THE MITRE CORPORATION

THE MAEC™ LANGUAGE VERSION 4.1 SPECIFICATION

MAEC PACKAGE VERSION 2.1

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.

Feedback
The MAEC development team welcomes any feedback regarding the MAEC Language Package 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 1
1.1 Additional Documents and Information 2
1.2 Data Model Conventions 3
1.2.1 Data Model Fields and Types 3
1.2.2 XML Attributes and Elements 3
1.2.3 Non-MAEC Data Models 4
1.2.4 Primitive Data Types 4
1.3 Controlled Vocabularies 4
1.4 ID Formats 5
1.5 XML Implementation 7
1.6 Document Conventions 7
1.6.1 Key Words 7
1.6.2 Fonts 7
1.6.3 Namespaces 8
1.6.4 UML Diagrams 8
1.6.5 Property Table Notation 8
2 MAEC Package Data Model 12
2.1 Package 12
2.1.1 PackageType 12
2.2 Malware Subject 13
2.2.1 MalwareSubjectType 13
2.2.2 Malware Instance Object Attributes 16
2.2.3 MalwareDevelopmentEnvironmentType 16
2.2.4 MalwareSubjectRelationshipType 17
2.3 Configuration Details 17
2.3.1 MalwareConfigurationDetailsType 18
2.3.2 MalwareConfigurationStorageDetailsType 18
2.3.3 MalwareBinaryConfigurationStorageDetailsType 19
2.3.4 MalwareConfigurationObfuscationDetailsType 19
2.3.5 MalwareConfigurationObfuscationAlgorithmType 20
2.3.6 MalwareConfigurationParameterType 20
2.4 Analysis 20
2.4.1 AnalysisType 21
2.4.2 AnalysisTypeEnum 23
2.4.3 AnalysisMethodEnum 23
2.4.4 DynamicAnalysisMetadataType 23
2.4.5 MalwareExceptionType 24
2.4.6 AnalysisEnvironmentType 24
2.4.7 HypervisorHostSystemType 25
2.4.8 AnalysisSystemType 25
2.4.9 InstalledProgramsType 25

2.4.10 NetworkInfrastructureType 26
2.4.11 CapturedProtocolType 26
2.4.12 InteractionLevelEnum 27
2.5 Findings Bundle 27
2.5.1 FindingsBundleListType 27
2.6 Grouping Relationship 28
2.6.1 GroupingRelationshipType 28
2.7 Clustering 29
2.7.1 ClusteringMetadataType 29
2.7.2 ClusteringAlgorithmParametersType 30
2.7.3 ClusterCompositionType 30
2.7.4 ClusterEdgeNodePairType 31
2.8 Meta-Analysis 32
2.8.1 MetaAnalysisType 32
2.8.2 ActionEquivalenceType 32
2.8.3 ObjectEquivalenceType 33
2.9 Shared Types 33
2.9.1 CommentType 33
2.9.2 SourceType 34
2.10 Shared Enumerations 34
2.10.1 Layer7ProtocolEnum 34
2.10.2 Layer4ProtocolEnum 35
2.11 Referential Types 35
2.11.1 MalwareSubjectReferenceType 35
2.12 List Types 36
2.12.1 ActionEquivalenceListType 36
2.12.2 AnalysisListType 36
2.12.3 AnalysisSystemListType 36
2.12.4 CapturedProtocolListType 36
2.12.5 CommentListType 37
2.12.6 GroupingRelationshipListType 37
2.12.7 MalwareSubjectListType 37
2.12.8 MalwareSubjectRelationshipListType 38
2.12.9 MinorVariantListType 38
2.12.10 ObjectEquivalenceListType 38
2.12.11 ToolListType 38
Appendix – References 39
A.1 MAEC Documents 39
A.2 MAEC Web Pages 39
A.3 MAEC Schema 40
A.4 MAEC Development 40
A.5 Other References 41

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).

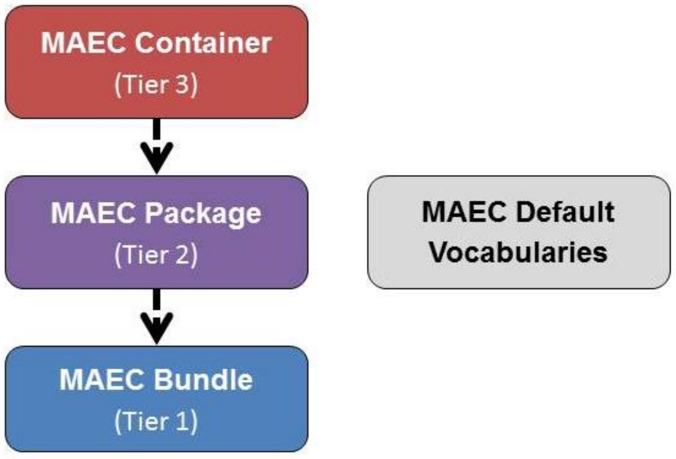


Figure 1-1. MAEC data models

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 Package data model. Before we present the Package 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 **Error! Reference source not found.** 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.

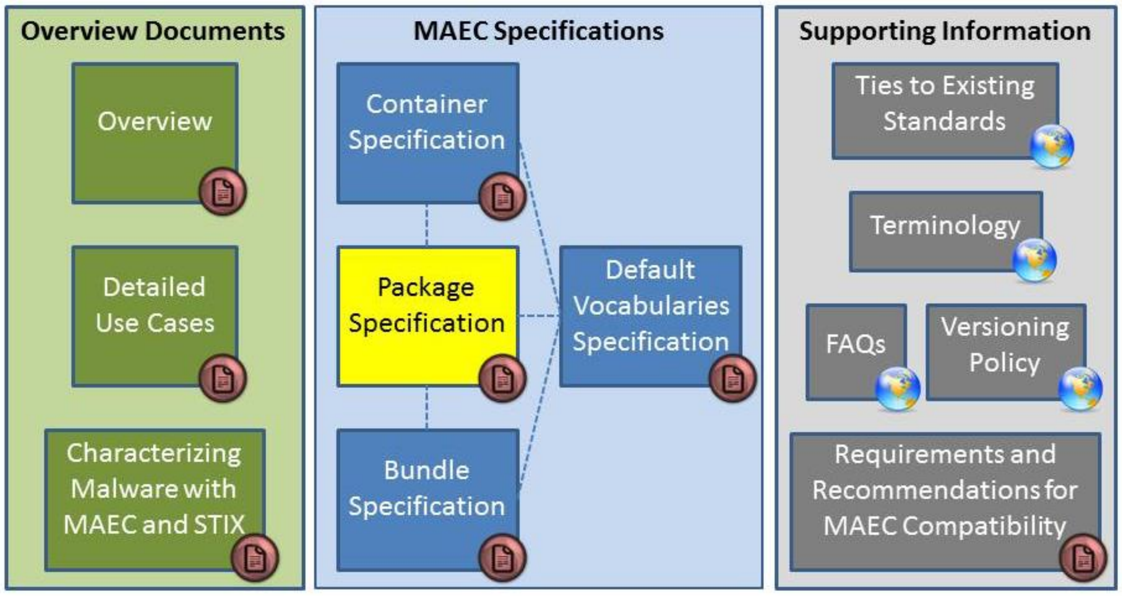


Figure 1-2. MAEC Language v4.1 documents

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].
- Package Specification: Specification for the MAEC Package data model [SPEC_P]. (This document.)

