THE MITRE CORPORATION

THE MAEC™ LANGUAGE VERSION 4.1 SPECIFICATION

MAEC BUNDLE VERSION 4.1

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 Bundle 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 **Error! Reference source not found.**, "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).

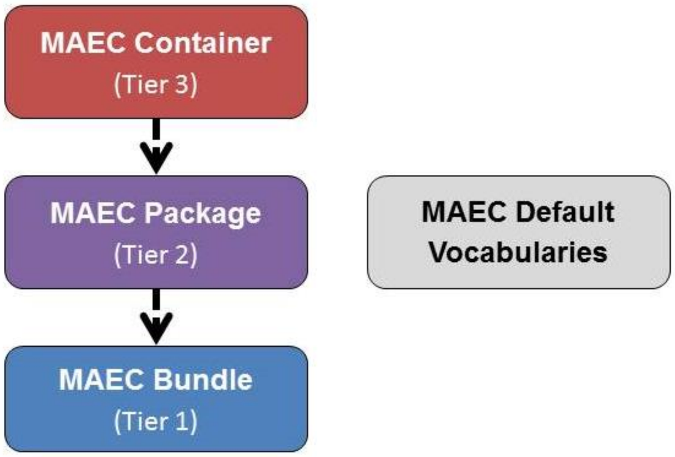


Figure 1-1. MAEC data models

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 in MAEC v4.1 via an XML schema. Other output formats, such as JSON, are being considered for future implementations.

This document serves as the specification for the MAEC Bundle data model. Before we present the Bundle 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.

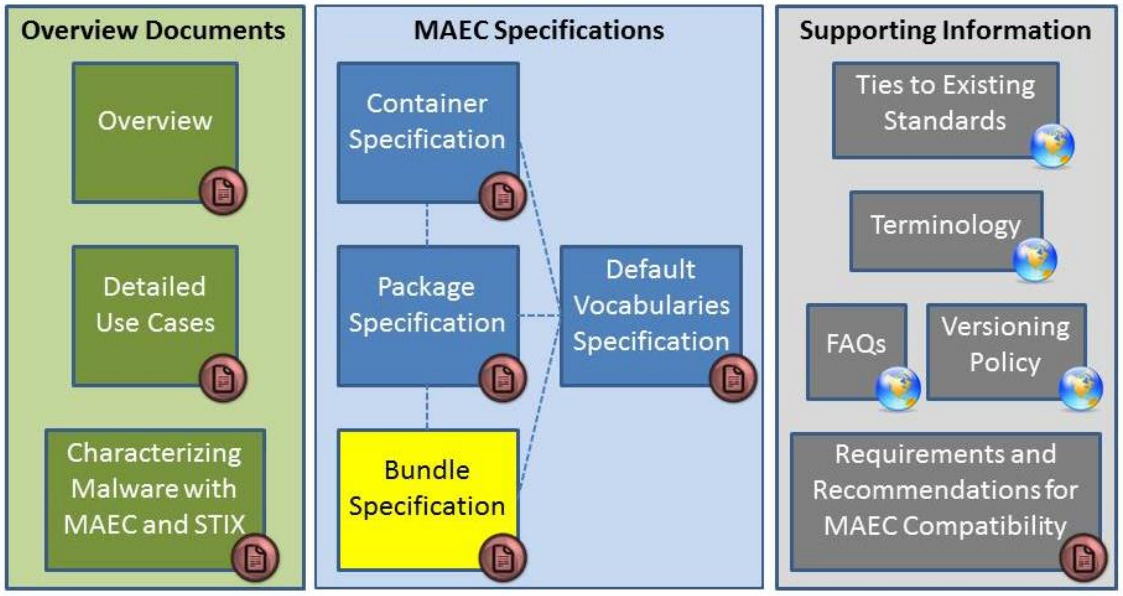


Figure 1-2. MAEC Language v4.1 documents

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 [$SPEC_P$].

- Bundle Specification: Specification for the MAEC Bundle data model [MAEC_B]. (This document.)
- 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 Bundle 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 Bundle data model.

[4] As stated in footnote 3, each data model and the default vocabularies are implemented 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 **Error! Reference source not found.** 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 Bundle Data Model

The root of the MAEC Bundle v4.1 data model is the `MAEC_Bundle` field of type `BundleType`. The `BundleType` and other types are defined below. Definitions have been organized by functional group (`MAEC_Bundle`, `Process_Tree`, `Capability`, `Behavior`, `Action`, `Object`, `Candidate_Indicator`, and `Collections`). Types shared by multiple functional groups appear in Section 2.11, "referential" types appear in Section 2.12, and "list" types appear in Section 2.13. All types originate from the MAEC Bundle schema, unless otherwise noted with a namespace prefix, e.g., "`cybox`" for the CybOX Core schema[8].

MAEC is designed to be very flexible, which means that a `MAEC_Bundle` containing analysis data can be created in a variety of ways. However, there are practices that best take advantage of MAEC's features; for example, to conserve space in a MAEC document, one may place all objects under the `Objects` root field (of type `ObjectListType`), all actions under the `Actions` root field (of type `ActionListType`), etc. The `Object` and `Action` fields, etc. can then be referenced as needed throughout the `MAEC_Bundle` via their `ID` attribute using the `IDREF` field on the corresponding reference structure, e.g., the `ObjectReferenceType`. In addition to reducing the amount of space that would be required if each object or action were defined multiple times in a `MAEC_Bundle`, the practice of having all entities of the same type defined in one location is advantageous from an organizational perspective.

Alternatively, in some situations, `Objects`, `Actions`, `Behaviors`, and `Candidate_Indicators` might be best grouped according to categories by leveraging the `Collections` field (`Object_Collections`, `Action_Collections`, etc.) For example, an analyst may find it easiest to define all IP addresses objects associated with a malware instance in one `Object_Collection` field and all URL objects in a second `Object_Collection` field. However, if there will be duplication between the collections, it might still be preferable to characterize the objects under the `Objects` field and then reference the relevant `Object` fields from each of the collections via their `ID` attribute. As MAEC is used more operationally, more information on best practices will be available on the MAEC Web site [MAEC].

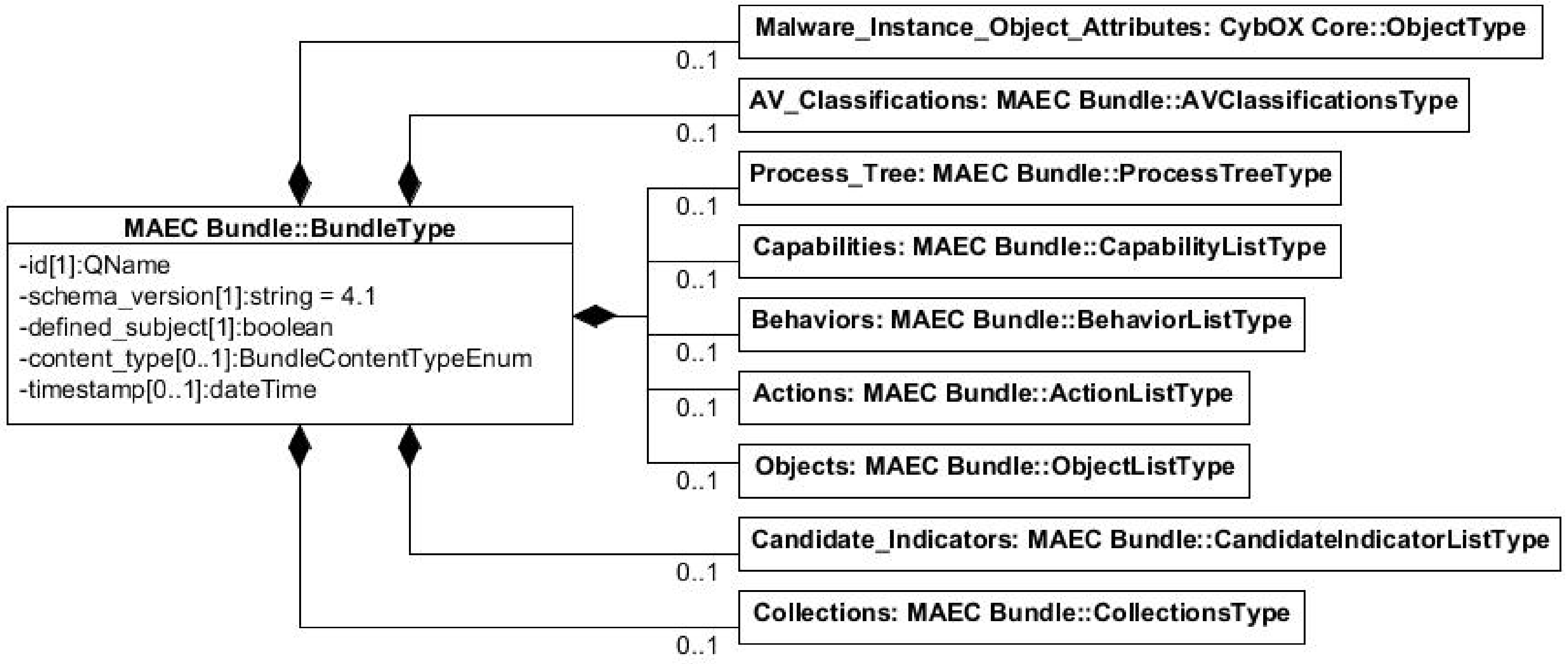
[8] As stated in [MAEC$_O$], MAEC draws several components from the CybOX Language; consequently, the reader is referred to [CYBOX] for the definitions of these entities. In this specification, we do not define any types that are part of a non-MAEC data model. Instead we make note of the referenced data 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.

