TELEMETRY GROUP

DOCUMENT 125-15

XML STYLE GUIDE

ABERDEEN TEST CENTER
DUGWAY PROVING GROUND
REAGAN TEST SITE
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

This page intentionally left blank.

DOCUMENT 125-15

XML STYLE GUIDE

July 2015

Prepared by

Telemetry Group

Published by

Secretariat
Range Commanders Council
White Sands Missile Range
New Mexico 88002-5110

This page intentionally left blank.

Table of Contents

Table of Figures

Acronyms

ASCII	American Standard Code for Information Interchange
DAU	data acquisition unit
DDML	data display markup language
IHAL	instrumentation hardware abstraction language
MDL	metadata description language
PCM	pulse code modulation
T&E	Test and Evaluation
TMATS	Telemetry Attributes Transfer Standard
URI	uniform resource identifier
W3C	World Wide Web Consortium
XML	extensible markup language
XSD	XML schema definition

This page intentionally left blank.

Preface

This standard is based on a document[1] presented at the 47th Annual International Telemetering Conference. The original document contained an overview of XML modeling with some initial guidelines. In response to RCC task TG-124, the document was expanded to establish guidelines that allow multiple schemas to be used in a mix-and-match fashion, identify best practices, and create a standard look and feel.

This standard was prepared by the Data Multiplex Committee of the Telemetry Group, Range Commanders Council. The XML Style Guide defines rules and guidelines for the development of modular XML schemas across the suite of schemas supported by the RCC (currently TMATS, MDL, IHAL, and DDML). The XML Style Guide is a common design reference for use by organizations that produce XML schemas, by ranges that receive XML instance documents that conform to the schemas, and by vendors who incorporate XML instance documents into their telemetry processing systems. The use of this style guide will ensure that all T&E XML schema standards share common design principles and a common look and feel in order to promote understanding, familiarity, and interoperability.

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

Telemetry Group 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

[1] Darr, Tim, John Hamilton, Ronald Fernandes, and Charles H. Jones. "Design Considerations for XML-Based T&E Standards." Paper presented during 47th Annual International Telemetering Conference, Las Vegas, NV. 24-27 October 2011.

This page intentionally left blank.

1. Introduction

The next generation of telemetry systems will rely heavily on extensible markup language (XML)-based standards. Multiple standards are currently being developed and reviewed by the Test and Evaluation (T&E) community. This document describes XML style guidelines to enable the development of modular XML schemas, which enables the ability to "mix and match" schema type, attribute, and element definitions from different schemas to facilitate reuse and interoperability and to use only parts of a schema that are needed. For example, this will enable the reuse of common schema structures such as a common units schema. In addition, these guidelines ensure that all T&E XML standards share common design principles and a common look and feel to promote understanding, familiarity, and interoperability.

Existing T&E standards (metadata description language [MDL][2], Telemetry Attributes Transfer Standard [TMATS] and data display markup language [DDML][3], instrumentation hardware abstraction language [IHAL][4]) cover a unique scope and define ways to describe several important T&E concepts. Ideally, future T&E systems and sub-systems will make use of concepts from several of these standards, combining portions of each standard that are relevant to the given sub-system. For example, a telemetry ground station may need to leverage concepts related to measurements (MDL), measurement packaging (MDL and TMATS XML), and data display (DDML). Similarly, an instrumentation engineer's system will need to make use of concepts related to instrumentation hardware (IHAL), measurements (MDL), and the relationships between instrumentation and measurements.

In the sections that follow, we provide a short introduction to XML for those that are not familiar with it, and then describe XML schema design guidelines that will enable the sharing of XML standards in existing and new standards. We illustrate the application of these guidelines using existing XML schemas.

Our intent is not to criticize the design of any existing standard. Rather, our purpose is threefold: (1) to document a common style for all IRIG 106 XML standards to promote consistency; (2) to highlight some common design practices that inhibit the integration and reuse of existing standards into new standards; and (3) to show alternative design practices that alleviate these issues. The XML schema design practices that prevent integration and reuse of existing standards include the following.

- Duplication of identical or nearly identical structures. It is not uncommon to repeat the same structure over and over again in the XML schema. It is sometimes easier to copy and paste structures instead of taking the time to design a proper schema.

---

[2] Moore, Michael S., Jeremy C. Price, Andrew R. Cormier, and William A. Malatesta. "Metadata description language: The iNET metadata standard language." Paper presented during 45th Annual International Telemetering Conference, Las Vegas, NV. 26-29 October 2009.

[3] Range Commanders Council. "Telemetry Attributes Transfer Standard," in *Telemetry Standards*. IRIG 106-15. June 2015. May be superseded by update. Retrieved 22 July 2015. Available at http://www.wsmr.army.mil/RCCsite/Documents/106-15_Telemetry_Standards/Chapter9.pdf.

[4] Hamilton, John, Ronald Fernandes, Paul Koola, and Charles H. Jones. "An overview of an instrumentation hardware abstraction language." Paper presented during 42nd Annual International Telemetering Conference, San Diego, CA. 23-26 October 2006.

- Duplication of elements. Similar to the previous practice, it is not uncommon to define local elements that refer to the same XML structures.
- Monolithic schemas. It is very common to design an XML schema as a single monolithic file. Decomposing a schema into logical components is a difficult task and requires careful design and thought but promotes reusability.

The next section provides an overview of XML and XML schemas for those that are not familiar with these technologies.

2. XML Overview

The XML standard is a specification produced by the World Wide Web Consortium (W3C), whose original intent was to provide a machine-readable format for describing documents.[5] Because of its popularity, wide adoption, and prevalence on the Internet, its use has expanded to describe arbitrary data structures such as web services and T&E metadata. The example in Figure 1 shows a portion of an XML document (an XML snippet) from an initial version of the TMATS XML schema.

```
<D:Measurement Name="WFA">
    <D:Parity>DE</D:Parity>
    <D:ParityTransferOrder>D</D:ParityTransferOrder>
    <D:MeasurementTransferOrder>D</D:MeasurementTransferOrder>
    <D:LocationType>WDFR</D:LocationType>
    <D:WordAndFrame>
        <D:IDCounterName></D:IDCounterName>
        <D:MeasurementLocation>
            <D:MeasurementFragments>
                <D:StartWord>1</D:StartWord>
                <D:WordInterval>0</D:WordInterval>
                <D:StartFrame>1</D:StartFrame>
                <D:FrameInterval>1</D:FrameInterval>
                <D:BitMask>FW</D:BitMask>
            </D:MeasurementFragments>
        </D:MeasurementLocation>
    </D:WordAndFrame>
</D:Measurement>
```

Figure 1. Example TMATS XML Snippet

In XML, each piece of data, or element, is surrounded by a "tag" such as `<D:Measurement>` and `<D:Parity>`. The structure of an XML file is such that tags can be enclosed in other tags to an arbitrary depth (`<D:MeasurementLocation>` is a sub-element of `<D:WordAndFrame>`, `<D:MeasurementFragments>` is a sub-element of `<D:MeasurementLocation>`, etc.). This is the basic idea behind the structure of an XML document.

The remainder of this section will get into more detail about the specific parts of an element and how a schema defines the rules for a specific XML document type.

[5] World Wide Web Consortium. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 26 November 2008. May be superseded by update. Retrieved 22 July 2015. Available at http://www.w3.org/TR/REC-xml/.

2.1 Anatomy of an XML Element

This section provides a brief overview of the structure of an XML element. The component parts of an XML element are identified in Figure 2. Each of these components is defined below the figure.

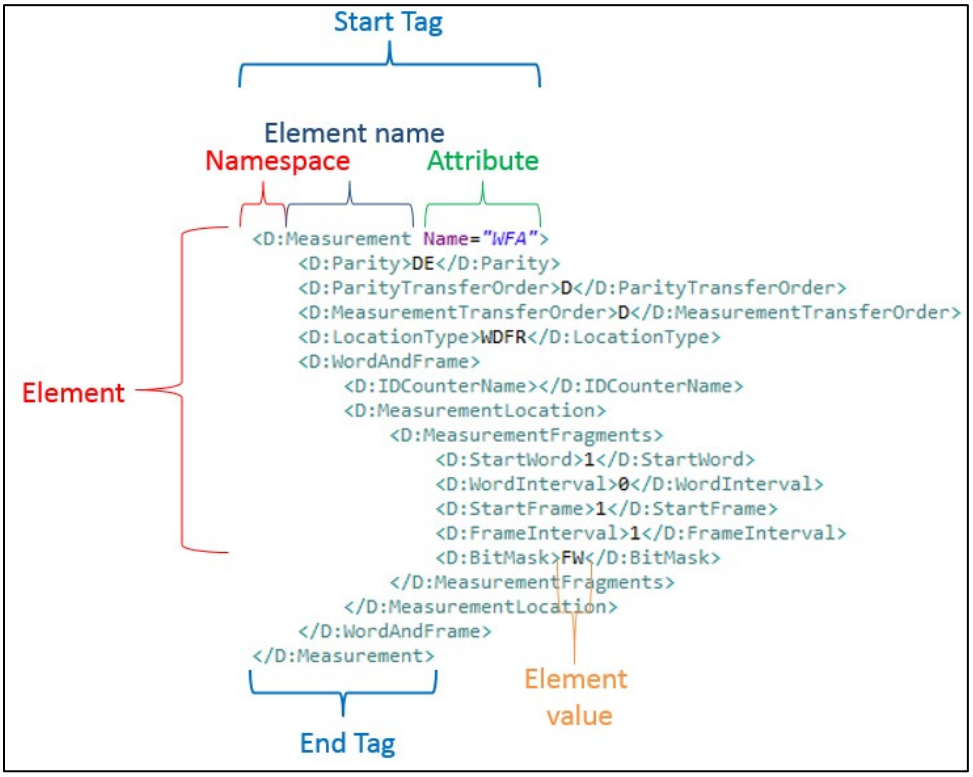

Figure 2. Components of an XML Element

- **Element:** "Element" is the term used to define a complete unit of XML information. It begins with a start tag and ends with an end tag. The value of an element can be a simple value or one or more sub-elements (children).
- **Start Tag:** The start tag identifies the beginning of the element and consists of the element's name (and possibly a namespace and attributes) included between a "<" and a ">" symbol.
- **End Tag:** The end tag identifies the end of the element and looks identical to the start tag, except it includes a "/" (forward-slash) after the "<" symbol. An end tag does not contain attributes.
- **Namespace:** The namespace is optional in XML, but can be used to define the scope within which the element is defined. In our TMATS example, we define a "d" namespace (for the TMATS D Group) and make all of the D Group elements members of it.
- **Element name:** The name of the element is what appears in the start and end tags and is what actually identifies the piece of information being defined.

