CHAPTER 23
Metadata Configuration

List of Tables

This page intentionally left blank.

Acronyms

HTML	Hypertext Markup Language
MDL	Metadata Description Language
PCM	pulse code modulation
TMATS	Telemetry Attributes Transfer Standard
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XSD	XML schema document

This page intentionally left blank.

CHAPTER 23

Metadata Configuration

23.1 Introduction

This chapter describes system configuration data for network-based telemetry systems in a common fashion, and provides a language for describing the configuration of all of the components in a telemetry system, as well as their logical and physical interrelationships. The language is intended to be expressive enough to describe a wide variety of systems: large and small, simple and complex, from the low-level transducer-to-measurement association for an acquisition card on a data acquisition unit up to network topology of multiple test mission networks.

This chapter defines the Metadata Description Language (MDL), which has a syntax that defines vocabulary and sentence structure, while the MDL semantics provide meaning. Using the MDL syntax and semantics, MDL instance documents can be created to describe telemetry systems. The descriptions may be used in various ways: to configure components, to predict the performance of the network, or to capture requirements for the telemetry system. Additionally, the MDL provides a common exchange language that facilitates the interchange of configuration information between telemetry system components.

23.2 Metadata Description Language

The MDL is described only as an eXtensible Markup Language (XML) schema. The MDL XML language schema consists of an XML schema document (XSD) file that defines the structure of valid MDL instance documents. Additionally, an automatic transformation process is applied to the schema in order to create a graphical depiction of the schema in Hyper Text Markup Language (HTML) format. The schema and HTML depiction are available here.

NOTE	The MDL XML-based schema is not intended to be read in a plain text fashion. The HTML graphical depiction is provided as an aid for those desiring to read the schema.

23.2.1 MDL Schema Concepts

The MDL schema defines a syntax, which includes a vocabulary, a set of types, and the rules for how an MDL instance document shall be structured. The syntax definition is realized using the XML Schema standard, which is maintained by the World Wide Web Consortium (W3C -- www.w3.org).

NOTE	The MDL builds upon the existing schema standard of the W3C. Readers unfamiliar with W3C schemas as a whole are encouraged to study basic concepts from the W3C prior to attempting to understand the MDL.

Constraints as defined by the W3C are a part of the MDL schema. In the MDL schema implementation, these constraints are distributed between statements that are syntax-related (encoded and enforced by the schema) and statements that are semantic-related (additional rules that are levied and provide meaning). The syntax of the language is enforced using W3C XML Schema constraints.

When possible, XML mechanisms are used to enforce semantic constraints. In cases not supported cleanly by XML, text has been added directly to the MDL schema documentation. In such cases, the text will typically include the keyword “shall”. The phrase “the value of the **`Name`** element of the **`Measurement`** element shall be unique” is one such example.

NOTE	Not all constraints that must be met in order for an MDL file to be valid for configuration can be expressed by W3C constraints. Plain text is used in the schema to describe such cases. Additionally, applications consuming or generating MDL instance documents are expected to assure that the files are valid.

23.2.1.1 Conditional elements in the MDL Schema Definition File

The MDL schema is a system-level description. Not all components require every element to properly configure. In such cases, these elements are conditional. The documentation specifies when the conditional elements shall be present and processed. Components not specifically called out in documentation of conditional elements shall not fail to configure if the particular conditional element is not present.

NOTE	Use of “conditional” over “optional” is an intentional style chosen for the MDL grammar. Conditional is preferred in order to remove ambiguity concerning grammar elements that must be considered by devices exchanging MDL files that are to be used for configuration.

23.2.1.2 Naming conventions in the MDL Schema Definition File

In the MDL schema definition file, MDL elements and attributes appear as instances of defined **`xsd:complexType`** and **`xsd:simpleType`** elements. Each declaration of these MDL-specific elements will specify a **`name`** attribute that is assigned a string that contains the name of the MDL element being described followed by a string suffix of “Type” or “Enum”. For example, the top-level element of the MDL schema is the **`MDLRoot`** element. In the MDL schema definition file, this element appears as an instance of an **`xsd:complexType`** element with a **`name`** attribute of “MDLRootType”. These **`name`** attribute strings that correspond to the defined MDL elements only appear in the MDL schema definition file.

23.2.1.3 Indexing policies

Numerical indices present in an MDL instance document are recommended to count starting at 1 and to increment by one (i.e., 1, 2, 3, 4,…).

23.2.1.4 Uniqueness of ID attributes

Values of **`ID`** attributes of any element within an MDL instance document shall be unique. The **`ID`** attributes are used to implement references.

23.2.1.5 Extending MDL with supplementary XML-based standards

The use of other XML-based standards (i.e., other schemas) in conjunction with the MDL schema is permitted. Potentially, the use of these external standards through their accompanying schemas leverages existing knowledge to describe concepts and domains beyond those in the MDL. The MDL does not explicitly constrain the available mechanisms to use external standards; however, the linking of external schemas to the MDL schema shall not result in the modification of the MDL schema.

