Telemetry Group

DOCUMENT 118-17 Vol. 6

TEST METHODS FOR TELEMETRY SYSTEMS AND SUBSYSTEMS
VOLUME 6: AUTOMATED TEST METHODS FOR XML METADATA

ABERDEEN TEST CENTER
DUGWAY PROVING GROUND
REAGAN TEST SITE
REDSTONE TEST CENTER
WHITE SANDS MISSILE RANGE
YUMA PROVING GROUND

NAVAL AIR WARFARE CENTER AIRCRAFT DIVISION
NAVAL AIR WARFARE CENTER WEAPONS DIVISION
NAVAL UNDERSEA WARFARE CENTER DIVISION, KEYPORT
NAVAL UNDERSEA WARFARE CENTER DIVISION, NEWPORT
PACIFIC MISSILE RANGE FACILITY

30TH SPACE WING
45TH SPACE WING
96TH TEST WING
412TH TEST WING
ARNOLD ENGINEERING DEVELOPMENT COMPLEX

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

DISTRIBUTION A: APPROVED FOR PUBLIC RELEASE
DISTRIBUTION IS UNLIMITED

This page intentionally left blank.

DOCUMENT 118-17 Volume 6

TEST METHODS FOR TELEMETRY SYSTEMS AND SUBSYSTEMS VOLUME 6: AUTOMATED TEST METHODS FOR XML METADATA

December 2017

Prepared by

Telemetry Group

Published by

Secretariat
Range Commanders Council
White Sands Missile Range
New Mexico 88002-5100

This page intentionally left blank.

Table of Contents

Table of Figures

This page intentionally left blank.

Preface

This document presents the results of efforts by the Range Commanders Council Telemetry Group under RCC Task TG-147. This document (Volume VI of the RCC Document 118 series) describes procedures used for evaluating XML metadata documents, including TMATS, MDL, IHAL, and DDML documents. These documents contain specifications or descriptions of artifacts and systems of importance to the collection and management of telemetry data. The methods defined in this report provide a means of evaluating the suitability of such a metadata document under the TMATS, MDL, IHAL, or DDML definitions.

The reader of this report should be familiar with the extensible markup language (XML) and its supporting form for describing XML file schema. For a quick overview, refer to the RCC Document 125-15 XML Style Guide.

The RCC gives special acknowledgement for production of this document to the TG Telemetry Data Processing Committee. Please direct any questions to the committee point of contact or to the RCC Secretariat as shown below.

This report should be of interest primarily to users of XML metadata documents, and may be of interest to developers of those documents. The methods described here may be used to evaluate XML metadata documents for use in describing telemetry artifacts and systems. Alternatively, through the use of gold-standard metadata documents, these methods may be used to evaluate evolving conventions, standards, guidelines, and application rules for compatibility with prior capabilities.

Telemetry Group Telemetry Data Processing Committee Chairman: Mr. Jon Morgan
412 TW, Edwards AFB
Bldg 1408 Room 5
301 East Yeager
Edwards AFB, CA 93524
Phone: DSN 527-8942 Com (661) 277-8942
Fax: DSN 527-8933 Com (661) 277 8933
email jon.morgan.2.ctr@us.af.mil

Secretariat, Range Commanders Council
ATTN: TEDT-WS-RCC
1510 Headquarters Avenue
White Sands Missile Range, New Mexico 88002-5110
Phone: DSN 258-1107 Com (575) 678-1107
Fax: DSN 258-7519 Com (575) 678-7519
email usarmy.wsmr.atec.list.rcc@mail.mil

This page intentionally left blank.

Acronyms

API	application programming interface
DDML	Data Display Markup Language
IHAL	Instrumentation Hardware Abstraction Language
MDL	Metadata Description Language
RCC	Range Commanders Council
TMATS	Telemetry Attributes Transfer Standard
XML	eXtensible Markup Language
XSD	XML schema definition

This page intentionally left blank.