2.1 MAEC Bundle

The root field of the MAEC Bundle schema is the `MAEC_Bundle` field of type `BundleType`. The `MAEC_Bundle` field represents the characterization of a single malware instance, whose identity is characterized in the top-level `Malware_Instance_Object_Attributes` field, via the CybOX `ObjectType`.

2.1.1 BundleType

The `BundleType` serves as the high-level construct that encapsulates all `MAEC_Bundle` fields and represents some characterized analysis data (from any arbitrary set of analyses) for a single malware instance in terms of its MAEC components (e.g., `Capabilities`, `Behaviors`, `Actions`, `Objects`, etc.).


MAEC Bundle::BundleType
-id[1]:QName
-schema_version[1]:string = 4.1
-defined_subject[1]:boolean
-content_type[0..1]:BundleContentTypeEnum
-timestamp[0..1]:dateTime
Malware_Instance_Object_Attributes: CybOX Core::ObjectType
AV_Classifications: MAEC Bundle::AVClassificationsType
Process_Tree: MAEC Bundle::ProcessTreeType
Capabilities: MAEC Bundle::CapabilityListType
Behaviors: MAEC Bundle::BehaviorListType
Actions: MAEC Bundle::ActionListType
Objects: MAEC Bundle::ObjectListType
Candidate_Indicators: MAEC Bundle::CandidateIndicatorListType
Collections: MAEC Bundle::CollectionsType
0..1

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for the `MAEC_Bundle`. The ID SHOULD follow the pattern defined in Section 1.4.
schema_version	`string`	1	Specifies the version of the MAEC Bundle schema that the document has been written in and that SHOULD be used for validation. The fixed value is '*4.1*.'
defined_subject	`boolean`	1	Specifies whether the fields that describe the properties of the malware instance characterized by the `MAEC_Bundle` are included inside this `MAEC_Bundle` (via the `Malware_Instance_Object_Attributes` field) or elsewhere (such as a `Malware_Subject` in a `MAEC_Package`).
content_type	`BundleContentTypeEnum`	0..1	Specifies the general type of content contained in the `MAEC_Bundle`, e.g., '*static analysis tool output*,' '*dynamic analysis tool output*,' etc.
timestamp	`dateTime`	0..1	Specifies the date/time that the `MAEC_Bundle` was generated.
Malware_Instance_Object_Attributes	`cybox:ObjectType`	0..1	Characterizes the properties of the malware instance (e.g., its MD5 hash) whose capabilities, behaviors, actions, objects, process tree, and candidate indicators are characterized in this `MAEC_Bundle`. This is equivalent to the `Malware_Instance_Object_Attributes` field inside of a `Malware_Subject` in the `MAEC_Package`, and it is therefore only REQUIRED if this `MAEC_Bundle` is to be used in a stand-alone fashion, i.e., without an accompanying `MAEC_Package` and with the `defined_subject` field set to '*true*'.
AV_Classifications	`AVClassificationsType`	0..1	Contains 1-n `AVClassificationType` fields, which capture any Anti-Virus scanner tool classifications of the

			malware instance.
Process_Tree	`ProcessTreeType`	0..1	Specifies the observed process tree of execution for the malware instance, along with references to any corresponding actions performed by each process, if applicable.
Capabilities	`CapabilityListType`	0..1	Contains 1-n fields of `CapabilityType`, which function as the MAEC representation for any capabilities that were observed for the malware instance.
Behaviors	`BehaviorListType`	0..1	Contains 1-n fields of `BehaviorType`, which function as the MAEC representation for any behaviors that were observed for the malware instance.
Actions	`ActionListType`	0..1	Contains 1-n fields of `ActionType`, which function as the MAEC representation for any lower-level actions that were observed for the malware instance.
Objects	`ObjectListType`	0..1	Contains 1-n fields of `ObjectType`, which function as the MAEC representation for any objects associated with the malware instance.
Candidate_Indicators	`CandidateIndicatorListType`	0..1	Contains 1-n fields of `CandidateIndicatorType`, which function as the MAEC representation of any candidate indicators associated with the malware instance.
Collections	`CollectionsType`	0..1	Contains the `Collection` fields for behaviors, actions, objects, and candidate indicators.

2.1.2 BundleContentTypeEnum

The `BundleContentTypeEnum` is a non-exhaustive enumeration of the general types of content that a `MAEC_Bundle` can contain.

Enumeration Value	Description
dynamic analysis tool output	Specifies that the `MAEC_Bundle` primarily captures some form of dynamic analysis tool output, such as from a sandbox.

static analysis tool output	Specifies that the MAEC_Bundle primarily captures some form of static analysis tool output, such as from a packer detection tool.
manual analysis output	Specifies that the MAEC_Bundle primarily captures some form of manual analysis output, which may or may not involve the use of tools.
extracted from subject	Specifies that the MAEC_Bundle primarily captures some data that extracted from the malware instance, such as some PE Header fields.
mixed	Specifies that the MAEC_Bundle captures some mixed forms of analysis or tool output for the malware instance, such as both dynamic and static analysis tool output.
other	Specifies that the MAEC_Bundle captures some other form of analysis or tool output that is not represented by the other enumeration values.

2.2 Malware Instance Object Attributes

The Malware_Instance_Object_Attributes field characterizes the properties (e.g., a file name and MD5 hash) and thus identity of the malware instance for which capabilities, behaviors, actions, objects, the process tree, and candidate indicators are characterized in the MAEC_Bundle. This field is equivalent to the Malware_Instance_Object_Attributes field inside of a Malware_Subject in the MAEC_Package, and it is therefore only REQUIRED if this MAEC_Bundle is to be used in a stand-alone fashion, i.e., without an accompanying MAEC_Package. In this case, the defined_subject field on the MAEC_Bundle MUST be set to '*true*' and the Malware_Instance_Object_Attributes field in the MAEC_Bundle SHOULD be used to characterize the fields of the object that represents the malware instance. Please see Section 2.2.2 in [SPEC$_P$] for discussion on how the Malware_Instance_Object_Attributes field can be used in a MAEC_Package.

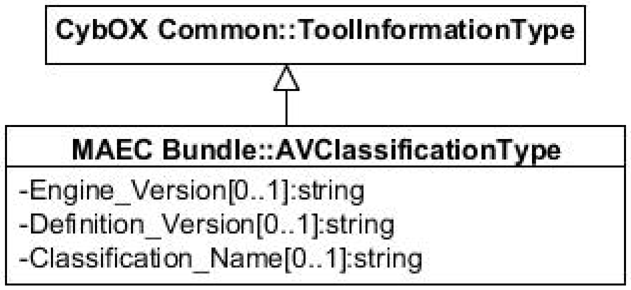
The Malware_Instance_Object_Attributes field is of type cybox:ObjectType, which will not be defined here (see [CYBOX]). While the id and idref fields of the CybOX ObjectType are OPTIONAL and have no required syntax, when the ObjectType is used in MAEC, the id field SHOULD always be used. The recommended format for the id field is given in Section 1.4.