23.2.1.6 Usage of Telemetry Attributes Transfer Standard in MDL

The MDL schema requires the **`tmatsP:PCMFormatAttributesType`** to describe pulse code modulation (PCM) measurements, and it is imported directly from the Telemetry Attributes Transfer Standard (TMATS) schema. The TMATS schema files are included with the MDL schema for convenience, and are also available in Chapter 9.

23.2.1.7 Usage of GenericParameter Element

The **`GenericParameter`** element allows the description of additional information outside the scope of the MDL, and may also be used to document decisions made to arrive at a vendor-specific configuration. **`GenericParameter`** shall be used to extend the MDL grammar when it cannot support those required concepts natively, or as a key so that vendor tools can arrive at the same configuration as in a previous run.

23.2.1.8 Recommended Best Practices

Appendix 23-C contains a table of recommended best practices to further enhance interoperability.

23.2.2 MDL Global Element Glossary

The MDL schema contains a large number of **`xsd:documentation`** elements that describe the intent, purpose, or usage of the elements in MDL. These embedded annotations collectively form the global element glossary for the MDL schema. The glossary can be viewed here (the MDL Schema Documentation.html file located in the compressed folder this link opens). It is automatically produced from the schema file by way of an eXtensible Stylesheet Language stylesheet, which renders the documentation as HTML.

This page intentionally left blank.

APPENDIX 23-A

MDL Examples

Example MDL files (XML instance documents) and associated documentation are here. As with most grammars, it is expected that the examples will be very useful in clarifying the expected use of MDL; however, the documentation of the schema takes precedence over concepts or details that may be inferred through the examples.

This page intentionally left blank.

APPENDIX 23-B

MDL Relationships to Chapters

The MDL is used to describe system configuration data for network-based telemetry systems; therefore, it includes elements to describe the concepts presented across Chapters 22 through 28. Table B-1 provides a mapping of the relevant top-level MDL elements and their relationships to these chapters.

Table B-1. MDL Mapping to RCC 106 Chapters

MDLRoot Element	Relationship	RCC 106 Chapter
TestMissions	Data Protocol	22: Network-Based Protocol Suite
	Management	26: TmNSDataMessage Transfer Protocol
MeasurementDomains	Message Internals	24: Message Formats
NetworkDomains	Network Shape	22: Network-Based Protocol Suite
	Network Use	25: Management Resources
	Network Management	26: TmNSDataMessage Transfer Protocol
RANConfigs	Radio Links	27: RF Network Access Layer
	Radio Management	28: RF Network Management
DSCPTable	Quality of Service	22: Network-Based Protocol Suite

Additionally, the spreadsheet inside the compressed folder linked here provides a detailed mapping of all MDL elements to Chapter 22 through 28.

This page intentionally left blank.

APPENDIX 23-C

MDL Recommended Best Practices

Table C-1 provides recommended best practices for creating MDL instance documents that will enhance interoperability.

Table C-1. MDL Recommended Best Practices

Scenario	MDL Syntax/Semantics	Related Scenarios
1. Measurement Name Scoping for External Usage	Any of the following can be used to identify a measurement. External tool representation examples: • Multiple Test Article (TA), Multiple Transport Case o TA/TRANSPORT/MEASUREMENT (e.g., Weapon1/PCM1/Airspeed) • Single TA, Multiple Transport Case o */TRANSPORT/MEASUREMENT (e.g., */PCM1/Airspeed) • Multiple TA, Single Transport Case o TA/*/MEASURMENT (e.g., Weapon1/*/Airspeed) • Single TA, Single Transport Case o */*/MEASUREMENT (e.g., */*/Airspeed) The recommended delimiter is the forward slash (/) character. When fields are not required to uniquely identify a measurement, the recommended wildcard is the asterisk (*) character. Measurement names should avoid embedded blanks or special characters other than _ and -.	None

2. Choosing the Correct DataProcessing Function Type	If a **`DataProcessing Function`** can be represented by a **`LookupTable`** element or a **`Polynomial`** element then these **`Function`** types should be used instead of the generic **`Algorithm`** element.
	Polynomial example:
	`<Function>`
	    `<Name>Polynomial Example</Name>`
	    `<Description>(5x**2 + 6x + 7) / (2.5x**3 + 3x*1.64)</Description>`
	    `<InputCount>1</InputCount>`
	    `<UpdateFrequency>IfAny</UpdateFrequency>`
	    `<Polynomial>`
	        `<Numerator>`
	            `<Term>`
	                `<Coefficient>5</Coefficient>`
	                `<Exponent>2</Exponent>`
	            `</Term>`
	            `<Term>`
	                `<Coefficient>6</Coefficient>`
		3. Specifying Points in a LookupTable