- **Attribute:** Attributes are another method of associating information with an XML element. An attribute consists of a name followed by a "=" sign followed by a value enclosed in quotes. There is currently some controversy among users of XML as to when it is appropriate to use an attribute instead of simply adding a child element with the same name and value.
- **Element value:** The value of the element is everything that lies between the start tag and the end tag. The value can EITHER be a single value (e.g. 7, "John", true, etc.) OR a collection of one or more sub-elements (children).

2.2 XML Schemas

An XML schema is a design document used to describe a specific language that is based on XML. The rules for formatting proper XML are very simple and unrestricted. A schema defines which element names are valid, which elements can have which children, and which values are valid for each element. Element types organize XML documents by defining the allowed structure for specific groupings of elements.

Even though an XML schema is itself a document, it is usually more useful to view the schema as a diagram. In this document, we use diagrams generated by the XMLSpy® tool. In order to understand these diagrams, we'll use the TMATS XML schema.

An example schema diagram for our TMATS example is shown in Figure 3. The TMATS example shown in Figure 2 is a valid XML instance document that conforms to this schema. In this diagram, XML elements appear as boxes with their names printed inside. Attributes appear in an aptly named "attributes" box. For both attributes and elements, a solid border indicates that it is required while a broken or "dotted" border indicates that the element is optional. You will notice, for example, that both the "TmatsCommon:TmatsVersion" attribute and the `<Parity>` sub-element are optional.

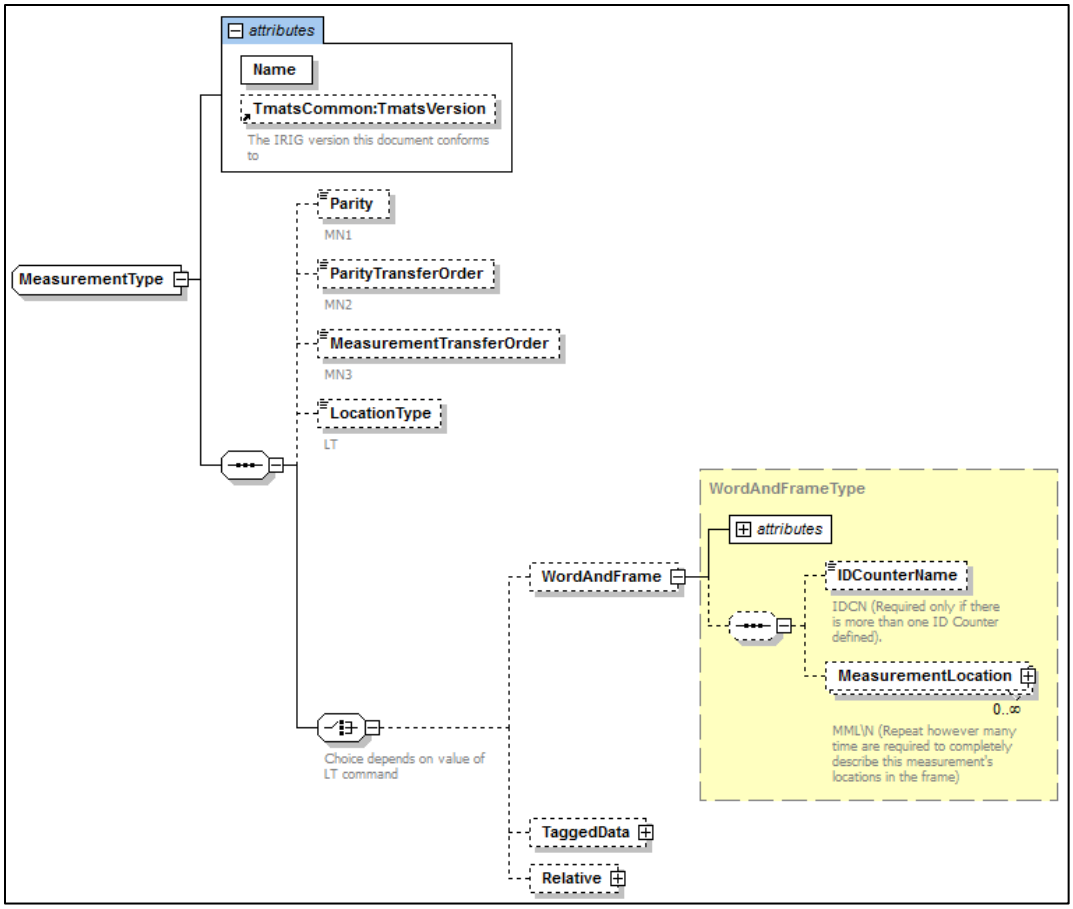

Figure 3. Example Schema Diagram

Sub-elements are connected to their parents with lines that pass through a special symbol. Each of these symbols specifies the rules for how the elements connected to the right side of it must appear in an instance document. A "sequence" symbol ( ) indicates that each of the child elements must appear in an instance document in the same order in which they appear in the schema. A "choice" symbol ( ) indicates that the instance document must contain exactly one of the child elements.

The XML and schema concepts presented in this section should be enough to understand all of the examples in the remainder of this paper.

2.3 Global vs. Local Definitions

Syntactically, global types and elements are defined as top-level elements in XML. Local types and elements are defined as sub-types or sub-elements of other XML components. Semantically, global types and elements can be reused throughout the rest of the schema while local types and elements can only be used in their local context.

3. The Case for Employing These Guidelines

There are costs for not employing the guidelines described in this paper and benefits for employing the guidelines. The costs include the following:

- Difficulty in understanding large schemas;
- Large schemas increase the size of other schemas that import them;
- Changing redundant schema components in multiple places is time-consuming and can lead to errors.

If a single, large, monolithic schema is used, it can be difficult for a human to understand the schema. This is especially true if the schema contains multiple naturally separable logical components.

Large schemas also cause problems when a user is interested in using a subset of that schema in another schema. In addition to the problem identified above in understanding the schema, importing (or including) a large schema has performance costs. The entirety of the schema must be imported (or included), which can cause problems in loading the schema or downloading the schema from the Internet.

If identical structures are defined multiple times and the user needs to change this structure in some way, the user must search for all instances of that structure and make the desired changes. This takes time, and if the user misses one of the structures, errors will be introduced into the schema. This is not so much of an issue for a structure that is not likely to change, but for evolving structures, this could be a major issue.

The benefits of following the guidelines include the following:

- Promotes the use of a schema in other schemas;
- Reduces the amount of time needed to understand a schema;
- Reduces the time needed to make changes;
- Reduces the errors when changing the schema;
- Allows composability, which is the ability to use only what you need.

The primary benefit of these guidelines is the reuse of schema structures in other schemas. During the development of the IHAL standard, it was necessary to represent measurement units. There were several candidate options: develop a custom representation of units, reuse the UnitsML schema standard developed by the National Institute of Standards and Technology[6], or reuse the units representation that is part of MDL. The IHAL team chose to reuse the MDL representation for two primary reasons: it promotes interoperability between MDL and IHAL and it reduces the amount of work that would have been required to create a new units representation.

By modularizing a schema, it becomes easier for a user to understand. In the example given previously, by separating out the TMATS R group schema, it would be easier for a new user of the schema to understand not only the R group, but also the other groups. If a user is interested in using only a small portion of a schema, such as the TMATS pulse code modulation (PCM) measurement structures, a modular schema provides a way to only use what you need.

[6] National Institute of Standards and Technology. Units Markup Language home page. http://unitsml.nist.gov/. Accessed 14 July 2015.

By defining XML types and elements globally and reusing these structures by reference, maintenance becomes much easier. Changing types only needs to be done in one place, reducing the time to make changes. Having to make changes to only one structure significantly reduces the potential for errors. Extending a type becomes much easier as well, as this only needs to be done in a single place.

4. Guidelines

This section describes the guidelines and illustrates their use via examples taken from existing T&E schemas.

4.1 Modularity

The following guidelines apply to schema modularity.[7]

- XML schema files should import and include other XML schema files rather than duplicating these elements locally
- Schemas should be specified in such a way that other schemas can leverage them. Details are provided in Section 4.5.

Attempt to organize the schema using logical modules as much as possible. This promotes composability of new schemas from existing schemas. Figure 4 shows the TMATS schema organized into modules that roughly correspond to the TMATS groups as documented in IRIG 106 Chapter 9. The dependencies between pairs of modular sub-schemas is captured using the XML import relationship. For example, the top-level Tmats.xsd schema depends only on the TmatsGGroup.xsd schema; the TmatsGGroup.xsd depends on all the other schemas since the G Group references all the other groups.

[7] David Stephenson. *XML Schema Best Practices*. December 2004. Retrieved 15 July 2015. Available at http://xml.coverpages.org/HP-StephensonSchemaBestPractices.pdf.