2.3 AV Classification

The AV_Classification field of type AVClassificationType captures information relating to anti-virus (AV) scanner classifications for a malware instance captured in the MAEC_Bundle or MAEC_Package.

2.3.1 AVClassificationType

The `AVClassificationType` characterizes AV-classification related data and extends the CybOX Common `ToolInformationType`. The extended fields are listed below.


CybOX Common::ToolInformationType
MAEC Bundle::AVClassificationType
-Engine_Version[0..1]:string
-Definition_Version[0..1]:string
-Classification_Name[0..1]:string

Field	Type	Multiplicity	Description
Engine_Version	`string`	0..1	Captures the version of the AV engine used by the AV scanner tool that assigned the classification to the malware instance.
Defintion_Version	`string`	0..1	Captures the version of the AV definitions used by the AV scanner tool that assigned the classification to the malware instance.
Classification_Name	`string`	0..1	Captures the classification assigned to the malware instance by the AV scanner tool characterized in the CybOX `Vendor` and (Product) `Name` fields.

2.4 Process Tree

The `Process_Tree` field of type `ProcessTreeType` specifies the observed process tree of execution for the malware instance, along with references to any corresponding `Action` entities, if applicable.

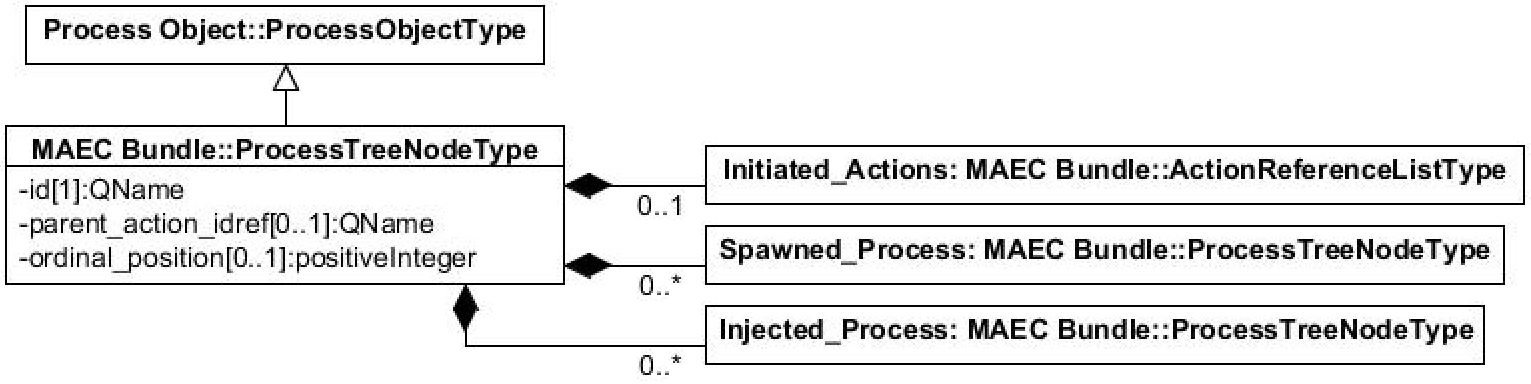
2.4.1 ProcessTreeType

The `ProcessTreeType` captures the process tree for the malware instance, including the parent process and processes spawned by it, along with any actions initiated by each process.

Field	Type	Multiplicity	Description
Root_Process	`ProcessTreeNodeType`	1	Captures the root process in the process tree.

2.4.2 ProcessTreeNodeType

The `ProcessTreeNodeType` captures a single process, or node, in the process tree. It extends the CybOX `ProcessObj:ProcessObjectType`. The extended fields are listed below.


Process Object::ProcessObjectType
MAEC Bundle::ProcessTreeNodeType
-id[1]:QName
-parent_action_idref[0..1]:QName
-ordinal_position[0..1]:positiveInteger
0..1
Initiated_Actions: MAEC Bundle::ActionReferenceListType
0..*
Spawned_Process: MAEC Bundle::ProcessTreeNodeType
0..*
Injected_Process: MAEC Bundle::ProcessTreeNodeType

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for the process node. The ID SHOULD follow the format described in Section 1.4.
parent_action_idref	`QName`	0..1	Specifies the ID of the `Action` that created or injected the process. The IDREF SHOULD follow the pattern defined in Section 1.4.
ordinal_position	`positiveInteger`	0..1	Specifies the ordinal position of the process with respect to other processes spawned or injected by the malware.
Initiated_Actions	`ActionReferenceListType`	0..1	Captures, via references, the `Action` fields (found inside the top-level `Actions` field, or an `Action_Collection` inside the top-

			level `Collections` field) initiated by the process.
Spawned_Process	`ProcessTreeNodeType`	0..*	Captures a single process spawned by this process.
Injected_Process	`ProcessTreeNodeType`	0..*	Captures a single process that was injected by this process.

2.5 Capability

The `Capability` field of type `CapabilityType` provides a standard way of capturing the set of high-level capabilities that a malware instance possesses. Examples of Capabilities include anti-detection, command and control, and privilege escalation.

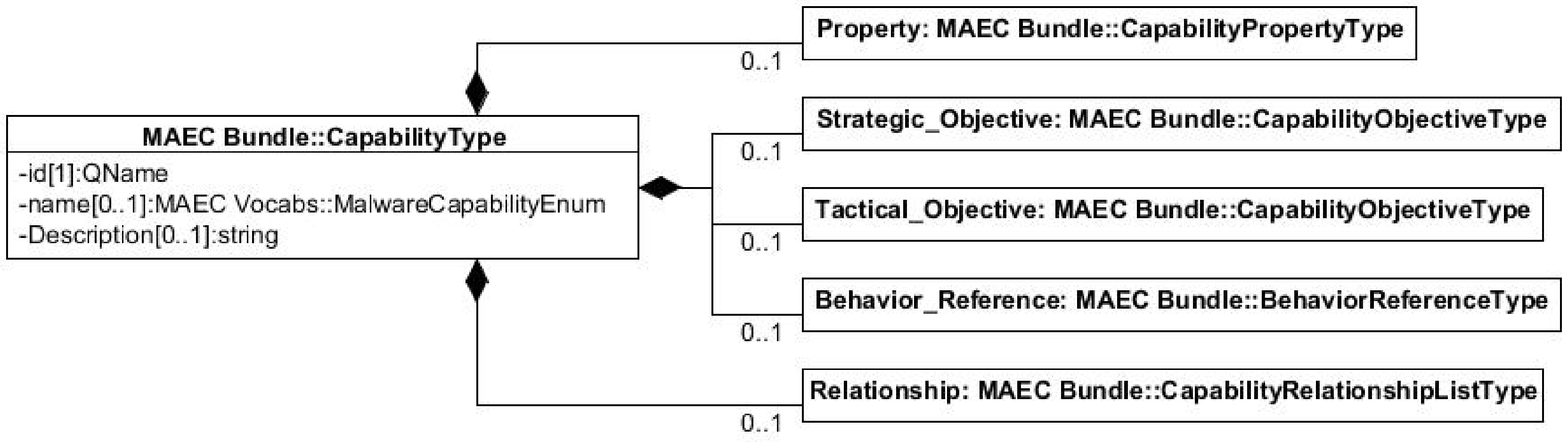
In addition, Strategic and Tactical Objectives have been defined for each Capability to more granularly capture the details of the Capability. More explicitly, a Capability can have one or more Strategic Objectives that the Capability attempts to carry out, and in a similar fashion, a Strategic Objective can have one or more Tactical Objectives. For example, a malware instance may possess a “persistence” Capability, which is further refined by having a Strategic Objective of “persist to continuously execute on system.” This Strategic Objective is in turn refined by having a Tactical Objective of “persist after system reboot.”

While Capabilities are intended to convey what a malware instance is capable of doing, there exists a clear link between Capabilities (i.e., “what” the malware is capable of doing) and the concrete ways they are implemented. We have supported this in MAEC by allowing for the linking between a Capability and/or one of its Strategic or Tactical Objectives with one or more MAEC Behaviors. These Behaviors in turn represent a particular implementation of a Capability or Strategic or Tactical Objective in the malware instance.

As detailed in [SPEC$_V$], “MAEC Default Vocabularies Specification,” individual vocabularies have been defined for Capabilities and for all Strategic and Tactical Objectives corresponding to a particular Capability. For some Capabilities, default vocabularies for properties pertaining to the Capability have also been defined.