- 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 **Error! Reference source not found.**.

1.2.1 Data Model Fields and Types

In Section **Error! Reference source not found.**, we define the types associated with the MAEC Package 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 Package 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 Package Data Model

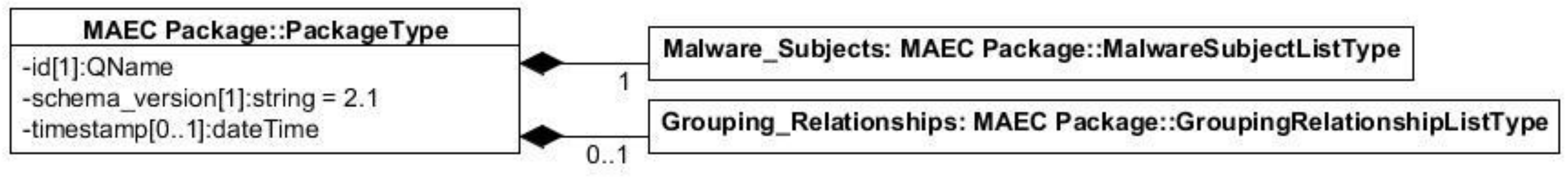
The root of the MAEC Package data model is the `MAEC_Package` field of type `PackageType`. Also key in the schema is the `Malware_Subject` field of type `MalwareSubjectType`. These and other types are described below, organized by functional group (`MAEC_Package`, `Malware_Subject`, `Analysis`, `Findings_Bundles`, and `Grouping_Relationship`). Types related to clustering appear in Section 2.7, meta-analysis types appear in Section 2.8, types shared by multiple functional groups appear in Section 2.9, shared enumerations appear in Section 2.10, "referential" types appear in Section 2.11, and "list" types appear in Section 2.12. All types originate from the MAEC Package schema, unless otherwise noted with a schema prefix, e.g., 'cybox:' for the CybOX Core schema.

2.1 Package

The root of the MAEC Package data model is the `MAEC_Package` field of type `PackageType`. A `MAEC_Package` encompasses one or more `Malware_Subjects` along with any associated metadata.

2.1.1 PackageType

The `PackageType` is the namesake type of the MAEC Package schema and captures either a single `Malware_Subject` or a collection of `Malware_Subjects` that are related in some way (even if exact details of the relationship are unknown). Unlike the `MAEC_Bundle`, which only captures the MAEC-characterized analysis results for a single malware instance, the `MAEC_Package` permits the capture of additional metadata relating to a `Malware_Subject` such as variant information, field data, and similar types of entities.


MAEC Package::PackageType
-id[1]:QName
-schema_version[1]:string = 2.1
-timestamp[0..1]:dateTime
1
Malware_Subjects: MAEC Package::MalwareSubjectListType
0..1
Grouping_Relationships: MAEC Package::GroupingRelationshipListType

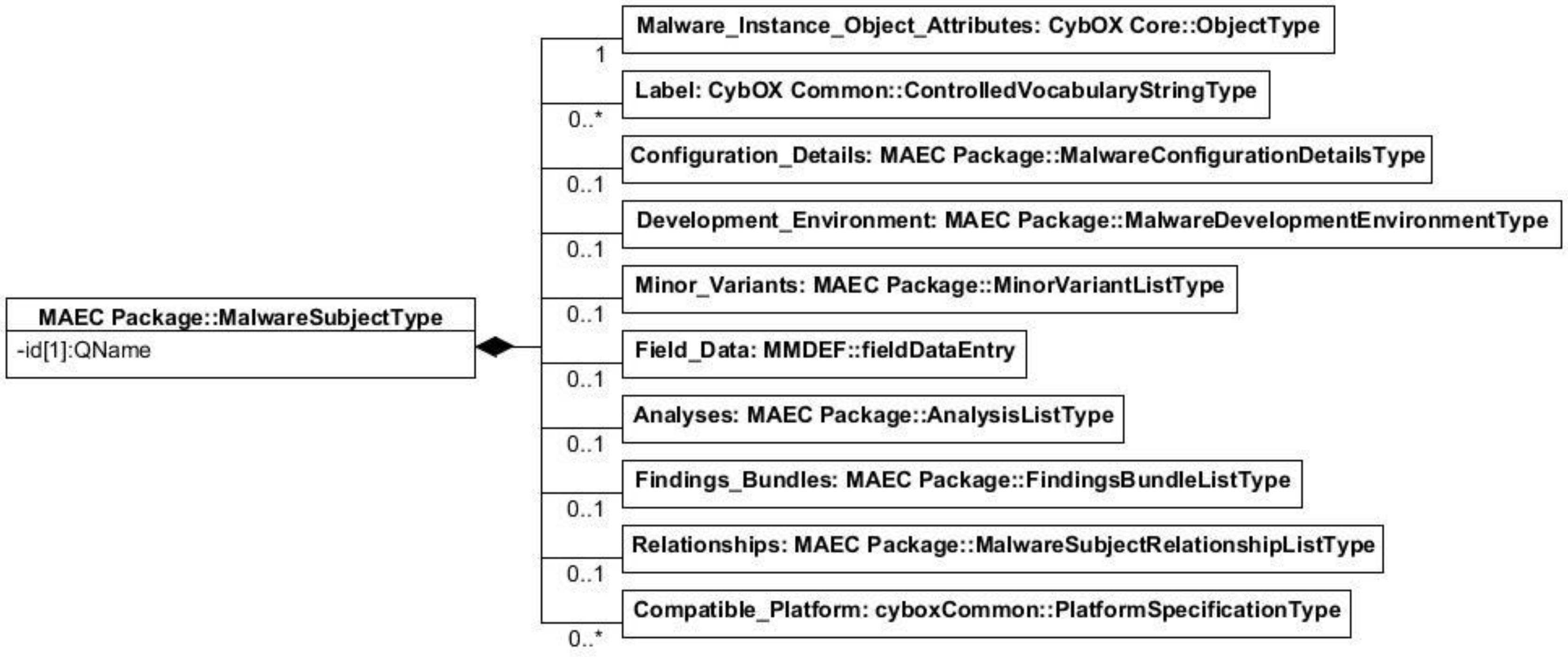
Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for the `MAEC_Package`. The IDREF SHOULD follow the pattern defined in Section 1.4.
schema_version	`string`	1	Specifies the version of the MAEC Package Schema that the document has been written in and that SHOULD be used for validation. The fixed value is '*2.1*.'
timestamp	`dateTime`	0..1	Specifies the date/time when the `MAEC_Package` was generated.
Malware_Subjects	`MalwareSubjectListType`	1	Captures each of the `Malware_Subjects` contained in the Package.
Grouping_Relationships	`GroupingRelationshipList Type`	0..1	Specifies the particular relationships that serve to group the `Malware_Subjects` encompassed by the `MAEC_Package`. Used only when more than one `Malware_Subject` is contained within the `MAEC_Package`.

