The CybOX™ Language Defined Objects Specification

Version 1.0(draft)

Sean Barnum, Robert Martin, Bryan Worrell, Ivan Kirillov, Penny Chase, Desiree Beck, Stelios Melachrinoudis

4/13/2012

The Cyber Observable eXpression (CybOX) is a standardized language, being developed in collaboration with any and all interested parties, for the specification, capture, characterization and communication of events or stateful properties that are observable in the operational domain. A wide variety of high-level cyber security use cases rely on such information including: event management/logging, malware characterization, intrusion detection, incident response/management, attack pattern characterization, etc. CybOX provides a common mechanism (structure and content) for addressing cyber observables across and among this full range of use cases improving consistency, efficiency, interoperability and overall situational awareness. To enable such an aggregate solution to be practical for any single use case, numerous flexibility mechanisms are designed into the language. In particular, almost everything is optional such that any single use case could leverage only the portions of CybOX that are relevant for it (from a single field to the entire language or anything in between) without being overwhelmed by the rest. This document defines the defined object types for the CybOX Language.

Acknowledgements

The authors Sean Barnum, Robert Martin, Bryan Worrell, Ivan Kirillov, Penny Chase, Desiree Beck and Stelios Melachrinoudis wish to thank the CybOX community for its assistance in contributing and reviewing this document.

Trademark Information

Terms of Use

Feedback

The MITRE Corporation welcomes any feedback regarding the CybOX Language Specification. Please send any comments, questions, or suggestions to cybox@mitre.org.[2]

[1] For more information see http://cybox.mitre.org/about/termsofuse.html
[2] For more information about the CybOX Language, please visit http://cybox.mitre.org

Table of Contents

1 Introduction

Information security is a complex function that consumes significant organizational resources, and is growing increasingly difficult to manage. One of the biggest problems is a lack of standardization among the various activities involved including between the sources of security information, and the tools that consume that information, as well as between the various tools themselves. Often, the exchange of security information is time critical, but is hampered by the variety of incompatible formats in which it is represented.

This lack of standardization gives rise to many challenges across the information security community. One such challenge is the ability to effectively understand and communicate observations in the cyber domain as well as meaningful patterns of potential observations that may indicate some sort of relevant event or state. The concept of observable events or properties in the operational cyber realm is a central, underlying element of a wide array of different activities involved in cyber security. Cyber observables are a critical element of event management, attack pattern & threat characterization, cyber threat indicator sharing, attack detection, incident investigation, malware analysis & management, digital forensics, etc.

Without a uniform, standard mechanism for specifying, capturing, characterizing, and communicating these cyber observables, each activity area, each use case, each organization, each sharing community and often each supporting tool vendor is forced to use its own unique approach for representing data that inhibits consistency, efficiency, interoperability, and overall situational awareness. This requires the IT Security Professional to translate the data produced by the various processes and tools in order to map between users and uses and to correlate all of this data in order to obtain a meaningful holistic situational awareness. It may also be necessary for the data to be manually converted into a format that is usable by another tool which can also be a tedious and error-prone process.

What the industry requires is a standardized method for representing cyber observables. The representation of this information must easily facilitate its generation, sharing, consumption and analysis by software tools. The advantage of such a standard is that it will:

- Bring consistency and transparency to cyber observables produced by sensors
- Bring consistency and transparency to the results produced by analysis tools.
- Enable new levels of correlation analysis heuristics
- Assist in the exchange of information between tools.
- Enable holistic exchange of cyber observables between differing activities and use cases
- Enable new levels of integrated situational awareness and operational understanding
- Reduce the need for IT Security Professionals to learn the proprietary languages of each of the processes and tools that they and their partners use, and instead allow them to learn a single language that is understood by all the processes and tools.

This document presents the Defined Objects portion of the CybOX Language as a standard that fulfills these needs and requirements.

1.1 The CybOX Language

The Cyber Observable eXpression (CybOX™) is an international, information security, community standard to promote consistent capture of cyber observable content, and to standardize the transfer of this information across the entire spectrum of security activities, tools and services.

The CybOX Language, developed by a broad spectrum of industry, academia, and government organizations from around the world, standardizes the encoding and communication of high-fidelity information about cyber observables, whether they are dynamic events or stateful measures observable in the operational cyber domain.

The CybOX Language adheres to three overarching principal objectives:

- Develop a common solution for all relevant use cases

  CybOX is not targeted at a single cyber security use case; rather it is intended to be flexible enough to offer a common solution for all cyber security use cases requiring the ability to deal with cyber observables. CybOX is targeted to support a wide range of relevant cyber security domains including: event management, attack pattern & threat characterization, cyber threat indicator sharing, attack detection, incident investigation, malware analysis & management, digital forensics, etc. ***To enable such an aggregate solution to be practical for any single use case, numerous flexibility mechanisms are designed into the language. In particular, almost everything is optional such that any single use case could leverage only the portions of CybOX that are relevant for it (from a single field to the entire language or anything in between) without being overwhelmed by the rest.***

- Develop a solution for both instances of observables as well as potential patterns

  CybOX is also intended to be flexible enough to allow both the ***high-fidelity description of cyber observable instances*** measured in an operational context as well as more ***abstract patterns for potential observables*** that may be targets for observation and analysis apriori. This flexibility has the potential to enable greater synergies between observation and interpretation.

- Develop a solution capable of supporting significant improvements in automation

  By specifying a common structured language mechanism for the cyber observables, the intent is to enable the potential for new levels of detailed ***automatation*** in sharing, mapping, detection and analysis heuristics.

By achieving these objectives the CybOX Language serves as a framework and vocabulary to provide:

- A comprehensive and flexible solution for characterizing cyber observables.
- A standard format that codifies the necessary range of cyber observable characteristics.
- An open alternative to closed, proprietary, and replicated efforts.
- An effort that is supported by a community of security experts, system administrators, and software developers from industry, government, and academia.

All of which leads to a common and structured format that facilitates collaboration and information sharing among the information security community as well as interoperability among security tools.

1.2 Document Conventions

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*.[16]

The following font and font style conventions are used throughout the remainder of this document:

- The `Courier New` font is used for writing constructs in the CybOX Language Data Model. Example: `generator`
- The *'italic, with single quotes'* font is used for noting values for CybOX Language properties. Example: *'does not exist'*

This document uses the concept of namespaces[3] to logically group CybOX constructs throughout both the Data Model section of the document, as well as other parts of the specification. The format of these namespaces is `prefix:element`, where the prefix is the namespace component, and the element is the name of the qualified construct. The following table lists the namespaces used in this document:

Data Model	Namespace	Description	Example(s)
CybOX Core	cybox	The CybOX Core data model that captures all of the foundational constructs used in CybOX.	`cybox:ObservableType`
CybOX Common	Common	The CybOX Common data model that captures all of the common constructs used across the various CybOX object data models	`Common:HashType`

CybOX Objects	<type>Obj	The CybOX Object data models construct representations of observable and stateful information. Each CybOX Object schema has its own defined namespace and can be used as an extension point for other domain-specific or organizational-specific models.	`FileObj:File_Object_Type`
			`MutexObj:Mutex`
			`MemoryObj:Memory_Block`

1.3 Specification Architecture

The CybOX language is defined within a set of specification documents as follows:

- **CybOX Language Core Specification**

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

Specifies the purpose, approach, conventions and usage of the CybOX language as well as the detailed language data models for the language core and set of common types.

- **CybOX Language Defined Objects Specfication**

  Restates some language basics from the CybOX Language Core Specification (to give context to readers of just the CybOX Language Defined Objects Specfication) as well as specifying the detailed language data models for the official set of CybOX defined objects.

- **CybOX Language Use Case Specification (coming soon)**

  Identifies and characterizes in summary the target use cases supported by the CybOX language.

1.4 CybOX Language Versioning Conventions

The accepted convention for CybOX Language versioning defines a single major and minor version that applies to the entire CybOX Language. These major and minor components are what allow changes to the language to be classified as either major or minor. Whenever a modification is made to the CybOX Language, the version of the language must change. Major versions are only needed when a change to the CybOX Language is made that is not backwards compatible. It is possible to introduce new capabilities to existing language constructs or make bug fixes within a minor revision regardless of whether backward compatibility is maintained though compatibility is always targeted. There is also the possibility for addressing critical defects that will result in breaking backward compatibility within a minor revision of the CybOX Language. The CybOX Language versioning convention also defines a single subminor version for the CybOX Language Defined Objects Specification that represents an independently incrementing version counter for any changes to the object specifications that are independent of changes to the Core language specification. Any implementation schemas should have their major and minor versions aligned with the major and minor versions of the corresponding Core language spec and should have a subminor version that represents an independently incrementing version counter for minor schema changes, feature additions or bug fixes occuring between specification releases.

Language Specifications

Core

Major version number = Major language changes

Minor version number = Feature additions and minor language changes (including bug fixes that could break backward compatability)

Objects

Major version number = Aligned with major version number of Core specification

Minor version number = Aligned with minor version number of Core specification

Subminor version number = Independently incrementing version counter (Any changes to object specifications independent of changes to the Core specification)

Implementation Schema Core & Common_Types

Major version number = Aligned with major version number of Core specification

Minor version number = Aligned with minor version number of Core specification

Subminor version number = Bug fixes without backward compatibility issues or between specification releases

Implementation Schema Defined Objects

Major version number = Aligned to major version number of related Core schema

Minor version number = Aligned with minor version number of Core specification

Subminor version number = Minor object schema changes, feature additions and bug fixes

Language Releases

Major version number = Aligned with major version number of Core specification

Minor version number = Aligned with minor version number of Core specification

1.5 CybOX Language Naming Conventions

The CybOX Language utilizes the following naming conventions.

Metadata Field Names

Convention: Lowercase with underscores (e.g. object_state)

Data Field Names

Convention: Capitalized with underscores (e.g. Defined_Object)

Type names

Convention: Camelcase upper start without pretype underscore (e.g. DefinedObjectType)

Enumeration Type names:

Convention: Camelcase upper start without pretype underscore with “Enum” appended (e.g. DefinedObjectTypeEnum)

Attribute Group names:

Convention:	Camelcase upper start without pretype underscore with “Group” appended (e.g. ObjectAttributeGroup)

Object Names

Convention:	Object specification file names: Capitalized with underscores (e.g. Win_Network_Route_Entry_Object)
Convention:	Object specification root element: Capitalized with underscores without trailing “Object” (e.g. Win_Network_Route_Entry)

Namespace names

Convention:	Camelcase upper start with entire object name with removed underscores (e.g. NetworkRouteEntryObject) The exceptions would be Common_Types which would just be “Common” and the core namespace would just be “cybox”

Namespace abbreviations

Convention:	Camelcase upper start with entire object name with removed underscores with Windows abbreviated to Win with Object abbreviated to Obj with Network abbreviated to Net (e.g. WinNetRouteEntryObj)

1.6 Document Structure

This document serves as the specification for the CybOX Language defining requirements, data model, and processing model which is organized into the following section:

- Section 1 – Introduction
- Section 2 –Use Cases for the CybOX Language
- Section 3 – Data Model for the CybOX Langauge Defined Objects
- Section 4 – Representations of the CybOX Language
- Appendix A – Leveraging the CybOX Language Data Model
- Appendix B – Extending the CybOX Language Data Model
- Appendix C – Normative References

- Appendix D – Change Log
- Appendix E –Acronyms

2 Use Cases for the CybOX Language

The following list identifies the key use cases that the CybOX language is targeted to support. These use cases will be further characterized and described within the CybOX Language Use Case Specification. Additional use cases will be documented as they emerge through the continued operational application of CybOX.

- **Use Case Area: Event Management**
  - Producing Event Data
  - Exchanging Event Data
  - Analyzing Event Data
  - Querying Event Data
  - Composing Events
- **Use Case Area: Attack Patterns and Threat Characterization**
  - Characterizing Observable Evidence of Granular Attacker Actions
  - Characterizing Observable Evidence of Attacker Preparatory Probing Techniques
  - Characterizing Observable Evidence of Attacker Obfuscation Techniques
  - Characterizing Observable Evidence of Abstract Attack Patterns
- **Use Case Area: Cyber Threat Indicator Sharing**
  - Generating Cyber Threat Indicators
  - Exchanging Cyber Threat Indicators
- **Use Case Area: Attack Detection**
  - Detecting Dynamic In-Progress Attacks
  - Detecting Past Attacks
- **Use Case Area: Incident Investigation**
  - Correlating Incident Initiation Data
  - Excavating Incident Context
- **Use Case Area: Malware Analysis & Management**
  - Analyzing Malware Instances
  - Analyzing Malware Patterns
  - Hunting Malware Artifacts
  - Metadata Indexing Malware Collections
  - Exchanging Malware Characterizations
- **Use Case Area: Digital Forensics**
  - Conducting Digital Forensic Analysis
  - Managing Evidentiary Process

3 Data Model

3.1 Data Model Conventions

The following conventions are used throughout this data model section.

3.1.1 Property Table Notation

Throughout the data model, tables are used to describe each data type. Each property table will consist of a column of property 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. Values in the type column are either primitive datatypes or other types defied in this document. These values will be cross referenced to the base definition of their types. Below is an example property table.

Table 3-1 Example Property Table

Property	Type	Multiplicity	Description
<PROPERTY NAME>	<DATA TYPE>	0..1	<DESCRIPTION OF THE PROPERTY AND ANY USAGE REQUIREMENTS FOR THE PROPERTY>

3.1.2 Primitive Data Types

The following primitive datatypes are used in the CybOX Language.

- hexBinary – Data of this type conforms to the World Wide Web Consortium (W3C) Recommendation for hex-encoded binary data [1].
- base64Binary – Data of this type conforms to the W3C Recommendation for base-64-encoded binary data [2].
- boolean – Data of this type conforms to the W3C Recommendation for boolean data [3].
- integer – Data of this type conforms to the W3C Recommendation for integer data [4].
- unsigned int – Data of this type represents an unsigned integer value that conforms to the W3C Recommendation for unsigned integer data [5].
- non-negative int – Data of this type represents a non-negative integer value that conforms to the W3C Recommendation for non-negative integer data [6].
- positive int – Data of this type represents a positive integer value that conforms to the W3C Recommendation for positive integer data [7].
- long – Data of this type represents a long integer value that conforms to the W3C Recommendation for long integer data [8].
- unsigned long – Data of this type represents an unsigned long value that conforms to the W3C Recommendation for unsigned long data [9].
- double – Data of this type represents a double value that conforms to the W3C Recommendation for double data [10].

- float – Data of this type represents a float value that confirms to the W3C Recommendation for float data [11].
- time – Data of this type represents a time value that conforms to the W3C Recommendation for time data [12].
- date – Data of this type represents a date value that conforms to the W3C Recommendation for date data [13].
- dateTime – Data of this type represents a date and time value that conforms with the W3C Recommendation for datetime data [14].
- duration – Data of this type represents a duration value that conforms to the W3C Recommendation for duration data [15].
- string – Data of this type conforms to the W3C Recommendation for string data [16].
- QName – Data of this type conforms to the W3C Recommendation for QName data[17]
- URI – Data of this type conforms to the W3C Recommendation for anyURI data [18].