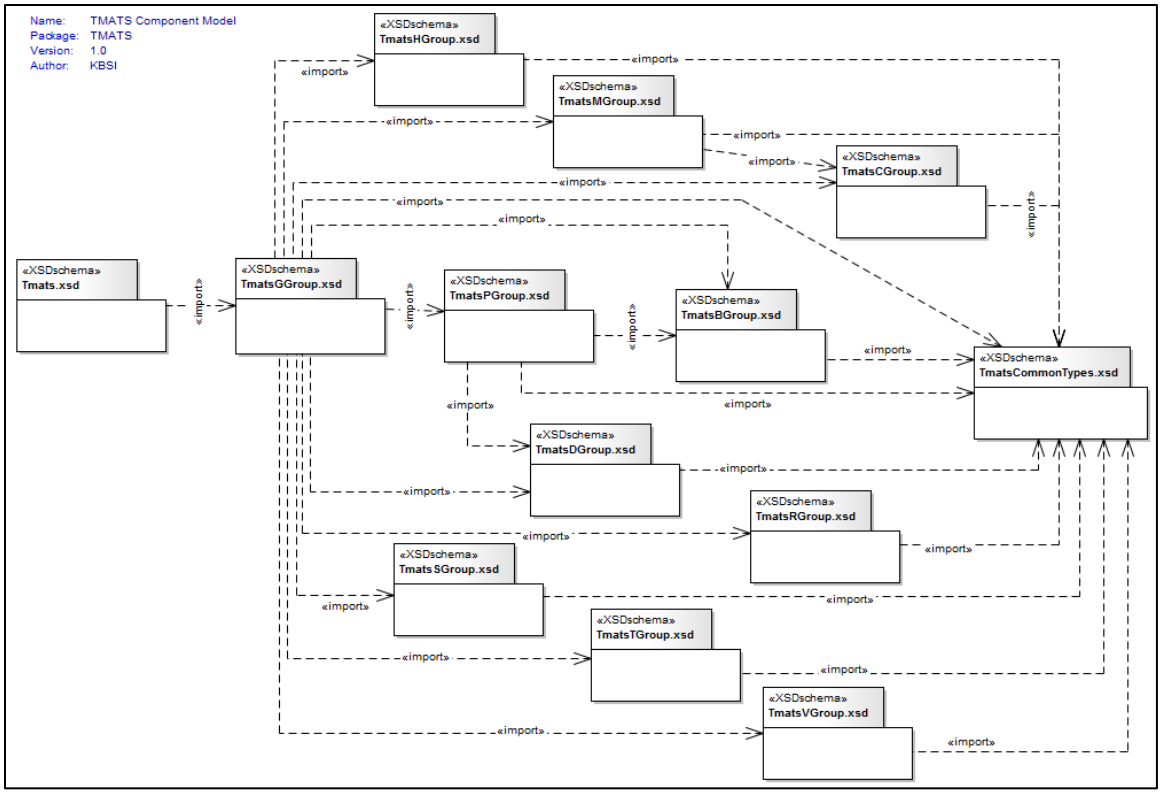

Figure 4. TMATS Schema Modularity

Figure 5 shows how each schema is imported into the TMATS G Group schema (TmatsGGroup.xsd in Figure 4) using the XML schema import statement. Each schema import includes the location of the schema and the namespace of the schema that is being imported. The location of the schema can be on the local file system or a remote location on the network. For example, the location of the TMATS common schema (TmatsCommonTypes.xsd in Figure 4) is TmatsCommonTypes.xsd in the local directory and the namespace is http://www.wsmr.army.mil/RCCsite/Documents/106-15_TelemetryStandards/TmatsCommon. The namespace is a unique identifier for the schema as a whole. The namespace of a given schema is defined in the schema itself. The important point is that the namespace that is in the import statement must match the namespaces as defined in the schema.

import	loc:TmatsCommonTypes.xsd	ns:http://www.wsmr.army.mil/RCCsite/Documents/106-13_Telemetry%20Standards/TmatsCommon
import	loc:TmatsBGroup.xsd	ns:http://www.wsmr.army.mil/RCCsite/Documents/106-13_Telemetry%20Standards/TmatsB
import	loc:TmatsCGroup.xsd	ns:http://www.wsmr.army.mil/RCCsite/Documents/106-13_Telemetry%20Standards/TmatsC
import	loc:TmatsDGroup.xsd	ns:http://www.wsmr.army.mil/RCCsite/Documents/106-13_Telemetry%20Standards/TmatsD
import	loc:TmatsHGroup.xsd	ns:http://www.wsmr.army.mil/RCCsite/Documents/106-13_Telemetry%20Standards/TmatsH
import	loc:TmatsMGroup.xsd	ns:http://www.wsmr.army.mil/RCCsite/Documents/106-13_Telemetry%20Standards/TmatsM
import	loc:TmatsPGroup.xsd	ns:http://www.wsmr.army.mil/RCCsite/Documents/106-13_Telemetry%20Standards/TmatsP
import	loc:TmatsRGroup.xsd	ns:http://www.wsmr.army.mil/RCCsite/Documents/106-13_Telemetry%20Standards/TmatsR
import	loc:TmatsSGroup.xsd	ns:http://www.wsmr.army.mil/RCCsite/Documents/106-13_Telemetry%20Standards/TmatsS
import	loc:TmatsTGroup.xsd	ns:http://www.wsmr.army.mil/RCCsite/Documents/106-13_Telemetry%20Standards/TmatsT
import	loc:TmatsVGroup.xsd	ns:http://www.wsmr.army.mil/RCCsite/Documents/106-13_Telemetry%20Standards/TmatsV
complexType	**DataLinkType**	ann:
complexType	**DataSourceType**	ann:
complexType	**RevisionAndUpdateType**	ann:
complexType	**TestInformationType**	ann:
complexType	**Tmats**	ann:TMATS G-Group

Figure 5. TMATS Schema Modularity Imports

Figure 6 shows how the namespace of a schema is defined within the schema itself, using TMATS as an example. The namespace of the schema is defined using the targetNamespace attribute as shown in the figure. Note that a namespace prefix is also defined for the TMATS schema using the xmlns:Tmats attribute. Namespace prefixes are described below. Best practices for using namespaces are given later in this document.

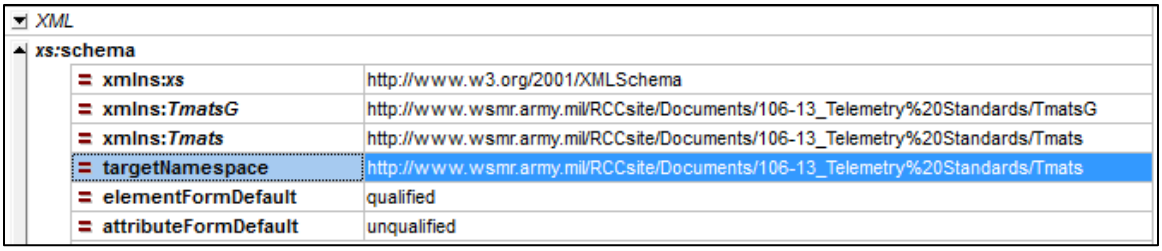

Figure 6. Example TMATS Namespace Definition

Figure 7 shows the modular IHAL use schema that is composed of other IHAL schemas as well as external schemas (XidML, TMATS, MDL). The IHAL InstrumentUse.xsd schema depends on types defined in the XidML Network-Transport.xsd schema, the TmatsRGroup.xsd and TmatsPGroup.xsd schemas, and the MDL_v0_8_17.xsd schema. The IHAL schema depends on these schema modules to facilitate interoperability, which guarantees that the same information is represented in the same way across schemas. The IHAL CommonUnitsSchema.xsd depends on the MDL units schema as a way to reuse XML schema implementations already defined in another schema.

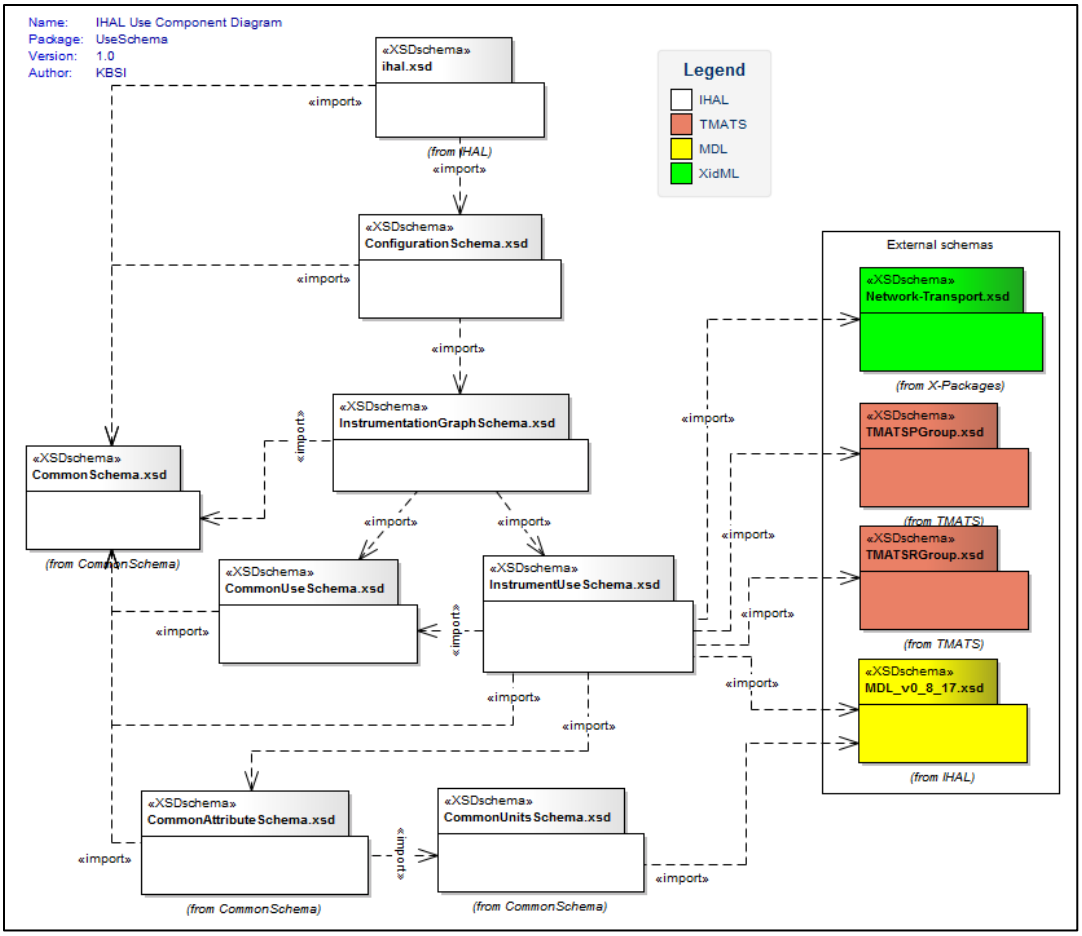

Figure 7. IHAL Use Schema Modularity and Composability

4.2 Schema Format

Define the schema according to the W3C XML schema definition (XSD).[8]

4.3 Naming Conventions

The following guidelines apply to naming conventions (Stephenson 2004).