2.2 Malware Subject

Also key to the MAEC Package data model is the `Malware_Subject` of type `MalwareSubjectType`. A `Malware_Subject` represents a single malware instance (most commonly a file) and its associated analysis and metadata information, such as `Analyses`, `MAEC_Bundles`, and `Relationships` to other `Malware_Subjects`.

2.2.1 MalwareSubjectType

The `MalwareSubjectType` captures all of the details pertaining to a single malware instance, including any corresponding `Analyses, Configuration_Details, Development_Environment` information, `Field_Data`, `Findings_Bundles, Label`(s), and `Relationships` to other `Malware_Subjects`.


MAEC Package::MalwareSubjectType
-id[1]:QName
Malware_Instance_Object_Attributes: CybOX Core::ObjectType
1
Label: CybOX Common::ControlledVocabularyStringType
0..*
Configuration_Details: MAEC Package::MalwareConfigurationDetailsType
0..1
Development_Environment: MAEC Package::MalwareDevelopmentEnvironmentType
0..1
Minor_Variants: MAEC Package::MinorVariantListType
0..1
Field_Data: MMDEF::fieldDataEntry
0..1
Analyses: MAEC Package::AnalysisListType
0..1
Findings_Bundles: MAEC Package::FindingsBundleListType
0..1
Relationships: MAEC Package::MalwareSubjectRelationshipListType
0..1
Compatible_Platform: cyboxCommon::PlatformSpecificationType
0..*

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for this `Malware_Subject`. The IDREF SHOULD follow the pattern defined in Section 1.4.
Malware_Instance_Object_Attributes	`cybox:ObjectType`	1	Characterizes the fields of the malware instance (most commonly a file) that is encompassed in the `Malware_Subject`, via its corresponding CybOX `Object`. For example, a file would be represented via a CybOX `File` field of type `FileObj:FileObjectType` and may have a file

			name, MD5 hash, etc.
Label	cyboxCommon:ControlledVocabularyStringType	0..*	Specifies a single commonly accepted label to describe the Malware_Subject, e.g. "worm". The default vocabulary for this field is the MAEC '*MalwareLabelVocab-1.0.*' More than one label may be specified through the use of multiple instances of this field.
Configuration_Details	MalwareConfigurationDetailsType	0..1	Captures details of the configuration specified for the Malware_Subject, such as configuration parameters.
Development_Environment	MalwareDevelopmentEnvironmentType	0..1	Captures details of the development environment used in the creation of the malware instance characterized by the Malware_Subject
Minor_Variants	MinorVariantListType	0..1	Captures any minor variants of the malware instance, such as the same file but with different filenames.
Field_Data	metadata:fieldDataEntry	0..1	Captures field data and prevalence information relating to the Malware_Subject. It uses the fieldDataEntry type from the MMDEF v1.2 schema.
Analyses	AnalysisListType	0..1	Captures any Analyses (including their associated metadata such as tools used, etc.) that were performed on the Malware_Subject.
Findings_Bundles	FindingsBundleListType	0..1	Specifies any MAEC_Bundles pertaining to the Malware_Subject, thus capturing any observed or discovered Behaviors, Actions, or Objects. These MAEC_Bundles can either be abstract or referenced as the result of an analysis that was performed on the malware.
Relationships	MalwareSubjectRelationshipListType	0..1	Captures any relationships between the Malware_Subject and other Malware_Subjects.

Compatible_Platform	cyboxCommon:Platform SpecificationType	0..*	Specifies a single platform that the Malware Subject is compatible with (i.e., can execute on). It uses the PlatformSpecificationType from the imported CybOX Common schema. More than one compatible platform can be specified by using multiple occurrences of this field.

2.2.2 Malware Instance Object Attributes

The Malware_Instance_Object_Attributes field captures the fields of the malware instance (most commonly a file) that is characterized in the Malware_Subject, via its corresponding CybOX Object. For example, a file would be represented via a CybOX File field of type FileObj:FileObjectType and may have a file name, MD5 hash, etc.

The Malware_Instance_Object_Attributes field is of type cybox:ObjectType, which will not be defined here (see [CYBOX]). While the id and idref fields of the CybOX ObjectType are OPTIONAL and have no required syntax, when the ObjectType is used in MAEC, the id field SHOULD always be used. The recommended format for the id field is given in Section 1.4. The Malware_Instance_Object_Attributes field is OPTIONAL in a MAEC_Bundle, and consequently was defined in [SPEC$_B$]; please refer to Section 2.2 in [SPEC$_B$] for details.

If a MAEC_Bundle is contained inside a Malware_Subject in a MAEC_Package, the defined_subject field of the MAEC_Bundle MUST be set to '*false*' and the fields of the object representing the malware instance MUST be captured in the Malware_Instance_Object_Attributes field at the MAEC_Package/Malware_Subject level. In this case, the content of the Malware_Instance_Object_Attributes field MAY remain in the MAEC_Bundle or MAY be removed; the MAEC_Bundle content will be superseded by the content of the Malware_Instance_Object_Attributes field at the MAEC_Package level.

2.2.3 MalwareDevelopmentEnvironmentType

The MalwareDevelopmentEnvironmentType captures details of the development environment used in developing the malware instance, such as information on any tools that were used.

Field	Type	Multiplicity	Description
Tools	`cyboxCommon: ToolsInformationType`	0..1	Captures the properties of one or more tools used in the development of the malware instance. For the `Type` field in each `Tool`, the MAEC default vocabulary '*MalwareDevelopmentToolVocab-1.0*' should be used.
Debugging_File	`FileObj:FileObjectType`	1..*	Captures the properties of a debugging file associated with the malware instance, such as a PDB file. More than one `Debugging_File` can be specified by using multiple instances of this field.

2.2.4 MalwareSubjectRelationshipType

The `MalwareSubjectRelationshipType` provides a mechanism for capturing the relationships between a `Malware_Subject` and one or more other `Malware_Subjects`.

Field	Type	Multiplicity	Description
Type	`cyboxCommon: ControlledVocabularyStringType`	1	Specifies the type of relationship being captured. The default vocabulary type for use in this field is the MAEC *'MalwareSubjectRelationshipTypeVocab-1.0.'*
Malware_Subject_Reference	`MalwareSubjectReferenceType`	1..*	Provides a reference to a single `Malware_Subject` to which the relationship pertains. More than one `Malware_Subject` may be referenced by using multiple instances of this field.

2.3 Configuration Details

The `Configuration_Details` field of type `MalwareConfigurationDetailsType` captures details of the configuration specified for the `Malware_Subject`, such as any configuration parameters.