3.1.3 CybOX Primitive Datatype Expansions

The CybOX language within the Common Types data model defines several datatypes to be used for CybOX object attributes in place of language-specific primitive data types. By leveraging a common foundation—`cybox:BaseObjectAttributeType`—each derivation is able to store metadata (e.g., regular expressions, ranges, entropy) to help characterize its stored data.

The following CybOX datatypes have been defined to expand language-specific primitives.

- AnyURIObjectAttributeType
- Base64BinaryObjectAttributeType
- DateObjectAttributeType
- DateTypeObjectAttributeType
- DoubleObjectAttributeType
- DurationObjectAttributeType
- FloatObjectAttributeType
- HexBinaryObjectAttributeType
- IntegerObjectAttributeType
- LongObjectAttributeType
- NameObjectAttributeType
- NonNegativeIntegerObjectAttributeType
- PositiveIntegerObjectAttributeType
- StringObjectAttributeType
- TimeObjectAttributeType
- UnsignedLongObjectAttributeType
- UnsignedIntegerObjectAttributeType

3.1.4 CybOX Identifier Conventions

The CybOX language defines identifier (id) fields as qualified names according to the W3C recommendation for QName data[17] with the added stipulation that the namespace prefix MUST be present.

The CybOX use of the QName type is a colon separated string construct where the nonoptional prefix before the colon is a namespace reference associated with a URI for the defining domain/scope and the postfix after the colon is an identifier string beginning with a letter whose format is specified by the associated namespace domain. Native CybOX content MUST utilize the "cybox" namespace prefix.

Examples:

cybox:guid-fce3cf95-2bc6-45de-b418-c5991e201196

maec:example-obj-1

capec:cybox-59cac3e5-a2bc-481a-9541-adafef920cc9

foo:bar-123

Utilizing this approach, CybOX both ensures global uniqueness of identifiers and enables the flexible use of CybOX content within various different contexts or other information standards that require their own particular identifier syntax.

Currently each specifying domain will define their own format locally. CybOX envisions a future independent registration of valid namespaces and associated domain formats under an organization such as IANA.

3.2 CybOX Object Types

3.2.1 APIObjectType (extends Common:DefinedObjectType)

The APIObjectType type is intended to characterize a specific Application Programming Interface.

Property	Type	Mult	Description
Description	Common: StructuredTextType	0..1	The Description element is intended for use in providing a brief description of the API.
Function_Name	Common: StringObject AttributeType	0..1	The function_name element contains the exact name of the API function called, e.g. CreateFileEx.
Normalized_Function_ Name	Common: StringObject AttributeType	0..1	The normalized_function_name element contains the normalized name of the API function called, e.g. CreateFile.
Platform	Common: CPESpecificationType	0..1	The Platform element specifies the relevant platform for this API, by way of a Common Platform Enumeration (CPE) identifier. For more information on CPE, go to http://cpe.mitre.org. With future releases of CPE we will have ability to cover the full range of platforms that an action implementation is tied into.

Address	Common: HexBinary ObjectAttributeType	0..1	The Address element contains the address of the API call in the binary.

3.2.2 AccountObjectType (extends Common:DefinedObjectType)

The AccountObjectType type is intended to characterize generic accounts.

Property	Type	Mult	Description
disabled	boolean	1..1	The disabled attribute specifies whether or not the account is disabled.
locked_out	boolean	1..1	The locked_out attribute specifies whether or not the account is locked out.
Description	Common: StringObject AttributeType	0..1	The Description element is used for providing a description of the account, if applicable.
Domain	Common: StringObject AttributeType	0..1	The Domain element is used for specifying the domain that the account belongs to.

3.2.3 AddressObjectType (extends Common:DefinedObjectType)

The AddressObjectType is intended to characterize cyber addresses. It is based on the IODEF address element.

Property	Type	Mult	Description
category	AddressObj: CategoryTypeEnum	1..1	The category attribute specifies the address category that is being defined.
is_destination	boolean	1..1	The is_destination attribute specifies if this is a "Destination" address
is_source	boolean	1..1	The is_source attribute specifies if this is a "Source" address
Address_Value	Common: StringObject AttributeType	1..1	The required Address_Value element specifies the actual value of the address.
Ext_Category	Common: StringObject AttributeType	0..1	The Ext_Category element defines a means by which to extend the Category element. For more information please see IETF RFC 5070 (http://www.ietf.org/rfc/rfc5070.txt).
VLAN_Name	Common: StringObject AttributeType	0..1	The VLAN_Name element specifies the name of the Virtual LAN to which the address belongs.
VLAN_Num	Common: IntegerObject AttributeType	0..1	The VLAN_Num element specifies the number of the Virtual LAN to which the address belongs.

3.2.3.1 CategoryTypeEnum

The CategoryTypeEnum type is an enumeration of address types.

Restriction base: NMTOKEN

Enumeration Value	Description
asn	The asn value specifies an identifier for an Autonomous System Number.
atm	The atm value specifies an Asynchronous Transfer Mode address.
cidr	The CIDR value specifies an address in Classless Interdomain Routing notation (the IP address and its associated routing prefix).

e-mail	The e-mail value specifies an e-mail address.
mac	The mac value specifies a system's MAC address.
ipv4-addr	The IPV4-addr value specifies an IPV4 address.
ipv4-net	
ipv4-net-mask	The IPV4-net-mask value specifies an IPV4 bitwise netmask.
ipv6-addr	The IPV4-addr value specifies an IPV6 address.
ipv6-net	
ipv6-net-mask	The IPV6-net-mask value specifies an IPV6 bitwise netmask.
ext-value	

3.2.4 CodeObjectType (extends Common:DefinedObjectType)

The CodeObjectType type is intended to characterize a body of computer code.

Property	Type	Mult	Description
Description	Common: StructuredTextType	0..1	The Description element is intended for use in providing a brief description of the code that is encapsulated in this element.
Type	CodeObj: CodeTypeType	0..1	The type element is intended to provide a way of specifying the type of code being characterized.
Purpose	CodeObj: CodePurposeType	0..1	The type element is intended to provide a way of specifying the purpose or flavor of code being characterized.
Code_Language	CodeObj: CodeLanguageType	0..1	The code_language element refers to the code language used in the code characterized in this element.
Targeted_Platforms	CodeObj: TargetedPlatformsType	0..1	The Targeted_Platforms element specifies a list platforms that this code is targeted for, by way of a Common Platform Enumeration (CPE) identifier. For more information on CPE, go to http://cpe.mitre.org. With future releases of CPE we will have ability to cover the full range of platforms that an action implementation is tied into.
Processor_Family	CodeObj: ProcessorTypeType	0..1	The processor_family element specifies the class of processor that the code snippet is targeting. Possible values: x86-32, x86-64, IA-64, PowerPC, ARM, Alpha, SPARC, z/Architecture, eSi-RISC, MIPS, Motorola 68k, Other.
Discovery_Method	Common: MeasureSourceType	0..1	The Discovery_Method element is intended to characterize the method and/or tool used to discover the code.
Start_Address	Common: HexBinary ObjectAttributeType	0..1	The start_address element can be used to reference the start address of the code, if it was discovered inside a binary.
Code_Segment	string	0..1	The Code_Segment element encompasses any arbitrary code segment in un-encoded (plaintext or binary) format. Code would typically be included here within a CDATA section.
XOR_Pattern	Common: HexBinary ObjectAttributeType	0..1	The xor_pattern element contains a 16-hexadecimal-character hex string, which represents the pattern that the Code_Segment element should be XORed with in order to recover the actual code.

			The default value is 55AA55AA55AA55BB, as specified by IETF RFC 5901.
Code_Segment_XOR	hexBinary	0..1	The Code_Segment_XOR element encompasses any arbitrary code segment of the type specified by the codetype attribute. Its contents should contain the actual code segment XORed with the pattern defined in the xorpattern attribute. This is so that the code contained in the pattern does not trigger IDS, AV, or other signature-based scanners. XOR'd Code would typically be included here within a CDATA section.
Digital_Signatures	CodeObj: DigitalSignaturesType	0..1	The Digital_Signatures element is optional and captures one or more digital signatures for the code.

3.2.4.1 CodeTypeType (restriction Common:BaseObjectAttributeType)

CodeTypeType specifies types of code, via a union of the CodeTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: CodeObj:CodeTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.4.2 CodeTypeEnum

CodeTypeEnum is a (non-exhaustive) enumeration of code types.

Restriction base: string

Enumeration Value	Description
Source_Code	The code represented is in the form of Source Code
Byte_Code	The code represented is in the form of Byte Code
Binary_Code	The code represented is in the form of Binar Code

3.2.4.3 CodePurposeType (restriction Common:BaseObjectAttributeType)

CodePurposeType specifies intended purposes of code, via a union of the CodePurposeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: CodeObj:CodePurposeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.4.4 CodePurposeEnum

CodePurposeEnum is a (non-exhaustive) enumeration of classes of code intended purposes.

Restriction base: string

Enumeration Value	Description

Application_Code	The code represented is intended as application code.
Library_Code	The code represented is intended as library code.
Shellcode	The code represented is intended as shell code.
Exploit_Code	The code represented is intended as exploit code.
Unknown	The code represented is intended for unknown purposes.
Other	The code represented is intended for a purpose other than thos listed in this enumeration.

3.2.4.5 CodeLanguageType (restriction Common:BaseObjectAttributeType)

CodeLanguageType specifies languages of code, via a union of the CodeLanguageEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: CodeObj:CodeLanguageEnum, string

Property	Type	Mult	Description
datatype	`Common:DatatypeEnum`	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.4.6 CodeLanguageEnum

The CodeLanguageEnum simple type is an (non-exhaustive) enumeration of computer code languages.

Restriction base: string

Enumeration Value	Description
C	Indicates the code is written in the C programming language
C++	Indicates the code is written in the C++ programming language
C#	Indicates the code is written in the C# programming language
Java	Indicates the code is written in the Java programming language
JSP	Indicates the code is written in the JSP (Java Server Pages) language
Javascript	Indicates the code is written in the Javascript programming language
ASP.NET	Indicates the code is written in the ASP.NET programming language
SQL	Indicates the code is written in SQL (Standard Query Language)
Python	Indicates the code is written in the Python programming language
Perl	Indicates the code is written in the Perl programming language
PHP	Indicates the code is written in the PHP programming language
SOAP	Indicates the code is written as a SOAP message
Ruby	Indicates the code is written in the Ruby programming language
Shell	Indicates the code is written as a Shell script
PseudoCode	Indicates the code is written as psuedo code
.NET	Indicates the code utilizes the .NET framework
Assembly	Indicates the code is written in an assembly language
XML	Indicates the code is written in XML (eXtensible Markup Language)
HTML	Indicates the code is written in HTML (HyperText Markup Language)
Other	Indicates the code is written in a language not found in this enumeration

3.2.4.7 ProcessorTypeType (restriction Common:BaseObjectAttributeType)

ProcessorTypeType specifies relevant processor families, via a union of the ProcessorTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: CodeObj:ProcessorTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.4.8 ProcessorTypeEnum

The ProcessorTypeEnum simple type is an (non-exhaustive) enumeration of computer processor architectures.

Restriction base: string

Enumeration Value	Description
x86-32	Indicates a x86 32bit processor
x86-64	Indicates a x86 64bit processor
IA-64	Indicates an IA (Intel Itanium) 64bit processor
PowerPC	Indicates a PowerPC processor
ARM	Indicates an ARM processor
Alpha	Indicates an Alpha processor
SPARC	Indicates a SPARC processor
z/Architecture	Indicates a z/Architecture (IBM) processor
eSi-RISC	Indicates an eSi-RISC processor
MIPS	Indicates a MIPS processor
Motorola 68k	Indicates a Motorola 68k processor
Other	Indicates a processor outside of this enumeration

3.2.4.9 TargetedPlatformsType

A list of targeted platforms

Property	Type	Mult	Description
Targeted_Platform	Common: CPESpecificationType	1..∞	The Targeted_Platform element specifies a particular platform that this code is targeted for, by way of a Common Platform Enumeration (CPE) identifier. For more information on CPE, go to http://cpe.mitre.org. With future releases of CPE we will have ability to cover the full range of platforms that an action implementation is tied into.

3.2.4.10 DigitalSignaturesType

A list of digital signatures

Property	Type	Mult	Description
Digital_Signature	Common: DigitalSignatureInfoType	0..1	The Digital_Signature element is optional and captures a single digital signature for the code.

3.2.5 DNSCacheObjectType (extends Common:DefinedObjectType)

The DNSCacheObjectType type is intended to characterize entries in a system's DNS cache.

Property	Type	Mult	Description
DNS_Cache_Entry	DNSCacheObj: DNSCacheEntryType	1..∞	The DNS_Cache_Entry element is intended to characterize a single domain name system cache entry.

3.2.5.1 DNSCacheEntryType

The DNSCacheEntryType type is intended to characterize a single entry in a system's DNS cache.

Property	Type	Mult	Description
DNS_Entry	DNSRecordObj: DNSRecordObjectType	1..1	The DNS_Entry element specifies the relevant DNS entry (including Domain Name and IP Address) for this DNS Cache Entry.
TTL	Common: PositiveIntegerObject AttributeType	0..1	The TTL element specifies the time-to-live value for the DNS cache entry, or in other words the number of seconds before the entry expires.

3.2.6 DNSRecordObjectType (extends Common:DefinedObjectType)

The DNSRecordObjectType type is intended to characterize an individual DNS record.

Property	Type	Mult	Description
Description	Common: StructuredTextType	0..1	The Description element provides a mechanism to specify a structured text description of this DNS_Entry.
Domain_Name	URIObj:URIObjectType	0..1	The Domain_Name element specifies the name of the domain to which the DNS cache entry points.
IP_Address	AddressObj: AddressObjectType	0..1	The IP_Address element specifies the IP address to which the domain name in the DNS cache entry resolves to.
Address_Class	Common: StringObject AttributeType	0..1	The Address_Class element specifies the address class (e.g. IN, TXT, ANY, etc.) for the DNS record
Entry_Type	Common: StringObject AttributeType	0..1	The Entry_Type element specifies the resource record type (e.g. SOA or A) for the DNS record.
Record_Name	Common: StringObject AttributeType	0..1	The Record_Name element is optional and specifies the name for the DNS record.
Record_Type	Common: StringObject AttributeType	0..1	The Record_Type element is optional and specifies the type of the DNS record.
TTL	Common: IntegerObject AttributeType	0..1	The TTL element is optional and specifies the time-to-live for the DNS record.
Flags	Common: HexBinary ObjectAttributeType	0..1	The Flags element is optional and specifies the relevant flags for the DNS record.
Data_Length	Common: IntegerObject AttributeType	0..1	The Data_Length element is optional and specifies the length of raw data to be captured in the Record_Data element.
Record_Data	anyType	0..1	The Record_Data element is optional and enables capture and expression of the raw record data.