1. Introduction

This document defines methods for evaluating Range Commanders Council (RCC) eXtensible Markup Language (XML) files with regard to XML-based metadata standards encoded in the XML schema definition (XSD) format and other standards and conventions. This method should be of interest primarily to parties having tools or applications that consume RCC metadata standard documents, and may be of interest to developers of tools or applications that produce RCC metadata standard documents. Evaluations defined in this method include formal schema definitions, conventional forms of modeling, established style guides, and ancillary application-specific usage rules.

The Telemetry Attributes Transfer Standard (TMATS) provides the common definition of information needed to fully describe data being transmitted from, or recorded on an item under test. A TMATS file serves as the medium of exchange between an information source (usually an instrumentation organization) and a user (usually a test range). For more details on TMATS/XML, see the TMATS Handbook.[1]

The Instrumentation Hardware Abstraction Language (IHAL) standard specifies an XML-based format for describing instrumentation hardware. The IHAL language is focused on the settings available on devices, referred to as "configurable attributes." The primary purpose of IHAL is to describe these attributes both in terms their currently configuration and their potential configuration. For more details on IHAL, see the IHAL Handbook.[2]

The Data Display Markup Language (DDML) standard specifies an XML-based format intended to serve as the "inter-lingua" - the common form of communication - among data displays. The DDML standard encompasses a variety of vendor-specific data display formats, providing a unifying scheme of expression. The DDML standard supports reusable concepts and is extensible to accommodate future objects. For more details on DDML, see the DDML Handbook.[3]

The Metadata Description Language (MDL) standard provides a common exchange language that facilitates the interchange of configuration information between telemetry system components. The MDL standard provides the means for describing the configuration of the components in a telemetry system, as well as their logical and physical interrelationships. The MDL syntax defines vocabulary and sentence structure, while the MDL semantics provide meaning. For details on MDL, see the RCC Telemetry Network Standard.[4]

---

[1] Range Commanders Council. *Telemetry Attributes Transfer Standard (TMATS) Handbook*. RCC 124-17. January 2017. May be superseded by update. Retrieved 1 December 2017. Available at http://www.wsmr.army.mil/RCCsite/Documents/124-17_TMATS_Handbook/124-17_TMATS_Handbook.pdf.

[2] Range Commanders Council. *Instrumentation Hardware Abstraction Language (IHAL) Handbook*. RCC 128-17. January 2017. May be superseded by update. Retrieved 1 December 2017. Available at http://www.wsmr.army.mil/RCCsite/Documents/128-17_IHAL_Handbook/128-17_IHAL_Handbook.pdf.

[3] Range Commanders Council. *Data Display Markup Language (DDML) Handbook*. RCC 127-17. January 2017. May be superseded by update. Retrieved 1 December 2017. Available at http://www.wsmr.army.mil/RCCsite/Documents/127-17_DDML_Handbook/127-17-DDML_Handbook.pdf.

[4] Range Commanders Council. "Telemetry Network Standard" in *Telemetry Standards*. RCC 106-17. Chapters 21-28. July 2017. May be superseded by update. Retrieved 28 December 2017. Available at http://www.wsmr.army.mil/RCCsite/Pages/Publications.aspx.

The XML standard is widely used for expressing complex data structures in a highly transportable manner. For information about XML, refer to RCC 125-15.[5]

The remainder of this document defines methods for evaluating RCC documents purporting to conform to an RCC standard (TMATS, MDL, IHAL, or DDML). The intent is that computer programs will be developed to automatically apply these methods. These methods go beyond the formal syntactic definition of those formats to allow evaluation against community conventions, style guides, and application-specific usage rules. The methods have been developed to conform to the RCC-118 family of documents (*Test Methods*).

2. Context

An RCC metadata document describes and defines artifacts and system configurations essential to the collection and management of telemetry data. The RCC has established standards for metadata documents for describing data being transmitted from or recorded on an item under test (TMATS); telemetry network configuration (MDL); instrumentation hardware (IHAL); and data display (DDML). This report describes test methods for the evaluation of RCC XML metadata documents. These documents are dynamic, adaptable, and often application-oriented. Accordingly, the described methods support formal schema definitions, community conventions, style guides, and application rules as sources of authority in evaluating metadata documents.

These methods are intended to be consistent with the needs of three distinct communities of users.

