THE MITRE CORPORATION

THE MAEC™ LANGUAGE VERSION 4.1

MAEC CONTAINER VERSION 2.1 SPECIFICATION

DESIREE BECK, IVAN KIRILLOV, PENNY CHASE, MITRE
JUNE 12, 2014

Malware Attribute Enumeration and Characterization (MAEC™) is a standardized language for sharing structured information about malware based upon attributes such as behaviors, artifacts, and attack patterns.

By eliminating the ambiguity and inaccuracy that currently exists in malware descriptions and by reducing reliance on signatures, MAEC aims to improve human-to-human, human-to-tool, tool-to-tool, and tool-to-human communication about malware; reduce potential duplication of malware analysis efforts by researchers; and allow for the faster development of countermeasures by enabling the ability to leverage responses to previously observed malware instances.

Acknowledgements

The authors would like to thank the MAEC Community for its input and help in reviewing this document.

Trademark Information

Warnings

MITRE PROVIDES MAEC "AS IS" AND MAKES NO WARRANTY, EXPRESS OR IMPLIED, AS TO THE ACCURACY, CAPABILITY, EFFICIENCY, MERCHANTABILITY, OR FUNCTIONING OF MAEC. IN NO EVENT WILL MITRE BE LIABLE FOR ANY GENERAL, CONSEQUENTIAL, INDIRECT, INCIDENTAL, EXEMPLARY, OR SPECIAL DAMAGES, RELATED TO MAEC OR ANY DERIVATIVE THEREOF, WHETHER SUCH CLAIM IS BASED ON WARRANTY, CONTRACT, OR TORT, EVEN IF MITRE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.[1]

Feedback

The MAEC development team welcomes any feedback regarding the MAEC Language Container Specification. Please send any comments, questions, or suggestions maec@mitre.org.[2]

[1] For detailed information see [TOU].

[2] For more information about the MAEC Language, please visit [MAEC].

Table of Contents

1 Overview

The Malware Attribute Enumeration and Characterization (MAEC) Language is defined by three data models and a set of default controlled vocabularies[3]. As illustrated in Figure 1-1, "MAEC Bundle" is the (lowest) Tier 1 data model; "MAEC Package" is the (middle) Tier 2 data model; and "MAEC Container" is the (highest) Tier 3 data model. All three data models offer a stand-alone output format, so a lower level model can be used without the higher tier data model (although each model level encompasses and makes use of all lower tiers).

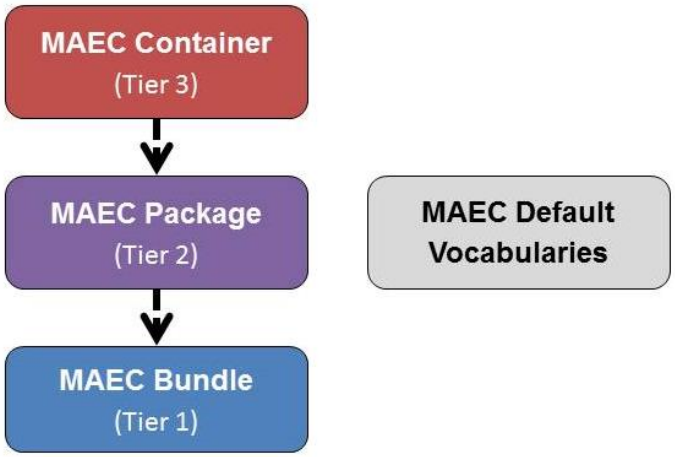


Figure 1-1. MAEC data models

A complete discussion of the structure of the MAEC language can be found in the MAEC Overview [$MAEC_O$]. In brief:

- MAEC Bundle – provides the ability to capture and share data obtained from the analysis of a single malware instance. Its underlying structure is formed by Actions, Behaviors, and Capabilities.
- MAEC Package – enables a user to capture and share MAEC characterized data for one or more Malware Subjects; in most such cases, the Malware Subjects are related. A Malware Subject is MAEC's representation of a malware instance and all of the known data associated with it, including data derived from analysis and metadata.
- MAEC Container – enables a user to share any collection of MAEC characterized data, including one or more Packages.