2.3.1 MalwareConfigurationDetailsType

The `MalwareConfigurationDetailsType` captures details of malware configuration parameters and associated metadata.

Field	Type	Multiplicity	Description
Storage	`MalwareConfigurationStorageDetailsType`	0..1	Captures details of the how the malware configuration parameters may be stored, e.g., in a separate file, in memory, etc.
Obfuscation	`MalwareConfigurationObfuscationDetailsType`	0..1	Captures details of how the malware configuration parameters may be obfuscated, if applicable.
Configuration_Parameter	`MalwareConfigurationParameterType`	0..*	Captures a single configuration parameter that may be defined for the `Malware Subject`. More than one configuration parameter may be specified by using multiple occurrences of this field.

2.3.2 MalwareConfigurationStorageDetailsType

The `MalwareConfigurationStorageDetailsType` captures details relating to the storage of malware configuration parameters.

Field	Type	Multiplicity	Description
Malware_Binary	`MalwareBinaryConfigurationStorageDetailsType`	0..1	Captures properties related to the storage of malware configuration parameters inside the malware binary captured in the `Malware_Instance_Object_Attributes` field.
File	`FileObj:FileObjectType`	0..1	Captures the properties of a configuration file, for cases where the `Malware Subject` stores its configuration parameters in a separate file.
URL	`URIObj:URIObjectType`	0..*	Captures a URL at which the configuration parameters for the `Malware Subject` may be stored. More than one such URL may be specified by using multiple occurrences of this field.

2.3.3 MalwareBinaryConfigurationStorageDetailsType

The `MalwareBinaryConfigurationStorageDetailsType` captures details relating to the storage of malware configuration parameters inside the malware binary itself.

Field	Type	Multiplicity	Description
File_Offset	`hexBinary`	0..1	Specifies the offset to the start of the malware configuration parameters in the malware binary.
Section_Name	`string`	0..1	Specifies the name of the PE section in the malware binary that contains the malware configuration parameters (for PE file malware binaries).
Section_Offset	`hexBinary`	0..1	Specifies the offset in the PE section in the malware binary that contains the malware configuration parameters to the start of the parameters themselves (for PE file malware binaries).

2.3.4 MalwareConfigurationObfuscationDetailsType

The `MalwareConfigurationObfuscationDetailsType` captures details relating to the obfuscation of malware configuration parameters.

Field	Type	Multiplicity	Description
is_encoded	`boolean`	0..1	Specifies that the malware configuration parameters are encoded with the algorithm captured in the `Algorithm_Details` field.
is_encrypted	`boolean`	0..1	Specifies that the malware configuration parameters are encrypted with the algorithm captured in the `Algorithm_Details` field.
Algorithm_Details	`MalwareConfigurationObfuscationAlgorithmType`	0..*	Captures the details of the algorithm used to encode or encrypt the malware configuration parameters, including the name of the algorithm and its key. More than one encryption or encoding algorithm may be specified by using multiple occurrences of this field.

2.3.5 MalwareConfigurationObfuscationAlgorithmType

The `MalwareConfigurationObfuscationDetailsType` captures of an algorithm used to encode or encrypt malware configuration parameters.

Field	Type	Multiplicity	Description
ordinal_position	`positiveInteger`	0..1	Specifies the explicit ordering of the usage of the algorithm with respect to the other algorithms used to encrypt or encode the malware configuration parameters, for cases where more than one algorithm was used.
Key	`hexBinary`	0..1	Captures the hexadecimal key used to decrypt the configuration parameters.
Algorithm_Name	`cyboxCommon:ControlledVocabularyStringType`	0..1	Captures the name of the encoding or encryption algorithm used to obfuscate the malware configuration parameters.

2.3.6 MalwareConfigurationParameterType

The `MalwareConfigurationParameterType` captures a single configuration parameter that may be defined for a malware instance, as a name/value pair.

Field	Type	Multiplicity	Description
Name	`cyboxCommon: ControlledVocabularyStringType`	0..1	Specifies the name of the configuration parameter being captured. The default vocabulary type for the `Name` field is the MAEC '*MalwareConfigurationParameterTypeVocab-1.0*.'
Value	`string`	0..1	Specifies the value of the malware configuration parameter.

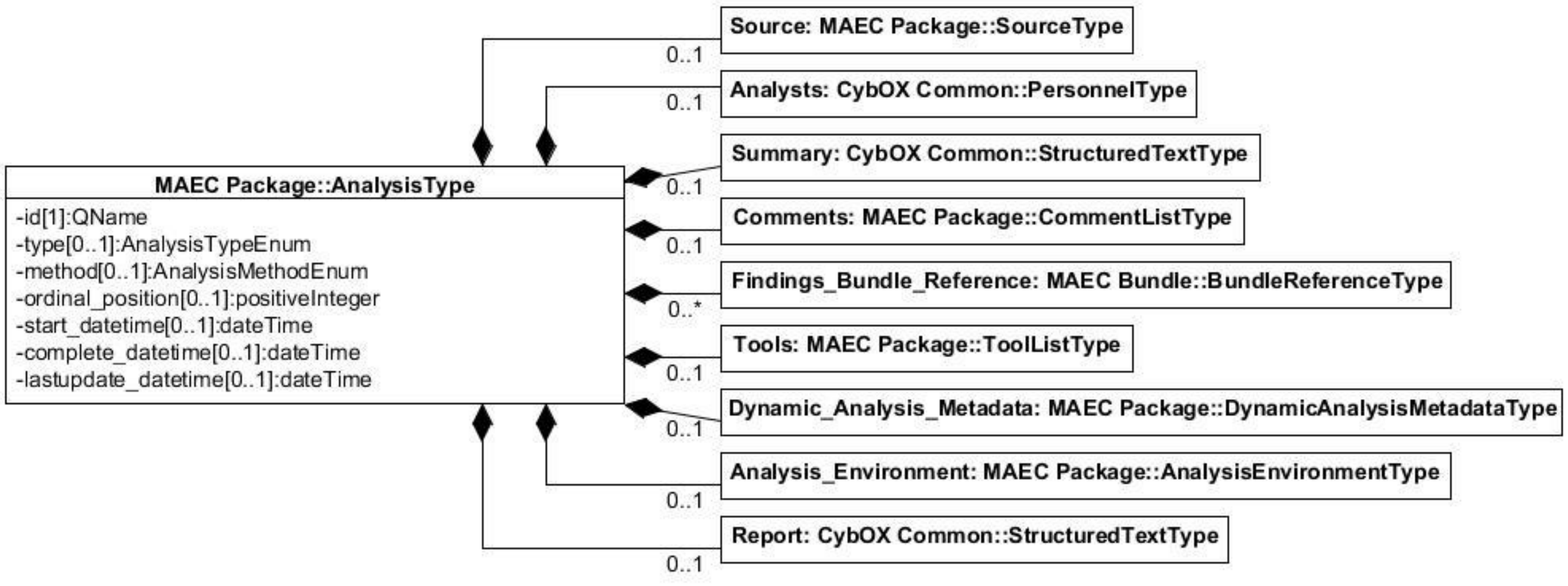
2.4 Analysis

The `Analysis` field of type `AnalysisType` captures details of any analysis (including associated metadata such as tools used, etc.) that was performed on the `Malware_Subject`.