- Telemetry system developers and operators need to know that metadata documents describing the artifacts and systems with which they are working are complete, consistent, and appropriate for use.
- Telemetry artifact and system developers (for example, developers of data acquisition units) need to be able to assure their customers and users that the metadata descriptions accompanying their products are complete, consistent, and appropriate for use.
- Developers of formal schemata, community conventions, style guides, and application rules (for example, the RCC-TG data multiplex committee, formal and informal user groups, etc.) need to assure (or at the very least to understand) the compatibility of emerging and evolving standards with prior standards and metadata documents.

3. XML Instance Document Syntax Testing

3.1 Purpose of Testing Syntax

This test determines that an XML instance document is syntactically correct; that is, that the document conforms to an XSD that represents the standard. Failure to pass this test will mean that consumers of the XML document will likely not be able to process it.

[5] Range Commanders Council. *XML Style Guide*. 125-15. July 2015. May be superseded by update. Retrieved 28 January 2016. Available at http://www.wsmr.army.mil/RCCsite/Documents/125-15_XML_Style_Guide/125-15_XML_Style_Guide.pdf.

3.2 Test Environment for Syntax

Figure 1 illustrates the environment for testing syntax.

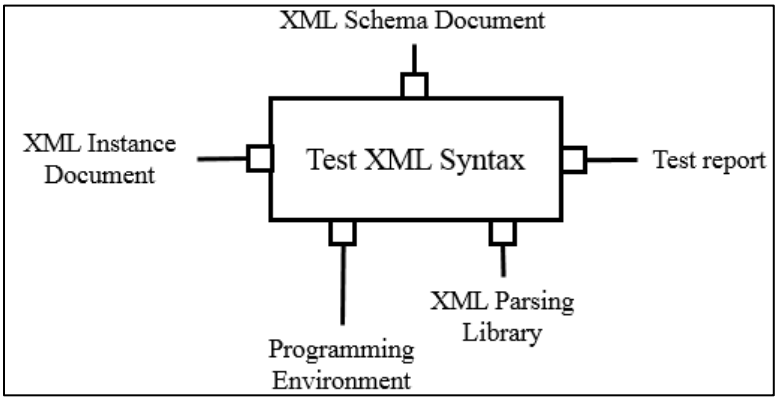

Figure 1. XML Instance Document Syntax Test Environment

The environment includes the following.

- The XSD for the standard;[6]
- The XML instance document;
- A suitable programming environment (C, C++, Java, etc.);
- An XML parsing library that conforms to the Simple API for XML[7] or the Document Object Model[8] for the selected programming environment. Some popular implementations are the Apache Xerces™ project[9], the Java Architecture for XML Binding[10], and xmllint[11].

3.3 Method for Testing Syntax

The test method is as follows.

1. Initialize the programming environment.
2. Write test application code to use the XML parsing library according to the instructions or the application programming interface (API) of the library. This application uses the XML parsing library to validate that an XML instance document conforms to the XSD. The output is a test report that describes the errors or warnings returned by the XML parsing library validation.
3. Compile and build the test application.

[6] In this test and the tests that follow, it is assumed that the XSD is an accurate representation of the standard and is syntactically valid. Testing whether or not the XSD accurately represents the standard is outside the scope of this document.
[7] http://www.saxproject.org/, last accessed August 3, 2017.
[8] https://www.w3.org/DOM/, last accessed August 3, 2017.
[9] http://xerces.apache.org/, last accessed August 3, 2017.
[10] http://www.oracle.com/technetwork/articles/javase/index-140168.html, last accessed August 3, 2017.
[11] http://xmlsoft.org/xmllint.html, last accessed August 25, 2017.

4. Run the test application to load the XML instance document and the XSD encoding of the standard into the XML parsing library per the instructions of the library (including relevant configuration parameters).
5. Observe the output of the test application to identify errors and warnings.

4. Style Guide Testing

4.1 Purpose of Testing Compliance to Style Guide

This test determines that an XSD or XML instance document complies with the RCC XML Style Guide. Failure to pass this test will mean that the documents being tested may be difficult for designers and developers to read and interpret; such documents will be more difficult and more expensive to maintain. This test applies to XML instance documents or XSDs.