2.5.1 CapabilityType

The `CapabilityType` is one of the foundational MAEC types and serves as a method for the characterization of capabilities possessed by malware.


MAEC Bundle::CapabilityType
-id[1]:QName
-name[0..1]:MAEC Vocabs::MalwareCapabilityEnum
-Description[0..1]:string
Property: MAEC Bundle::CapabilityPropertyType
Strategic_Objective: MAEC Bundle::CapabilityObjectiveType
Tactical_Objective: MAEC Bundle::CapabilityObjectiveType
Behavior_Reference: MAEC Bundle::BehaviorReferenceType
Relationship: MAEC Bundle::CapabilityRelationshipListType
0..1
0..1
0..1
0..1
0..1

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for this `Capability`. The ID SHOULD follow the pattern defined in Section 1.4.
name	`maecVocabs: MalwareCapabilityEnum`	0..1	Specifies the name of the `Capability`. It uses the '*MalwareCapabilityEnum-1.0*' enumeration from the MAEC Vocabularies schema.
Description	`string`	0..1	Specifies a basic textual description of the `Capability`.
Property	`CapabilityPropertyType`	0..*	Specifies a single property of the `Capability` as a key/value pair. More than one property can be specified via multiple occurrences of this field.
Strategic_Objective	`CapabilityObjectiveType`	0..*	Specifies a single strategic objective that the `Capability` attempts to achieve. A `Strategic_Objective` is a more granular way of capturing the `Capabilities` present in the malware instance. More than one

			Strategic_Objective can be specified via multiple occurrences of this field.
Tactical_Objective	CapabilityObjectiveType	0..*	Specifies a single tactical objective that the Capability attempts to achieve, typically in the context of a broader Strategic_Objective. A Tactical_Objective can be considered as a way of expounding upon strategic objectives to capture the Capabilities of the malware instance in more detail. More than one Tactical_Objective can be specified via multiple occurrences of this field.
Behavior_Reference	BehaviorReferenceType	0..*	Specifies a reference to a Behavior that serves as an implementation of the Capability. For Behaviors that serve as implementations of specific strategic or tactical objectives, the Behavior_Reference field under the Strategic_Objective or Tactical_Objective fields should be used, respectively. More than one Behavior can be referenced via multiple occurrences of this field.
Relationship	CapabilityRelationshipType	0..*	Specifies any relationships between this Capability and any other Capabilities. More than one Relationship can be specified via multiple occurrences of this field.

2.5.2 CapabilityPropertyType

The CapabilityPropertyType captures a single property of a Capability or Capability Objective.

Field	Type	Multiplicity	Description
Name	cyboxCommon: ControlledVocabularyStringType	0..1	Specifies the name of the property being captured. The default vocabulary type for a property of Capability *X* field is the MAEC *'XPropertiesVocab-1.0.'*

Value	`cyboxCommon: StringObjectPropertyType`	0..1	Specifies the value of the property being captured.

2.5.3 CapabilityObjectiveType

The `CapabilityObjectiveType` captures details of a `Strategic_Objective or Tactical_Objective` field that is associated with a `Capability`.

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for the `Strategic_Objective` or `Tactical_Objective` field. The ID SHOULD follow the pattern defined in Section 1.4.
Name	`cyboxCommon: ControlledVocabularyStringType`	0..1	Specifies the name of the `Capability` objective. There are several default vocabularies for this use included in the MAEC Default Vocabularies schema. The default vocabulary type for use with a `Strategic_Objective` or `Tactical_Objective` field associated with `Capability` *X* is the MAEC '*XStrategicObjectivesVocab-1.0*' or '*XTacticalObjectivesVocab-1.0*', respectively.
Description	`string`	0..1	Specifies a basic textual description of the `Capability` objective.
Property	`CapabilityPropertyType`	0..*	Permits the capture of a single property of the `Capability` objective, as a key/value pair. More than one property can be specified via multiple occurrences of this field.
Behavior_Reference	`BehaviorReferenceType`	0..*	Specifies a reference to a `Behavior` that functions as an implementation of the `Capability` objective. More than one `Behavior` can be referenced via multiple occurrences of this field.
Relationship	`CapabilityObjective RelationshipType`	0..*	Specifies a relationship from the `Capability` objective to one or more other `Capability` objectives. More than one relationship can be specified via multiple occurrences of this

			field.

2.5.4 CapabilityObjectiveRelationshipType

The `CapabilityObjectiveRelationshipType` captures a relationship between a `Capability` objective (a `Strategic_Objective` or `Tactical_Objective`) and one or more other `Capability` objectives.

Field	Type	Multiplicity	Description
Relationship_Type	`cyboxCommon: ControlledVocabularyStringType`	0..1	Specifies the type of relationship being expressed between objectives (either strategic or tactical). The default vocabulary type for use in this field is the MAEC '*CapabilityObjectiveRelationshipTypeVocab-1.0.*'
Objective_Reference	`CapabilityObjectiveReferenceType`	1..*	References a single `Capability` objective (either strategic or tactical) in the relationship. More than one objective can be referenced via multiple occurrences of this field.

2.5.5 CapabilityRelationshipType

The `CapabilityRelationshipType` captures a relationship between a `Capability` and one or more other `Capabilities.`

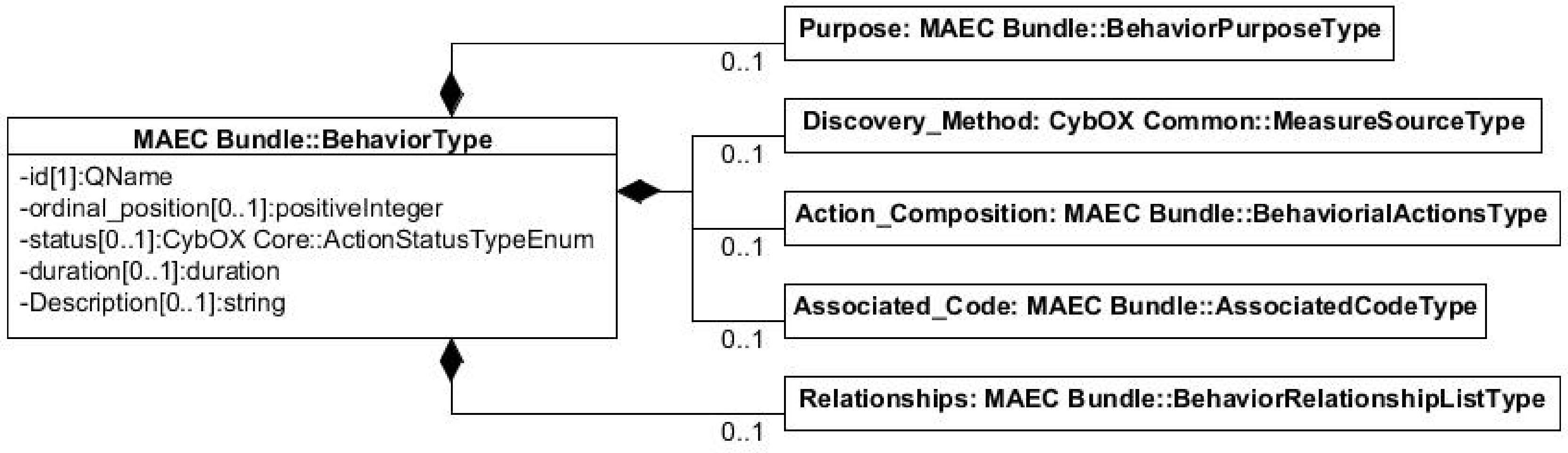
Field	Type	Multiplicity	Description
Relationship_Type	`cyboxCommon: ControlledVocabularyStringType`	0..1	Specifies the type of relationship between Capabilities. A default vocabulary type has not yet been defined in v4.1.
Capability_Reference	`CapabilityReferenceType`	1..*	Specifies a reference to a single `Capability` in the relationship. More than one `Capability` can be referenced via multiple occurrences of this field.

2.6 Behavior

The `Behavior` field of type `BehaviorType` can be thought of as capturing the intent behind groups of `Action` entities and is therefore used to represent distinct portions of higher-level malware functionality. Thus, while a malware instance may perform some multitude of actions, it is likely that these actions represent only a few distinct behaviors. Some examples include vulnerability exploitation, email address harvesting, the disabling of a security service, etc. `Behavior` entities can represent discrete components of malware functionality at a level that is useful for analysis, triage, and detection.