3.2.7 DeviceObjectType (extends Common:DefinedObjectType)

The DeviceObjectType type is intended to characterize a specific Device.

Property	Type	Mult	Description
Description	Common: StructuredTextType	0..1	The Description element is intended for use in providing a brief description of the Device.
Device_Type	Common:	0..1	The Device_Type element specifies the type of the

	StringObjectAttributeType		device.
Manufacturer	Common:StringObjectAttributeType	0..1	The Manufacturer element specifies the manufacturer of the device.
Model	Common:StringObjectAttributeType	0..1	The Model element specifies the model identifier of the device.
Serial_Number	Common:StringObjectAttributeType	0..1	The Serial_Number element specifies the serial number of the Device.

3.2.8 DiskObjectType (extends Common:DefinedObjectType)

The DiskObjectType type is intended to characterize disk drives.

Property	Type	Mult	Description
Disk_Name	Common:StringObjectAttributeType	0..1	The Disk_Name element specifies the name of the disk.
Disk_Size	Common:UnsignedLongObjectAttributeType	0..1	The Disk_Size element specifies the size of the disk, in bytes.
Free_Space	Common:UnsignedLongObjectAttributeType	0..1	The Free_Space element specifies the amount of free space on the disk, in bytes.
Partition_List	DiskObj:PartitionListType	0..1	The Partition_List element specifies the partitions that reside on the disk.
Type	DiskObj:DiskType	0..1	The Type element specifies the type of disk being characterized, e.g. removable.

3.2.8.1 PartitionListType

The PartionListType type specifies a list of partitions.

Property	Type	Mult	Description
Partition	DiskPartitionObj:DiskPartitionObjectType	1..∞	The Partition element specifies a single partition that resides on the disk.

3.2.8.2 DiskType (restriction Common:BaseObjectAttributeType)

DiskType specifies disk types, via a union of the DiskTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: DiskObj:DiskTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.8.3 DiskTypeEnum

The DiskTypeEnum type contains a non-exhaustive enumeration of disk types.

Restriction base: string

Enumeration Value	Description

Removable	Indicates the removable disk type.
Fixed	Indicates the fixed disk type.
Remote	Indicates the remote disk type.
CDRom	Indicates the CDRom disk type.
RAMDisk	Indicates the RAMDisk disk type.

3.2.9 DiskPartitionObjectType (extends Common:DefinedObjectType)

The DiskPartitionType type is intended to characterize partitions of disk drives.

Property	Type	Mult	Description
Created	Common: DateTimeObject AttributeType	0..1	The Created element specifies the date/time the partition was created.
Device_Name	Common: NameObject AttributeType	0..1	The Device_Name element specifies the name of the device on which the partition resides.
Mount_Point	Common: StringObject AttributeType	0..1	The Mount_Point element specifies the mount point of the partition.
Partition_ID	Common: IntegerObject AttributeType	1..1	The Partition_ID element specifies the numerical identifier of the partition.
Partition_Length	Common: UnsignedLong ObjectAttributeType	0..1	The Partition_Length element specifies the length of the partition, in bytes.
Partition_Offset	Common: UnsignedLong ObjectAttributeType	0..1	The Partition_Offset element specifies the starting offset of the partition, in bytes.
Space_Left	Common: UnsignedLong ObjectAttributeType	0..1	The Space_Left element specifies the amount of space left on the partition, in bytes.
Space_Used	Common: UnsignedLong ObjectAttributeType	0..1	The Space_Used element specifies the amount of space used on the partition, in bytes.
Total_Space	Common: UnsignedLong ObjectAttributeType	0..1	The Total_Space element specifies the total amount of space available on the partition, in bytes.
Type	DiskPartitionObj: PartitionType	0..1	The Type element specifies the type of partition being characterized.

3.2.9.1 PartitionType (restriction Common:BaseObjectAttributeType)

PartitionType specifies partition types, via a union of the PartitionTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: DiskPartitionObj:PartitionTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.9.2 PartitionTypeEnum

The PartitionTypeEnum type is a non-exhaustive enumeration of partition types. See http://www.win.tue.nl/~aeb/partitions/partition_types-1.html for more information about the various partition types.

Restriction base: string

Enumeration Value	Description
PARTITION_ENTRY_UNUSED	Indicates an unused partition entry.
PARTITION_FAT_12	Indicates a FAT 12 partition.
PARTITION_XENIX_1	Indicates a XENIX type 1 partition.
PARTITION_XENIX_2	Indicates a XENIX type 2 partition.
PARTITION_FAT_16	Indicates a XENIX FAT 16 partition.
PARTITION_EXTENDED	Indicates a XENIX extended partition.
PARTITION_HUGE	Specifies an MS-DOS V4 huge partition. This value indicates that there is no Microsoft file system on the partition. Use this value when creating a logical volume.
PARTITION_IFS	Indicates an IFS partition.
PARTITION_OS2BOOTMGR	Indicates an OS/2 boot manager partition.
PARTITION_FAT32	Indicates a FAT32 partition.
PARTITION_FAT32_XINT13	Indicates a FAT32 Extended-INT13 equivalent partition to the FAT32 partition.
PARTITION_XINT13	Indicates an XINT13 partition.
PARTITION_XINT13_EXTENDED	Indicates an extended XINT13 partition.
PARTITION_PREP	Indicates a PReP (Power PC Reference Platform) partition.
PARTITION_LDM	Indicates an LDM partition.
PARTITION_UNIX	Indicates a UNIX partition.
VALID_NTFT	Specifies a valid NTFT partition. The high bit of a partition type code indicates that a partition is part of an NTFT mirror or striped array.
PARTITION_NTFT	Specifies an NTFT partition.
UNKNOWN	Refers to an unknown partition or a partition other than those listed.

3.2.10 EmailMessageObjectType (extends Common:DefinedObjectType)

The EmailMessageObjectType type is intended to characterize an individual email message.

Property	Type	Mult	Description
Attachments	EmailMessageObj: AttachmentsType	0..1	The Attachments element specifies any files that were attached to the email message. It imports and uses the CybOX FileObjectType from the File_Object to do so.
Header	EmailMessageObj: EmailHeaderType	1..1	The Header element specifies a variety of common headers that may be included in the email message.
Optional_Header	EmailMessageObj: EmailOptionalHeaderType	0..1	The Optional_Header element specifies a variety of optional headers that may be included in the email message.
Email_Server	Common: StringObject AttributeType	0..1	The Email_Server element is optiona and specifies the relevant email server.
Raw_Body	Common: StringObject AttributeType	0..1	The Raw_Body element specifies the complete (raw) body of the email message.
Raw_Header	Common: StringObject AttributeType	0..1	The Raw_Header element specifies the complete (raw) headers of the email message.

3.2.10.1 AttachmentsType

A list of attachments for an email message

Property	Type	Mult	Description
File	FileObject:FileObjectType	1..∞	The File element specifies a file that was attached to the email message. It uses the File_ObjectType of the CybOX File_Object.

3.2.10.2 EmailHeaderType

A representation of a standard email header

Property	Type	Mult	Description
To	EmailMessageObj: EmailRecipientsType	0..1	The To element specifies the email addresses of the recipients of the email message.
CC	EmailMessageObj: EmailRecipientsType	0..1	The CC element specifies the email addresses of any recipients that were included in the carbon copy header of the email message.
BCC	EmailMessageObj: EmailRecipientsType	0..1	The BCC element specifies the email addresses of any recipients that were included in the blind carbon copy header of the email message.
From	AddressObject: AddressObjectType	1..1	The From element specifies the email address of the sender of the email message.
Subject	Common: StringObject AttributeType	0..1	The Subject element specifies the subject (a brief summary of the message topic) of the email message.
In_Reply_To	Common: StringObject AttributeType	0..1	The In_Reply_To element specifies the message ID of the message that this email is a reply to.
Date	Common: DateTimeObject AttributeType	0..1	The Date element specifies the date/time that the email message was sent.
Message_ID	Common: StringObject AttributeType	0..1	The Message_ID element specifies the automatically generated ID of the email message.
Sender	AddressObject: AddressObjectType	0..1	The Sender element specifies the email address of the sender who is acting on behalf of the author listed in the From: field.
Reply_To	AddressObject: AddressObjectType	0..1	The Reply_To element specifies the email address that should be used when replying to the email message.
Errors_To	Common: StringObject AttributeType	0..1	The Errors_To element specifies the entry in the (deprecated) errors_to header of the email message.

3.2.10.3 EmailOptionalHeaderType

A representation of optional email header members

Property	Type	Mult	Description
Boundary	Common: StringObject AttributeType	0..1	The Boundary element specifies a boundary tag that may be included in a MIME multipart message. This boundary tag is used to indicate the parts of a multipart message.
Content-Type	Common: StringObject AttributeType	0..1	The Content-Type element specifies the internet media, or MIME, type of the email message content.
MIME-Version	Common:	0..1	The MIME-Version element specifies the version of

	StringObject AttributeType		the MIME formatting used in the email message.
Precedence	Common: StringObject AttributeType	0..1	The Precedence element specifies the (non-standard) priority value of the message, which can influence transmission speed and delivery. Use of this field is typically discouraged, as per IETF RFC2076 (http://www.faqs.org/rfcs/rfc2076.html).
X-Mailer	Common: StringObject AttributeType	0..1	The X-Mailer element specifies the software used to send the email message. This field is non-standard.
X-Originating-IP	AddressObject: AddressObjectType	0..1	The X-Originating-IP element specifies the originating IP Address of the email sender, in terms of their connection to the mail server used to send the email message. This field is non-standard.
X-Priority	Common: PositiveIntegerObject AttributeType	0..1	The X-Priority element specifies the numerical priority of the email message. This is a non-standard field, but typically a value of '1' is considered the highest priority, '3' is normal, and '5' is the lowest priority.

3.2.10.4 EmailRecipientsType

A list of recipients for an email message

Property	Type	Mult	Description
Recipient	AddressObject: AddressObjectType	1..∞	The Recipient element represents a single recipient for an email message.

3.2.11 FileObjectType (extends Common:DefinedObjectType)

The File_ObjectType type is intended to characterize generic files.

Property	Type	Mult	Description
is_packed	boolean	1..1	The ispacked attribute is used to indicate whether the file is packed or not.
File_Name	Common: StringObject AttributeType	0..1	The File_Name element specifies the name of the file.
File_Path	FileObj:FilePathType	0..1	The File_Path element specifies the path to the file, not including the device. Whether the path is relative or fully-qualified can be specified via the 'type' attribute.
Device_Path	Common: StringObject AttributeType	0..1	The Device_Path element specifies the path to the device on which the file resides.
Full_Path	Common: StringObject AttributeType	0..1	The Full_Path element specifies the complete path to the file, including the device path.
File_Extension	Common: StringObject AttributeType	0..1	The File_Extension element specifies the file extension of the file.
Size_In_Bytes	Common: UnsignedLong ObjectAttributeType	0..1	The Size_In_Bytes element specifies the size of the file, in bytes.
Hashes	Common:HashListType	0..1	The Hashes element specifies any hashes of the file.
Digital_Signatures	FileObj: DigitalSignatures	0..1	The Digital_Signatures element is optional and

	Type		captures one or more digital signatures for the file.
Modified_Time	Common: StringObject AttributeType	0..1	The Modified_Time element specifies the date/time the file was last modified.
Accessed_Time	Common: StringObject AttributeType	0..1	The Accessed_Time element specifies the date/time the file was last accessed.
Created_Time	Common: DateTimeObject AttributeType	0..1	The Created_Time element specifies the date/time the file was created.
File_Attributes_List	FileObj: FileAttributes ListType	0..1	The File_Attributes_List element specifies the particular special attributes set for the file. Since this is a platform-specific attribute, it is defined here as an abstract type and then implemented in any platform specific derived file objects.
Permissions	FileObj: FilePermissionsType	0..1	The Permissions element specifies that particular permissions that a file may have. Since this is a platform-specific attribute, it is defined here as an abstract type and then implemented in any platform specific derived file objects.
User_Owner	Common: StringObject AttributeType	0..1	The User_Owner element specifies the name of the user that owns the file.
Packer_List	FileObj: PackerListType	0..1	The Packer_List element specifies any packers that the file may be packed with. The term 'packer' here refers to packers, as well as things like archivers and installers.
Peak_Entropy	Common: DoubleObject AttributeType	0..1	The Peak_Entropy element specifies the calculated peak entropy of the file.
Sym_Links	FileObj: SymLinksListType	0..1	The Sym_Links element specifies any symbolic links that may exist for the file.
Extracted_Features	Common: Extracted FeaturesType	0..1	The Extracted_Features element is optional and enables description of features extracted from an this object.
Byte_Runs	Common:ByteRunsType	0..1	The Byte_Runs element contains a list of byte runs from the raw file or its storage medium.

3.2.11.1 FilePathType (extends Common:StringObjectAttributeType)

The FilePathType type specifies the path to the file, not including the device. Whether the path is relative or fully-qualified can be specified via the 'type' attribute.

Property	Type	Mult	Description
fully_qualified	boolean	1..1	The fully_qualified attribute specifies whether the path is fully qualified.

3.2.11.2 DigitalSignaturesType

A list of digital signatures

Property	Type	Mult	Description
Digital_Signature	Common: DigitalSignatureInfoType	0..1	The Digital_Signature element is optional and captures a single digital signature for the file.

3.2.11.3 FileAttributesListType

The FileAttributesListType specifies a list of file attributes.

Property	Type	Mult	Description
File_Attribute	FileObj:FileAttributeType	1..∞	The FileAttributeType specifies a single file attribute.

3.2.11.4 FileAttributeType (abstract)

The FileAttributeType type specifies an attribute of a file. Since this attribute is platform-specific, it is defined here as an abstract type.

3.2.11.5 FilePermissionsType (abstract)

The FilePermissionsType type specifies a permission of a file. Since this is a platform-specific attribute, it is defined here as an abstract type and then implemented in any platform specific derived file objects.

3.2.11.6 PackerListType

The PackerListType type specifies a list of file packers.

Property	Type	Mult	Description
Packer	FileObj: PackerAttributesType	1..∞	The Packer element specifies a single file packer.

3.2.11.7 PackerAttributesType

The PackerAttributesType type specifies the elements that characterize a particular file packer, such as name and version.

Property	Type	Mult	Description
Name	Common: StringObject AttributeType	1..1	The Name element specifies the name of the packer.
Version	Common: StringObject AttributeType	0..1	The Version element specifies the version of the packer.
PEiD	Common: StringObject AttributeType	0..1	The PEiD element specifies the PEiD signature for the packer, if applicable.
Type	FileObj:PackerType	0..1	The Type element specifies the type of packer being characterized.