4.2 Environment for Testing Compliance

Figure 2 illustrates the environment for testing an XML instance document's compliance with the style guide.

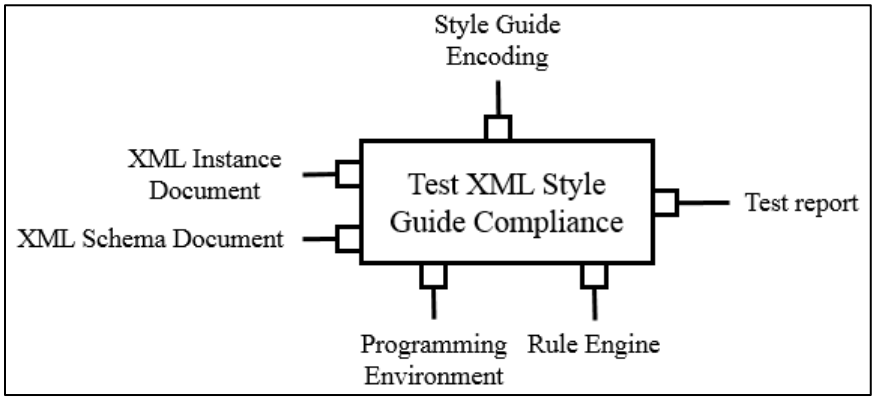

Figure 2. Style Guide Test Environment

The environment for this test includes the following.

- The XSD for the standard or the XML instance document to test.
- A suitable programming environment (C, C++, Java, etc.).
- A rule engine for executing the style guide rules that is compatible with the programming environment. Preferably, this will be an open-source rule engine or a rule engine that can be obtained by the government and other stakeholders at no cost.
- An encoding of the style guide rules that can be read and executed by the selected rule engine.
  - At the time of this writing, a recommended encoding of these rules does not exist.
  - Encodings shall be selected that can be interpreted by the rule engine – custom or proprietary encodings are strongly discouraged.
  - In the future, the relevant communities will recommend a rule encoding and rules engine for this test.

4.3 Method for Testing Compliance

The test method is as follows.

1. Initialize the programming environment.
2. Write test application code to use the rule engine according to the instructions or API of the engine. This application loads the XML instance document or XSD and the style guide rule encodings into the rule engine's internal objects and then invokes the appropriate rule engine API to validate the XML instance document or XSD against the encoded style guide rules. The output is a test report that describes the errors or warnings returned by the rule engine validation.
3. Compile and build the test application.
4. Run the test application to load the XML instance document or XSD and the encodings of the style guide rules into the rule engine per the instructions of the library (including relevant configuration parameters).
5. Run the test application to evaluate the XML instance document or XSD against the encodings of the style guide rules.
6. Observe the output of the test application to identify errors and warnings.

5. Community Conventions Testing

5.1 Purpose of Testing against Community Conventions

This test determines that an XML instance document complies with conventions established by a community of interest. A community can be a standards committee, range, test program, etc. Community conventions can be naming schemes for measurements, property lengths, non-syntactic rules documented in the standard, etc. Failure to pass this test will mean that the documents being tested may be difficult for users to understand, may consume excessive memory, may violate rules and constraints established by the community, and may not be understood by software or in applications sponsored by that community. Such documents will fall short of achieving the intended use. This test applies to XML instance documents.

An example of community conventions is contained in the TMATS document.[12] Property lengths - for example, the data source ID parameter - are limited to no more than 32 characters. There are also non-syntactic rules, for example, that the property *tape width* is allowed when the *media type* is *analog* or *cassette*.

5.2 Environment for Testing against Conventions

Figure 3 illustrates the environment for testing an instance document against community conventions.

[12] Range Commanders Council. "Telemetry Attributes Transfer Standard" in *Telemetry Standards*. RCC 106-17. July 2017. May be superseded by update. Retrieved 6 December 2017. Available at http://www.wsmr.army.mil/RCCsite/Documents/106-17_Telemetry_Standards/Chapter9.pdf.