2.6.1 BehaviorType

The `BehaviorType` is one of the foundational MAEC types and serves as a method for the characterization of malicious behaviors found or observed in malware.


MAEC Bundle::BehaviorType
-id[1]:QName
-ordinal_position[0..1]:positiveInteger
-status[0..1]:CybOX Core::ActionStatusTypeEnum
-duration[0..1]:duration
-Description[0..1]:string
Purpose: MAEC Bundle::BehaviorPurposeType
0..1
Discovery_Method: CybOX Common::MeasureSourceType
0..1
Action_Composition: MAEC Bundle::BehaviorialActionsType
0..1
Associated_Code: MAEC Bundle::AssociatedCodeType
0..1
Relationships: MAEC Bundle::BehaviorRelationshipListType
0..1

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for this `Behavior`. The ID SHOULD

			follow the pattern defined in Section 1.4.
ordinal_position	`positiveInteger`	0..1	Specifies the ordinal position of the `Behavior` with respect to the execution of the malware.
status	`cybox:ActionStatusTypeEnum`	0..1	Specifies the execution status of the `Behavior` being characterized.
duration	`duration`	0..1	Specifies the duration of the `Behavior`. One way to derive such a value may be to calculate the difference between the timestamps of the first and last `Actions` that compose the `Behavior`.
Purpose	`BehaviorPurposeType`	0..1	Specifies the intended purpose of the `Behavior`. Because a `Behavior` is not always successful, and may not be fully observed, this is meant as way to state the nature of the `Behavior` apart from its constituent `Action` entities.
Description	`string`	0..1	Specifies a prose textual description of the `Behavior`.
Discovery_Method	`cyboxCommon:MeasureSourceType`	0..1	Specifies the method used to discover the `Behavior`.
Action_Composition	`BehavioralActionsType`	0..1	Captures the `Action` entities that compose the `Behavior`.
Associated_Code	`AssociatedCodeType`	0..1	Specifies any code snippets that are associated, or are likely associated, with the `Behavior`.
Relationships	`BehaviorRelationshipListType`	0..1	Specifies any relationships between this `Behavior` and any other `Behaviors`.

2.6.2 BehaviorPurposeType

The B`ehaviorPurposeType` captures the purpose behind a `Behavior`.

Field	Type	Multiplicity	Description
Description	`string`	0..1	Contains a prose text description of the purpose of the `Behavior`, whether it was successful or not.
Vulnerability_Exploit	`VulnerabilityExploitType`	0..1	Characterizes any vulnerability (known or unknown) that a `Behavior` may have attempted to exploit.

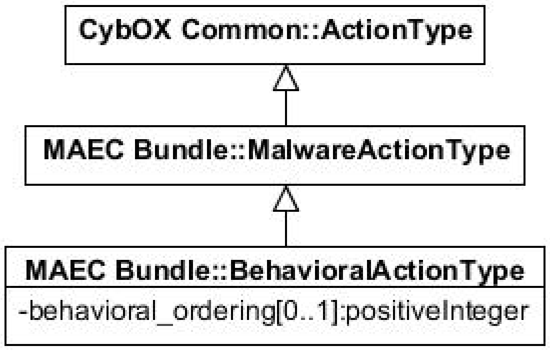
2.6.3 BehavioralActionsType

The BehavioralActionsType is intended to capture the Action entities or Action_Collection entities that make up a Behavior.

Field	Type	Multiplicity	Description
Action_CollectionA	ActionCollectionType	A(1)| B(1)| C(1)| D(1)	Specifies an Action_Collection that is part of the behavioral composition.
ActionB	BehavioralActionType		Specifies a single Action that is part of the behavioral composition.
Action_ReferenceC	BehavioralActionReferenceType		Specifies a reference to a single Action that is part of the behavioral composition.
Action_Equivalence_ReferenceD	BehavioralActionEquivalenceReferenceType		Specifies a reference to a single Action equivalence that is part of the behavioral composition.

2.6.4 BehavioralActionType

The BehavioralActionType defines an Action field that can be used as part of a Behavior. It extends the MAEC MalwareActionType, which in turn extends the CybOX ActionType. The extended field is listed below.


CybOX Common::ActionType
MAEC Bundle::MalwareActionType
MAEC Bundle::BehavioralActionType
-behavioral_ordering[0..1]:positiveInteger

Field	Type	Multiplicity	Description
behavioral_ordering	`positiveInteger`	0..1	Defines the ordering of the `Action` with respect to the other `Actions` that make up the `Behavior`. So an `Action` with a `behavioral_ordering` of '*1*' would come before an `Action` with a `behavioral_ordering` of '*2*', etc.

2.6.5 BehaviorRelationshipType

The `BehaviorRelationshipType` serves to characterize relationships between `Behavior` entities.

Field	Type	Multiplicity	Description
type	`restriction of cyboxVocabs: ActionRelationshipTypeEnum-1.0`	0..1	Specifies the nature of the relationship between `Behaviors` that is being captured. The original enumeration is restricted to `*Preceded_By*', `*Followed_By*', `*Related_To*', and '*Dependent_On*'.
Behavior_Reference	`BehaviorReferenceType`	1..*	Specifies a reference to a single `Behavior` in the relationship. More than one `Behavior` can be referenced via multiple occurrences of this field.

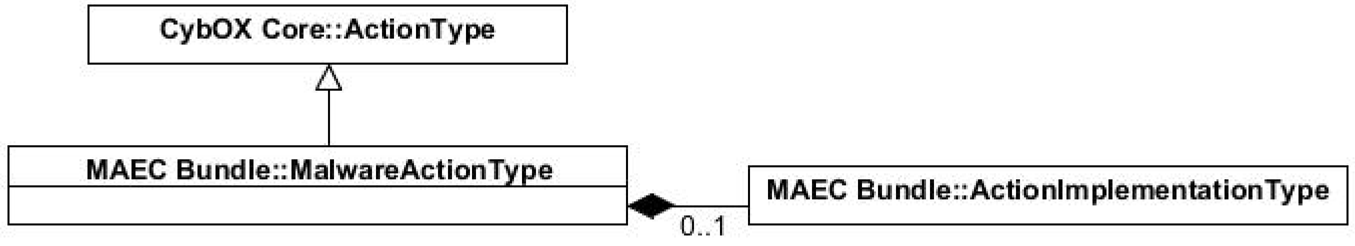
2.7 Action

`Action` entities of type `MalwareActionType` can be thought of as system state changes and similar operations that represent the fundamental low-level functionality of malware. Some examples include the creation of a file, deletion of a registry key, and the sending of some data on a socket.

2.7.1 MalwareActionType

The `MalwareActionType` is one of the foundational MAEC types and serves as a method for the characterization of `Action` entities found or observed in malware. The `MalwareActionType` extends the CybOX `ActionType`. The extended field is listed below. While the `id` and `idref` fields of the CybOX `ActionType` are OPTIONAL and have no required syntax, when The

ActionType is used in MAEC, the id or idref field SHOULD always be used. The MAEC-recommended format for the id field is provided in Section 1.4.


CybOX Core::ActionType
MAEC Bundle::MalwareActionType
MAEC Bundle::ActionImplementationType
0..1

Field	Type	Multiplicity	Description
Implementation	ActionImplementationType	0..1	Serves to capture fields that are relevant to how the Action is implemented in the malware, such as the specific API call that was used.

2.7.2 ActionImplementationType

The ActionImplementationType serves as a method for the characterization of action Implementation entities. Currently supported are implementations achieved through API function calls and abstractly defined code snippets.

Field	Type	Multiplicity	Description
id	QName	0..1	Specifies a unique ID for this action Implementation. The ID SHOULD follow the pattern defined in Section **Error! Reference source not found.**1.4.
type	ActionImplementationTypeEnum	1	Specifies the type of action Implementation being characterized in this field.
Compatible_Platforms	PlatformListType	0..1	Specifies the specific platform(s) that the Action is compatible with, or in other words, capable of being successfully executed on.
API_Call$_A$	APICallType	A(0..1)|B(0..*)	Allows for the characterization of a system-level API call

			that was used to implement the `Action`. Software typically must make use of such calls to talk to hardware and perform system-specific functions.
Code$_B$	`CodeObj:CodeObjectType`		Contains any form of code that was used to implement the `Action`.

2.7.3 ActionImplementationTypeEnum

The `ActionImplementationTypeEnum` represents an enumeration of action `Implementation` types.