3.2.11.8 PackerType (restriction Common:BaseObjectAttributeType)

PackerType specifies packer types, via a union of the PackerTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: FileObj:PackerTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.11.9 PackerTypeEnum

The PackerTypeEnum type is a (non-exhaustive) enumeration of packer types.

Restriction base: string

Enumeration Value	Description
Archiver	Indicates that the packer is an archiver.
Installer	Indicates that the packer is an installer.
Self-Extracting Archiver	Indicates that the packer is a self-extracting archiver.
Crypter	Indicates that the packer is a crypter.
Packer	Indicates a packer.
Protector	Indicates that the packer is a protector.
Bundler	Indicates that the packer is a bundler.
Other	Indicates a different type of packer from the ones listed.

3.2.11.10 SymLinksListType

The SymLinksListType specifies a list of symbolic links.

Property	Type	Mult	Description
Sym_Link	Common: StringObject AttributeType	1..∞	The Sym_Link element specifies a single symbolic link.

3.2.12 GUIDialogboxObjectType (extends GUIObj:GUIObjectType)

The GUIDialogboxObjectType type is intended to characterize GUI dialog boxes.

Property	Type	Mult	Description
Box_Caption	Common: StringObject AttributeType	0..1	The Box_Caption element specifies the caption associated with the dialog box.
Box_Text	Common: StringObject AttributeType	0..1	The Box_Text element specifies the text contained inside the dialog box.

3.2.13 GUIObjectType (extends Common:DefinedObjectType)

The GUIObjectType type is intended to characterize generic GUI objects.

Property	Type	Mult	Description
Height	Common: IntegerObject AttributeType	0..1	The Height element specifices the height of the GUI object.
Width	Common: IntegerObject AttributeType	0..1	The Width element specifies the width of the GUI object.

3.2.14 GUIWindowObjectType (extends GUIObj:GUIObjectType)

The GUIWindowObjectType is intended to characterize GUI windows.

Property	Type	Mult	Description
Owner_Window	Common: StringObject AttributeType	0..1	The Owner_Window specifies the owner window of the window object.
Parent_Window	Common: StringObject AttributeType	0..1	The Parent_Window element contains the parent window of the window object.
Window_Display_Name	Common: StringObject AttributeType	0..1	The Window_Display_Name element specifies the display name or title bar text of the window object.

3.2.15 LibraryObjectType (extends Common:DefinedObjectType)

The LibraryObjectType type is intended to characterize software libraries.

Property	Type	Mult	Description
Name	Common: StringObject AttributeType	0..1	The Name element specifies the full file name of the library. Example: abcd.dll.
Path	Common: StringObject AttributeType	0..1	The Path element specifies the fully-qualified path to the library.
Size	Common: UnsignedLong ObjectAttributeType	0..1	The Size element specifies the size of the library, in bytes.
Type	LibraryObj: LibraryType	0..1	The Type element specifies the type of library being characterized.
Version	Common: StringObject AttributeType	0..1	The Version element specifies the library version.
Base_Address	Common: HexBinary ObjectAttributeType	0..1	The Base_Address element specifies the default virtual address into which the library is loaded.

3.2.15.1 LibraryType (restriction Common:BaseObjectAttributeType)

LibraryType specifies library types, via a union of the LibraryTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: LibraryObj:LibraryTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.15.2 LibraryTypeEnum

The LibraryTypeEnum type is an enumeration of library types.

Restriction base: string

Enumeration Value	Description
Dynamic	Indicates a dynamic library.
Static	Indicates a static library.
Remote	Indicates a remote library.
Shared	Indicates a shared library.
Other	Indicates a different type of library than those listed above.

3.2.16 LinuxPackageObjectType (extends Common:DefinedObjectType)

The LinuxPackageObjectType type is intended to characterize Linux packages.

Property	Type	Mult	Description
Architecture	LinuxPackageObj: ArchitectureType	0..1	The Architecture element specifies the architecture for which the package was built.
Category	Common: StringObject AttributeType	0..1	The Category element specifies the categories under which a package may be displayed.

Description	Common: StringObject AttributeType	0..1	The Description element specifies an in-depth description of a package.
Epoch	Common: StringObject AttributeType	0..1	The Epoch element specifies the epoch number of the package.
EVR	Common: StringObject AttributeType	0..1	The EVR element specifies the epoch, version, and release fields of the package as a single version string.
Name	Common: StringObject AttributeType	1..1	The Name element specifies the name of the package.
Release	Common: StringObject AttributeType	0..1	The Release element specifies the release number of the package build.
Vendor	Common: StringObject AttributeType	0..1	The Vendor element specifies the vendor that holds the software copyright of the package.
Version	Common StringObject AttributeType	0..1	The Version element specifies the version number of the package build.

3.2.16.1 ArchitectureType (restriction Common:BaseObjectAttributeType)

ArchitectureType specifies CPU architecture types, via a union of the ArchitectureTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: LinuxPackageObj:ArchitectureTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.16.2 ArchitectureTypeEnum

The ArchitectureTypeEnum type is a non-exhaustive enumeration of CPU architectures.

Restriction base: string

Enumeration Value	Description
i386	Indicates an i386 architecture.
PPC	Indicates an PPC architecture.
SPARC	Indicates an SPARC architecture.
noarch	Indicates no particular architecture.

3.2.17 MemoryObjectType (extends Common:DefinedObjectType)

The MemoryObjectType type is intended to characterize generic memory objects.

Property	Type	Mult	Description
is_injected	boolean	1..1	The isinjected attribute specifies whether or not the particular memory object has had data/code injected into it by another process.
is_mapped	boolean	1..1	The ismapped attribute specified whether or not the particular memory object has been assigned a byte-for-byte correlation with some portion of a

			file or file-like resource.
is_protected	boolean	1..1	The isprotected attribute specifies whether or not the particular memory object is protected (read/write only from the process that allocated it).
Hashes	Common:HashListType	0..1	The Hashes element specifies any hashes of the particular memory object.
Name	Common: StringObject AttributeType	0..1	The Name element specifies the name of the particular memory object, if applicable.
Region_Size	Common: UnsignedLong ObjectAttributeType	0..1	The Region_Size element specifies the size of the particular memory region, in bytes.
Region_Start_Address	Common: HexBinaryObject AttributeType	0..1	The Region_Start_Address element specifies the starting address of the particular memory region.

3.2.18 MutexObjectType (extends Common:DefinedObjectType)

The MutexObjectType type is intended to characterize generic mutual exclusion (mutex) objects.

Property	Type	Mult	Description
named	boolean	1..1	The named attribute specifies whether the Mutex is named.
Name	Common: StringObject AttributeType	0..1	The Name element specifies the name for a named mutex object.

3.2.19 NetworkFlowObjectType (extends Common:DefinedObjectType)

Defines the fields necessary to summarize network traffic, expressed as flows of multiple packets. Does not include the packet payload data (i.e. the actual data that was uploaded/downloaded to and from the Dest IP to Source IP as included in packet monitoring tools, such as Wireshark).

Property	Type	Mult	Description
Network_Flow_Label	NetFlowObj: NetworkFlowLabelType	0..1	Represents elements common to all flow records formats - either expressed as a 5-tuple or an extended 7-tuple (actually an 8-tuple because for organizational reasons, we include the egress interface index). Because these fields are defined here, they are excluded from the fields associated directly with each different flow record format type.
Unidirectional_Flow_Record	NetFlowObj: UnidirectionalRecordType	0..1	Represents flow-record formats that capture data in one direction only (e.g., Netflow v9).
Bidirectional_Flow_Record	NetFlowObj: BidirectionalRecordType	0..1	Represents flow-record formats that capture data in both directions (e.g., YAF).

3.2.19.1 NetworkLayerInfoType

Network layer information (relative to the OSI network model) which is typically captured all types of network flow records.

Property	Type	Mult	Description
Src_IP	AddressObj:	0..1	Represents the source IP address for the network

	AddressObjectType		flow expressed as an IPv4 or IPv6 address. Note that not all flow protocols support IPv6 addresses.
Dest_IP	AddressObj: AddressObjectType	0..1	Represents the destination IP address for the network flow expressed as an IPv4 or IPv6 address. Note that not all flow protocols support IPv6 addresses.
Src_Port	PortObj:PortObjectType	0..1	The source port of the network flow (0-65535).
Dest_Port	PortObj:PortObjectType	0..1	The destination port of the network flow (0 to 65535).
IP_Protocol	PacketObj: IANAAssignedIP NumbersType	0..1	The IP Protocol of the network flow. This is usually TCP, UDP, or SCTP, but can include others as represented in NetFlow as an integer from 0 to 255. Please refer to http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml for reference.

3.2.19.2 NetworkFlowLabelType (extends NetFlowObj:NetworkLayerInfoType)

The NetworkFlowLabelType contains elements that are common to all flow record formats. It builds off of network layer information (a 5-tuple that commonly defines a flow) and includes ingress and egress interface indexes and IP protocol information (not present if all flow record formats). Egress information is usually not thought of as part of the extended 7-tuple, but we include it for organizational purposes. Because these fields are defined here, they are excluded from the fields associated directly with each different flow record format type.

Property	Type	Mult	Description
Ingress_Interface_Index	Common: IntegerObject AttributeType	0..1	Represents the index (in SNMP, by default) of the network interface card where the flows entered the router.
Egress_Interface_Index	Common: IntegerObject AttributeType	0..1	Represents the index (in SNMP, by default) of the network interface card where the flows leave the router.
IP_Type_Of_Service	Common: HexBinaryObject AttributeType	0..1	Type of service field from the IP header. Specifies the IP Type of Service (ToS). See RFC 1349 for more information.

3.2.19.3 UnidirectionalRecordType

Netflow record formats that capture traffic in one direction.

Property	Type	Mult	Description
IPFIX_Message	NetFlowObj: IPFIXMessageType	0..1	Represents an Internet Protocol Flow Information eXport (IPFIX) protocol. IPFIX is based on NetFlow v9. Has several extensions such as Enterprise-defined fields types and variable length fields. See RFC 5101 for more information.
NetflowV9_Export_Packet	NetFlowObj: NetflowV9Export PacketType	0..1	Represents the Netflow V9 flow record format. See RFC 3954 (Netflow v9) for more information.
NetflowV5_Packet	NetFlowObj: NetflowV5PacketType	0..1	Represents the NetFlow v5 flow record format, which is commonly used to represent network flow data.

SiLK_Record	NetFlowObj: SiLKRecordType	0..1	Represents a network flow record in the System for Internet-Level Knowledge (SiLK) format, developed by CERT at Carnegie Mellon University (CMU)'s Software Engineering Institute (SEI) as part of the NetSA security suite. See http://tools.netsa.cert.org/silk/analysis-handbook.pdf for more information.

3.2.19.4 BidirectionalRecordType

Network record formats that capture traffic in both directions. Later, we plan to add Argus as a network flow format type. Argus supports bidirectional flows, and as such, is usually used as an alternative to NetFlow v5 analysis via SiLK (http://www.qosient.com/argus/).

Property	Type	Mult	Description
YAF_Record	NetFlowObj: YAFRecordType	0..1	Represents flow records generated via YAF (Yet Another Flowmeter), a bidirectional network flow meter. See http://www.usenix.org/event/lisa10/tech/full_papers/Inacio.pdf or http://tools.netsa.cert.org/yaf/index.html for more information.

3.2.19.5 IPFIXMessageType

The IPFIX protocol provides IP flow information. http://tools.ietf.org/html/rfc5101

Property	Type	Mult	Description
Message_Header	NetFlowObj: IPFIXMessage HeaderType	0..1	The Message Header is the first part of an IPFIX Message, which provides basic information about the message, such as the IPFIX version, length of the message, message sequence number, etc. http://tools.ietf.org/html/rfc5101
Set	NetFlowObj: IPFIXSetType	0..∞	Set is a generic term for a collection of records that have a similar structure. In an IPFIX Message, one or more Sets follow the Message Header. http://tools.ietf.org/html/rfc5101

3.2.19.6 IPFIXMessageHeaderType

This type represents the message header for the IPFIX format. For more information about each of the fields, please refer to RFC 5101 (http://tools.ietf.org/html/rfc5101) under the heading, "Message Header Field Descriptions." Note that common elements are included in the Network_Flow_Label.

Property	Type	Mult	Description
Version	Common: HexBinary ObjectAttributeType	0..1	Indicates the version number of Flow Record format exported in this message. The value of this field is 0x000a for the current version, incrementing by one the version used in the NetFlow services export version 9 [see RFC3954].
Byte_Length	Common: HexBinary ObjectAttributeType	0..1	Indicates the total byte length of the IPFIX Message, measured in octets, including Message Header and Set(s).

Export_Timestamp	Common: IntegerObject AttributeType	0..1	Indicates the time, in seconds, since 0000 UTC Jan 1, 1970, at which the IPFIX message header leaves the Exporter.
Sequence_Number	Common: IntegerObject AttributeType	0..1	Indicates the incremental sequence counter modulo 2^32 of all IPFIX Data Records sent on this PR-SCTP stream from the current Observation Domain by the Exporting Process. This value SHOULD be used by the Collecting Process to identify whether any IPFIX Data Records have been missed. Template and Options Template Records do not increase the Sequence Number.
Observation_Domain_ID	Common: IntegerObject AttributeType	0..1	Indicates a 32-bit identifier of the Observation Domain that is locally unique to the Exporting Process. See RFC 5101 under Observation Domain ID for more information.

3.2.19.7 IPFIXSetType

Represents the possible sets of records that can be represented in an IPFIX message. See RFC 5101 and look for the terms "Template Set", "Options Template Set", and "Data Set", for more information.

Property	Type	Mult	Description
Template_Set	NetFlowObj: IPFIXTemplateSetType	0..1	Indicates a collection of one or more Template Records that have been grouped together in an IPFIX message.
Options_Template_Set	NetFlowObj: IPFIXOptions TemplateSetType	0..1	Indicates a collection of one or more Options Template Records that have been grouped together in an IPFIX message.
Data_Set	NetFlowObj: IPFIXDataSetType	0..1	Indicates one or more Data Records, of the same type, that have been grouped together in an IPFIX message. Each Data Record is previously defined by a Template Record or an Options Template Record.

3.2.19.8 IPFIXTemplateSetType

Specifies the regions of a Template Set, of which there are three: the Set Header, the collection of Template Records, and the optional padding at the end of the Template Set. See RFC 5101 under Set Format, which is section 3.3.1, for more information.