2.4.1 AnalysisType

The `AnalysisType` provides a way of capturing the information associated with the analysis of a malware instance, such as the method(s) used, authors, the date/time when it was started and finished, and other relevant data. The data model is meant to be flexible in its ability to capture analysis data, although best practices suggest the following use of the fields when capturing textual information associated with an analysis:

- `Summary` – the `Summary` SHOULD be high-level and concise. It SHOULD summarize the contents of the `Report` field, if present, and otherwise SHOULD provide a brief synopsis of the analysis that was performed and any highlights.
- `Comments` – a comment SHOULD be attributable to a specific analyst and SHOULD reflect particular insights of the author that are significant from an analysis standpoint. The contents of comments are typically not contained in the `Report`.
- `Report` – the `Report` SHOULD correspond to the human-readable prose document that captures key aspects and outcomes of the analysis.


MAEC Package::AnalysisType
-id[1]:QName
-type[0..1]:AnalysisTypeEnum
-method[0..1]:AnalysisMethodEnum
-ordinal_position[0..1]:positiveInteger
-start_datetime[0..1]:dateTime
-complete_datetime[0..1]:dateTime
-lastupdate_datetime[0..1]:dateTime
Source: MAEC Package::SourceType
0..1
Analysts: CybOX Common::PersonnelType
0..1
Summary: CybOX Common::StructuredTextType
0..1
Comments: MAEC Package::CommentListType
0..1
Findings_Bundle_Reference: MAEC Bundle::BundleReferenceType
0..*
Tools: MAEC Package::ToolListType
0..1
Dynamic_Analysis_Metadata: MAEC Package::DynamicAnalysisMetadataType
0..1
Analysis_Environment: MAEC Package::AnalysisEnvironmentType
0..1
Report: CybOX Common::StructuredTextType
0..1

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for the MAEC `Analysis`. The IDREF SHOULD follow the pattern defined in Section 1.4.
type	`AnalysisTypeEnum`	0..1	Specifies the type of malware analysis being performed.
method	`AnalysisMethodEnum`	0..1	Characterizing the type of analysis method used in the analysis field.
ordinal_position	`positiveInteger`	0..1	Specifies ordering of the `Analysis` with respect to other `Analyses` performed on the `Malware_Subject`.
start_datetime	`dateTime`	0..1	Specifies the date/time the analysis was started.
complete_datetime	`dateTime`	0..1	Specifies the date/time the analysis was completed.
lastupdate_datetime	`dateTime`	0..1	Specifies the date/time the analysis was updated.
Source	`SourceType`	0..1	Specifies information about the internal or external source of the analysis, if applicable.
Analysts	`cyboxCommon:PersonnelType`	0..1	Specifies the analyst(s) who performed the analysis.
Summary	`cyboxCommon:StructuredTextType`	0..1	Specifies a summary of the analysis that was performed.
Comments	`CommentsListType`	0..1	Specifies any comments regarding the analysis that was performed.
Findings_Bundle_Reference	`maecBundle:BundleReferenceType`	0..*	Specifies a reference to a `MAEC_Bundle` contained in the `Malware_Subject/Findings_Bundles` field that encompasses the results and output of the Analysis in terms of the corresponding MAEC entities, such as `Behaviors` and `Actions`. More than one Bundle may be referenced by using multiple occurrences of this field.
Tools	`ToolListType`	0..1	Specifies information about the tool(s) used in the analysis, via the CybOX `ToolInformationType`. If only a single `Tool` is specified, then this implies that this tool was responsible for all of the findings contained in the `MAEC_Bundle` referenced by the

			`Findings_Bundle_Reference` field.
Dynamic_Analysis_Metadata	`DynamicAnalysisMetadataType`	0..1	Specifies metadata pertaining to the dynamic analysis of the subject binary, such as the command line used, the duration of the analysis, etc.
Analysis_Environment	`AnalysisEnvironmentType`	0..1	Specifies fields for characterizing the analysis environment in which the analysis was performed.
Report	`cyboxCommon:StructuredTextType`	0..1	Specifies the textual report regarding the analysis performed on the malware.

2.4.2 AnalysisTypeEnum

The `AnalysisTypeEnum` is an enumeration of types of malware analyses.

Enumeration Value	Description
triage	Specifies a cursory, or triage type of malware analysis, commonly automated in conjunction with one or more tools.
in-depth	Specifies a detailed type of malware analysis that is typically performed by a human analyst.

2.4.3 AnalysisMethodEnum

The `AnalysisMethodEnum` is an enumeration of malware analysis methods.

Enumeration Value	Description
static	Specifies a static malware analysis method, which is achieved by inspecting but not executing the malware instance.
dynamic	Specifies a dynamic malware analysis method, which is achieved by executing but not inspecting the malware instance.
combination	Specifies a combination of dynamic and static malware analysis, achieved by both inspecting and executing the malware instance.

2.4.4 DynamicAnalysisMetadataType

The `DynamicAnalysisMetadataType` captures any metadata specific to the dynamic analysis of a malware instance.

Field	Type	Multiplicity	Description
Command_Line	`string`	0..1	Specifies the command line used to launch the subject binary.
Analysis_Duration	`float`	0..1	Specifies the duration of the overall dynamic analysis process, in seconds.
Exit_Code	`integer`	0..1	Specifies the exit code with which the subject binary exited.
Raised_Exception	`MalwareExceptionType`	0..*	Captures a single exception that was raised (or thrown) during the execution of the malware instance. More than one exception may be captured through the use of multiple instances of this field.

2.4.5 MalwareExceptionType

The `MalwareExceptionType` captures details of exceptions that may be raised as a result of a malware instance executing on a system. This complex type extends `ErrorType` defined in CybOX Common. The extended fields are shown below.

Field	Type	Multiplicity	Description
is_fatal	`boolean`	0..1	Specifies whether the exception is fatal; that is, whether it caused the malware instance to terminate.
Exception_Code	`string`	0..1	Specifies the particular code that identifies the type of exception that occurred.
Faulting_Address	`hexBinary`	0..1	Specifies the memory address where the exception occurred.
Description	`short`[8]	0..1	Captures the textual description of the exception.

2.4.6 AnalysisEnvironmentType

The `AnalysisEnvironmentType` provides mechanisms for characterizing the particular hardware/software environment used in the analysis of a `Malware_Subject`.

Field	Type	Multiplicity	Description
Hypervisor_Host_System	`HypervisorHostSystemType`	0..1	Characterizes the (physical) host system used in the analysis on which the VM Hypervisor runs.

[8] The correct type for this field should be `string`. This is a documented bug and will be addressed in the next MAEC release.