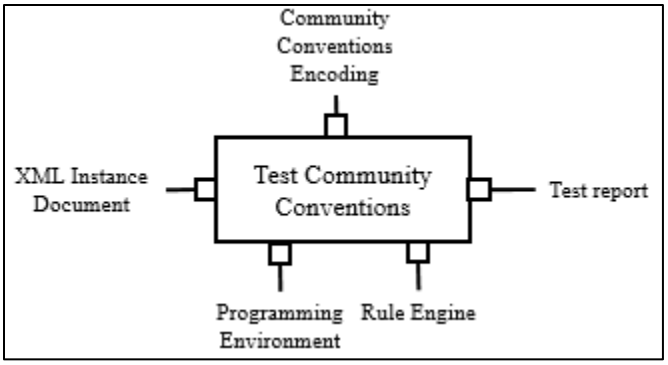

Figure 3. Community Conventions Test Environment

The environment for this test includes the following.

- The XML instance document to test.
- A suitable programming environment (C, C++, Java, etc.).
- A rule engine for executing the community conventions that is compatible with the programming environment. Preferably, this will be an open-source rule engine or a rule engine that can be obtained by the government and other stakeholders at no cost.
- An encoding of the community conventions that can be read and executed by the selected rule engine.
  - At the time of this writing, a recommended encoding of these rules does not exist.
  - Encodings shall be selected that can be interpreted by the rule engine – custom or proprietary encodings are strongly discouraged.
  - In the future, the relevant communities will recommend a rule encoding and rules engine for this test.

5.3 Method for Testing against Conventions

The test method is as follows.

1. Initialize the programming environment.
2. Write test application code to use the rule engine according to the instructions or API of the engine. This application loads the XML instance document and the community convention rule encodings into the rule engine's internal objects and then invokes the appropriate rule engine API to validate the XML instance document against the encoded community convention rules. The output is a test report that describes the errors or warnings returned by the rule engine validation.
3. Compile and build the test application.
4. Run the test application to load the XML instance document and the encodings of the community conventions into the rule engine per the instructions of the library (including relevant configuration parameters).
5. Run the test application to evaluate the XML instance document against the encodings of the community conventions.
6. Observe the output of the test application to identify errors and warnings.

6. Local Application Testing

6.1 Purpose of Testing against Local Requirements

This test determines that an XML instance document satisfies requirements for a local application. A local application is, for example, a specific range event or series of range events. Requirements within the context of a specific event may include hardware compatibility constraints, physical constraints, business rules, configuration rules, etc. Failure to pass this test will mean that the document is not compatible with the expectations of the event and should not be used to support the event. This test applies to XML instance documents.

6.2 Environment for Testing against Local Requirements

Figure 4 illustrates the environment for testing an instance document against requirements specific to local applications.

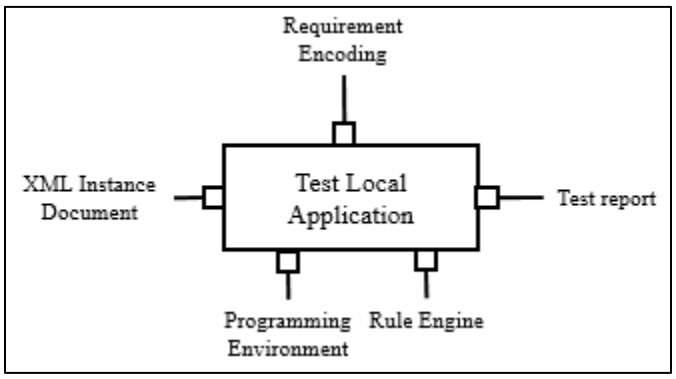

Figure 4. Local Application Test Environment

The environment for this test includes the following.

- The XML instance document to test.
- A suitable programming environment (C, C++, Java, etc.).
- A rule engine for executing the requirements that is compatible with the programming environment. Preferably, this will be an open-source rule engine or a rule engine that can be obtained by the government and other stakeholders at no cost.
- An encoding of the requirements that can be read and executed by the selected rule engine.
  - At the time of this writing, a recommended encoding of these rules does not exist.
  - Encodings shall be selected that can be interpreted by the rule engine – custom or proprietary encodings are strongly discouraged.
  - In the future, the relevant communities will recommend a rule encoding and rules engine for this test.