Enumeration Value	Description
api call	Specifies that the action was implemented using some particular API call, details of which MAY be captured in the `API_Call` field.
code	Specifies that the `Action` was implemented using some particular code snippet, details of which MAY be captured in the `Code` field.

2.8 Object

An `Object` field captures the characteristics of a specific cyber-relevant entity (e.g., a file, a registry key, or a process). Note that a MAEC `Object` is of type `cybox:ObjectType`, which will not be defined here (see [CYBOX]), but MAEC-specific types related to Objects are defined in the Collections, Reference Types, and List Types sections below.

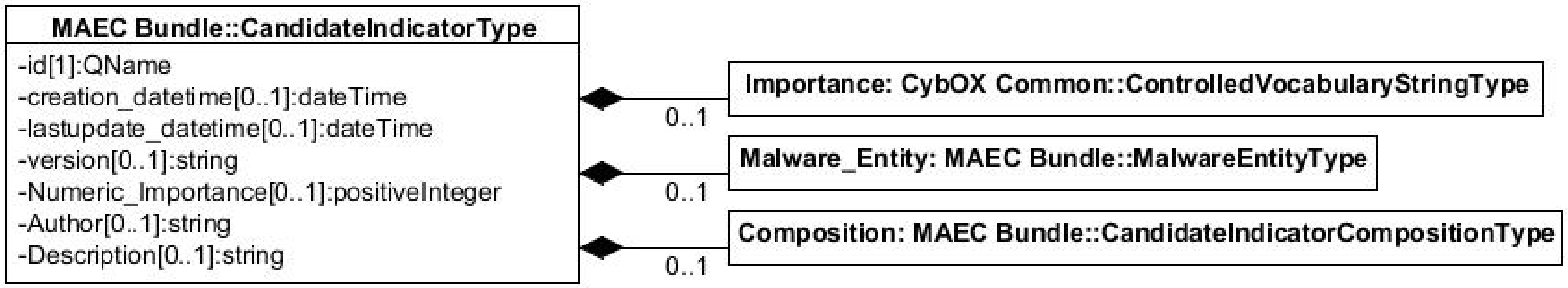
While the `id` and `idref` fields of the CybOX `ObjectType` are OPTIONAL and have no required syntax, when `ObjectType` is used in MAEC, the `id` field SHOULD always be used. Instead of using the `idref` field for referencing existing `Object` entities in the MAEC document, we recommend using the MAEC-specific `ObjectReferenceType`, defined in Section 2.12.7 below. The recommended format for the `id` field is given in Section 1.4.

2.9 Candidate Indicator

A MAEC entity-based `Candidate_Indicator` field of type `CandidateIndicatorType` captures the particular components that may signify the presence of the malware instance on a host system or network.

2.9.1 CandidateIndicatorType

The `CandidateIndicatorType` defines a MAEC entity-based `Candidate_Indicator`.


MAEC Bundle::CandidateIndicatorType
-id[1]:QName
-creation_datetime[0..1]:dateTime
-lastupdate_datetime[0..1]:dateTime
-version[0..1]:string
-Numeric_Importance[0..1]:positiveInteger
-Author[0..1]:string
-Description[0..1]:string
0..1
Importance: CybOX Common::ControlledVocabularyStringType
0..1
Malware_Entity: MAEC Bundle::MalwareEntityType
0..1
Composition: MAEC Bundle::CandidateIndicatorCompositionType

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for this `Candidate_Indicator`. The ID SHOULD follow the pattern defined in Section 1.4.
creation_datetime	`dateTime`	0..1	Specifies the date/time that the `Candidate_Indicator` was created.
lastupdate_datetime	`dateTime`	0..1	Specifies the last date/time that the `Candidate_Indicator` was updated.
version	`string`	0..1	Specifies the version of the `Candidate_Indicator`.
Importance	`cyboxCommon: ControlledVocabularyStringType`	0..1	Specifies the relative importance of the `Candidate_Indicator`. The default vocabulary type is the MAEC *'ImportanceTypeVocab-1.0.'*
Numeric_Importance	`positiveInteger`	0..1	Specifies the specific numeric importance of the `Candidate_Indicator`.
Author	`string`	0..1	Specifies the author of the `Candidate_Indicator`.
Description	`string`	0..1	Provides a brief description of the `Candidate_Indicator`.
Malware_Entity	`MalwareEntityType`	0..1	Specifies the particular malware entity that the

			`Candidate_Indicator` is written against, whether it be a malware instance, family, etc.
Composition	`CandidateIndicatorComposition Type`	1	Specifies the actual observables that the `Candidate_Indicator` is composed of, via a reference to one or more MAEC entities contained in the `MAEC_Bundle`.

2.9.2 CandidateIndicatorCompositionType

The `CandidateIndicatorCompositionType` captures the composition of a `Candidate_Indicator`, via references to any corresponding MAEC entities contained in the `MAEC_Bundle`.

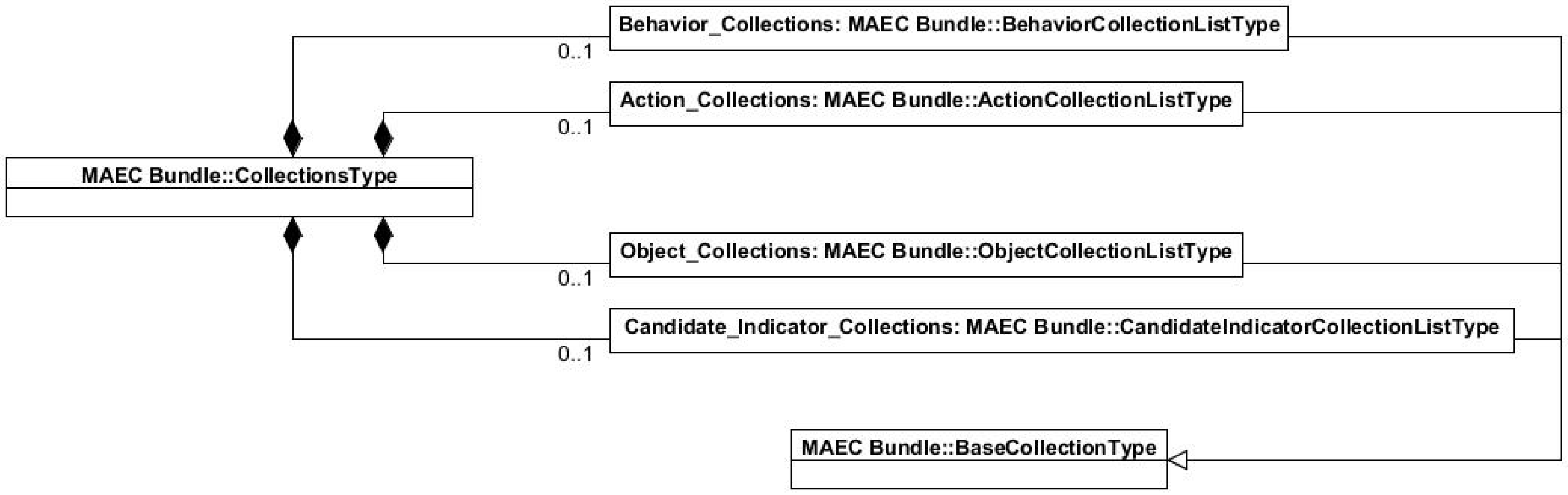
Field	Type	Multiplicity	Description
operator	`cybox:OperatorTypeEnum`	0..1	Specifies the boolean operator for this level of the `Candidate_Indicator's` composition.
Behavior_Reference$_A$	`BehaviorReferenceType`	A(0..1)| B(0..1)| C(0..1)	Specifies a reference to a single `Behavior` in the `MAEC_Bundle` that is part of the `Candidate_Indicator's` composition.
Action_Reference$_B$	`cybox:ActionReferenceType`		Specifies a reference to a single `Action` in the `MAEC_Bundle` that is part of the `Candidate_Indicator's` composition.
Object_Reference$_C$	`ObjectReferenceType`		Specifies a reference to a single `Object` in the `MAEC_Bundle` that is part of the `Candidate_Indicator's` composition.
Sub_Composition	`CandidateIndicatorCompositionType`	0..*	Captures any sub-compositions in this `Candidate_Indicator`, for expressing more complex `Candidate_Indicators`.