- Elements and attributes should be named consistently using camel case.
- Simple and complex types should be named consistently using Pascal case.

Enumerated values and names of types, elements, and attributes should be concise and informative (less than 25 characters). Standard abbreviations (i.e., mA) and domain acronyms (i.e., TandE) should be used with care.[9] Names should be Pascal case or camel case. Pascal case is a naming convention in which the first letter of a name and the first letter of each concatenated

[8] Gao, Shudi, C. M. Sperberg-McQueen, and Henry S. Thompson. *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. 5 April 2012. Retrieved 22 July 2015. Available at http://www.w3.org/TR/xmlschema11-1.
[9] Google. *Google XML Document Format Style Guide*. 2008. Retrieved 21 July 2015. Available at https://google-styleguide.googlecode.com/svn/trunk/xmlstyle.html.

word is capitalized (i.e., MeasurementList, RecorderType).[10] Camel case is a naming convention in which the first letter of a name is lower-case and the first letter of each concatenated word is capitalized (i.e., signalConditioningCard, busAttributes) (Google 2008).

Names of element types must only contain ASCII letters and digits and should start with an ASCII letter. Simple and complex type names must be Pascal case. Figure 8 illustrates positive and negative examples of this convention. In Figure 8(a), the type name is Pascal case. In Figure 8(b), the type name is not Pascal case.

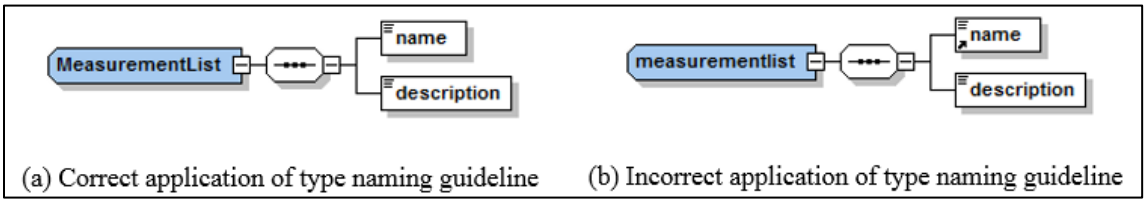

Figure 8. Application of Type Naming Convention

Element names must only contain ASCII letters and digits and should start with an ASCII letter. Elements names must be camel case. Figure 9 illustrates positive and negative examples of this convention. In Figure 9(a), the element name is camel case. In Figure 9(b), the type name is not camel case.

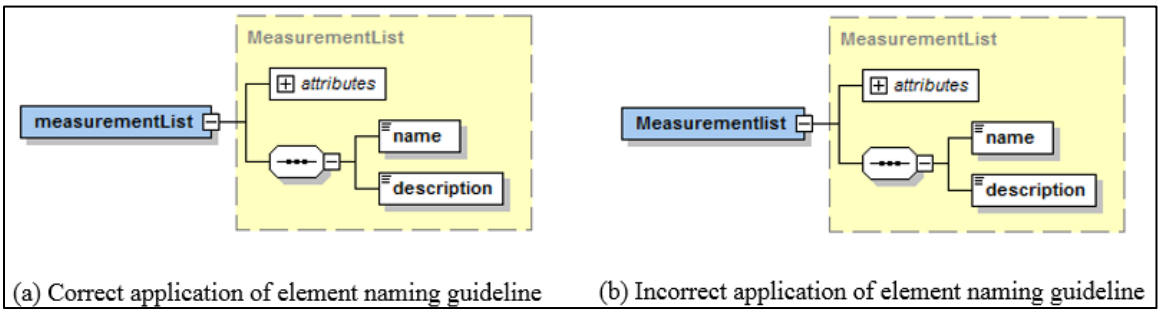

Figure 9. Application of Element Naming Convention

Attribute names must only contain ASCII letters and digits and should start with an ASCII letter. Element and attribute names must be camel case. Figure 10 illustrates positive and negative examples of this convention. In Figure 10(a), the attribute name is camel case. In Figure 10(b), the attribute name is not camel case.

[10] Microsoft Developer Network. *Capitalization Styles*. n.d. Retrieved 21 July 2015. Available at https://msdn.microsoft.com/en-us/library/x2dbyw72(v=vs.71).aspx.

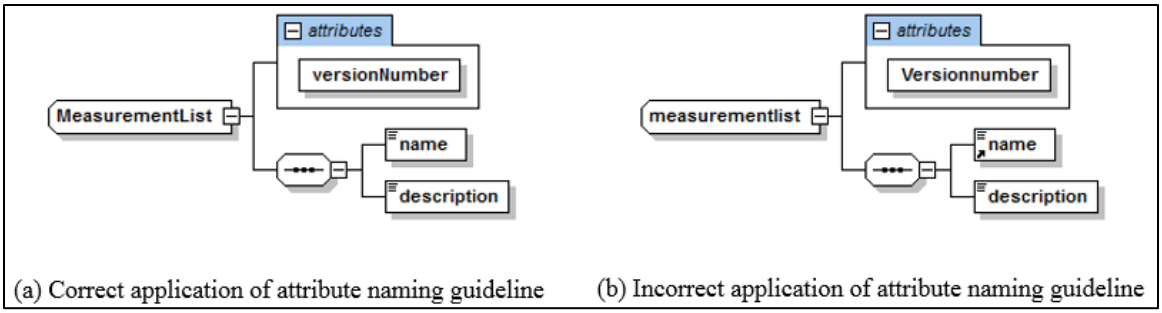

Figure 10. Application of Attribute Naming Convention

Enumerated values must only contain ASCII letters, digits, and whitespace and should start with an ASCII letter or digit.

4.4 Namespaces

The following is a summary of the best practices for using namespaces (Stephenson 2004).

- A target namespace shall always be specified for a schema.
- The target namespace and the schema namespace shall be mapped to prefixes.
- A default namespace shall not be used.
- Element names shall always be qualified.
- Attribute names shall not be qualified.

The XML namespaces are used to uniquely identify the schema types and elements within an XML schema to avoid name conflicts. As mentioned in a previous section, the targetNamespace attribute in a schema definition defines the uniform resource identifier (URI) for the schema. Conflicts can appear when an XML schema imports another XML schema (the modularity principle). For example, both TMATS and MDL include a representation of a measurement. When creating an integrated schema that includes elements of both TMATS and MDL, it is necessary to have a way to distinguish the TMATS measurement from the MDL measurement. This is done using namespaces.

Namespaces fully qualify the types or elements within a schema or instance document using the syntax *namespace:SchemaType* or *namespace:elementName*. For example, the namespace of the TMATS G Group is http://www.wsmr.army.mil/RCCsite/Documents/106-15_TelemetryStandards/TmatsG. The fully qualified name of the DataLinkType schema type is http://www.wsmr.army.mil/RCCsite/Documents/106-15_TelemetryStandards/TmatsG:DataLinkType. This is an unwieldy way to reference a type or element, so the XML schema provides the prefix concept as a way to short-hand the namespace. If the user defines the TMATS G Group namespace prefix as TmatsG, then the fully qualified reference to the DataLinkType becomes TmatsG:DataLinkType. In an XML schema, namespaces are defined by the xmlns attribute in the top-level element and followed by the element prefix.

Figure 11 shows a root element from a TMATS XML file. The attributes in the top-level Tmats:Tmats element define the namespace prefixes for the instance document. For example,

the attribute xmlns:Common defines the namespace prefix for the TMATS Common Types schema.

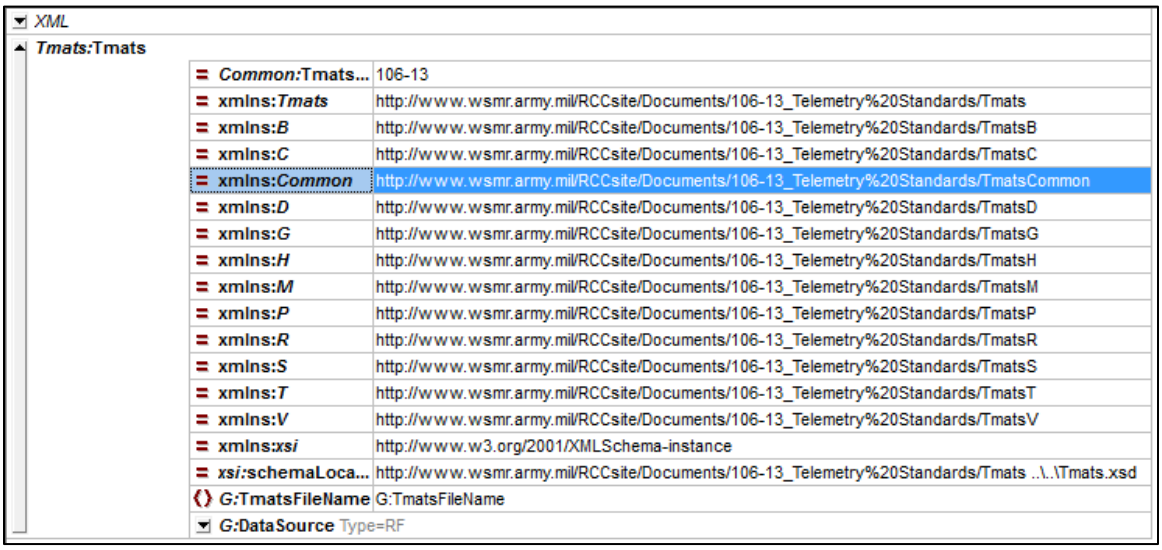

Figure 11. TMATS XML Example Root Element, Prefixes, and Namespaces

Figure 12 shows how namespace prefixes are used in a snippet of a TMATS XML instance document. Each element in the file includes the respective namespace prefix so that there is no ambiguity about the source of the element. For example, the TMATS data source is defined in the G group, so its namespace prefix is G (G:DataSource). Navigating down through the data source element, we eventually come to an element from a different namespace for the PCM measurements element defined in the P group (P:PCMMeasurements).