Analysis_Systems	AnalysisSystemList Type	0..1	Captures a list of the systems, physical or virtual, used in the analysis of a Malware_Subject.
Network_Infrastructure	NetworkInfrastructure Type	0..1	Captures details of the network infrastructure used in the analysis environment, such as any network protocols that are captured or manipulated.

2.4.7 HypervisorHostSystemType

The HypervisorHostSystemType characterizes the Hypervisor_Host_System field used in the malware analysis environment. This complex type extends the CybOX SystemObj:SystemObjectType. The extended field is shown below.

Field	Type	Multiplicity	Description
VM_Hypervisor	cyboxCommon:PlatformSpecificationType	0..1	Refers to the name of the VM Hypervisor that hosts the operating system(s) on which the analysis was performed.

2.4.8 AnalysisSystemType

The AnalysisSystemType is intended to characterize the system(s) (real or virtual) on which the actual analysis was performed, including information about both the hardware and software, such as the properties of its BIOS, processor architecture, and operating system. It extends the CybOX SystemObj:SystemObjectType. The extended field is shown below.

Field	Type	Multiplicity	Description
Installed_Programs	InstalledProgramsType	0..1	Specifies the programs installed on the OS that was used to perform the analysis. This can be useful for clarifying the nature of the analysis environment, for instance for determining whether an exploited piece of software was present, as well as for specifying any tools that may have been installed.

2.4.9 InstalledProgramsType

The InstalledProgramsType captures the programs installed on a particular operating system image.

Field	Type	Multiplicity	Description
Program	`cyboxCommon:PlatformSpecificationType`	1..*	Specifies a single program that is installed on the system. It uses `PlatformSpecificationType` from the CybOX Common schema. Multiple installed programs may be specified by using multiple occurrences of this field.

2.4.10 NetworkInfrastructureType

The `NetworkInfrastructureType` captures specific details about the network infrastructure used in the malware analysis environment.

Field	Type	Multiplicity	Description
Captured_Protocols	`CapturedProtocolListType`	1	Specifies a list of network protocols, along with the particular level of interaction that the malware analysis environment captures or interacts with in some fashion.

2.4.11 CapturedProtocolType

The `CapturedProtocolType` specifies the details of a network protocol that may be captured or otherwise manipulated in the malware analysis environment.

Field	Type	Multiplicity	Description
layer7_protocol	`Layer7ProtocolEnum`	0..1	Specifies the name of the Layer 7 network protocol (OSI model) captured or manipulated by the analysis environment.
layer4_protocol	`Layer4ProtocolEnum`	0..1	Specifies the name of the Layer 4 network protocol (OSI model) captured or manipulated by the analysis environment.
port_number	`positiveInteger`	0..1	Specifies the port number for this network protocol that is captured or manipulated by the analysis environment.
interaction_level	`InteractionLevelEnum`	0..1	Specifies the relative level of interaction that the analysis environment has with the specified network protocol.

2.4.12 InteractionLevelEnum

The `InteractionLevelEnum` is a non-exhaustive enumeration of interaction levels for network protocols in a malware analysis environment.

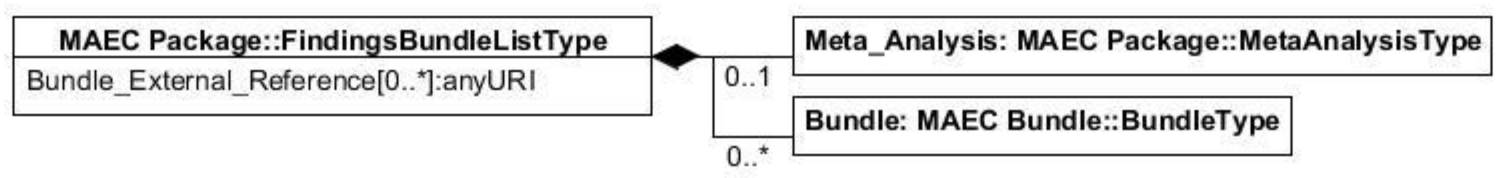
Enumeration Value	Description
high	Specifies that, for the specified protocol, the analysis environment will establish the connection and attempt to decode/identify any common protocols used by the malware. The level of decode/protocol support can be subjective and dependent on the particular environment.
low	Specifies that, for the specified protocol, the analysis environment will accept the packets and will identify the initial connection request. No further interaction is performed.
honeytrap	Specifies that, for the specified protocol, the analysis environment will establish the connection and attempt to interact with outgoing requests. The level of interaction can be subjective and dependent on the particular environment.
live	Specifies that, for the specified protocol, the analysis environment allows the malware to connect out to the real (un-emulated) IP.
none	Specifies that, for the specified protocol, the analysis environment does not support or perform any level of interaction.

2.5 Findings Bundle

The `Findings_Bundles` field of type `FindingsBundleListType` contains a set of `MAEC_Bundles` (or external references to `MAEC_Bundles`), along with any related meta-analysis entities, such as equivalencies.

2.5.1 FindingsBundleListType

The `FindingsBundleListType` captures a list of `MAEC_Bundles` or external references to `MAEC_Bundles`, along with any related meta-analysis entities.


MAEC Package::FindingsBundleListType
Bundle_External_Reference[0..*]:anyURI
0..1
0..*
Meta_Analysis: MAEC Package::MetaAnalysisType
Bundle: MAEC Bundle::BundleType

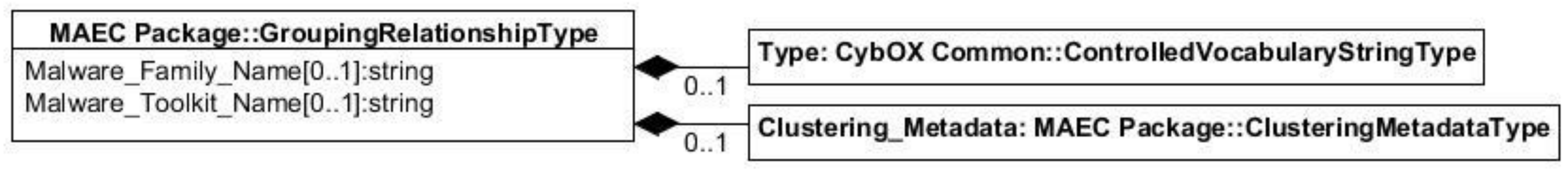
Field	Type	Multiplicity	Description
Meta_Analysis	`MetaAnalysisType`	0..1	Captures any meta-analysis related entities for the `MAEC_Bundles` captured for a `Malware_Subject`, such as equivalencies.
Bundle	`maecBundle: BundleType`	0..*	Captures a single `MAEC_Bundle`, representing some set of characterized entities resulting from an analysis performed on the `Malware_Subject`. Multiple `MAEC_Bundles` can be captured by using multiple occurrences of this field.
Bundle_External_Reference	`anyURI`	0..*	Specifies a single externally located `MAEC_Bundle` (such as a file or URL) via a URI, representing some set of results from analysis of the `Malware_Subject`. Multiple external `MAEC_Bundles` can be referenced by using multiple occurrences of this field.