6.3 Method for Testing against Local Requirements

The test method is as follows.

1. Initialize the programming environment.

2. Write test application code to use the rule engine according to the instructions or API of the engine. This application loads the XML instance document and the requirements rule encodings into the rule engine's internal objects and then invokes the appropriate rule engine API to validate the XML instance document against the encoded requirements rules. The output is a test report that describes the errors or warnings returned by the rule engine validation.
3. Compile and build the test application.
4. Run the test application to load the XML instance document and the encodings of the requirements into the rule engine per the instructions of the library (including relevant configuration parameters).
5. Run the test application to evaluate the XML instance document against the encodings of the requirements.
6. Observe the output of the test application to identify errors and warnings.

7. Regression Testing

7.1 Purpose of Regression Testing

This test determines that an XSD, style guide encoding, community conventions encoding, or requirements encoding does not introduce regression errors into the application of a standard to a specific test or suite of tests. Regression in this context means the corpus of pre-existing rules or expectations; regression testing evaluates new rules or definitions to see if they preserve prior expectations. This test also determines that an XSD, style guide encoding, community conventions encoding, or requirements encoding is compatible with all the programming environments, XML parsing libraries, and rule engines that the community has determined should be supported. Failure to pass this test will mean that changes to the environment have introduced errors that may be backward-incompatible with prior instances of the XSD, community conventions encoding, or requirements encoding. This test applies to XSDs or XML instance documents.

7.2 Environment for Performing Regression Tests

Figure 5 illustrates the environment for performing regression tests.

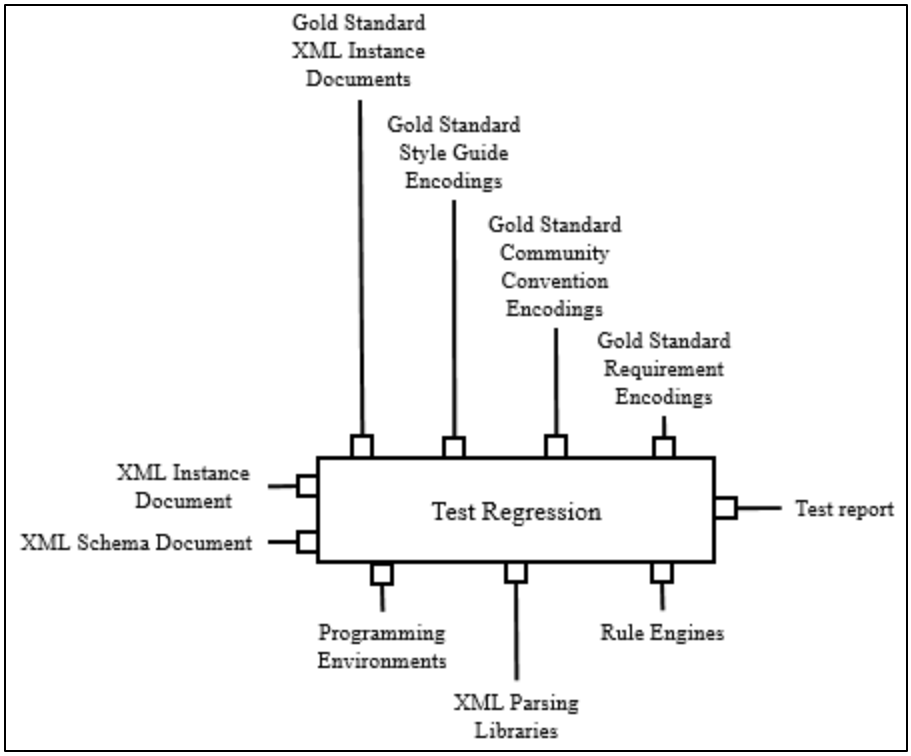

Figure 5. Regression Test Environment

The environment for this test includes the following.