[3] Each data model and the default vocabularies are implemented by an XML schema. Other output formats, such as JSON, are being considered for future implementations.

This document serves as the specification for the MAEC Container data model. Before we present the Container data model in Section 2, we provide relevant background information in Subsections 1.1 through 1.6.

1.1 Additional Documents and Information

Numerous overview, specification, and supporting documents are available for the MAEC Language. All documents are shown in Figure 1-2. Icons are used to indicate whether the material is contained in an actual document ( ) or captured on a Web page ( ). This document is highlighted in yellow.

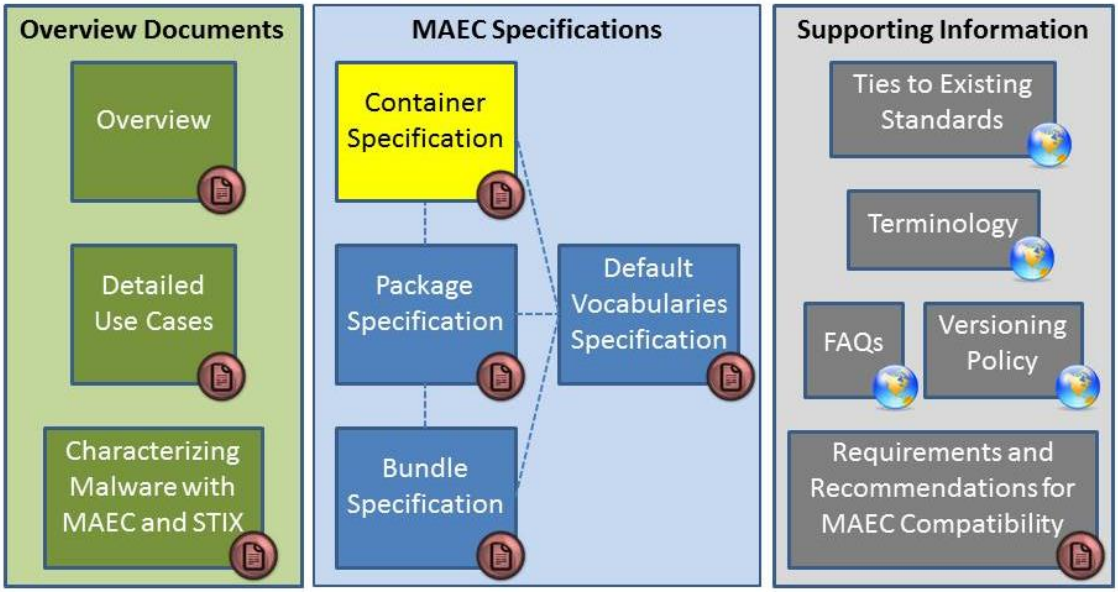


Figure 1-2. MAEC Language v4.1 documents

All documents can be found on the MAEC Website [MAEC], and a summary and link to each is provided below:

- Overview: Introduces and motivates MAEC, provides an overview of the MAEC language, and presents a collection of high level use cases [$MAEC_O$].
- Detailed Use Cases: Provides explicit examples to illustrate how MAEC can be used to capture malware information stemming from various forms of malware analysis [$EXAM_D$].
- Characterizing Malware with MAEC and STIX: Describes the use of MAEC and STIX in the context of malware characterization and malware metadata exchange [$MAEC_S$].
- Container Specification: Specification for the MAEC Container data model [$SPEC_C$]. (This document.)
- Package Specification: Specification for the MAEC Package data model [$SPEC_P$].