Property	Type	Mult	Description
Set_Header	NetFlowObj: IPFIXSetHeaderType	0..1	Indicates the Set Header region, which is 32-bit region containing the 16-bit fields Set ID and Length.
Template_Record	NetFlowObj: IPFIXTemplate RecordType	0..∞	Indicates the region of Template Records. These are the same fields referenced in the IPFIXTemplateRecordType.
Padding	Common: HexBinary ObjectAttributeType	0..1	Indicates the optional Padding at the end of a Template Set. As mentioned in RFC 5101, the Exporting Process MAY insert some padding octets, so that the subsequent Set starts at an aligned boundary. For security reasons, the padding octet(s) MUST be composed of zero (0) valued octets, and the padding length MUST be shorter than any allowable record in this Set. For more information see RFC 5101

			under Padding.

3.2.19.9 IPFIXOptionsTemplateSetType

Specifies the regions of an Options Template Set, of which there are three: the Set Header, the collection of Options Template Records, and the optional padding at the end of the Options Template Set. See RFC 5101 under Set Format, which is section 3.3.1, for more information.

Property	Type	Mult	Description
Set_Header	NetFlowObj: IPFIXSetHeaderType	0..1	Indicates the Set Header region, which is 32-bit region containing the 16-bit fields Set ID and Length, in that order. These are the same fields referenced in the IPFIXSetHeaderType.
Options_Template_ Record	NetFlowObj: IPFIXOptions TemplateRecordType	0..∞	Indicates the region of Options Template Records. These are the same fields referenced in the IPFIXOptionsTemplateRecordType.
Padding	Common: HexBinary ObjectAttributeType	0..1	Indicates the optional Padding at the end of an Options Template Set. As mentioned in RFC 5101, the Exporting Process MAY insert some padding octets, so that the subsequent Set starts at an aligned boundary. For security reasons, the padding octet(s) MUST be composed of zero (0) valued octets, and the padding length MUST be shorter than any allowable record in this Set. For more information see RFC 5101 under Padding.

3.2.19.10 IPFIXDataSetType

Specifies the regions of a Data Set, of which there are three: the Set Header, the collection of Data Records, and the optional padding at the end of the Data Set. See RFC 5101 under Set Format, which is section 3.3.1, for more information.

Property	Type	Mult	Description
Set_Header	NetFlowObj: IPFIXSetHeaderType	0..1	Indicates the Set Header region, which is 32-bit region containing the 16-bit fields Set ID and Length, appended in that order. These are the same fields referenced in the IPFIXSetHeaderType.
Data_Record	NetFlowObj: IPFIXDataRecordType	0..∞	Indicates the region of Data Records, which consist of a series of field values without a header, according to RFC 5101, section 3.4.3.
Padding	Common: HexBinaryObject AttributeType	0..1	Indicates the optional Padding at the end of a Data Set. As mentioned in RFC 5101, the Exporting Process MAY insert some padding octets, so that the subsequent Set starts at an aligned boundary. For security reasons, the padding octet(s) MUST be composed of zero (0) valued octets, and the padding length MUST be shorter than any allowable record in this Set. For more information see RFC 5101 under Padding.

3.2.19.11 IPFIXSetHeaderType

Defines the elements of the IPFIX set header.

Property	Type	Mult	Description
Set_ID	Common: IntegerObject AttributeType	0..1	Indicates a 16-bit value that identifies the set. The values of 0 and 1 are not used for historical reasons according to RFC 3954. Otherwise, a value of 2 is reserved for the Template Set and 3 is reserved for the Option Template Set. All other values from 4 to 255 are reserved for future use.
Length	Common: IntegerObject AttributeType	0..1	Total length of the set, in octets, including the set header, all records, and the optional padding. Because an individual Set MAY contain multiple records, the Length value MUST be used to determine the position of the next Set. http://tools.ietf.org/html/rfc5101

3.2.19.12 IPFIXTemplateRecordType

Specifies the regions of a Template Record, of which there are two: the Template Record Header, and the Field Specifiers. See RFC 5101 under Template Record Format, section 3.4.1, for more information.

Property	Type	Mult	Description
Template_Record_Header	NetFlowObj: IPFIXTemplate RecordHeaderType	0..1	Indicates the Template Record Header region, which is a 32-bit region containing the 16-bit fields Template ID (> 255) and Field Count, appended in that order. These are the same fields referenced in the IPFIXTemplateRecordHeaderType.
Field_Specifier	NetFlowObj: IPFIXTemplateRecord FieldSpecifiersType	0..∞	Indicates the region of Field Specifiers. These are the same fields referenced in the IPFIXTemplateRecordFieldSpecifiersType.

3.2.19.13 IPFIXTemplateRecordHeaderType

Specifies the fields in a Template Record Header, Template_ID and Field_Count, as explained in RFC 5101, section 3.4.1.

Property	Type	Mult	Description
Template_ID	Common: IntegerObject AttributeType	0..1	Specifies a unique Template ID which is numbered 256-65535 since IDs 0-255 are reserved for Template Sets, Options Template Sets, and other reserved Sets yet to be created.
Field_Count	Common: HexBinary ObjectAttributeType	0..1	Specifies the number of fields in this Template Record.

3.2.19.14 IPFIXTemplateRecordFieldSpecifiersType

Specifies the fields in a Template Record Field Specifier, as explained in RFC 5101, section 3.2.

Property	Type	Mult	Description
Enterprise_Bit	boolean	0..1	Specifies the Enterprise bit, either 0 or 1. If this bit is zero, the Information Element Identifier identifies an IETF-specified Information Element, and the four-octet Enterprise Number field SHOULD NOT be present. If this bit is one, the Information Element identifier identifies an enterprise-specific Information

			Element, and the Enterprise Number filed SHOULD be present. NOTE: While it is legal to use "true" and "false" here, this value SHOULD be set to 0 or 1 for consistency with RFC 5101.
Information_Element_ID	Common: StringObject AttributeType	0..1	Specifies the 15-bit (NOT 16-bit) Information Element ID referring to the type of Information Element, as shown in RFC 5102.
Field_Length	Common: StringObject AttributeType	0..1	Specifies the 16-bit Field Length, in octets, of the corresponding encoded Information Element as defined in RFC 5102. The field length may be smaller than the definition in RFC 5102 if the reduced size encoding is used (see Section 6.2 of RFC 5101). The value 65535 is reserved for variable length Information Elements.
Enterprise_Number	Common: StringObject AttributeType	0..1	Specifies the 32-bit IANA Enterprise Number of the authority defining the Information Element identifier in this Template Record. Information Element Identifiers 1.2 and 2.1 are defined by the IETF (Enterprise bit = 0) and, therefore, do not need an Enterprise Number to identify them.

3.2.19.15 IPFIXOptionsTemplateRecordType

Specifies the regions of an Options Template Record, of which there are two: the Options Template Record Header, and the Field Specifiers. See RFC 5101 under Options Template Record Format, section 3.4.2.2, for more information.

Property	Type	Mult	Description
Options_Template_Record_Header	NetFlowObj: IPFIXOptionsTemplate RecordHeaderType	0..1	Indicates the Options Template Record Header region, which is a 48-bit region containing the 16-bit fields Template ID, Field Count, and Scope Field Count, appended in that order.
Field_Specifier	NetFlowObj: IPFIXOptionsTemplate RecordField SpecifiersType	0..∞	Indicates the region of Field Specifiers. These are the same fields referenced in the IPFIXOptionsTemplateRecordFieldSpecifiersType.

3.2.19.16 IPFIXOptionsTemplateRecordHeaderType

Defines the ehader of an options template record.

Property	Type	Mult	Description
Template_ID	Common: IntegerObject AttributeType	0..1	Specifies a unique Template ID which is numbered 256-65535 since IDs 0-255 are reserved for Template Sets, Options Template Sets, and other reserved Sets yet to be created.
Field_Count	Common: HexBinary ObjectAttributeType	0..1	Specifies the number of fields in this Options Template Record, INCLUDING the Scope Fields.
Scope_Field_Count	Common: PositiveInteger ObjectAttributeType	0..1	Specifies the number of scope fields in this Options Template Record, which is NONZERO. The Scope Fields are normal Fields except that they are interpreted as scope at the Collector.

3.2.19.17 IPFIXOptionsTemplateRecordFieldSpecifiersType

Specifies the fields in an Options Template Record Field Specifier, as explained in RFC 5101, sections 3.2 and 3.4.2.2. It consists of two sequences: Scope Fields and Option Fields, appended together.

Property	Type	Mult	Description
Scope_Enterprise_Bit	boolean	0..1	Specifies the Scope Enterprise bit, either 0 or 1. If this bit is zero, the Information Element Identifier identifies an IETF-specified Information Element, and the four-octet Enterprise Number field SHOULD NOT be present. If this bit is one, the Information Element identifier identifies an enterprise-specific Information Element, and the Enterprise Number filed SHOULD be present. NOTE: While it is legal to use "true" and "false" here, this value SHOULD be set to 0 or 1 for consistency with RFC 5101.
Scope_Information_Element_ID	Common: StringObject AttributeType	0..1	Specifies the 15-bit (NOT 16-bit) Scope Information Element ID referring to the type of Information Element, as shown in RFC 5102.
Scope_Field_Length	Common: IntegerObject AttributeType	0..1	Specifies the 16-bit Scope Field Length, in octets, of the corresponding encoded Information Element as defined in RFC 5102. The field length may be smaller than the definition in RFC 5102 if the reduced size encoding is used (see Section 6.2 of RFC 5101). The value 65535 is reserved for variable length Information Elements.
Scope_Enterprise_Number	Common: StringObject AttributeType	0..1	Specifies the 32-bit IANA Scope Enterprise Number of the authority defining the Information Element identifier in this Template Record. Information Element Identifiers 1.2 and 2.1 are defined by the IETF (Enterprise bit = 0) and, therefore, do not need an Enterprise Number to identify them.
Option_Enterprise_Bit	boolean	0..1	Specifies the Option Enterprise bit, either 0 or 1. If this bit is zero, the Information Element Identifier identifies an IETF-specified Information Element, and the four-octet Enterprise Number field SHOULD NOT be present. If this bit is one, the Information Element identifier identifies an enterprise-specific Information Element, and the Enterprise Number filed SHOULD be present. NOTE: While it is legal to use "true" and "false" here, this value SHOULD be set to 0 or 1 for consistency with RFC 5101.
Option_Information_Element_ID	Common: StringObject AttributeType	0..1	Specifies the 15-bit (NOT 16-bit) Option Information Element ID referring to the type of Information Element, as shown in RFC 5102.

Option_Field_Length	Common: IntegerObject AttributeType	0..1	Specifies the 16-bit Option Field Length, in octets, of the corresponding encoded Information Element as defined in RFC 5102. The field length may be smaller than the definition in RFC 5102 if the reduced size encoding is used (see Section 6.2 of RFC 5101). The value 65535 is reserved for variable length Information Elements.
Option_Enterprise_Number	Common: StringObject AttributeType	0..1	Specifies the 32-bit IANA Option Enterprise Number of the authority defining the Information Element identifier in this Template Record. Information Element Identifiers 1.2 and 2.1 are defined by the IETF (Enterprise bit = 0) and, therefore, do not need an Enterprise Number to identify them.

3.2.19.18 IPFIXDataRecordType

Data records are sent in data sets. A data record consists of only one more more Field values.

Property	Type	Mult	Description
Field_Value	Common: StringObject AttributeType	0..∞	Indicates the individual Field Value, which need not be 16-bit. The Template ID to which the Field Values belong to is encoded in the Data Set Header field "Set ID", i.e. "Set ID" = "Template ID".

3.2.19.19 NetflowV9ExportPacketType

Netflow v9 was developed by Cisco and provides acess to IP flow information.
http://www.ietf.org/rfc/rfc3954.txt

Property	Type	Mult	Description
Packet_Header	NetFlowObj: NetflowV9PacketHeaderType	0..1	Specifies the Packet Header, which is the first part of an Export Packet. The Packet Header provides basic information about the packet such as the NetFlow version, number of records contained within the packet, and sequence numbering. See RFC 3954 for more information.
Flow_Set	NetFlowObj: NetflowV9FlowSetType	0..∞	Specifies a FlowSet, which is a collection of Flow Records that have similar structure. In an Export Packet, one or more FlowSets follow the Packet Header. There are three differnet types of FlowSets, as defined in RFC 3954: a Template FlowSet, Options Template FlowSet and Data FlowSet.

3.2.19.20 NetflowV9PacketHeaderType

Header fields defined for Netflow v9. Note that common elements are included in the Network_Flow_Label. http://www.ietf.org/rfc/rfc3954.txt

Property	Type	Mult	Description
Version	Common: HexBinary	0..1	Specifies the version of flow record format exported in this packet. The value of this field is 9 for the

	ObjectAttributeType		Netflow v9.
Record_Count	Common: IntegerObject AttributeType	0..1	Specifies the total number of records in the Export Packet, which is the sum of Options FlowSet records, Template FlowSet records, and Data FlowSet records. http://www.ietf.org/rfc/rfc3954.txt
Sys_Up_Time	Common: IntegerObject AttributeType	0..1	Specifies the time in milliseconds since this device was first booted.
Unix_Secs	Common: IntegerObject AttributeType	0..1	Specifies the time in seconds since 0000 UTC 1970 at which the Export Packet leaves the Exporter.
Sequence_Number	Common: IntegerObject AttributeType	0..1	Incremental sequence counter of all Export Packets sent from the current Observation Domain by the Exporter. This value MUST be cumulative, and SHOULD be used by the Collector toidentify whether any Export Packets have been missed. http://www.ietf.org/rfc/rfc3954.txt
Source_ID	Common: HexBinaryObject AttributeType	0..1	Specifies a 32-bit value that identifies the Exporter Observation Domain. NetFlow Collectors SHOULD use the combination of the source IP address and the Source ID field to separate different export streams originating from the same Exporter.

3.2.19.21 NetflowV9FlowSetType

In an Export Packet, one or more FlowSets follow the Packet Header. There are three differnet types of FlowSets, as defined in RFC 3954: a Template FlowSet, Options Template FlowSet and Data FlowSet.

Property	Type	Mult	Description
Template_Flow_Set	NetFlowObj: NetflowV9Template FlowSetType	0..1	One of the essential elements in the NetFlow format is the Template FlowSet. Templates greatly enhance the flexibility of the Flow Record format because they allow the NetFlow Collector to process Flow Records without necessarily knowing the interpretation of all the data in the Flow Record. http://www.ietf.org/rfc/rfc3954.txt
Options_Template_Flow_Set	NetFlowObj: NetflowV9Options TemplateFlowSetType	0..1	Specifies an Options Template FlowSet, which is one or more Options Template Records that have been grouped together in an Export Packet.
Data_Flow_Set	NetFlowObj: NetflowV9Data FlowSetType	0..1	Specifies a Data FlowSet, which is one or more records, of the same type, that are grouped together in an Export Packet. Each record is either a Flow Data Record or an Options Data Record previously defined by a Template Record or an Options Template Record.

3.2.19.22 NetflowV9TemplateFlowSetType

Provides the format of the Template FlowSet.