Table C-1. MDL Recommended Best Practices

Scenario	MDL Syntax/Semantics	Related Scenarios

	`<Exponent>1</Exponent>`
	`</Term>`
	`<Term>`
	`<Coefficient>7</Coefficient>`
	`<Exponent>0</Exponent>`
	`</Term>`
	`</Numerator>`
	`<Denominator>`
	`<Term>`
	`<Coefficient>2.5</Coefficient>`
	`<Exponent>3</Exponent>`
	`</Term>`
	`<Term>`
	`<Coefficient>3</Coefficient>`
	`<Exponent>1.64</Exponent>`
	`</Term>`
	`</Denominator>`
	`</Polynomial>`
	`</Function>`	

3. Specifying Points in a LookupTable	When defining a **`LookupTable`** in a **`DataProcessing Function`** the complete set of points should be provided, including +/-Inf. If all points are not provided, then the **`LookupTable`** output is undefined for those input points.
	LookupTable example:
	`<Function>`
	`<Name>LookupTable Example</Name>`
	`<Description>Example showing a lookup table</Description>`
	`<InputCount>1</InputCount>`
	`<UpdateFrequency>IfAny</UpdateFrequency>`
	`<LookupTable>`
	`<LookupTableEntry>`
	
	`<InputValue>NegativeInfinity</InputValue>`
	`<OutputValue>0.0</OutputValue>`
	`</LookupTableEntry>`
	`<LookupTableEntry>`
	`<InputValue>0</InputValue>`
	`<OutputValue>0.0</OutputValue>`
	`</LookupTableEntry>`
	`<LookupTableEntry>`
	`<InputValue>1</InputValue>`
	`<OutputValue>1.0</OutputValue>`
	`</LookupTableEntry>`
	`<LookupTableEntry>`
	`<InputValue>2</InputValue>`
		None

Table C-1. MDL Recommended Best Practices

Scenario	MDL Syntax/Semantics	Related Scenarios

	`<OutputValue>5.0</OutputValue>`
	`</LookupTableEntry>`
	`<LookupTableEntry>`
	
	`<InputValue>PositiveInfinity</InputValue>`
	`<OutputValue>6.0</OutputValue>`
	`</LookupTableEntry>`
	`</LookupTable>`
	`</Function>`	

4. Using MeasurementTimeRefs in Packages	Packages should only contain one instance of time (a single **`MeasurementTimeRef`**) per set of measurements taken at that time.
	MeasurementTimeRef example:
	`<DataMaps>`
	`<DataWordToFieldMap>`
	`<DataWord>`
	`<Name>Measurement 1</Name>`
	`<Description>Simultaneously sampled measurement</Description>`
	`<DataWordWidth>`
	`<Value>16</Value>`
	`<BaseUnit>Bit</BaseUnit>`
	`</DataWordWidth>`
	`<MeasurementRef IDREF="Meas1"/>`
	`<Syllables>`
	`<Syllable>`
	`<Name>Measurement 1 Syllable</Name>`
	`</Syllable>`
	`</Syllables>`
	`</DataWord>`
	`<DataStructureFieldRef IDREF="Package1_Field1"/>`
	
	`<TimeOrder>IncreasingTemporal</TimeOrder>`
	`</DataWordToFieldMap>`
	`<DataWordToFieldMap>`
	`<DataWord>`
	`<Name>Measurement 2</Name>`
	`<Description>Simultaneously sampled measurement</Description>`
	`<DataWordWidth>`
	`<Value>16</Value>`
	`<BaseUnit>Bit</BaseUnit>`
	`</DataWordWidth>`
	`<MeasurementRef IDREF="Meas2"/>`
		None

Table C-1. MDL Recommended Best Practices

Scenario	MDL Syntax/Semantics	Related Scenarios

```
            <Syllables>
                <Syllable>
                    <Name>Measurement 2
Syllable</Name>
                </Syllable>
            </Syllables>
        </DataWord>
        <DataStructureFieldRef
IDREF="Package1_Field2"/>

<TimeOrder>IncreasingTemporal</TimeOrder>
    </DataWordToFieldMap>
    <DataWordToFieldMap>
        <DataWord>
            <Name>Time Measurement</Name>
            <Description>Time that the
simultaneously sampled measurements were
taken</Description>
            <DataWordWidth>
                <Value>16</Value>
                <BaseUnit>Bit</BaseUnit>
            </DataWordWidth>
            <MeasurementRef IDREF="Time"/>
            <Syllables>
                <Syllable>
                    <Name>Time Measurement
Syllable</Name>
                </Syllable>
            </Syllables>
        </DataWord>
        <DataStructureFieldRef
IDREF="Package1_Field3"/>

<TimeOrder>IncreasingTemporal</TimeOrder>
    </DataWordToFieldMap>
</DataMaps>
```

Note: If repeating fields are required, then packages without headers can be used to create the same resulting output as one defined by repeating fields. To accomplish this, the first package would use standard package headers and contain a measurement value and **`MeasurementTimeRef`**. Subsequent repeating packages would not use a header and contain the repeated measurement values each with its own **`MeasurementTimeRef`**.