- Bundle Specification: Specification for the MAEC Bundle data model [$MAEC_B$].
- Default Vocabulary Specification: Specification for the MAEC Default Vocabularies [$SPEC_V$].
- Ties to Existing Standards: Provides an overview of how MAEC is related to MMDEF, CybOX, CPE, CVE, and STIX [TIES].
- Terminology: Contains terms associated with malware and malware analysis, as well as terminology that is specific to MAEC [TERM].
- FAQs: Frequently asked questions about MAEC including questions about the language, use, relationships to other efforts, and the MAEC community [FAQ].
- Versioning Policy: Details the current methodology for determining whether a revision will require a major version change, a minor version change, or an update version change. Note that the MAEC schemas and default vocabularies are versioned independently of the MAEC Language, and their version numbers may or may not coincide with each other or with that of the MAEC Language [VER].
- Requirements and Recommendations for MAEC Compatibility: Specifies requirements for MAEC-compatible tools, services, and repositories [REQ].

1.2 Data Model Conventions

The following information and conventions are used to define the MAEC data models, and may or may not apply to the particular MAEC data model documented in Section 2.

1.2.1 Data Model Fields and Types

In Section 2, we define the types associated with the MAEC Container data model fields. It is important to understand that "fields" correspond to the malware-related properties captured in a MAEC document and "types" are used to define and express the underlying data model used in the fields.

1.2.2 XML Attributes and Elements

Our methodology for representing a field as either an attribute or an element in the XML implementation[4] is based primarily on the determination of the complexity of the field. Generally, simple fields such as identifiers, data types, and timestamps are represented as attributes. Complex fields, for example, those that have multiplicity greater than one (such as lists), are represented as elements. However, in this specification we have attempted, as much as possible, to abstract away these XML-specific implementation details to provide a more general view of the MAEC Container data model.

[4] Each data model and the default vocabularies are implemented in MAEC v4.1 via an XML schema.

1.2.3 Non-MAEC Data Models

MAEC draws several components from the CybOX Language (see [$MAEC_O$]); consequently, the reader is referred to [CYBOX] for the definitions of these entities. In this specification, we do not define any types that are part of a non-MAEC data model. Instead we make note of the referenced data model's specification and explicitly define only the extensions (i.e., new fields and types) that have been made as an extension of the base type.

1.2.4 Primitive Data Types

The following primitive datatypes are used in the MAEC Language.

- binary – Data of this type conforms to the World Wide Web Consortium (W3C) Recommendation for hex-encoded binary data [$W3C_1$].
- boolean – Data of this type conforms to the W3C Recommendation for boolean data [$W3C_2$].
- double – Data of this type conforms to the W3C Recommendation for double data [$W3C_3$].
- float – Data of this type conforms to the W3C Recommendation for float data [$W3C_4$].
- int – Data of this type conforms to the W3C Recommendation for integer data [$W3C_5$].
- QName – Data of this type conforms to the W3C Recommendation for an XML namespace-qualified name [$W3C_6$].
- string – Data of this type conforms to the W3C Recommendation for string data [$W3C_7$].
- unsigned int – Data of this type conforms to the W3C Recommendation for unsigned int data [$W3C_8$].
- URI – Data of this type conforms to the W3C Recommendation for anyURI data [$W3C_9$].
- dateTime – Data of this type represents a time value that conforms to the yyyy-mm-ddThh:mm:ss format.

1.3 Controlled Vocabularies

Some of the fields defined in the MAEC schemas are of type `cyboxCommon:ControlledVocabularyStringType`. A field of this type is implemented through the `xsi:type` XML abstract type extension mechanism. The default vocabulary applicable to the particular type will be provided in the "Description" column of the property table. Default vocabularies are defined in the maec_default_vocabularies.xsd file available at [REL_D]. Please see the MAEC Default Vocabularies Specification document [$SPEC_V$] for more information.