Property	Type	Mult	Description

Flow_Set_ID	Common: HexBinaryObject AttributeType	0..1	Specifies the FlowSet ID, which is fixed to 0 for the Template FlowSet.
Length	Common: IntegerObject AttributeType	0..1	Length is the sum of the lengths of the FlowSet ID, the Length itself, and all Template Records within this FlowSet.
Template_Record	NetFlowObj: NetflowV9Template RecordType	0..∞	Specifies the Template Record region, which includes the template ID, field count, field type, and field length.

3.2.19.23 NetflowV9TemplateRecordType

Specifies the Template Record region, which includes the template ID, field count, field type, and field length.

Property	Type	Mult	Description
Template_ID	Common: IntegerObject AttributeType	0..1	Specifies a unique Template ID for the Template Record. IDs in the range 0-255 are reserved for Template FlowSets, Options FlowSets, and other reserved Sets yet to be created. http://www.ietf.org/rfc/rfc3954.txt
Field_Count	Common: IntegerObject AttributeType	0..1	Specifies the number of fields in this Template Record.
Field_Type	NetFlowObj: NetflowV9FieldType	0..1	Specifies a numeric value that represents the type of the field. Refer to the "Field Type Definitions" section in RFC 3954 for descriptions of these types.
Field_Length	Common: HexBinary ObjectAttributeType	0..1	Specifies the length of the corresponding field type, in bytes.

3.2.19.24 NetflowV9FieldType (restriction Common:BaseObjectAttributeType)

NetflowV9FieldType specifies possible fields types for Netflow v9, via a union of the NetflowV9FieldTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: NetFlowObj:NetflowV9FieldTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.19.25 NetflowV9FieldTypeEnum

This enumeration describe the field types in NetFlow Version 9. Only the first 20 have been enumerated so far. Please see Section 8 in http://www.ietf.org/rfc/rfc3954.txt for the complete list (79 in total).

Restriction base: string

Enumeration Value	Description
IN_BYTES(1)	The IN_BYTES(1) field represents the incoming counter with length N x 8 bits for number of bytes associated with an IP Flow.
IN_PKTS(2)	The IN_PKTS(2) field represents the incoming counter with length N x 8 bits for the number of packets associated with an IP Flow.
FLOWS(3)	The FLOWS(3) field represents the number of flows that were aggregated; default for N

	is 4.
PROTOCOL(4)	The PROTOCOL(4) field represents the IP protocol byte.
SRC_TOS(5)	The TOS(5) field represents the Type of Service byte setting when entering incoming interface.
TCP_FLAGS(6)	The TCP_FLAGS(6) field is cumulative of all the TCP flags seen for this flow.
L4_SRC_PORT(7)	The L4_SRC_PORT(7) field represents the TCP/UDP source port number i.e.: FTP, Telnet, or equivalent.
IPV4_SRC_ADDR(8)	The IPV4_SRC_ADDR(8) field represents the IPv4 source address.
SRC_MASK(9)	The SRC_MASK(9) field represents the number of contiguous bits in the source address subnet mask i.e.: the submask in slash notation.
INPUT_SNMP(10)	The INPUT_SNMP(10) field represents the number of contiguous bits in the source address subnet mask i.e.: the submask in slash notation.
L4_DST_PORT(11)	The LP_DST_PORT(11) field represents the TCP/UDP destination port number i.e.: FTP, Telnet, or equivalent.
IPV4_DST_ADDR(12)	The IPV4_DST_ADDR(12) field represents the IPv4 destination address.
DST_MASK(13)	The DST_MASK(13) field represents the number of contiguous bits in the destination address subnet mask i.e.: the submask in slash notation.
OUTPUT_SNMP(14)	The OUTPUT_SNMP(14) field represents the output interface index; default for N is 2 but higher values could be used.
IPV4_NEXT_HOP(15)	The IPV4_NEXT_HOP(15) field represents the IPv4 address of next-hop router.
SRC_AS(16)	The SRC_AS(16) field represents the source BGP autonomous system number where N could be 2 or 4.
DST_AS(17)	The DST_AS(17) field represents the destination BGP autonomous system number where N could be 2 or 4.
BGP_IPV4_NEXT_HOP(18)	The BGP_IPV4_NEXT_HOP(18) field represents the next-hop router's IP in the BGP domain.
MUL_DST_PKTS(19)	The MUL_DST_PKTS(19) field represents the IP multicast outgoing packet counter with length N x 8 bits for packets associated with the IP Flow.
MUL_DST_BYTES(20)	The MUL_DST_BYTES(20) field represents the IP multicast outgoing byte counter with length N x 8 bits for bytes associated with the IP Flow.

3.2.19.26 NetflowV9OptionsTemplateFlowSetType

Specifies an Options Template FlowSet, which is one or more Options Template Records that have been grouped together in an Export Packet.

Property	Type	Mult	Description
Flow_Set_ID	Common: HexBinary ObjectAttributeType	0..1	Specifies the FlowSet ID, which is fixed to 1 for the Options Template FlowSet.
Length	Common: IntegerObject AttributeType	0..1	Specifies the total length of this FlowSet, in octets, including the set header, all records, and the optional padding.
Options_Template_Record	NetFlowObj: NetflowV9Options TemplateRecordType	0..∞	Specifies the Options Template Record region, which includes the Option Scope Length, Option Length, and fields specifying the Scope field type and Scope field length.
Padding	Common: HexBinary ObjectAttributeType	0..1	Specifies the number of padding bytes to be inserted so that the subsequent FlowSet starts at a 4-byte aligned boundary. It is important to note that the Length field includes the padding bytes. Padding SHOULD be using zeros.

3.2.19.27 NetflowV9OptionsTemplateRecordType

Specifies the Options Template Record region, which includes the Option Scope Length, Option Length, and fields specifying the Scope field type and Scope field length.

Property	Type	Mult	Description
Template_ID	Common: IntegerObject AttributeType	0..1	Specifies the template ID of this Options Template, which must be greater than 255.
Option_Scope_Length	Common: HexBinary ObjectAttributeType	0..1	Specifies the length of bytes of any Scope field definition contained in the Options Template Record.
Option_Length	Common: HexBinary ObjectAttributeType	0..1	Specifies the length of bytes of any options field definitions contained in this Options Template Record.
Scope_Field_Type	NetFlowObj: NetflowV9Scope FieldType	0..1	Specifies the relevent portion of the Exporter/NetFlow process to which the Options Template Record refers. Currently defined values include 1 for System, 2 for Interface, 3 for Line Card, 4 for Cache, and 5 for Template. More information can be found in RFC 3954.
Scope_Field_Length	Common: HexBinary ObjectAttributeType	0..1	Specifies the length (in bytes) of the Scope field as it would appear in an Options Data Record.
Option_Field_Type	NetFlowObj: NetflowV9FieldType	0..1	Specifies the type of field that would appear in the Options Template Record. More information can be found in RFC 3954.
Option_Field_Length	Common: HexBinary ObjectAttributeType	0..1	Specifies the length (in bytes) of the Option field.

3.2.19.28 NetflowV9ScopeFieldType (restriction Common:BaseObjectAttributeType)

NetflowV9ScopeFieldType specifies scope field types for Netflow v9, via a union of the NetflowV9ScopeFieldTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: NetFlowObj:NetflowV9ScopeFieldTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.19.29 NetflowV9ScopeFieldTypeEnum

These describe the scope field types, found in the relevant portion of the NetFlow process to which the options record refers. http://www.ietf.org/rfc/rfc3954.txt

Restriction base: string

Enumeration Value	Description
System(1)	Indicates the System scope field type.
Interface(2)	Indicates the Interface scope field type.
LineCard(3)	Indicates the Line Card scope field type.
Cache(4)	Indicates the NetFlow Cache scope field type.
Template(5)	Describes the Template scope field type.

3.2.19.30 NetflowV9DataFlowSetType

Specifies a Data FlowSet, which is one or more records, of the same type, that are grouped together in an Export Packet. Each record is either a Flow Data Record or an Options Data Record previously defined by a Template Record or an Options Template Record. http://www.ietf.org/rfc/rfc3954.txt

Property	Type	Mult	Description
Flow_Set_ID_Template_ID	Common: IntegerObject AttributeType	0..1	Specifies the FlowSet ID, which corresponds to the Template ID from a Template Flow Set or an Options Template Flow Set.
Length	Common: IntegerObject AttributeType	0..1	Specifies the length of this FlowSet.
Data_Record	NetFlowObj: NetflowV9DataRecordType	0..∞	The remainder of the Data FlowSet is a collection of Flow Data Record(s), each containing a set of field values. The Type and Length of the fields have been previously defined in the Template Record referenced by the FlowSet ID or Template ID. Specifies either a template flow set or an options template flow set. http://www.ietf.org/rfc/rfc3954.txt
Padding	Common: HexBinary ObjectAttributeType	0..1	Specifies the padding bytes used so that the subsequent FlowSet starts at a 4-byte aligned boundary. It is important to note that the Length field includes the padding bytes. Padding SHOULD be using zeros.

3.2.19.31 NetflowV9DataRecordType

A Data FlowSet is one or more records, of the same type, that are grouped together in an Export Packet. Each record is either a Flow Data Record or an Options Data Record previously defined by a Template Record or an Options Template Record. http://www.ietf.org/rfc/rfc3954.txt

Property	Type	Mult	Description
Flow_Data_Record	NetFlowObj: FlowDataRecordType	0..∞	Specifies a Flow Data Record, which corresponds to a FieldType defined in the Template Record. Each one will have multiple values associated with it.
Options_Data_Record	NetFlowObj: OptionsDataRecordType	0..∞	Specifies an Options Data Record, which Corresponds to a previously defined Options Template Record.

3.2.19.32 FlowDataRecordType

A Flow Data Record is a data record that contains values of the Flow parameters corresponding to a Template Record.

Property	Type	Mult	Description
Flow_Record_Collection_Element	NetFlowObj: FlowCollectionElementType	0..∞	For each flow record, field values are listed.

3.2.19.33 FlowCollectionElementType

Field values are associated with each record in the collection of a flow data record.

Property	Type	Mult	Description
Flow_Record_Field_Value	Common: StringObject AttributeType	0..∞	Set of fields values for a given Flow Data Record.

3.2.19.34 OptionsDataRecordType

The data record that contains values and scope information of the Flow measurement parameters, corresponding to an Options Template Record.

Property	Type	Mult	Description
Scope_Field_Value	Common: StringObject AttributeType	0..1	Corresponds to a previously defined Options Template Record.
Option_Record_Collection_Element	NetFlowObj: OptionCollection ElementType	0..∞	For each option data record, field values are listed.

3.2.19.35 OptionCollectionElementType

Field values are associatedwith each option in the collection of an option data record.

Property	Type	Mult	Description
Option_Record_Field_Value	Common: StringObject AttributeType	0..∞	Set of field values for a given Options Data Record.

3.2.19.36 NetflowV5PacketType

Defines the contents of a Netflow v5 packet. As of 2012, Netflow v5 is still the most commonly used network flow format. Netflow v5 was developed by Cisco. http://netflow.caligare.com/netflow_v5.htm

Property	Type	Mult	Description
Flow_Header	NetFlowObj: NetflowV5FlowHeaderType	0..1	Elements of a netflow v5 header.
Flow_Record	NetFlowObj: NetflowV5FlowRecordType	1..30	See Network_Flow_Label for other common fields. Padding of 0-bytes is not captured. REF: http://netflow.caligare.com/netflow_v5.htm REF: http://tools.netsa.cert.org/silk/faq.html#ipfix-fields

3.2.19.37 NetflowV5FlowHeaderType

Defines elements of a netflow v5 header. http://netflow.caligare.com/netflow_v5.htm

Property	Type	Mult	Description
Version	Common: HexBinary ObjectAttributeType	0..1	Specifies the NetFlow export format version number, which defaults to 5 in this case.
Count	Common: IntegerObject AttributeType	0..1	Specifies the number of flows exported in the packet (1-30).
Sys_Up_Time	Common: IntegerObject AttributeType	0..1	Specifies the current time in milliseconds since the export device booted.
Unix_Secs	Common: IntegerObject AttributeType	0..1	Specifies the current time in milliseconds since 0000 UTC 1970.
Unix_Nsecs	Common: IntegerObject	0..1	Specifies the residual in nanoseconds since 0000

	AttributeType		UTC 1970.
Flow_Sequence	Common: IntegerObject AttributeType	0..1	Specifies the sequence counter of total flows seen.
Engine_Type	Common: StringObject AttributeType	0..1	Specifies the type of flow-switching engine.
Engine_ID	Common: IntegerObject AttributeType	0..1	Specifies the slot number of the flow-switching engine.
Sampling_Interval	Common: HexBinary ObjectAttributeType	0..1	Specifies the sampling interval field, which consists of the first two bits holding the sampling mode, with the remaining 14 bits holding the value of the sampling interval.

3.2.19.38 NetflowV5FlowRecordType

Defines elements of a Netflow v5 flow record. Recall that the seven elements that define the flow itself (e.g., source IP address) are provided in NetworkFlowLabelType.
https://bto.bluecoat.com/packetguide/8.6/info/netflow5-records.htm

Property	Type	Mult	Description
Nexthop_IPv4_Addr	AddressObj: AddressObjectType	0..1	Represents the IP address of the next hop router.
Packet_Count	Common: IntegerObject AttributeType	0..1	Represents the number of packets in the flow.
Byte_Count	Common: IntegerObject AttributeType	0..1	Represents the total number of bytes in the flow.
SysUpTime_Start	Common: IntegerObject AttributeType	0..1	Represents the SysUpTime at start of flow: the total time in milliseconds starting from when the first packet in the flow was seen.
SysUpTime_End	Common: IntegerObject AttributeType	0..1	Represents the SysUpTime at end of flow: when the last packet in the flow was seen.
Padding1	Common: HexBinary ObjectAttributeType	0..1	One byte of padding.
TCP_Flags	Common: HexBinary ObjectAttributeType	0..1	Specifies the union of all TCP flags observed over the life of the flow.
Src_Autonomous_System	Common: IntegerObject AttributeType	0..1	Specifies the source autonomous system number, either origin or peer.
Dest_Autonomous_System	Common: IntegerObject AttributeType	0..1	Specifies the destination autonomous system number, either origin or peer.
Src_IP_Mask_Bit_Count	Common: StringObject AttributeType	0..1	Specifies the source address prefix mask bits.
Dest_IP_Mask_Bit_Count	Common: StringObject AttributeType	0..1	Specifies the destination address prefix mask bits.
Padding2	Common: HexBinary ObjectAttributeType	0..1	Unused (zero) bytes, which is used for purposes of padding.