2.10 Collections

The `Collections` field of type `CollectionsType` contains the collection field types for `Behavior`, `Action`, `Object`, and `Candidate_Indicator` entities. Because the associated collection types are particular to being part of a collection, they are listed in this section, rather than in the functional groupings section (e.g., the `BehaviorCollectionType` is listed below instead of in the Behavior section (Section 2.6)).

2.10.1 CollectionsType

The `CollectionsType` captures the various types of MAEC field collections. As shown in the UML, each of the different collection list types extend the MAEC Bundle `BaseCollectionType`.


Behavior_Collections: MAEC Bundle::BehaviorCollectionListType
0..1
Action_Collections: MAEC Bundle::ActionCollectionListType
0..1
MAEC Bundle::CollectionsType
Object_Collections: MAEC Bundle::ObjectCollectionListType
0..1
Candidate_Indicator_Collections: MAEC Bundle::CandidateIndicatorCollectionListType
0..1
MAEC Bundle::BaseCollectionType

Field	Type	Multiplicity	Description
Behavior_Collections	`BehaviorCollectionListType`	0..1	Captures any collections of MAEC `Behaviors` in the `MAEC_Bundle`.
Action_Collections	`ActionCollectionListType`	0..1	Captures any collections of MAEC `Actions` in the

			MAEC_Bundle.
Object_Collections	ObjectCollectionListType	0..1	Captures any collections of MAEC Objects in the MAEC_Bundle.
Candidate_Indicator_Collections	CandidateIndicatorCollectionListType	0..1	Captures any collections of MAEC Candidate_Indicators in the MAEC_Bundle.

2.10.2 BaseCollectionType

The BaseCollectionType is the base type for other MAEC collection types.

Field	Type	Multiplicity	Description
name	string	0..1	Specifies the name of the Collection.
Affinity_Type	string	0..1	Provides an abstract way of characterizing how the Object entities in a Collection are related.
Affinity_Degree	string	0..1	Intended to provide an abstract way of characterizing the degree to which the Object entities in a Collection are related.
Description	string	0..1	Contains a textual description of the Collection.

2.10.3 BehaviorCollectionType

The BehaviorCollectionType provides a mechanism for characterizing collections of behaviors. It extends the MAEC Bundle BaseCollectionType (defined in Section 2.10.2). The BehaviorListType is defined in Section 2.13.6.

Field	Type	Multiplicity	Description
id	QName	1	Specifies a unique ID for this Behavior_Collection. The ID SHOULD follow the pattern defined in Section 1.4.
Purpose	string	0..1	States the intended purpose of the collection of Behavior entities. Because Behaviors are not always successful, and may not be fully observed, this is meant as way of abstracting the nature of the collection of Behaviors away from its constituent Actions.

Behavior_List	`BehaviorListType`	1	Specifies a list of `Behaviors` that make up the collection.

2.10.4 ActionCollectionType

The `ActionCollectionType` provides a method for characterizing collections of `Actions`. This can be useful for organizing `Action` entities that may be related and where the exact relationship is unknown, as well as `Actions` whose associated `Behavior` has not yet been established. It extends the `BaseCollectionType` (defined in Section 2.10.2). The `ActionListType` is defined in Section 2.13.2.

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for this `Action_Collection`. The ID SHOULD follow the pattern defined in Section 1.4.
Action_List	`ActionListType`	1	Specifies a list of `Actions` that make up the collection.

2.10.5 ObjectCollectionType

The `ObjectCollectionType` provides a mechanism for characterizing collections of `Objects`. For instance, it can be used to group all of the `Objects` that are associated with a specific `Behavior`. It extends the MAEC Bundle `BaseCollectionType` (defined in Section 2.10.2). The `ObjectListType` is defined in Section 2.13.12.

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for this `Object_Collection`. The ID SHOULD follow the pattern defined in Section 1.4.
Object_List	`ObjectListType`	1	Specifies a list of `Objects` that make up the collection.

2.10.6 CandidateIndicatorCollectionType

The `CandidateIndicatorCollectionType` provides a mechanism for characterizing collections of `Candidate_Indicators`. It extends the MAEC Bundle `BaseCollectionType` (defined in Section 2.10.2). The `CandidateIndicatorListType` is defined in Section 2.13.9.

Field	Type	Multiplicity	Description
id	`QName`	1	Specifies a unique ID for this `Candidate_Indicator_Collection`. The ID SHOULD follow the pattern defined in Section 1.4.
Candidate_Indicator_List	`CandidateIndicatorListType`	1	Specifies a list of `Candidate_Indicators` that make up the collection.

2.11 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 included here because they could be used more generally.

2.11.1 APICallType

The `APICallType` provides a method for the characterization of API calls, including functions and their parameters.

Field	Type	Multiplicity	Description
function_name	`string`	0..1	Contains the exact name of the API function called, e.g., '*CreateFileEx*.'
normalized_function_name	`string`	0..1	Contains the normalized name of the API function called, e.g., '*CreateFile*.'
Address	`hexBinary`	0..1	Contains the code address of the API call in the binary.
Return_Value	`string`	0..1	Contains the return value of the API call.
Parameters	`ParameterListType`	0..1	Captures any name/value pairs of the parameters passed into the API call.

2.11.2 AssociatedCodeType

The `AssociatedCodeType` serves as generic way of specifying any code snippets associated with a MAEC field, such as a `Behavior`.

Field	Type	Multiplicity	Description
Code_Snippet	`CodeObj:CodeObjectType`	1..*	Captures a single snippet of code, via the CybOX `CodeObjectType`.

2.11.3 CVEVulnerabilityType

The `CVEVulnerabilityType` provides a way of referencing specific vulnerabilities that malware exploits or attempts to exploit via a Common Vulnerabilities and Exposures (CVE) identifier. For more information on CVE please see [CVE].

Field	Type	Multiplicity	Description
cve_id	`string`	1	Contains the ID of the CVE that is being referenced, e.g., '*CVE-1999-0002*.'
Description	`string`	0..1	Specifies the textual description of the vulnerability referenced by the cve_id.

2.11.4 MalwareEntityType

The `MalwareEntityType` provides a mechanism for characterizing the particular entity that an indicator or signature is written against, such as a particular malware instance, family, etc.

Field	Type	Multiplicity	Description
Type	`cyboxCommon:ControlledVocabularyStringType`	0..1	Refers to the specific type of malware entity that the indicator or signature is written against. The default vocabulary type for use in this field is the MAEC *'MalwareEntityTypeVocab-1.0*.'
Name	`string`	0..1	Refers to the name of the malware instance, malware family, or malware class that the indicator or signature is written against.
Description	`string`	0..1	Intended to provide a brief description of the entity that the indicator or signature is written against.

2.11.5 ParameterType

The `ParameterType` characterizes function parameters.

Field	Type	Multiplicity	Description
ordinal_position	`positiveInteger`	0..1	Refers to the ordinal position of the parameter with respect to the function where it is used.
name	`string`	0..1	Specifies the name of the parameter.
value	`string`	0..1	Specifies the actual value of the parameter.

2.11.6 VulnerabilityExploitType

The `VulnerabilityExploitType` characterizes any vulnerability that may be exploited by malware through a `Behavior`.

Field	Type	Multiplicity	Description
known_vulnerability	`boolean`	0..1	Specifies whether the vulnerability that the malware is exploiting has been previously identified. If so, it SHOULD be referenced via a CVE ID in the CVE field. If not, the platform(s) targeted by the vulnerability exploitation behavior MAY be specified in the `Targeted_Platforms` field.
CVE	`CVEVulnerabilityType`	0..1	Specifies the CVE ID and description of the vulnerability targeted by the exploit, if available.
Targeted_Platforms	`PlatformListType`	0..1	Specifies the platforms(s) targeted by the vulnerability exploit.

2.12 Referential Types

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

2.12.1 BehavioralActionEquivalenceReferenceType

The `BehavioralActionEquivalenceReferenceType` defines an `Action_Equivalence_Reference` that can be used as part of a `Behavior`. Because `Action_Equivalence_Reference` equates two or more `Actions`, this can be thought of as specifying one of the aforementioned `Actions` as part of the composition of the `Behavior`.