1.4 ID Formats

In MAEC v4.1, all MAEC IDs are captured and formatted as XML QNames[5]. Each such ID includes both a namespace portion (optional) and an ID portion (required), separated by a colon (":"). The recommended approach to creating a MAEC ID is to define a producer namespace and namespace prefix and then use the form:

```
[ns prefix]:[construct type]-[GUID]
```

The "ns prefix" SHOULD be a namespace prefix bound to a namespace owned/controlled by the producer of the content. For consistency across MAEC documents, the "construct type" SHOULD correspond to the labels provided in Table 1-1 below (datatypes are defined in MAEC v4.1 unless otherwise indicated). Finally, the "GUID" SHOULD correspond to a globally unique ID. For example, a MAEC Bundle could have the following ID:

```
somecompany:bundle-2f44522e-8164-4050-8e13-e01f9a
```

In order to use this approach, the namespace and prefix MUST be defined in the head of the XML document, e.g.,
`xmlns:somecompany="http://company.example.com"`.

This format provides high assurance that IDs will be both meaningful and unique. Meaning comes from the producer namespace, which denotes who is producing it, as well as the construct type, which denotes to what the ID pertains. Uniqueness is achieved when the meaningful portion is combined with a globally unique ID.

[5] In MAEC v4.1, restrictions on ID syntax have been lifted in all IDs used in MAEC types so that all MAEC IDs are now compatible with the implementations used in CybOX and STIX. Consequently, the additional schematron and XSL files used in earlier MAEC versions primarily for ID syntax validation have been deprecated.

Table 1-1. Recommended construct type labels

Construct Name	Datatype (defining ID)	Construct Type (in ID)
BUNDLE IDs and IDREFs		
action_collection	ActionCollectionType	action_collection
action_implementation	ActionImplementationType	action_implementation
action_equivalence_reference	BehavioralAction EquivalenceReferenceType	action_equivalence
action	cybox:ActionType	action
behavior	BehaviorType	behavior
behavior_collection	BehaviorCollectionType	behavior_collection
maec_bundle	BundleType	bundle
candidate_indicator_collection	CandidateIndicatorCollectionType	candidate_indicator_collection
candidate_indicator	CandidateIndicatorType	candidate_indicator
capability	CapabilityType	capability
malware_instance_object_attributes	cybox:ObjectType	object
strategic_objective	CapabilityObjectiveType	objective
tactical_objective	CapabilityObjectiveType	objective
object_collection	ObjectCollectionType	object_collection
process_tree_node	ProcessTreeNodeType	process_tree
object	cybox:ObjectType	object
PACKAGE IDs and IDREFs		
action_equivalence	ActionEquivalenceType	action_equivalence
analysis	AnalysisType	analysis
malware_subject	MalwareSubjectType	malware_subject
object_equivalence	ObjectEquivalenceType	object_equivalence
maec_package	PackageType	package
malware_instance_object_attributes	cybox:ObjectType	object
CONTAINER IDs		
maec_container	ContainerType	container

1.5 XML Implementation

The XML implementation of the MAEC Language data model is documented in a series of XML Schemas.[6] These schemas describe how the information presented in this Specification is formatted and represented as XML. Please refer to the appropriate Schema for more information about a specific XML implementation.

MAEC Container Model
https://maec.mitre.org/language/version4.1/maec-container-schema.xsd

MAEC Package Model
https://maec.mitre.org/language/version4.1/maec-package-schema.xsd

MAEC Bundle Model
https://maec.mitre.org/language/version4.1/maec-bundle-schema.xsd

MAEC Default Vocabularies
https://maec.mitre.org/language/version4.1/maec-default-vocabularies.xsd

The complete listing of XML representation resources can be found on the MAEC website [REL4].

1.6 Document Conventions

The following conventions are used in this document.

1.6.1 Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in *RFC 2119* [RFC2119].

1.6.2 Fonts

The following font and font style conventions are used in the document:

- Capitalization is used for MAEC high level concepts, which are defined as basic components in the MAEC Overview document [$MAEC_O$] (see Section 2 in [$MAEC_O$]).

  Examples: Bundle, Strategic Objective, Malware Subject

[6] XML Schema Part 0: Primer Second Edition http://www.w3.org/TR/xmlschema-0

- The `Courier New` font is used for writing constructs in the MAEC Language Data Model (and related data models).