3.2.19.39 SiLKRecordType

System for Internet-Level Knowledge (CMU/SEI). The fields are taken from a list shown in http://tools.netsa.cert.org/silk/rwcut.html. Fields common to all network flows are defined in NetworkFlowLabelType (e.g., source IP, SNMP ingress, etc.). For additional references, see http://tools.netsa.cert.org/silk/analysis-handbook.pdf, http://tools.netsa.cert.org/silk/faq.html#ipfix-fields.

Property	Type	Mult	Description
Packet_Count	Common: IntegerObject AttributeType	0..1	Represents the number of packets in the flow.
Byte_Count	Common: IntegerObject AttributeType	0..1	Represents the number of Layer 3 bytes in the packets of the flow.
TCP_Flags	Common: HexBinary ObjectAttributeType	0..1	Specifies the union of all TCP flags observed over the life of the flow.
Start_Time	Common: IntegerObject AttributeType	0..1	Represents the SysUpTime at start of flow, i.e. the total time in milliseconds starting from when the router booted. There is another element "Start_Time+msec" which is the starting time of flow including milliseconds, but milliseconds are the resolution of Start_Time unless the -legacy-timestamps switch is specified, so "Start_Time+msec" is not defined separately.
Duration	Common: IntegerObject AttributeType	0..1	Specifies the duration of the flow. There is another element "Duration+msec" which is the starting time of flow including milliseconds, but milliseconds are the resolution of Duration unless the -legacy-timestamps switch is specified, so "Duration+msec" is not defined separately.
End_Time	Common: IntegerObject AttributeType	0..1	Represents the SysUpTime at end of flow. There is another element "End_Time+msec" which is the starting time of flow including milliseconds, but milliseconds are the resolution of End_Time unless the -legacy-timestamps switch is specified, so "End_Time+msec" is not defined separately.
Sensor_Info	NetFlowObj: SiLKSensorInfoType	0..1	Defines the fields associated with the sensor at the collection point.
ICMP_Type	Common: IntegerObject AttributeType	0..1	ICMP type for ICMP flows. Empty for non-ICMP flows.
ICMP_Code	Common: IntegerObject AttributeType	0..1	ICMP code for ICMP flows. Empty for non-ICMP flows.
Router_Next_Hop_IP	AddressObj: AddressObjectType	0..1	Router next hop IP.
Initial_TCP_Flags	PacketObj:TCPFlagsType	0..1	TCP flags on first packet in the flow.
Session_TCP_Flags	Common: HexBinary ObjectAttributeType	0..1	bit-wise OR of TCP flags over all packets except the first in the flow
Flow_Attributes	NetFlowObj: SiLKFlowAttributesTy	0..1	Flow attributes set by the flow generator.

	pe		
Flow_Application	PacketObj: IANAPortNumber RegistryType	0..1	Based on an examination of payload contents, this value = the port number traditionally used for that type of traffic (21 for FTP traffic even if actually routed over port 80). Documentation (http://tools.netsa.cert.org/silk/rwcut.html) says this is a "guess as to the content of the flow".
Src_IP_Type	NetFlowObj: SiLKAddressType	0..1	The type of the source IP in terms of whether the address is routable, external, etc.
Dest_IP_Type	NetFlowObj: SiLKAddressType	0..1	The type of the destination IP in terms of whether the address is routable, external, etc.
Src_Country_Code	NetFlowObj:SiLKCountryCodeType	0..1	A two-letter country code denoting the country of location of the source IP address.
Dest_Country_Code	NetFlowObj: SiLKCountryCodeType	0..1	A two-letter country code denoting the country of location of the destination IP address.
Src_MAPNAME	Common: StringObject AttributeType	0..1	User defined string for integrating external information into SiLK records. See documentation on SiLK pmap filter for details (defined in the prefix map associated with MAPNAME).
Dest_MAPNAME	Common: StringObject AttributeType	0..1	User defined string for integrating external information into SiLK records. See documentation on SiLK pmap filter for details (defined in the prefix map associated with MAPNAME).

3.2.19.40 SiLKFlowAttributesType (restriction Common:BaseObjectAttributeType)

SiLKFlowAttributesType specifies SiLK flow attributes, via a union of the SiLKFlowAttributesTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: NetFlowObj:SiLKFlowAttributesTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.19.41 SiLKFlowAttributesTypeEnum

The SiLKFlowAttributesTypeEnum specifies the flow attributes set by the flow generator. This is field 28 of the rwstats options. See http://tools.netsa.cert.org/silk/rwstats.html for more information.

Restriction base: string

Enumeration Value	Description
F (FIN flag)	Indicates that the flow generator saw additional packets in this flow following a packet with a FIN flag (excluding ACK packets).
T (Timeout)	Indicates that the flow generator prematurely created a record for a long-running connection due to a timeout. (When the flow generator yaf(1) is run with the --silk switch, it will prematurely create a flow and mark it with T if the byte count of the flow cannot be stored in a 32-bit value.)
C (Continuation)	Indicates that the flow generator created this flow as a continuation of long-running connection, where the previous flow for this connection met a timeout (or a byte threshold in the case of yaf).

3.2.19.42 SiLKAddressType (restriction Common:BaseObjectAttributeType)

SiLKAddressType specifies SiLK address types, via a union of the SiLKAddressTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: NetFlowObj:SiLKAddressTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.19.43 SiLKAddressTypeEnum

Environment variable allows user to specify the address type mapping file. A partial, typical list is currently given--see http://tools.netsa.cert.org/silk/addrtype.html for more information.

Restriction base: string

Enumeration Value	Description
non-routable (0)	Denotes a (non-routable) IP address.
internal(1)	Denotes an IP address internal to the monitored network.
routable_external(2)	Denotes an IP address external to the monitored network.

3.2.19.44 SiLKCountryCodeType (restriction Common:BaseObjectAttributeType)

SiLKCountryCodeType specifies country codes used by SiLK, via a union of the SiLKCountryCodeTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: NetFlowObj:SiLKCountryCodeTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.19.45 SiLKCountryCodeTypeEnum

Environment variable allows user to specify a country code mapping file. No enumerations are currently defined.

3.2.19.46 SiLKSensorInfoType

Defines elements associated with a SiLK sensor.

Property	Type	Mult	Description
Sensor_ID	Common: StringObject AttributeType	0..1	Name or ID of sensor at the collection point.
Class	NetFlowObj: SiLKSensorClassType	0..1	By default, only one "all" class. Others can be configured.
Type	NetFlowObj: SiLKDirectionType	0..1	Specifies the direction of traffic, which is enumerated by SiLKDirectionType.

3.2.19.47 SiLKDirectionType (restriction Common:BaseObjectAttributeType)

SiLKType specifies direction of SiLK traffic, via a union of the SiLKDirectionTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: NetFlowObj:SiLKDirectionTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.19.48 SiLKDirectionTypeEnum

Enumerates direction of traffic. Not all are currently enumerated.

Restriction base: string

Enumeration Value	Description
in	Denotes inbound traffic relative to a sensor.
inweb	Denotes inbound web traffic relative to a sensor. SiLK categorizes a flow as web if the protocol is TCP and either the source port or destination port is one of 80, 443, or 8080.
innull	Denotes null inbound traffic relative to a sensor.
out	Denotes outbound traffic relative to a sensor.
outweb	Denotes outbound web traffic relative to a sensor. SiLK categorizes a flow as web if the protocol is TCP and either the source port or destination port is one of 80, 443, or 8080.
outnull	Denotes null outbound traffic relative to a sensor.

3.2.19.49 SiLKSensorClassType (restriction Common:BaseObjectAttributeType)

SiLKSensorClassType specifies the sensor class, via a union of the SiLKSensorClassTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: NetFlowObj:SiLKSensorClassTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.19.50 SiLKSensorClassTypeEnum

Enumerates SiLK sensor classes. Currently just one class (all) is defined.

Restriction base: string

Enumeration Value	Description
all	Defines sensor class "all".

3.2.19.51 YAFRecordType

YAF (Yet Another Flowmeter) is bidirectional network flow meter. It processes packet data from pcap(3) dumpfiles as generated by tcpdump(1) or via live capture from an interface using pcap(3) into bidirectional flows, then exports those flows to IPFIX. (REF: http://www.usenix.org/event/lisa10/tech/full_papers/Inacio.pdf)

Property	Type	Mult	Description

Flow	NetFlowObj: YAFFlowType	0..1	The elements in a YAF record have been separated based on flow direction. These elements are defined for the general forward flow.
Reverse_Flow	NetFlowObj: YAFReverseFlowType	0..1	Some elements in a YAF record correspond to the reverse flow. These elements are given here.

3.2.19.52 YAFFlowType

These elements of a YAF record correspond to the flow generally or to the forward portion of the flow. Elements common to all network flow objects are defined in the NetworkFlowLabelType (src ip address, ingress/egress interface).

Property	Type	Mult	Description
Flow_Start_Milliseconds	Common: IntegerObject AttributeType	0..1	Flow start time in milliseconds since 1970-01-01 00:00:00 UTC
Flow_End_Milliseconds	Common: IntegerObject AttributeType	0..1	Flow end time in milliseconds since 1970-01-01 00:00:00 UTC
Octet_Total_Count	Common: IntegerObject AttributeType	0..1	Number of octets in packets in forward direction of flow. May be encoded in 4 octets using IPFIX reduced-length encoding.
Packet_Total_Count	Common: IntegerObject AttributeType	0..1	Number of packets in forward direction of flow.
Flow_End_Reason	Common: HexBinary ObjectAttributeType	0..1	The reason for Flow termination. It may contain SiLK-specific tags. The range of values may include the following: 0x01: idle timeout (the Flow was terminated because it was considered to be idle). 0x02: active timeout (the Flow was terminated for reporting purposes while it was still active, for example, after the maximum lifetime of unreported Flows was reached). 0x03: end of Flow detected (the Flow was terminated because the Metering Process detected signals indicating the end of the Flow, for example, the TCP FIN flag.) 0x04: forced end (the Flow was terminated because of some external event, for example, a shutdown of the Metering Process initiated by a network management application.) 0x05: lack of resources (the Flow was terminated because of lack of resources available to the Metering Process and/or the Exporting Process.) See http://www.iana.org/assignments/ipfix/ipfix.xml for more information.
SiLK_App_Label	Common: IntegerObject AttributeType	0..1	The SiLK_App_Label is the port number that is traditionally used for that type of traffic (see the /etc/services file on most UNIX systems). For example, traffic that the flow generator recognizes as FTP will have a value of 21, even if that traffic is being routed through the standard HTTP/web port (80).
Payload_Entropy	Common: IntegerObject	0..1	Shannon Entropy calculation of the forward payload data. The calculation generates a real

	AttributeType		number value between 0.0 and 8.0. That number is then converted into an 8-bit integer value between 0 and 255. Roughly, numbers above 230 are generally compressed (or encrypted) and numbers centered around approximately 140 are English text. Lower numbers carry even less information content.
ML_App_Label	Common: HexBinary ObjectAttributeType	0..1	Machine-learning app label
TCP_Flow	NetFlowObj: YAFTCPFlowType	0..1	Contains TCP-related information of the network flow.
Vlan_ID_MAC_Addr	AddressObj: AddressObjectType	0..1	The MAC adress.
Passive_OS_FIngerprinting	NetFlowObj: YAFOSInformationType	0..1	OS name and version.
First_Packet_Banner	Common: HexBinary ObjectAttributeType	0..1	First forward packet IP payload.
Second_Packet_Banner	Common: HexBinary ObjectAttributeType	0..1	Second forward packet IP payload.
N_Bytes_Payload	Common: HexBinary ObjectAttributeType	0..1	Initial n bytes of forward direction of applications payload.

3.2.19.53 YAFReverseFlowType

These elements correspond to the reverse flow captured by in YAF record.

Property	Type	Mult	Description
Reverse_Octet_Total_Count	Common: IntegerObject AttributeType	0..1	Number of octets in packets in reverse direction of flow. May be encoded in 4 octets using IPFIX reduced-length encoding.
Reverse_Packet_Total_Count	Common: IntegerObject AttributeType	0..1	Number of packets in reverse direction of flow.
Reverse_Payload_Entropy	Common: IntegerObject AttributeType	0..1	Shannon Entropy calculation of the reverse payload data. The calculation generates a real number value between 0.0 and 8.0. That number is then converted into an 8-bit integer value between 0 and 255. Roughly, numbers above 230 are generally compressed (or encrypted) and numbers centered around approximately 140 are English text. Lower numbers carry even less information content.
Reverse_Flow_Delta_Milliseconds	Common: IntegerObject AttributeType	0..1	RTT of initial handshake.
TCP_Reverse_Flow	NetFlowObj:YAFTCPFlowType	0..1	The associated elements relate to the reverse packets of the flow.
Reverse_Vlan_ID_MAC_Addr	AddressObj: AddressObjectType	0..1	Reverse MAC address.
Reverse_Passive_OS_Fingerprinting	NetFlowObj:YAFOSInformationType	0..1	OS name and version of the reverse flow.
Reverse_First_Packet	Common:	0..1	First reverse packet IP payload.

	HexBinary ObjectAttributeType		
Reverse_N_Bytes_Payload	Common: HexBinary ObjectAttributeType	0..1	Initial n bytes of reverse direction of flow payload

3.2.19.54 YAFTCPFlowType

Contains TCP-related information of the network flow.

Property	Type	Mult	Description
TCP_Sequence_Number	Common: IntegerObject AttributeType	0..1	TCP sequence number.
Initial_TCP_Flags	PacketObj:TCPFlagsType	0..1	TCP flags of the first packet.
Union_TCP_Flags	Common: HexBinary ObjectAttributeType	0..1	The union of the TCP flags of the 2...nth packet.

3.2.19.55 YAFOSInformationType

This represents information about an operating system gathered through OS fingerprinting techniques. We need to reference CPE at some point.

Property	Type	Mult	Description
OS_Name	Common: StringObject AttributeType	0..1	The name of the operating system
OS_Version	Common: StringObject AttributeType	0..1	The version of the operating system

3.2.20 NetworkPacketType (extends Common:DefinedObjectType)

The definition of network packet is based on the TCP/IP model/Internet protocol suite. In the TCP/IP stack, "packet" is generally defined as IP header plus payload, but we also include the LinkLayer from the OSI model, which defines the physical network interfaces and routing protocols. Protocol fields are provided but requirements are not enforced/captured; all fields are optional.

Property	Type	Mult	Description
Link_Layer	PacketObj:LinkLayerType	0..1	The Link Layer is the lowest layer of the TCP/IP network stack and is comprised of physical and logical protocols that operate between adjacent nodes of a network segment or a WAN connection.
Internet_Layer	PacketObj:InternetLayerType	0..1	Internet layer characterizes information about the network layer of this Network Packet. The network layer is one layer from the 7-layer OSI Model.
Transport_Layer	PacketObj:TransportLayerType	0..1	Transport layer characterizes information about the transport layer of this Network Packet. The transport layer is one layer from the 7-layer OSI Model.