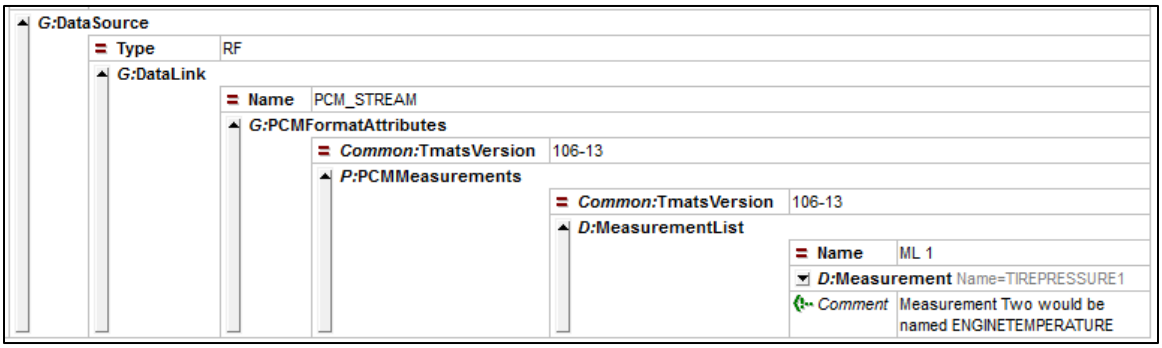

Figure 12. XML Example File

Namespace prefixes in XML should be short and contain only lower-case letters (Google 2008). Child elements with the same prefix are associated with namespaces defined for parent elements. Namespace prefixes should not be used in attribute names since it can make the namespaces difficult to read (this is achieved by setting attributeFormDefault=unqualified, described below).

Namespace names should be URIs that preferably resolve to an actual URI. The namespace should have some indication of the version to which the schema refers. For example, the root IHAL namespace is http://www.wsmr.army.mil/RCCsite/Documents/106-

15_TelemetryStandards/ihal, where the version is given by "106-15_Telemetry Standards." Namespaces must not be changed unless there is significant change to the referred schema. Changing a namespace has significant effects on all schemas that refer to it.

There is some flexibility in how namespace prefixes are used in the context of *local* elements and attributes only. Global elements always include namespace prefixes. The designer may not want XML instance documents to show the namespace prefix for local elements and attributes for a variety of reasons (our best practice is to show namespace prefixes for elements but not attributes). This is done by specifying the element and attribute form qualification in the schema definition using the attributeFormDefault and elementFormDefault attributes, respectively.[11] We illustrate these concepts using three cases: unqualified elements and attributes, qualified elements and attributes, and qualified elements and unqualified attributes.

Figure 13 shows an example schema definition for unqualified attributes and elements. Note that the elementFormDefault and attributeFormDefault attributes are both set to "unqualified." Note also that there are three global elements defined: MeasurementList, Measurement, and PCMMeasurements.

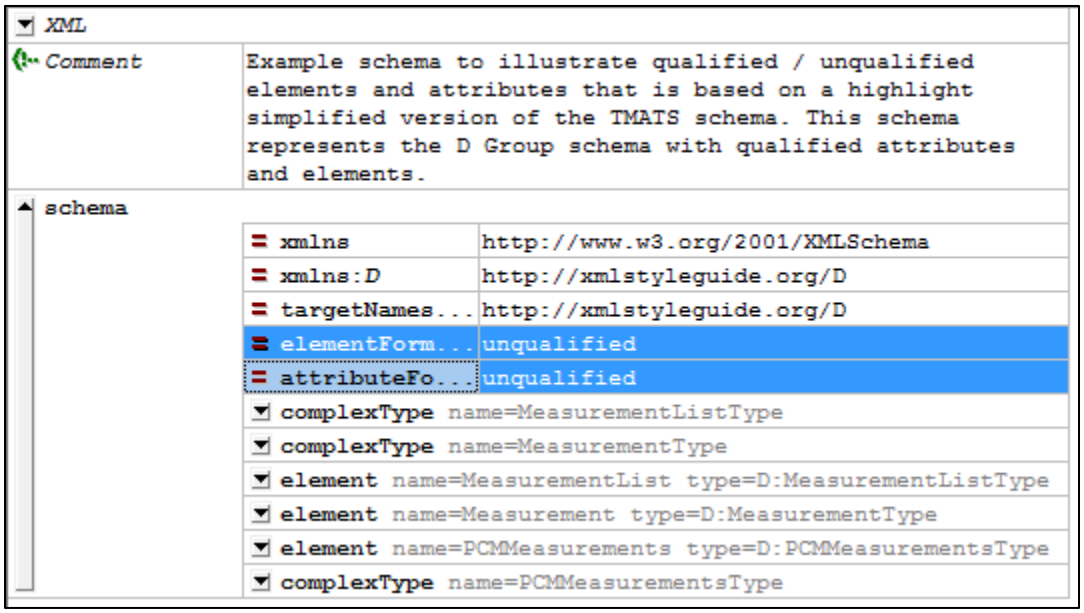

Figure 13. Example Schema with Unqualified Elements and Attributes

Figure 14 shows an example instance document snippet that conforms to the schema above. Note that the global elements (PCMMeasurements, MeasurementList, and Measurement) are qualified with a namespace prefix (D:PCMMeasurements, D:MeasurementList, and D:Measurement) since the element qualification does not apply to global elements. Neither the attributes (ID) nor the local elements (Name and Parity under the D:Measurement element) are qualified with a prefix.

[11] W3Schools. *XML Schema Element*. n.d. May be superseded by update. Retrieved 22 July 2015. Available at http://www.w3schools.com/schema/el_schema.asp.

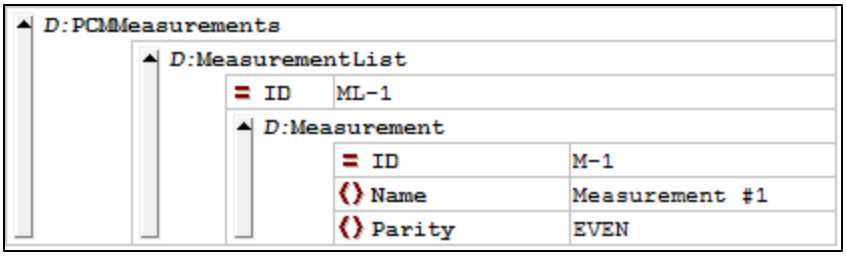

Figure 14. Example Instance Snippet with Unqualified Elements and Attributes

Figure 15 shows an example instance document snippet that conforms to the schema above with the modification that both elements and attributes are qualified. Both the attributes (D:ID) and the local elements (D:Name and D:Parity under the D:Measurement element) are qualified with a prefix.

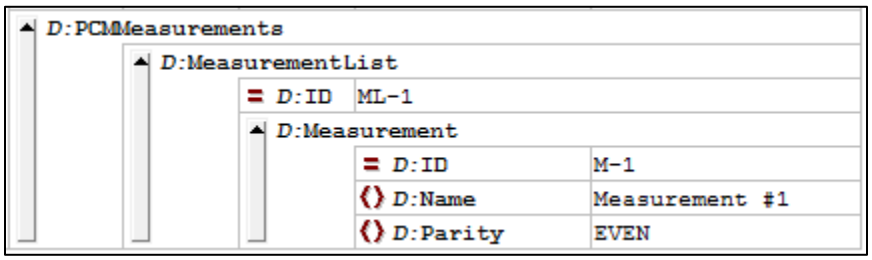

Figure 15. Example Instance Snippet with Qualified Elements and Attributes

Figure 16 shows an example instance document snippet that conforms to the schema above with the modification that the elements are qualified and attributes are unqualified. The attributes (ID) are not qualified with a prefix; the local elements (D:Name and D:Parity under the D:Measurement element) are qualified with a prefix.

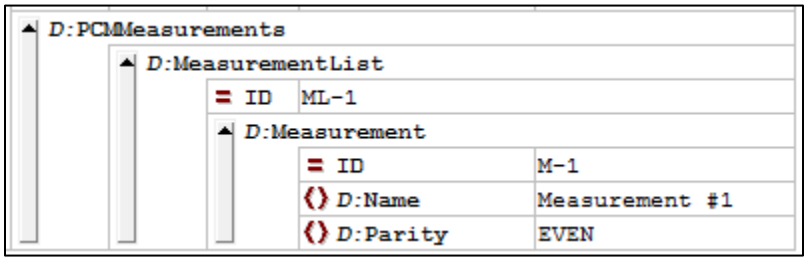

Figure 16. Example Instance Snippet with Qualified Elements and Unqualified Attributes

The rationale behind the best practice for qualified elements and unqualified attributes is the following.

- Qualified elements make the namespace where the element is defined explicit. Since global elements are always qualified, it might be confusing to the user of the schema to see some elements qualified and some not.

- Unqualified attributes make the attribute names more concise. Since there is no distinction between global and local attributes, there is no possible confusion as in the case for elements.

4.5 Global vs. Local Types

The following is a summary of the best practices for global and local types (Stephenson 2004).

- Elements should be defined globally.
- Complex and simple types shall always be defined globally.

A component (element, complex type, or simple type) is global if it is an immediate child of the top-level <schema/> element; it is local if it is nested within another component.[12] There are three patterns that are commonly used to illustrate the global vs. local decision, as described in the following sub-sections. Our best practice is to use the Venetian Blind pattern.