2.6 Grouping Relationship

The `Grouping_Relationship` field of type `GroupingRelationshipType` specifies a particular relationship that serves to group the `Malware_Subject` entities within the `MAEC_Package`. It is used solely in cases where more than one `Malware_Subject` is contained within the `MAEC_Package`.

2.6.1 GroupingRelationshipType

The `GroupingRelationshipType` provides a mechanism for specifying the relationship that groups together the `Malware_Subjects` in a `MAEC_Package`.


MAEC Package::GroupingRelationshipType
Malware_Family_Name[0..1]:string
Malware_Toolkit_Name[0..1]:string
0..1
Type: CybOX Common::ControlledVocabularyStringType
0..1
Clustering_Metadata: MAEC Package::ClusteringMetadataType

Field	Type	Multiplicity	Description
Type	`cyboxCommon: ControlledVocabularyStringType`	1	Specifies the type of relationship that groups the `Malware_Subjects` in the `MAEC_Package`. The default vocabulary type for use in this field is the MAEC *'GroupingRelationshipTypeVocab-1.0.'*
Malware_Family_Name	`string`	0..1	Specifies the name of the malware family when the Type field is set to the value of '*same malware family.*'
Malware_Toolkit_Name	`string`	0..1	Specifies the name of the malware toolkit when the Type field is set to the value of '*same malware toolkit.*'
Clustering_Metadata	`ClusteringMetadataType`	0..1	Specifies any metadata regarding the algorithm and/or methods used for clustering the `Malware_Subjects` in this `MAEC_Package` when the `Type` field is *'clustered together*.'

2.7 Clustering

This section contains clustering-related components to be used when specifying the `clustered_together` grouping relationship type.

2.7.1 ClusteringMetadataType

The `ClusteringMetadataType` specifies metadata regarding a particular method used to cluster malware.

Field	Type	Multiplicity	Description
Algorithm_Name	`string`	0..1	Specifies the name of the clustering algorithm used to cluster the malware.
Algorithm_Version	`string`	0..1	Specifies the version of the algorithm used to cluster the malware.
Algorithm_Parameters	`ClusteringAlgorithmParametersType`	0..1	Specifies any parameters that may have been used in the clustering algorithm.
Cluster_Size	`positiveInteger`	0..1	Specifies the size of the malware cluster.
Cluster_Description	`string`	0..1	Specifies a textual description of the malware cluster, such as information about its composition, etc.
Cluster_Composition	`ClusterCompositionType`	0..1	Specifies the composition of the malware cluster, including the similarity indices between its members, as a collection of edges and their corresponding nodes.

2.7.2 ClusteringAlgorithmParametersType

The `ClusteringAlgorithmParametersType` captures any parameters that may have been used in a malware clustering algorithm.

Field	Type	Multiplicity	Description
Distance_Threshold	`decimal`	0..1	Specifies the minimum distance threshold for the cluster, or the minimum distance between nodes in order for them to belong to the same cluster.
Number_of_Iterations	`positiveInteger`	0..1	Specifies the number of times that the algorithm was executed in order to produce the cluster.

2.7.3 ClusterCompositionType

The `ClusterCompositionType` captures the composition of a malware cluster via its edges and their respective connected nodes, as in an undirected graph.

Field	Type	Multiplicity	Description
score_type	`string`	0..1	Specifies the type of score used to define the composition of the malware cluster (clustering algorithms may capture different types of scores).
Edge_Node_Pair	`ClusterEdgeNodePairType`	1..*	Specifies a single edge and its connected nodes in the malware cluster, representing the similarity index between two `Malware_Subjects`.

2.7.4 ClusterEdgeNodePairType

The `ClusterEdgeNodePairType` captures a single edge-node pair in a malware cluster, which is composed of the two `Malware_Subjects` that correspond to the nodes connected to the edge (via references), and represents the similarity index between the two `Malware_Subjects`.

Field	Type	Multiplicity	Description
similarity_index	`decimal`	0..1	Specifies the similarity index between the two `Malware_Subjects` being referenced (indicating how similar they are), as a decimal value. This value SHOULD be equivalent to 1 minus the similarity distance value (if included).
similarity_distance	`decimal`	0..1	Specifies the similarity distance between the two `Malware_Subjects` being referenced (indicating how dissimilar they are), as a decimal value. This value SHOULD be equivalent to 1 minus the similarity index value (if included).
Malware_Subject_Node_A	`MalwareSubjectReferenceType`	1	Represents a node connected to the edge via a reference to a `Malware_Subject` that is part of a malware cluster.
Malware_Subject_Node_B	`MalwareSubjectReferenceType`	1	Represents a node connected to the edge via a reference to a `Malware_Subject` that is part of a malware cluster.

2.8 Meta-Analysis

This section captures types reflecting meta-analysis information.

2.8.1 MetaAnalysisType

The `MetaAnalysisType` captures meta-analysis entities associated with the `MAEC_Bundles` that were captured for a `Malware_Subject`, such as `Action_Equivalencies`.

Field	Type	Multiplicity	Description
Action_Equivalences	`ActionEquivalenceListType`	0..1	Captures any equivalences between `Actions` contained in one or more `MAEC_Bundles`.
Object_Equivalences	`ObjectEquivalenceListType`	0..1	Captures any equivalences between `Objects` contained in one or more `MAEC_Bundles`.

2.8.2 ActionEquivalenceType

The `ActionEquivalenceType` relates any `Actions` that are equivalent to each other, e.g., those that were found for the same `Malware_Subject` when using different analysis tools. For example, two different dynamic analysis tools may execute a binary and report an identical '*create file*' `Action`, which would then be stored in the `Findings_Bundles` field that corresponds to the findings of each respective tool. The `ActionEquivalenceType` can be used as a means of explicitly specifying that these `Actions` were reported in two unique `MAEC_Bundles`, which can be useful as a means of generating `Candidate_Indicators`, specifying entities that may merit further analysis, and so forth. It can also be used as a way of referencing equivalent `Actions` as a single unit, such as for specifying the `Action` composition of a `Behavior`.

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies the ID for the `Action_Equivalence`. The ID SHOULD follow the pattern defined in Section 1.4.
Action_Reference	`cybox:ActionReferenceType`	1..*	Specifies a reference to a single `Action` that is part of the `Action_Equivalence`.

2.8.3 ObjectEquivalenceType

The `ObjectEquivalenceType` relates any `Objects` that are equivalent to each other, e.g., those that were found for the same `Malware_Subject` when using different analysis tools; it extends the MAEC Bundle `ObjectReferenceListType`. The extended field is shown below.

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies the ID for the `Object_Equivalence`. The ID SHOULD follow the pattern defined in Section 1.4.

2.9 Shared Types

These types are used by a variety of fields. They are listed in alphabetical order. Note that some types in this section are currently used by only one particular field; however, they are placed here because they could be used more generally.