  Examples: `CandidateIndicatorType`, `Malware_Subject`

  Note that all high level concepts have a corresponding data model construct (e.g., Malware Subject → `Malware_Subject`).

- The '*italic, with single quotes*' font is used for noting values for MAEC Language properties.

  Examples: '*2.1*', '*MAEC Default Device Driver Action Names*'

1.6.3 Namespaces

This document uses the concept of namespaces[7] to logically group MAEC constructs throughout the Data Model section of the document, as well as other parts of the specification. The format of these namespaces is `prefix:namespace`, where the prefix is the namespace component, and the namespace is the actual namespace URI. Table 1-2 on page 10 provides a listing of the default namespaces used in MAEC to help provide context as to the particular source data model or vocabulary used in a field. Table 1-2 also lists the relevant version of each of the data models. These namespaces are compatible with XML Namespaces [$W3C_0$], though the MAEC language is not restricted to XML serialization.

1.6.4 UML Diagrams

The Data Model makes use of Unified Modeling Language (UML) diagrams where appropriate, to visually depict relationships for the MAEC Language constructs. Diagrams are included for any construct that inherits from other constructs or has a compositional relationship.

1.6.5 Property Table Notation

Throughout the data model, tables are used to describe each data type and its properties. Each property table will consist of a column of field names to identify the property, a type column to reflect the datatype of the property, a multiplicity column to reflect the allowed number of occurrences of the property, and a description column that will describe the property. In addition:

- Fields that are part of a "choice" relationship (e.g., Field1 OR Field2 is used but not both) will be denoted by a unique letter subscript (e.g., API_Call_A, $Code_B$) and single logic expression in the Multiplicity column. For example, if there is a choice of field

[7] Namespaces (computer science): http://en.wikipedia.org/wiki/Namespace_(computer_science)

`API_Call`$_A$ and `Code`$_B$, the expression "A(1)|B(0..1)" will indicate that the `API_Call` field can be chosen with multiplicity 1 or the `Code` property can be chosen with multiplicity 0..1.

Values in the type column are either primitive datatypes or other types defined in this document. These values will be cross referenced to the base definition of their types.

Table 1-2. Namespace prefixes used by MAEC

Data Model / Vocab	Namespace Prefix	Description	Example
MAEC Bundle v4.1	maecBundle	The MAEC Bundle data model captures the constructs used in a MAEC Bundle.	`maecBundle:ActionType`
MAEC Package v2.1	maecPackage	The MAEC Package data model captures the constructs used in a MAEC Package.	`maecPackage:MalwareSubjectType`
MAEC Container v2.1	maecContainer	The MAEC Container data model captures all MAEC characterized data.	`maecContainer:PackageListType`
MAEC Default Vocabularies v1.1	maecVocabs	The MAEC default vocabularies define types for default controlled vocabularies used within MAEC.	`maecVocabs:FileActionNameVocab`
Malware Metadata Exchange Format (MMDEF) v1.2	metadata	The MMDEF data model captures some constructs used in exchanging malware sample data.	`metadata:fieldDataEntry`
CybOX Core v2.1	cybox	The CybOX core data model captures all the core constructs used in CybOX.	`cybox:ObjectType`
CybOX Common v2.1	cyboxCommon	The CybOX common data model captures common constructs used across CybOX objects and other types.	`cyboxCommon:MeasureSourceType`
CybOX Default Vocabularies v2.1	cyboxVocabs	The CybOX default vocabularies define types for default controlled vocabularies used within CybOX.	`cyboxVocabs:HashNameVocab`
Code Object v2.1	CodeObj	The CybOX Code Object data model is intended to characterize a body of computer code.	`CodeObj:CodeObjectType`
System Object v2.1	SystemObj	The CybOX System Object data model is intended to characterize computer	`SystemObj:SystemObjectType`

		systems (as a combination of both software and hardware).	
Process Object v2.1	ProcessObj	The CybOX Process Object data model is intended to characterize system processes.	`ProcessObj:ProcessObjectType`