4.5.1 Russian Doll Design

In this pattern, the schema structure mirrors the element structure; that is, the schema components are defined locally to the component that it is contained in. This is the "local" end of the global vs. local type spectrum. Figure 17 and Figure 18 illustrate this pattern for TMATS measurements from an early version of the TMATS XML schema (Tmats_05-2007.xsd). Note that the elements and the structure of the Measurement location element are defined local to the Measurement element. Figure 17 graphically illustrates the Russian Doll design. Figure 18 illustrates the Russian Doll design in XML text.

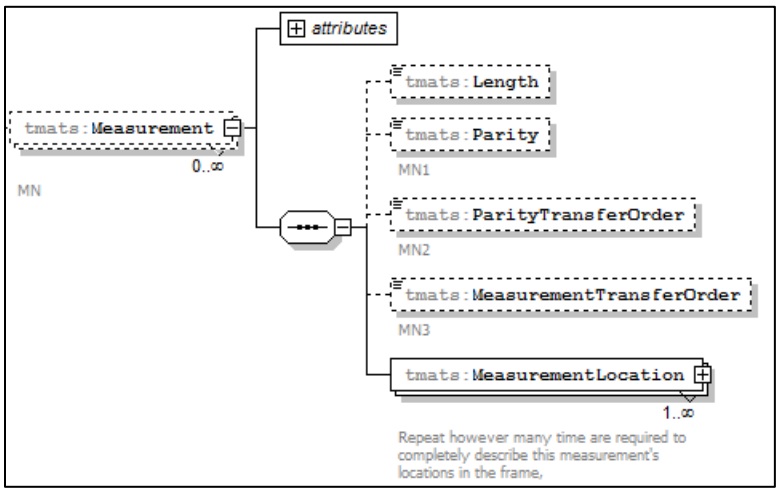

Figure 17. Graphical Illustration of Russian Doll Design - TMATS Measurement

[12] Costello, et al. "Global versus Local", in *XML Schemas: Best Practices*. 1 November 2006. Retrieved 22 July 2015. Available at http://www.xfront.com/GlobalVersusLocal.html.

```
<xs:element name="Measurement" minOccurs="0" maxOccurs="unbounded">
    <xs:annotation>
        <xs:documentation>MN</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Length" type="xs:positiveInteger" minOccurs="0"/>
            <xs:element name="Parity" default="Default" minOccurs="0">
            <xs:element name="ParityTransferOrder" default="Default" minOccurs="0">
            <xs:element name="MeasurementTransferOrder" default="Default" minOccurs="0">
            <xs:element name="MeasurementLocation" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Repeat however many time are required to
                    completely describe this measurement's locations in the
                    frame,</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="MeasurementFragments" maxOccurs="8">
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="Name" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
```

Figure 18. Textual Illustration of Russian Doll Design - TMATS Measurement

Characteristics of the Russian Doll design include the following (Costello et al. 2006).

- Opaque – the content of each component is opaque to other parts of the schema and cannot be reused.
- Localized scope – if the schema has a default element form unqualified, then the namespaces of the contained elements are hidden; this makes namespace management easier.
- Compact – everything is bundled together.
- Decoupled – changes to the component will have limited impact; the contents can be changed with minimal (if any) impact to other parts of the schema or other schemas that might include it.
- Cohesive – all data is grouped into self-contained components.

The Russian Doll design pattern hides namespace complexities and limits or prohibits reuse.

4.5.2 Salami Slice Design

In this pattern, the schema is disassembled into individual components. Each component is defined as an element declaration and then assembled together as needed. This is the "global" end of the global vs. local type spectrum. Figure 19 and Figure 20 illustrate this pattern for the IHAL instrumentation graph. Note that the instrumentation graph element is global, the instrumentation graph type is global, and the sub-elements instrument use and connection are

references to globally defined elements. Figure 19 graphically illustrates the Salami Slice design. Figure 20 illustrates the Salami Slice design in XML text.

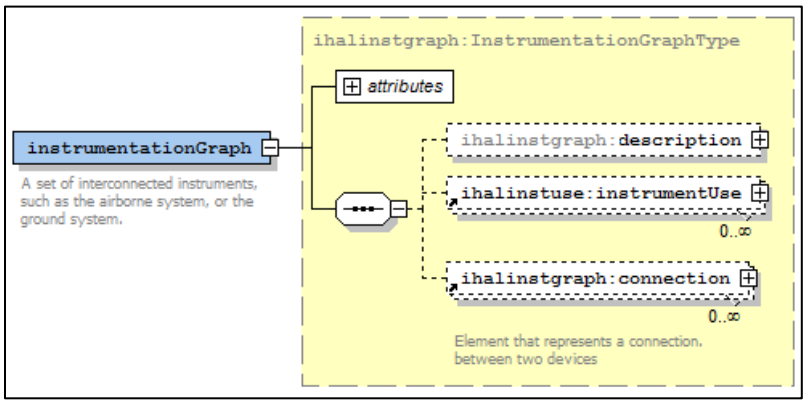

Figure 19. Graphical Illustration of Salami Slice Design - IHAL Instrumentation Graph

```
<xs:element name="instrumentationGraph" type="ihalinstgraph:InstrumentationGraphType">
    <xs:annotation>
        <xs:documentation>A set of interconnected instruments,
        such as the airborne system, or the ground system.
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:complexType name="InstrumentationGraphType">
    <xs:sequence>
        <xs:element name="description" type="ihalcommon:DescriptionType" minOccurs="0"/>
        <xs:element ref="ihalinstuse:instrumentUse" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="ihalinstgraph:connection" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute ref="ihalcommon:ID" use="required"/>
    <xs:attribute ref="ihalcommon:ihalVersion"/>
</xs:complexType>
```

Figure 20. Textual Illustration of Salami Slice Design - IHAL Instrumentation Graph

Characteristics of the Salami Slice design include the following (Costello et al. 2006).

- Transparent – the content of each component is visible to other parts of the schema and can be reused.
- Global scope – regardless of the element form default value, the namespaces of the components will be visible in instance documents.
- Verbose – every component is visible.
- Coupled – changes to the component(s) can impact other components that use it; the components are interconnected.
- Cohesive – all data is grouped into self-contained components.

The Salami Slice design pattern facilitates component reuse of the maximum extent, but requires coordination and careful thought in the naming and design of global elements and types.

4.5.3 Venetian Blind Design

In this pattern, the schema design consists of components (like the Salami Slice design), but elements can be defined locally. This design maximizes reuse and the potential for local namespace hiding. Figure 21 and Figure 22 illustrate this pattern for MDL measurements. Note that the elements of these types are defined local to the enclosing structure. Figure 21 graphically illustrates the Venetian Blind design. Figure 22 illustrates the Venetian Blind design in XML text.