2.9.1 CommentType

The `CommentType` captures a comment relating to some MAEC field. It is of base type `cyboxCommon:StructuredTextType` and is extended with the following fields.

Field	Type	Multiplicity	Description
author	`string`	0..1	Specifies the name of the author that added the comment.
timestamp	`dateTime`	0..1	Specifies the date/time that the comment was added.
observation_name	`string`	0..1	Captures the name, type, or identifier of an observation for comments that refer to the observation of particular entities. For example, a comment that refers to a command and control (C2) encryption key could have an `observation_name` of "C2 Encryption Key".

2.9.2 SourceType

The `SourceType` provides a way of characterizing the external source of a relevant MAEC field, such as an `Analysis`.

Field	Type	Multiplicity	Description
Name	`string`	0..1	Refers to the name of the person linked to the source.
Method	`string`	0..1	Provides an abstract way of specifying the method used to obtain the data that the Source field refers to. Examples methods include '*organizational affiliation*', '*personal contact*', and '*open source*.'
Reference	`string`	0..1	Provides an abstract way of specifying a reference name or ID for the source.
Organization	`string`	0..1	Specifies the name of the organization from which the source originated.
URL	`anyURI`	0..1	Specifies the Uniform Resource Locator (URL) of the external source, if applicable.

2.10 Shared Enumerations

These enumerations are used by a variety of fields. They are listed in alphabetical order. Note that some enumerations in this section are currently used by only one particular field; however, they are placed here because they could be used more generally.

2.10.1 Layer7ProtocolEnum

The `Layer7ProtocolEnum` is a non-exhaustive enumeration of OSI model Layer 7 (application layer) network protocols.

Enumeration Value	Description
http	Specifies the Hypertext Transfer Protocol (HTTP).
https	Specifies the Hypertext Transfer Protocol Secure (HTTPS).
ftp	Specifies the File Transfer Protocol (FTP).
ftps	Specifies the File Transfer Protocol Secure (FTPS).
smtp	Specifies the Simple Mail Transfer Protocol (SMTP).
smtps	Specifies the Simple Mail Transfer Protocol Secure (SMTPS).
pop3	Specifies the Post Office Protocol version 3 (POP3).

pop3s	Specifies the Post Office Protocol version 3 Secure (POP3S).
irc	Specifies the Internet Relay Chat (IRC) protocol.
dns	Specifies the Domain Name System (DNS) protocol.
rdp	Specifies the Remote Desktop Protocol (RDP).
rpc	Specifies some Remote Procedure Call (RPC) protocol, such as MSRPC.
ssh	Specifies the Secure Shell (SSH) protocol.
telnet	Specifies the Telnet protocol.

2.10.2 Layer4ProtocolEnum

The `Layer4ProtocolEnum` is a non-exhaustive enumeration of OSI model Layer 4 (transport layer) network protocols.

Enumeration Value	Description
tcp	Specifies the Transport Control Protocol (TCP).
udp	Specifies the User Datagram Protocol (UDP).

2.11 Referential Types

This section defines the types of the MAEC Package data model whose sole purpose is to reference other types.

2.11.1 MalwareSubjectReferenceType

The `MalwareSubjectReferenceType` provides a mechanism for specifying a reference to a `Malware_Subject` contained in the `MAEC_Package`.

Field	Type	Multiplicity	Description
malware_subject_idref	`MalwareSubjectIDREFPattern`	1	Provides a reference to a `Malware_Subject` contained in the `MAEC_Package`, via its ID. The IDREF SHOULD follow the pattern defined in Section 1.4.

2.12 List Types

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

2.12.1 ActionEquivalenceListType

The `ActionEquivalenceListType` captures a list of `Action_Equivalences`.

Field	Type	Multiplicity	Description
Action_Equivalence	`ActionEquivalenceType`	1..*	Captures a single `Action_Equivalence` in the list.

2.12.2 AnalysisListType

The `AnalysisListType` captures a list of analyses that were performed on a `Malware_Subject`.

Field	Type	Multiplicity	Description
Analysis	`AnalysisType`	1..*	Represents the metadata regarding a single analysis that was performed on a `Malware_Subject`.

2.12.3 AnalysisSystemListType

The `AnalysisSystemListType` captures a list of the systems, physical or virtual, used in the analysis of a `Malware_Subject`.

Field	Type	Multiplicity	Description
Analysis_System	`AnalysisSystemType`	1..*	Captures a single analysis system.

2.12.4 CapturedProtocolListType

The `CapturedProtocolListType` captures a list of network protocols that a malware analysis environment may capture or interact with.

Field	Type	Multiplicity	Description
Protocol	`CapturedProtocolType`	1..*	Specifies a single layer 4 or layer 7 network protocol captured or interacted with by the analysis environment.

2.12.5 CommentListType

The `CommentListType` captures any comments relating to MAEC entities, such as `Analyses`.

Field	Type	Multiplicity	Description
Comment	`CommentType`	1..*	Specifies a single comment pertaining to a particular MAEC field.

2.12.6 GroupingRelationshipListType

The `GroupingRelationshipListType` captures a list of grouping relationships relating the `Malware_Subjects` in a `MAEC_Package`.

Field	Type	Multiplicity	Description
Grouping_Relationship	`GroupingRelationshipType`	1..*	Specifies a single grouping relationship in the list.

2.12.7 MalwareSubjectListType

The `MalwareSubjectListType` captures a list of `Malware_Subjects`.

Field	Type	Multiplicity	Description
Malware_Subject	`MalwareSubjectType`	1..*	Represents a single `Malware_Subject` (most commonly a file) and its associated metadata, such as `Analyses`, `MAEC_Bundles`, relationships to other `Malware_Subjects`, etc.

2.12.8 MalwareSubjectRelationshipListType

The MalwareSubjectRelationshipListType captures a list of relationships between a Malware_Subject and other Malware_Subjects.

Field	Type	Multiplicity	Description
Relationship	MalwareSubjectRelationshipType	1..*	Specifies a relationship that relates the Malware_Subject to one or more other Malware_Subjects contained in the MAEC_Package.

2.12.9 MinorVariantListType

The MinorVariantListType captures a list of minor variants of a Malware_Subject's malware instance (e.g., the same binary with but with different filenames).

Field	Type	Multiplicity	Description
Minor_Variant	cybox:ObjectType	1..*	Captures a single minor variant of the malware instance.

2.12.10 ObjectEquivalenceListType

The ObjectEquivalenceListType captures a list of Object_Equivalences.

Field	Type	Multiplicity	Description
Object_Equivalence	ObjectEquivalenceType	1..*	Specifies a single Object Equivalence in the list.

2.12.11 ToolListType

The ToolsType characterizes one or more tools, such as those used in the analysis of a Malware_Subject.

Field	Type	Multiplicity	Description
Tool	cyboxCommon:ToolInformationType	1..*	Specifies a single Tool in the list.

Appendix – 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