- The XSD or XML instance document to test.
- All programming environments (C, C++, Java, etc.) that the community has determined need to be supported.
- All XML parsing libraries supported by the community that conform to the Simple API for XML or the Document Object Model that is compatible with the programming environment.
- A set of gold-standard encodings:[13]
  - style guide rules that can be read and executed by the selected rule engine;
  - community conventions that can be read and executed by the selected rule engine;
  - requirements that can be read and executed by the selected rule engine in the previous bullet;
  - XML instance documents.
- Rule engine(s)[14] for:
  - executing the style guide rules that are compatible with the programming environment;
  - executing the community conventions that are compatible with the programming environment;

[13] By gold standard we mean rules that exercise functionality of the community; ideally with full coverage.
[14] The rule engine may or may not be the same for all types of validation.

- executing the requirements that are compatible with the programming environment.

7.3 Method for Performing Regression Tests

The test method is as follows for each compatible combination of programming environment, XML parsing library, style guide rule engine, community conventions rule engine, and requirements rule engine. This test method uses the gold-standard XML instance document and encodings to verify that the rules engines and other tools work together.

1. Initialize the programming environment.
2. Write test application code to use the rule engine(s) and XML parsing library according to the instructions or API of the engine. This test application includes the functionality of the test applications described in sections 3.3, 4.3, 5.3, and 6.3.
3. Compile and build the test application.
4. Run the test application to load each XML instance document and the XSD encoding of the standard into the XML parsing library per the instructions of the library (including relevant configuration parameters).
5. Observe the output of the test application to identify errors and warnings.
6. Run the test application to load each XML instance document and XSD and the encodings of the style guide rules into the rule engine per the instructions of the library (including relevant configuration parameters).
7. Run the test application to evaluate the XML instance document and XSD against the encodings of the style guide rules.
8. Observe the output of the test application to identify errors and warnings.
9. Run the test application to load the XML instance documents and the encodings of the community conventions into the rule engine per the instructions of the library (including relevant configuration parameters).
10. Run the test application to evaluate the XML instance document against the encodings of the community conventions.
11. Observe the output of the test application to identify errors and warnings.
12. Run the test application to load the XML instance documents and the encodings of the requirements into the rule engine per the instructions of the library (including relevant configuration parameters).
13. Run the test application to evaluate the XML instance document against the encodings of the requirements.
14. Observe the output of the test application to identify errors and warnings.

This page intentionally left blank.

APPENDIX A

Citations

Range Commanders Council. *Data Display Markup Language (DDML) Handbook*. RCC 127-17. January 2017. May be superseded by update. Retrieved 1 December 2017. Available at http://www.wsmr.army.mil/RCCsite/Documents/127-17_DDML_Handbook/127-17-DDML_Handbook.pdf.

———. *Instrumentation Hardware Abstraction Language (IHAL) Handbook*. RCC 128-17. January 2017. May be superseded by update. Retrieved 1 December 2017. Available at http://www.wsmr.army.mil/RCCsite/Documents/128-17_IHAL_Handbook/128-17_IHAL_Handbook.pdf.

———. "Telemetry Attributes Transfer Standard" in *Telemetry Standards*. RCC 106-17. July 2017. May be superseded by update. Retrieved 6 December 2017. Available at http://www.wsmr.army.mil/RCCsite/Documents/106-17_Telemetry_Standards/Chapter9.pdf.

———. "Telemetry Network Standard" in *Telemetry Standards*. RCC 106-17. Chapters 21-28. July 2017. May be superseded by update. Retrieved 28 December 2017. Available at http://www.wsmr.army.mil/RCCsite/Pages/Publications.aspx.

———.*Telemetry Attributes Transfer Standard (TMATS) Handbook*. RCC 124-17. January 2017. May be superseded by update. Retrieved 1 December 2017. Available at http://www.wsmr.army.mil/RCCsite/Documents/124-17_TMATS_Handbook/124-17_TMATS_Handbook.pdf.

———. *XML Style Guide*. 125-15. July 2015. May be superseded by update. Retrieved 28 January 2016. Available at http://www.wsmr.army.mil/RCCsite/Documents/125-15_XML_Style_Guide/125-15_XML_Style_Guide.pdf.

*** * * END OF DOCUMENT * * ***