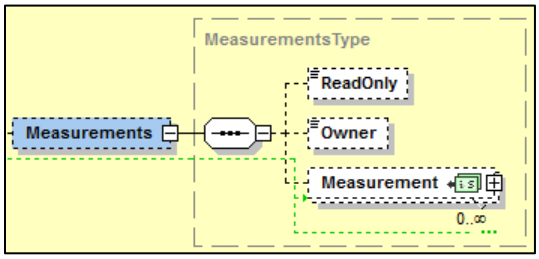

Figure 21. Graphical Illustration of Venetian Blind Design - MDL Measurements

```
<xsd:complexType name="MeasurementsType">
  <xsd:annotation>
  <xsd:sequence>
    <xsd:element name="ReadOnly" type="xsd:boolean" default="false" minOccurs="0"/>
    <xsd:element name="Owner" type="xsd:token" minOccurs="0"/>
    <xsd:element name="Measurement" type="MeasurementType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Figure 22. Graphical Illustration of Venetian Blind Design - MDL Measurements

Characteristics of the Venetian Blind design include the following.

- Maximum reuse – type definitions are defined globally.
- Maximum namespace hiding – element declarations are enclosed within type definitions.
- Easy exposure switching – the elementFormDefault attribute controls namespace visibility.
- Coupled – global type definitions generate interconnectedness.
- Cohesive – all data is grouped into self-contained components.

Following the Venetian Blind design, the following guidelines shall be followed (Costello et al. 2006).

- Use global type definitions as the main form of component reuse.
- Global element definitions are encouraged but not required.

4.6 Attribute Guidelines

The following guidelines apply to XML attributes (Google 2008).

- An ordering of the attributes defined for an element in an instance document shall not be assumed since XML parsers may not enforce this.
- Attributes should be used for IDs and IDREFs only.
- Attributes that might contain a value with a line break shall not be used in an instance document since XML parsers might not process this correctly.
- A difference between single or double quotes around the attribute value in an instance document shall not be assumed since most XML parsers treat them the same.

4.7 Value Guidelines

The following guidelines apply to XML element and attribute values (Google 2008).

- Numeric values shall be 32-bit signed integers (xsd:int), 64-bit signed integers (xsd:long), or 64-bit IEEE doubles (xsd:double), all expressed in base 10.
- Boolean values shall be typed to xsd:Boolean and only have the values true and false.
- Dates shall be represented in RFC 3339 format (xsd:dateTime).[13] The UTC shall be used rather than local times.

4.8 XML Instance Document Guidelines

The following guidelines apply to XML instance documents (Google 2008).

- UTF-8 character encoding shall be used.
- Namespaces shall be declared in the root element of the document.
- Namespace URIs shall be mapped to prefixes one time in the document.
- Standard namespaces prefixes for standard URIs shall be used.
- Standard namespace prefixes for IRIG 106 schemas shall be used.
    - tmats, tmatsa, tmatsb, etc. for the TMATS schema, mdl for the MDL schema, etc.
- Redundant whitespace in component tags should be minimized.
- CDATA shall not be used.
- Comments shall be written to support automated documentation generation.

4.9 Guideline for Using Attributes or Elements

The following guideline applies for when to use an attribute or element to represent a piece of information (Google 2008). Attributes are more restrictive than elements, so an all-element design is the simplest. Elements use more memory than attributes internally and make the XML file itself larger because of the use of begin and end tags. In some programming languages or libraries, the processing of attributes and elements is very different, so there could

[13] Internet Engineering Task Force. *Date and Time on the Internet: Timestamps*. RFC 3339. July 2002. May be superseded by update. Retrieved 16 July 2015. Available at http://www.rfc-editor.org/rfc/pdfrfc/rfc3339.txt.pdf.

be performance implications to the choice of attributes vs. elements. Take into consideration the processing of the XML instance documents when making this decision.

The specific guideline for attributes is to use them only if the data being modeled is an ID or an IDREF to some other piece of data (Google 2008). This is not a hard and fast rule, but should be discussed and reviewed at design time.

4.10 Other Guidelines

The following other guidelines apply.

- Binary data shall not be used.
- An ID shall only be used as a reference for an IDREF.[14]
- Namespaces shall contain a version.
- Schemas shall be compatible with the following libraries: XML Beans[15], JAXB[16].

5. Illustration of Use of Guidelines

This section illustrates the use of the guidelines in a semi-realistic modeling situation. The modeling problem is as follows: define an XML schema using the best practices to model T&E instrumentation hardware including data acquisition units (DAUs), bus controllers, recorders, signal conditioning cards, and measurements. The intent of this exercise is to illustrate the best practices and not necessarily model a real T&E scenario.

The modeling requirements are as follows.

- DAUs are defined by a name, description, bus type, and recorder type.
- Bus controllers are defined by a name, description, and bus type.
- Recorders are defined by a name, description, and recorder type.
- Signal conditioning cards are defined by a name, description, and amplification.
- Measurements are defined by a name, description, and sample rate.

Figure 23 is a schema that meets the requirements of the model, but it violates several of the best practices. There is no consistent naming convention. There is no reuse of common structures. There is no modularity. We will incrementally modify this design to illustrate the best practices.

[14] Dare Obasanjo. *W3C XML Schema Design Patterns: Avoiding Complexity.* January 2003. Retrieved 17 July 2015. Available at http://msdn.microsoft.com/en-us/library/aa468564.aspx#xmlscmavdcmplx_topic13.
[15] "Welcome to XMLBeans." The Apache XML Project. Last published 15 August 2012. Retired 23 May 2014. http://xmlbeans.apache.org/.
[16] "Project JAXB." Java.net. Last published 14 October 2014. https://jaxb.java.net/.

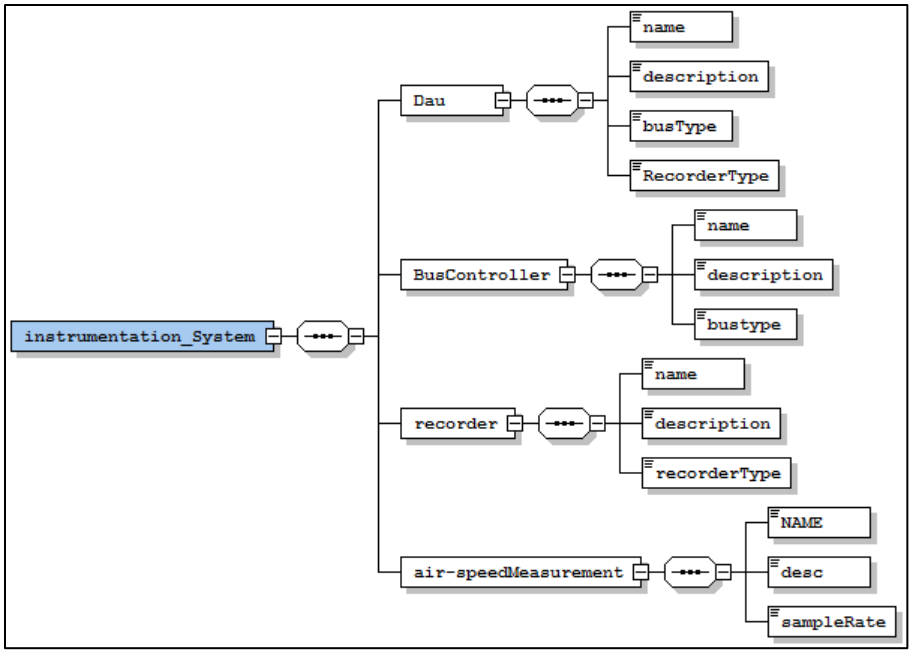

Figure 23. Initial Example Schema Design

Figure 24 shows the initial schema with the naming convention applied to the schema elements. The readability of the schema is improved by using the naming convention best practice. This is also an example of the Russian Doll design pattern.

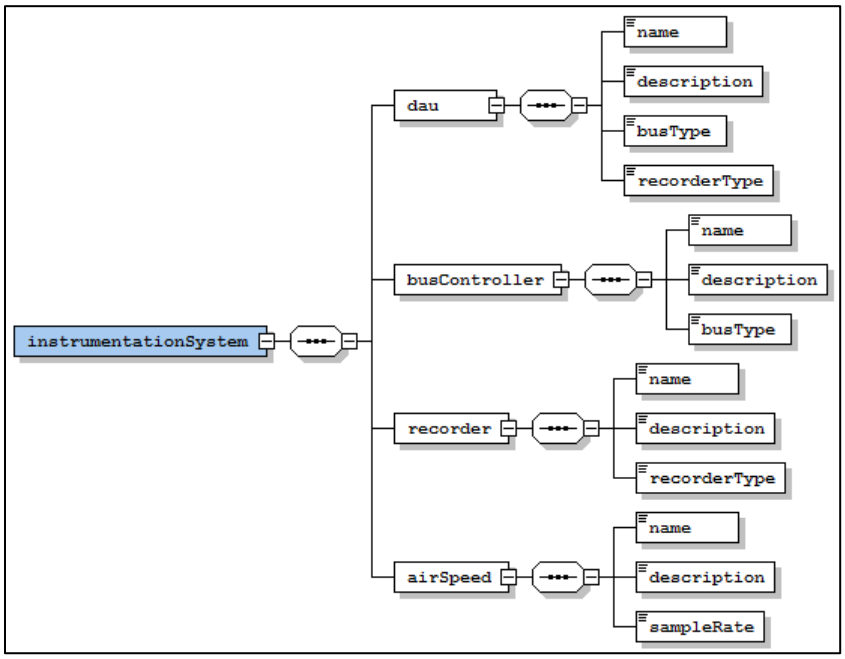

Figure 24. Example Schema with Naming Convention Best Practice

There are still several problems with this schema, such as: each element is defined locally; even though the "name" element is used in multiple places, it is defined in a single, global way; there is no reuse of common structures. To remedy this, we create global simple types for names, descriptions, bus types, etc., and use those type definitions in each of the

structures in our schema. When we do this, we make sure that the naming convention for types is followed. Figure 25 shows the schema with global simple types defined for name, description, and bus type. This modification promotes reuse of these types. With this modification, we partially achieve the Venetian Blind design pattern.

```
<xs:simpleType name="NameType">
    <xs:restriction base="xs:string">
        <xs:length value="16"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="DescriptionType">
    <xs:restriction base="xs:string">
        <xs:length value="40"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="BusType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="MIL-STD 1553"/>
        <xs:enumeration value="ARINC 129"/>
    </xs:restriction>
</xs:simpleType>

