from standards, physical and logical functional limitations
Transfer speed	Clock speed, throughput speed
Signal definition	Description of functionality and intended use
Physical Implementation	
Technology	e.g. GPP, DSP, FPGA, ASIC and description
Connectors	Model number, pin out (incl. unused pins)
Data plane	Width, speed, timing, data encoding, protocols
Control plane	Control signals, control messages or commanding, interrupts, message protocol
Functional Implementation	
Models	Data plane model, control plane model, test bench model
Power	Voltages, currents, noise, conducted immunity, susceptibility
APIs	Custom or standard, particular to OS environment
Software	Device drivers, development environment & tool chain
Logical Implementation	
Addressing	Method, schemes
Channels	Open, close
Connection type	Forward, terminate, test

- 5. (STRS-8) The STRS platform provider shall describe in the HID document, the control, telemetry, and data mechanisms of each modular component (i.e. how to program or control each modular component of the platform, and how to use or access each device or software component, noting any proprietary aspects).

Although not in the requirements, some things to look for are:

- i. Connector type
- ii. Connector pinout (including unused pins)
- iii. Electrical signaling specifications (logic standard, terminations, etc.)
- iv. Signal timing (setup and hold times, clock rates, clock accuracy, etc.)
- v. Electrical isolation
- vi. Data encoding (Non-return to zero (NRZ), Manchester, etc.)
- vii. Data transfer protocol

- 6. (STRS-9) The STRS platform developer shall describe in the HID document, the behavior and performance of any power supply or power converter modular component(s).

Although not in the requirements, some things to look for are:

- i. Minimum, maximum, and nominal voltages required
- ii. Standby and maximum current availability and consumptions

- iii. Connector type and pinout
 - iv. Voltage ripple tolerance
7. (STRS-92) The STRS platform provider shall provide the STRS platform HAL documentation that includes the following:
- For each method/function, its calling sequence, return values, an explanation of its functionality, any preconditions for using the method/function, and the postconditions after using the method/function.
 - Information required to address the underlying hardware, including interrupt input and output, memory mapping, and the configuration data necessary to operate in the STRS platform environment.
8. (STRS-12) Application development artifacts shall be submitted to the NASA STRS Repository. Although these aren't required until the final submittal, some things to look for are:
- i. License agreements for software use and reuse.
 - ii. High level system or component software model
 - iii. Documentation of application firmware external interfaces (e.g. signal names and descriptions, signal polarity and format, timing constraints of signals)
 - iv. Documentation of STRS application behavior
 - v. Description of application function sources
 - vi. Description of application libraries, if applicable
 - vii. Documentation of application development environment and tool suite
 - viii. Include application name, purpose, developer, version, and configuration specifics
 - ix. Include the hardware on which the application is executed, its OS, OS developer, OS version, and OS configuration specifics
 - x. Test plan and results documentation
 - xi. Identification of Flight Software Development Standards used

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188		
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY) 01-12-2011		2. REPORT TYPE Technical Memorandum		3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE Space Telecommunications Radio System (STRS) Compliance Testing			5a. CONTRACT NUMBER		
			5b. GRANT NUMBER		
			5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Handler, Louis, M.			5d. PROJECT NUMBER		
			5e. TASK NUMBER		
			5f. WORK UNIT NUMBER WBS 439432.04.07.01		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration John H. Glenn Research Center at Lewis Field Cleveland, Ohio 44135-3191			8. PERFORMING ORGANIZATION REPORT NUMBER E-18021		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001			10. SPONSORING/MONITOR'S ACRONYM(S) NASA		
			11. SPONSORING/MONITORING REPORT NUMBER NASA/TM-2011-217266		
12. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified-Unlimited Subject Categories: 17, 38, and 61 Available electronically at http://www.sti.nasa.gov This publication is available from the NASA Center for AeroSpace Information, 443-757-5802					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The Space Telecommunications Radio System (STRS) defines an open architecture for software defined radios. This document describes the testing methodology to aid in determining the degree of compliance to the STRS architecture. Non-compliances are reported to the software and hardware developers as well as the NASA project manager so that any non-compliances may be fixed or waivers issued. Since the software developers may be divided into those that provide the operating environment including the operating system and STRS infrastructure (OE) and those that supply the waveform applications, the tests are divided accordingly. The static tests are also divided by the availability of an automated tool that determines whether the source code and configuration files contain the appropriate items. Thus, there are six separate step-by-step test procedures described as well as the corresponding requirements that they test. The six types of STRS compliance tests are: STRS application automated testing, STRS infrastructure automated testing, STRS infrastructure testing by compiling WFCCN with the infrastructure, STRS configuration file testing, STRS application manual code testing, and STRS infrastructure manual code testing. Examples of the input and output of the scripts are shown in the appendices as well as more specific information about what to configure and test in WFCCN for non-compliance. In addition, each STRS requirement is listed and the type of testing briefly described. Attached is also a set of guidelines on what to look for in addition to the requirements to aid in the document review process.					
15. SUBJECT TERMS Telecommunications; Computer programming; Software engineering; Software development tools; Software reliability					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email:help@sti.nasa.gov)
U	U	U	UU	51	19b. TELEPHONE NUMBER (include area code) 443-757-5802