2 MAEC Container Data Model

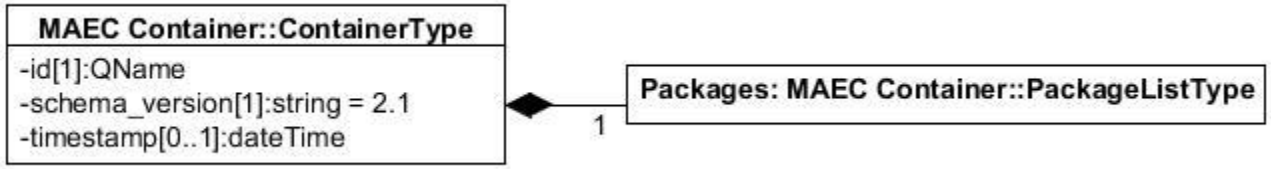
We describe the types presently used in the MAEC Container data model (the MAEC Container data model is not yet fully defined). The root field of the MAEC Container schema is the `MAEC_Container` field of type `ContainerType`. All types originate from the MAEC Container schema, unless otherwise noted with a schema prefix, e.g., 'cybox:' for the CybOX Core schema.

2.1 Container

The root field of the MAEC Container schema is the `MAEC_Container` field of type `ContainerType`. A `MAEC_Container` captures one or more `MAEC_Packages`.

2.1.1 ContainerType

The `ContainerType` encompasses all forms of MAEC data. Currently, this entails a list of `MAEC_Packages`.



Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for this `Container`. The ID SHOULD follow the pattern defined in Section 1.4.
schema_version	`string`	1	Specifies the version of the MAEC Container Schema that the document has been written in and that SHOULD be used for validation. The fixed value is '*2.1*.'
timestamp	`dateTime`	0..1	Specifies the date/time that the `MAEC_Container` was generated.

Packages	PackageListType	1	Captures a list of MAEC_Packages.

2.2 List Types

This section contains an alphabetical list of types that are lists of fields used in the MAEC Container data model.

2.2.1 PackageListType

The PackageListType captures a list of MAEC_Packages.

Field	Type	Multiplicity	Description
Package	maecPackage:PackageType	1..*	Specifies a single MAEC_Package, which encompasses 1-n Malware_Subjects and any associated metadata.

References

References made in this document are listed below.

A.1 MAEC Documents

[$MAEC_O$] MAEC Overview
http://maec.mitre.org/about/docs/MAEC_Overview.pdf

[$MAEC_S$] Characterizing Malware with MAEC and STIX
http://maec.mitre.org/about/docs/Characterizing_Malware_MAEC_and_STIX_v1.0.pdf

[$SPEC_B$] MAEC Bundle Specification
http://maec.mitre.org/language/version4.1/MAEC_Bundle_Spec_v4_1.pdf

[$SPEC_P$] MAEC Package Specification
http://maec.mitre.org/language/version4.1/MAEC_Package_Spec_v2_1.pdf

[$SPEC_C$] MAEC Container Specification
http://maec.mitre.org/language/version4.1/MAEC_Container_Spec_v2_1.pdf

[$SPEC_V$] MAEC Default Vocabularies Specification
http://maec.mitre.org/language/version4.1/MAEC_Vocabs_Spec_v1_1.pdf

[REQ] Requirements and Recommendations for MAEC Compatibiity
http://maec.mitre.org/compatible/Requirements_for_MAEC_Compatibility_V1.1.pdf

A.2 MAEC Web Pages

[$EXAM_W$] MAEC v4.1 Release Examples
http://maec.mitre.org/language/version4.1/#samples

[$EXAM_G$] MAEC Examples (GitHub repository)
https://github.com/MAECProject/schemas/tree/master/examples

[MAEC] MAEC Web Site
https://maec.mitre.org

[$MAEC_C$] MAEC Community
https://maec.mitre.org/community/index.html

[$MAEC_L$] MAEC Discussion List Signup
http://maec.mitre.org/community/discussionlist.html