<xs:element name="name" type="NameType"/>
<xs:element name="description" type="DescriptionType"/>
<xs:element name="busType" type="BusType"/>
<xs:element name="recorderType" type="recorder-type"/>
```

Figure 25. Example Schema with Global Types

While the modifications made so far improve the initial design in several ways, there is still opportunity to improve modularity and achieve a fully Venetian Blind design pattern. In the schema as it now is, the DAU, bus controller, recorder, and air speed measurement share a name and description structure. To achieve a more modular schema, we create a GenericInstrumentType complex type that consists of a name and a description and extend this GenericInstrumentType for the DAU, bus controller, recorder, and measurement types. Figure 26 shows this schema modification.

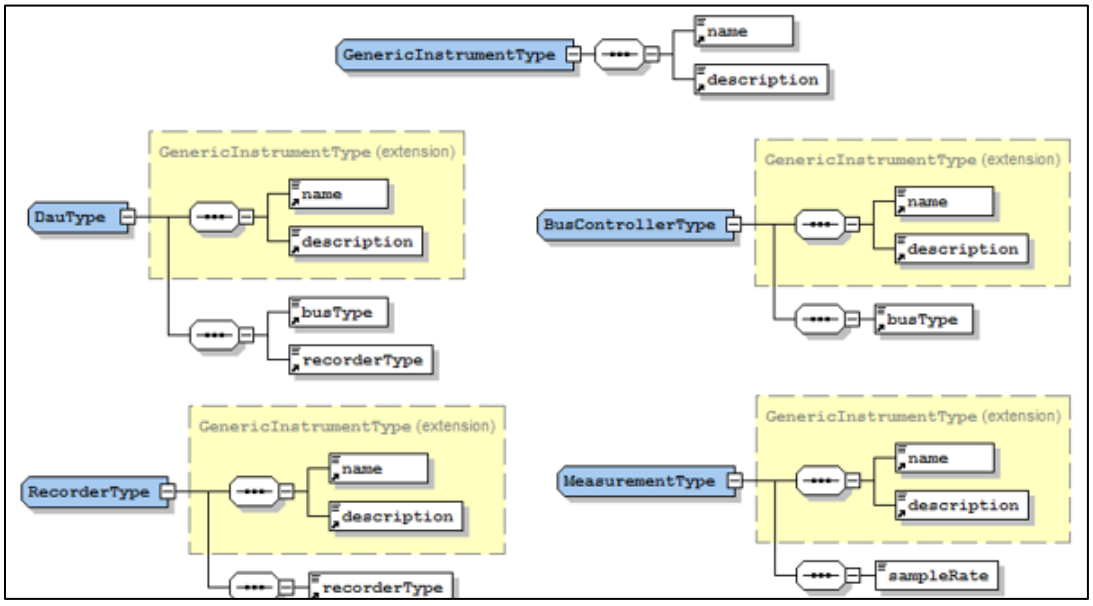

Figure 26. Example Schema with Modularity

Figure 27 shows the schema with all the modifications. This is a Venetian Blind design pattern since some elements are local (sample rate element of the air speed measurement) and some are global (bus type and recorder type elements of the DAU). If all elements were global,

the schema would be an instance of the Salami Slice design pattern. The Venetian Blind design pattern should be used since it may not be necessary to make all elements global; for example, since the sample rate is only used in the air speed measurement it is not necessary to make this global (it is not shared by multiple elements).

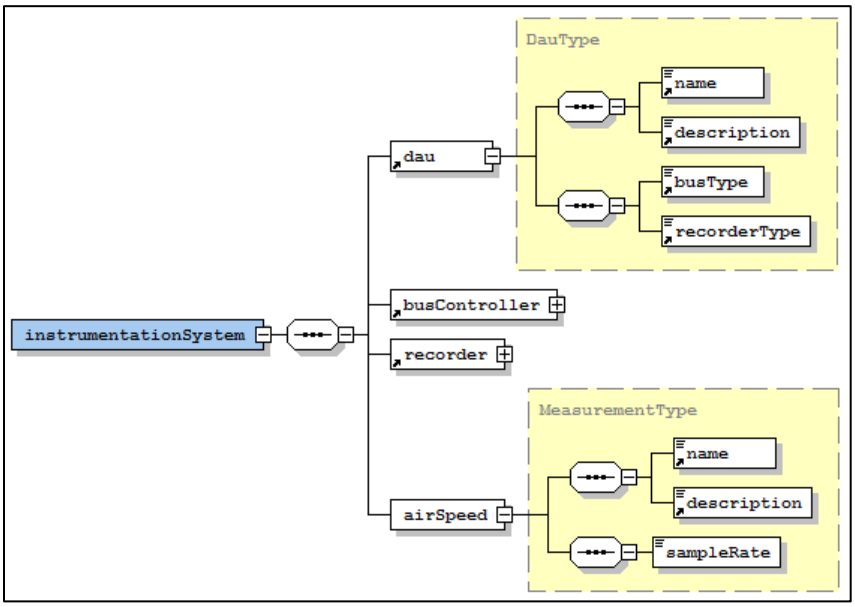

Figure 27. Venetian Blind Design Pattern

The modularity of our schema so far does not allow us to use these type definitions in other schemas. We would like to be able to use the GenericInstrumentType definition in multiple schemas. To use the structures in a schema (the *imported schema*) in some other schema (the *importing schema*), we must do the following.

1. Uniquely identify the imported schema by assigning a namespace to that schema; since namespaces can be long, we associate the schema with a namespace prefix.
2. Disambiguate the imported and importing schema elements by qualifying the schema attributes, elements, and types; this is done using the namespace prefixes.
3. Import the imported schema into the importing schema and use the structures of the imported schema in the importing schema.

To illustrate this, we will separate out the GenericInstrumentType definition into its own schema and use it in a new card schema. Figure 28 shows the GenericInstrumentType schema. The type definition is moved into its own schema file and assigned the namespace http://www.rcc.org/generic in the XML schema header (attribute targetNamespace) and assigned a prefix "generic" (attribute xmlns:generic). In the schema definition, note that the name and description elements are associated with the generic namespace to remove the possibility of any ambiguity.

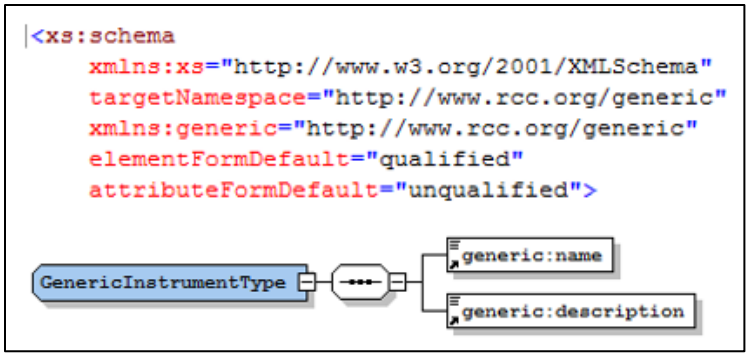

Figure 28. Generic Instrument Type Schema

A card schema can now be created that imports the GenericInstrumentType schema and defines card types based on the GenericInstrumentType definition. Figure 29 shows the signal conditioning card schema. A new schema file is created and assigned the namespace http://www.rcc.org/card in the XML schema header (attribute targetNamespace) and assigned a prefix "card" (attribute xmlns:card). The XML schema header also includes the definition of the generic namespace (attribute xmlns:generic). The type for a signal conditioning card extends the generic instrument type by adding an amplification element. The generic instrument schema is imported using the import statement with the location and namespace of the imported schema. Note that the generic instrument type name and description are associated with the "generic" namespace prefix and the amplification element is associated with the "card" prefix to remove any ambiguity.

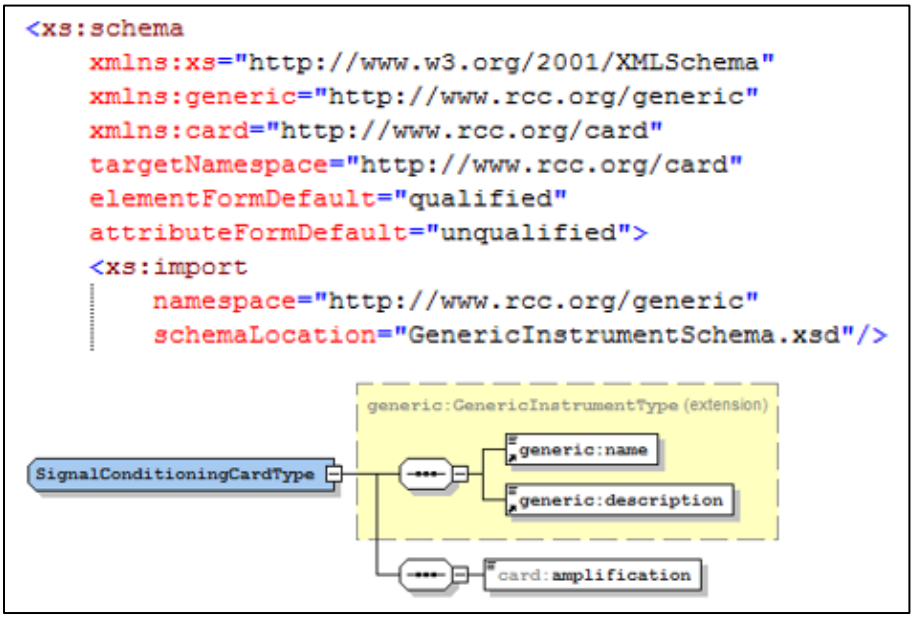

Figure 29. Signal Conditioning Card Type Schema

This page intentionally left blank.

APPENDIX A

Citations

The Apache Software Foundation. "Welcome to XMLBeans." The Apache XML Project. Last published 15 August 2012. Retired 23 May 2014. http://xmlbeans.apache.org/.

Costello, et al. "Global versus Local", in *XML Schemas: Best Practices*. 1 November 2006. Retrieved 22 July 2015. Available at http://www.xfront.com/GlobalVersusLocal.html.

Dare Obasanjo. *W3C XML Schema Design Patterns: Avoiding Complexity*. January 2003. Retrieved 17 July 2015. Available at http://msdn.microsoft.com/en-us/library/aa468564.aspx#xmlscmavdcmplx_topic13.

Darr, Tim, John Hamilton, Ronald Fernandes, and Charles H. Jones. "Design Considerations for XML-Based T & E Standards." Paper presented during 47th Annual International Telemetering Conference, Las Vegas, NV. 24-27 October 2011.

David Stephenson. *XML Schema Best Practices*. December 2004. Retrieved 15 July 2015. Available at http://xml.coverpages.org/HP-StephensonSchemaBestPractices.pdf.

Gao, Shudi, C. M. Sperberg-McQueen, and Henry S. Thompson. *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. 5 April 2012. Retrieved 22 July 2015. Available at http://www.w3.org/TR/xmlschema11-1.

Google. *Google XML Document Format Style Guide*. 2008. Retrieved 21 July 2015. Available at https://google-styleguide.googlecode.com/svn/trunk/xmlstyle.html.

Hamilton, John, Ronald Fernandes, Paul Koola, and Charles H. Jones. "An overview of an instrumentation hardware abstraction language." Paper presented during 42nd Annual International Telemetering Conference, San Diego, CA. 23-26 October 2006.

Internet Engineering Task Force. *Date and Time on the Internet: Timestamps*. RFC 3339. July 2002. May be superseded by update. Retrieved 16 July 2015. Available at http://www.rfc-editor.org/rfc/pdfrfc/rfc3339.txt.pdf.

Microsoft Developer Network. *Capitalization Styles*. n.d. Retrieved 21 July 2015. Available at https://msdn.microsoft.com/en-us/library/x2dbyw72(v=vs.71).aspx.

Moore, Michael S., Jeremy C. Price, Andrew R. Cormier, and William A. Malatesta. "Metadata description language: The iNET metadata standard language." Paper presented during 45th Annual International Telemetering Conference, Las Vegas, NV. 26-29 October 2009.

National Institute of Standards and Technology. Units Markup Language home page. http://unitsml.nist.gov/. Accessed 14 July 2015.

"Project JAXB." Java.net. Last published 14 October 2014. https://jaxb.java.net/.

Range Commanders Council. "Telemetry Attributes Transfer Standard," in *Telemetry Standards*. IRIG 106-15. June 2015. May be superseded by update. Retrieved 22 July 2015. Available at http://www.wsmr.army.mil/RCCsite/Documents/106-15_Telemetry_Standards/Chapter9.pdf.

W3Schools. *XML Schema Element*. n.d. May be superseded by update. Retrieved 22 July 2015. Available at http://www.w3schools.com/schema/el_schema.asp.

World Wide Web Consortium. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 26 November 2008. May be superseded by update. Retrieved 22 July 2015. Available at http://www.w3.org/TR/REC-xml/.

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