Field	Type	Multiplicity	Description

action_equivalence_idref	`QName`	1	Specifies the ID of an `Action_Equivalence` contained in the same MAEC document as the `Behavior` that utilizes it. The IDREF SHOULD follow the pattern defined in Section 1.4.
behavioral_ordering	`positiveInteger`	0..1	Defines the ordering of the `Action` with respect to the other `Actions` that make up the `Behavior`. For example, an `Action` with a `behavioral_ordering` of '*1*' would come before an `Action` with a `behavioral_ordering` of '*2*', etc.

2.12.2 BehavioralActionReferenceType

The `BehavioralActionReferenceType` defines an `Action` reference that can be used as part of a `Behavior`. It extends the CybOX `ActionReferenceType`.

Field	Type	Multiplicity	Description
behavioral_ordering	`positiveInteger`	0..1	Defines the ordering of the `Action` with respect to the other `Actions` that make up the `Behavior`. So an `Action` with a `behavioral_ordering` of '*1*' would come before an `Action` with a `behavioral_ordering` of '*2*', etc.

2.12.3 BehaviorReferenceType

The `BehaviorReferenceType` serves as a method for referencing existing `Behaviors` contained in the `MAEC_Bundle`.

Field	Type	Multiplicity	Description
behavior_idref	`QName`	1	Specifies the ID of the `Behavior` being referenced; this `Behavior` MUST be present in the current `MAEC_Bundle`. The IDREF SHOULD follow the pattern defined in Section 1.4.

2.12.4 BundleReferenceType

The BundleReferenceType serves as a method for linking to MAEC_Bundle entities embedded in other locations. It MAY be used in a MAEC_Package.

Field	Type	Multiplicity	Description
bundle_idref	QName	1	References the ID of a MAEC_Bundle contained inside the current MAEC document. The IDREF SHOULD follow the pattern defined in Section 1.4.

2.12.5 CapabilityObjectiveReferenceType

The CapabilityObjectiveReferenceType serves as a method for referencing existing Capability objectives (Strategic_Objective or Tactical_Objective fields) contained in a MAEC document.

Field	Type	Multiplicity	Description
objective_idref	QName	1	Specifies the ID of a Capability objective (either a Strategic_Objective or Tactical_Objective) contained inside the current MAEC document. The IDREF SHOULD follow the pattern defined in Section 1.4.

2.12.6 CapabilityReferenceType

The CapabilityReferenceType serves as a method for referencing existing Capability contained in a MAEC document.

Field	Type	Multiplicity	Description
capability_idref	QName	1	Specifies the ID of a Capability contained inside the current MAEC document. The IDREF SHOULD follow the pattern defined in Section 1.4.

2.12.7 ObjectReferenceType

The ObjectReferenceType serves as a method for linking to CybOX Objects embedded in the MAEC_Bundle.

Field	Type	Multiplicity	Description

object_idref	`QName`	1	Specifies the ID of a CybOX `Object` being referenced in the current `MAEC_Bundle`. The IDREF SHOULD follow the pattern defined in Section 1.4.

2.13 List Types

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

2.13.1 ActionCollectionListType

The `ActionCollectionListType` captures a list of `Action_Collections`.

Field	Type	Multiplicity	Description
Action_Collection	`ActionCollectionType`	1..*	Specifies a single collection of `Actions` in the `MAEC_Bundle`.

2.13.2 ActionListType

The `ActionListType` captures a list of `Actions`.

Field	Type	Multiplicity	Description
Action	`MalwareActionType`	1..*	Specifies a single `Action` in the list.

2.13.3 ActionReferenceListType

The `ActionReferenceListType` captures a list of `Action_References`.

Field	Type	Multiplicity	Description
Action_Reference	`cybox:ActionReferenceType`	1..*	Specifies a reference to a single `Action`.

2.13.4 AVClassificationsType

The `AVClassificationsType` captures a list of `AV_Classifications`.

Field	Type	Multiplicity	Description
AV_Classification	AVClassificationType	1..*	Captures a single AV_Classification of the malware instance.

2.13.5 BehaviorCollectionListType

The BehaviorCollectionListType captures a list of Behavior_Collections.

Field	Type	Multiplicity	Description
Behavior_Collection	BehaviorCollectionType	1..*	Specifies a single collection of MAEC Behaviors in the MAEC_Bundle.

2.13.6 BehaviorListType

The BehaviorListType captures a list of Behaviors.

Field	Type	Multiplicity	Description
Behavior	BehaviorType	1..*	Specifies a single MAEC Behavior in the list of Behaviors.

2.13.7 BehaviorRelationshipListType

The BehaviorRelationshipListType captures any relationships between a Behavior and other Behaviors.

Field	Type	Multiplicity	Description
Relationship	BehaviorRelationshipType	1..*	Specifies a single Relationship between a single Behavior and one or more other Behaviors.

2.13.8 CandidateIndicatorCollectionListType

The `CandidateIndicatorCollectionListType` captures a list of `Candidate_Indicator_Collections`.

Field	Type	Multiplicity	Description
Candidate_Indicator_Collection	`CandidateIndicatorCollectionType`	1..*	Specifies a single collection of `Candidate_Indicators` in the `MAEC_Bundle`.

2.13.9 CandidateIndicatorListType

The `CandidateIndicatorListType` captures a list of `Candidate_Indicators`.

Field	Type	Multiplicity	Description
Candidate_Indicator	`CandidateIndicatorType`	1..*	Specifies a single `Candidate_Indicator` in the list.

2.13.10 CapabilityListType

The `CapabilityListType` captures a list of `Capabilities`.

Field	Type	Multiplicity	Description
Capability	`CapabilityType`	1..*	Specifies a single `Capability` in the list and represents a single capability possessed by the malware instance.
Capability_Reference	`CapabilityReferenceType`	1..*	References a single `Capability` defined elsewhere in the MAEC document.

2.13.11 ObjectCollectionListType

The ObjectCollectionListType captures a list of Object_Collections.

Field	Type	Multiplicity	Description
Object_Collection	ObjectCollectionType	1..*	Specifies a single collection of CybOX Objects in the MAEC_Bundle.

2.13.12 ObjectListType

The ObjectListType captures a list of CybOX Objects.

Field	Type	Multiplicity	Description
Object	cybox:ObjectType	1..*	Specifies a single CybOX Object in the list. For use in MAEC, the ID field at the top level of the Object MUST be utilized.

2.13.13 ObjectReferenceListType

The ObjectReferenceListType captures a list of references to CybOX Objects. Note that this type is not currently used inside of a MAEC_Bundle, but is used by a MAEC_Package field (maecPackage:ObjectEquivalenceType). It is listed in the MAEC Bundle data model section for consistency given that the ObjectReferenceType is also defined in this section (Section 2.12.7).

Field	Type	Multiplicity	Description
Object_Reference	ObjectReferenceType	1..*	Specifies a reference to a single CybOX Object.

2.13.14 ParameterListType

The ParametersType captures a list of function Parameter entities.

Field	Type	Multiplicity	Description

Parameter	`ParameterType`	1..*	Specifies a single function `Parameter`.

2.13.15 PlatformListType

The `PlatformListType` captures a list of software or hardware `Platform` entities.

Field	Type	Multiplicity	Description
Platform	`cyboxCommon: PlatformSpecificationType`	1..*	Specifies a single `Platform` in the list via a common platform enumeration ID. Uses `PlatformSpecificationType` from the CybOX Common schema v2.0.1.

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

[W3C0]	W3C Namespaces in XML 1.0 (Third Edition) http://www.w3.org/TR/REC-xml-names/
[W3C1]	W3C Recommendation for Hex-Encoded Binary Data http://www.w3.org/TR/xmlSchema-2/#hexBinary
[W3C2]	W3C Recommendation for Boolean Data http://www.w3.org/TR/xmlSchema-2/#boolean
[W3C3]	W3C Recommendation for Double Data http://www.w3.org/TR/xmlschema-2/#double
[W3C4]	W3C Recommendation for Float Data http://www.w3.org/TR/xmlSchema-2/#float
[W3C5]	W3C Recommendation for Integer Data http://www.w3.org/TR/xmlSchema-2/#integer
[W3C6]	W3C Recommendation for XML Qualified Names http://www.w3.org/TR/xmlSchema-2/#QName
[W3C7]	W3C Recommendation for String Data http://www.w3.org/TR/xmlSchema-2/#string
[W3C8]	W3C Recommendation for unsigned int Data http://www.w3.org/TR/xmlschema-2/#unsignedInt
[W3C9]	W3C Recommendation for URI Data http://www.w3.org/TR/xmlschema-2/#anyURI