[$MAEC_H$] MAEC Handshake (send email to maec@mitre.org for access)
https://handshake.mitre.org/

[REL4] MAEC v4.1 Release
https://maec.mitre.org/language/version4.1/

[TERM] MAEC Terminology
http://maec.mitre.org/about/terminology.html

[TIES] Ties to Existing Standards
http://maec.mitre.org/about/standards.html

[FAQ] MAEC FAQ
http://maec.mitre.org/about/faqs.html

[TOU] MAEC Terms of Use
https://maec.mitre.org/about/termsofuse.html

[VER] Versioning Policy
http://maec.mitre.org/language/versioning_policy.html

A.3 MAEC Schema

[REL_B] MAEC Bundle Model
https://maec.mitre.org/language/version4.1/maec_bundle_schema.xsd

[REL_P] MAEC Package Model
https://maec.mitre.org/language/version4.1/maec_package_schema.xsd

[REL_C] MAEC Container Model
https://maec.mitre.org/language/version4.1/maec_container_schema.xsd

[REL_D] MAEC Default Vocabularies
https://maec.mitre.org/language/version4.1/maec_default_vocabularies.xsd

A.4 MAEC Development

[DEV] MAEC GitHub Repositories
https://github.com/MAECProject/

[DEV_P] MAEC Python Library
https://github.com/MAECProject/python-maec

[DEV_S] MAEC Schema Development
https://github.com/MAECProject/schemas

[DEV_U] MAEC Utilities
https://github.com/MAECProject/utils

A.5 Other References

[CPE] Common Platform Enumeration (CPE)
http://nvd.nist.gov/cpe.cfm (Official CPE Dictionary)
http://csrc.nist.gov/publications/PubsNISTIRs.html (CPE Specifications)

[CUCKOO] Cuckoo Sandbox
http://www.cuckoosandbox.org/

[CVE] Common Vulnerabilities and Exposures (CVE)
http://cve.mitre.org

[CVSS] Common Vulnerability Scoring System
http://www.first.org/cvss

[CYBOX] Cyber Observable eXpression (CybOX)
http://cybox.mitre.org

[IOC] Open Indicators of Compromise (OpenIOC)
http://openioc.org/

[MMDEF] IEEE ICSG's Malware Metadata Exchange Format
http://standards.ieee.org/develop/indconn/icsg/mmdef.html

[OVAL] Open Vulnerability and Assessment Language (OVAL)
http://oval.mitre.org

[RFC2119] RFC 2119 – Key words for use in RFCs to Indicate Requirement Levels
http://www.ietf.org/rfc/rfc2119.txt

[STIX] Structured Threat Information eXpression (STIX)
http://stix.mitre.org

[$W3C_0$] W3C Namespaces in XML 1.0 (Third Edition)
http://www.w3.org/TR/REC-xml-names/

[$W3C_1$] W3C Recommendation for Hex-Encoded Binary Data
http://www.w3.org/TR/xmlSchema-2/#hexBinary

[$W3C_2$] W3C Recommendation for Boolean Data
http://www.w3.org/TR/xmlSchema-2/#boolean

[$W3C_3$] W3C Recommendation for Double Data
http://www.w3.org/TR/xmlschema-2/#double

[$W3C_4$] W3C Recommendation for Float Data
http://www.w3.org/TR/xmlSchema-2/#float

[$W3C_5$] W3C Recommendation for Integer Data
http://www.w3.org/TR/xmlSchema-2/#integer

[$W3C_6$] W3C Recommendation for XML Qualified Names
http://www.w3.org/TR/xmlSchema-2/#QName

[$W3C_7$] W3C Recommendation for String Data
http://www.w3.org/TR/xmlSchema-2/#string

[$W3C_8$] W3C Recommendation for unsigned int Data
http://www.w3.org/TR/xmlschema-2/#unsignedInt

[$W3C_9$] W3C Recommendation for URI Data
http://www.w3.org/TR/xmlschema-2/#anyURI