3.2.20.1 LinkLayerType

A link layer protocol is a hardware interface protocol, such as Ethernet, or a logical link routing protocol, such as ARP.

Property	Type	Mult	Description
Physical_Interface	PacketObj:PhysicalInterfaceType	0..1	Physical Interface characterizes one hardware interface of a link layer connection.
Logical_Protocols	PacketObj:LogicalProtocolType	0..1	Logical Protocols characterizes the logical protocol of a link layer connection. One example of a logical protocol is ARP.

3.2.20.2 PhysicalInterfaceType

Multiple interface types exist - only most common (Ethernet) included now. Others will be added later as needed.

Property	Type	Mult	Description
Ethernet	PacketObj:EthernetInterfaceType	0..1	Ethernet sends network packets from the sending host to one or more receiving hosts. (REF: IEEE 802.3; http://wiki.wireshark.org/Ethernet)

3.2.20.3 LogicalProtocolType

Logical Protocols characterizes the logical protocol of a link layer connection. One example of a logical protocol is ARP.

Property	Type	Mult	Description
ARP_RARP	PacketObj:ARPType	0..1	ARP is a logical protocol used for resolution of network layer addresses (e.g., IP addresses) into link layer addresses (e.g., MAC addresses). RARP is a logical protocol used by a host computer to request its network layer address when it has its link layer address.
NDP	PacketObj:NDPType	0..1	Neighbor Discovery Protocol (NDP) is used with IPv6 to determine the link-layer addresses for neighbors. Corresponds to combination of IPv4 protocols: ARP, ICMP Router Discovery, and ICMP Redirect.

3.2.20.4 EthernetInterfaceType

Ethernet sends network packets from the sending host to one or more receiving hosts. (REF: IEEE 802.3; http://wiki.wireshark.org/Ethernet)

Property	Type	Mult	Description
Ethernet_Header	PacketObj:EthernetHeaderType	0..1	The ethernet header includes information such as source MAC address, destination MAC address, and more.

3.2.20.5 EthernetHeaderType

Ethernet header characterizes and ethernet header and includes information such as source MAC address, destination MAC address, and more.

Property	Type	Mult	Description
Destination_MAC_Addr	AddressObj: AddressObjectType	0..1	Destination MAC Addr characterizes the destination MAC Address of the ethernet frame.
Source_MAC_Addr	AddressObj: AddressObjectType	0..1	Source MAC Addr characterizes the source MAC Address of the ethernet frame.
Type_Or_Length	PacketObj:TypeLengthType	0..1	Type or Length characterizes either the length of the ethernet frame or the protocol type of the network layer.
Checksum	Common: HexBinary ObjectAttributeType	0..1	Checksum characterizes the Frame Check sequence of an ethernet frame.

3.2.20.6 TypeLengthType

0-1500 then it is a length field. Otherwise, it defines the protocol type of the Internet layer.

Property	Type	Mult	Description
Length	Common: HexBinary ObjectAttributeType	0..1	Length characterizes the length of the ethernet frame.
Internet_Layer_Type	PacketObj:IANAEtherType	0..1	two-octet field in an Ethernet frame. specifies protocol encapsulated in the payload of ethernet frame.

3.2.20.7 ARPType

The Address Resolution Protocol is a request and reply protocol that runs encapsulated by the line protocol. It is communicated within the boundaries of a single network, never routed across internetwork nodes. This property places ARP into the Link Layer. It is encapsulated. REF: http://www.comptechdoc.org/independent/networking/guide/netarp.html

Property	Type	Mult	Description
Hardware_Addr_Type	PacketObj:IANAHardwareType	0..1	Characterizes the type of hardware address specified in an ARP message.
Proto_Addr_Type	PacketObj:IANAEtherType	0..1	ProtoAddrType characterizes the type of protocol address being mapped. For IPv4 addresses, value = 0x0800.
Hardware_Addr_Size	Common: HexBinary ObjectAttributeType	0..1	Harware_Addr_Size represents the byte size of the hardware address. For Ethernet or other IEEE 802 MAC addresses, the value is 6.
Protol_Addr_Size	Common: HexBinary ObjectAttributeType	0..1	Proto_Addr_Size represents the byte size of the protocol address. IPv4 addresses = 4.
Op_Type	PacketObj:ARPOpType	0..1	Op_Type characterizes the type of operation. 1 = ARP request, 2=ARP reply, 3=RARP request, 4=RARP reply.
Sender_Hardware_Addr	AddressObj: AddressObjectType	0..1	Sender_Hardware_Addr characterizes the sender's hardware address (e.g., MAC address).
Sender_Protocol_Addr	AddressObj: AddressObjectType	0..1	Sender_Protocol_Addr characterizes the

			sender's IP address.
Recip_Hardware_Addr	AddressObj: AddressObjectType	0..1	Recip_Sender_Hardware Addr characterizes the recipients's hardware address (e.g., MAC address).
Recip_Protocol_Addr	AddressObj: AddressObjectType	0..1	Recip Protocol Addr characterizes the recipient's IP address.

3.2.20.8 ARPOpType (restriction Common:BaseObjectAttributeType)

ARPOpType specifies types of ARP operations, via a union of the ARPOpTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:ARPOpTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.9 ARPOpTypeEnum

ARPOpTypeEnum contains the various ARP Operation Types.

Restriction base: string

Enumeration Value	Description
ARP request(1)	Indicates the ARP request operation, or value 1 in the OPER field of an ARP packet.
ARP reply(2)	Indicates the ARP reply operation, or value 2 in the OPER field of an ARP packet.
RARP request(3)	Indicates the RARP request operation, or value 3 in the OPER field of an ARP packet.
RARP reply(4)	Indicates the RARP reply operation, or value 4 in the OPER field of an ARP packet.

3.2.20.10 NDPType

NDP Type characterizes NDP (Neighbor Discover Protocol) IPv6 packets. NDP defines five ICMPv6 packet types. RFC 2461: http://tools.ietf.org/html/rfc4861

Property	Type	Mult	Description
ICMPv6_Header	PacketObj:ICMPv6HeaderType	0..1	ICMPv6 Header characterizes an ICMPv6 header.
Router_Solicitation	PacketObj: RouterSolicitationType	0..1	Hosts send Router Solicitations in order to prompt routers to generate Router Advertisements quickly (type=133; code=0).
Router_Advertisement	PacketObj: RouterAdvertisementType	0..1	Routers send out Router Advertisement messages periodically, or in response to Router Solicitations (type=134; code=0).
Neighbor_Solicitation	PacketObj: NeighborSolicitationType	0..1	Nodes send Neighbor Solicitations to request the link-layer address of a target node while also providing their own link-layer address to the target. Neighbor Solicitations are multicast when the node needs to resolve an address and unicast when the node seeks to verify the reachability of a neighbor (type=135; code=0).
Neighbor_Advertisement	PacketObj: NeighborAdvertisementType	0..1	A node sends Neighbor Advertisements in response to Neighbor Solicitations and sends

			unsolicited Neighbor Advertisements in order to (unreliably) propagate new information quickly (type=136; code=0).
Redirect	PacketObj:RedirectType	0..1	Routers send Redirect packets to inform a host of a better first-hop node on the path to a destination. Hosts can be redirected to a better first-hop router but can also be informed by a redirect that the destination is in fact a neighbor. The latter is accomplished by setting the ICMP Target Address equal to the ICMP Destination Address (type=137; code=0).

3.2.20.11 RouterSolicitationType

Hosts send Router Solicitations in order to prompt routers to generate Router Advertisements quickly.(type=133; code=0)

Property	Type	Mult	Description
Options	PacketObj: RouterSolicitationOptionsType	0..∞	Router Solicitation messages include zero or more options, some of which may appear multiple times in the same message.

3.2.20.12 RouterSolicitationOptionsType

Neighbor Discovery messages include zero or more options, some of which may appear multiple times in the same message.

Property	Type	Mult	Description
Src_Link_Addr	PacketObj:NDPSrcLinkAddrType	0..1	Src Link Addr characterizes the Source Link-Layer Address option.

3.2.20.13 RouterAdvertisementType

Routers send out Router Advertisement messages periodically, or in response to Router Solicitations. (type=134; code=0)

Property	Type	Mult	Description
managed_address_config_flag	boolean	1..1	1-bit "Managed address configuration" flag. When set, it indicates that addresses are available via Dynamic Host Configuration Protocol. If the M flag is set, the O flag is redundant and can be ignored because DHCPv6 will return all available configuration information.
other_config_flag	boolean	1..1	1-bit "Other configuration" flag. When set, it indicates that other configuration information is available via DHCPv6. Examples of such information are DNS-related information or information on other servers within the network.
Cur_Hop_Limit	Common: IntegerObject AttributeType	0..1	8-bit unsigned integer. The default value that should be placed in the Hop Count field of the IP header for outgoing IP packets. A value of zero

			means unspecified (by this router).
Router_Lifetime	Common: IntegerObject AttributeType	0..1	16-bit unsigned integer. The lifetime associated with the default router in units of seconds. The field can contain values up to 65535 and receivers should handle any value, while the sending rules in Section 6 limit the lifetime to 9000 seconds.
Reachable_Time	Common: IntegerObject AttributeType	0..1	32-bit unsigned integer. The time, in milliseconds, between retransmitted Neighbor Solicitation messages. Used by address resolution and the Neighbor Unreachability Detection algorithm. A value of zero means unspecified (by this router).
Retrans_Timer	Common: IntegerObject AttributeType	0..1	32-bit unsigned integer. The time, in milliseconds, between retransmitted Neighbor Solicitation messages. Used by address resolution and the Neighbor Unreachability Detection algorithm. A value of zero means unspecified (by this router)
Options	PacketObj: RouterAdvertisement OptionsType	0..1	Neighbor Discovery messages include zero or more options, some of which may appear multiple times in the same message.

3.2.20.14 RouterAdvertisementOptionsType

Router Advertisement messages include zero or more options, some of which may appear multiple times in the same message.

Property	Type	Mult	Description
Src_Link_Addr	PacketObj:NDPSrcLinkAddrType	0..1	Src Link Addr characterizes the Source Link-Layer Address option.
MTU	PacketObj:NDPMTUType	0..1	32-bit unsigned integer. The recommended MTU for the link.
Prefix_Info	PacketObj:NDPPrefixInfoType	0..1	Prefix Info characterizes Prefix Information for Router Advertisement Options.

3.2.20.15 NeighborSolicitationType

Nodes send Neighbor Solicitations to request the link-layer address of a target node while also providing their own link-layer address to the target. Neighbor Solicitations are multicast when the node needs to resolve an address and unicast when the node seeks to verify the reachability of a neighbor. (type=135; code=0)

Property	Type	Mult	Description
Target_IPv6_Addr	AddressObj: AddressObjectType	0..1	The IP address of the target of the solicitation.
Options	PacketObj: NeighborSolicitationOptionsType	0..1	Neighbor Solicitation messages include zero or more options, some of which may appear multiple times in the same message.

3.2.20.16 NeighborSolicitationOptionsType

Neighbor Solicitation messages include zero or more options, some of which may appear multiple times in the same message.

Property	Type	Mult	Description

Src_Link_Addr	PacketObj:NDPSrcLinkAddrType	0..1	Src Link Addr characterizes the Source Link-Layer Address option.

3.2.20.17 NeighborAdvertisementType

A node sends Neighbor Advertisements in response to Neighbor Solicitations and sends unsolicited Neighbor Advertisements in order to (unreliably) propagate new information quickly. (type=136; code=0)

Property	Type	Mult	Description
override_flag	boolean	1..1	Override flag. When set, the O-bit indicates that the advertisement should override an existing cache entry and update the cached link-layer address.
router_flag	boolean	1..1	Router flag. When set, the R-bit indicates that the sender is a router. The R-bit is used by Neighbor Unreachability Detection to detect a router that changes to a host.
solicited_flag	boolean	1..1	Solicited flag. When set, the S-bit indicates that the advertisement was sent in response to a Neighbor Solicitation from the Destination address. The S-bit is used as a reachability confirmation for Neighbor Unreachability Detection.
Target_IPv6_Addr	AddressObj: AddressObjectType	0..1	The IP address of the target of the advertisement.
Options	PacketObj:NeighborOptionsType	0..1	Neighbor Advertisement messages include zero or more options, some of which may appear multiple times in the same message.

3.2.20.18 NeighborOptionsType

Neighbor Advertisement messages include zero or more options, some of which may appear multiple times in the same message.

Property	Type	Mult	Description
Target_Link_Addr	PacketObj: NDPTargetLinkAddrType	0..1	Target Link Addr characterizes the Target Link-Layer Address option.

3.2.20.19 RedirectType

Routers send Redirect packets to inform a host of a better first-hop node on the path to a destination. Hosts can be redirected to a better first-hop router but can also be informed by a redirect that the destination is in fact a neighbor. The latter is accomplished by setting the ICMP Target Address equal to the ICMP Destination Address. (type=137; code=0)

Property	Type	Mult	Description
Target_IPv6_Addr	AddressObj: AddressObjectType	0..1	An IP address that is a better first hop to use for the ICMP Destination Address.
Dest_IPv6_Addr	AddressObj: AddressObjectType	0..1	The IP address of the destination that is redirected to the target.
Options	PacketObj: RedirectOptionsType	0..1	Redirect messages include zero or more options,

			some of which may appear multiple times in the same message.

3.2.20.20 RedirectOptionsType

Redirect messages include zero or more options, some of which may appear multiple times in the same message.

Property	Type	Mult	Description
Target_Link_Addr	PacketObj: NDPTargetLinkAddrType	0..1	The linnk-layer address for the target.
Redirected_Header	PacketObj: NDPRedirectedHeaderType	0..1	As much as possible of the IP packet that triggered the sending of the Redirect message without making the redirect packet exceed the minimum MTU specified in the IPv6 protocol.

3.2.20.21 NDPSrcLinkAddrType

Src Link Addr characterizes the Source Link-Layer Address option. (type=1)

Property	Type	Mult	Description
Length	Common: IntegerObject AttributeType	0..1	The length of the option (including the type and length fields) in units of 8 octets.
Link_Layer_MAC_Addr	AddressObj: AddressObjectType	0..1	The variable length link-layer address. The content and format of this field (including byte and bit ordering) is expected to be specified in specific documents that describe how IPv6 operates over different link layers.

3.2.20.22 NDPTargetLinkAddrType

Target Link Addr characterizes the Target Link-Layer Address option. (type=2)

Property	Type	Mult	Description
Length	Common: IntegerObject AttributeType	0..1	The length of the option (including the type and length fields) in units of 8 octets.
Link_Layer_MAC_Addr	AddressObj: AddressObjectType	0..1	The variable length link-layer address. The content and format of this field (including byte and bit ordering) is expected to be specified in specific documents that describe how IPv6 operates over different link layers.

3.2.20.23 NDPPrefixInfoType

Prefix Info characterizes Prefix Information for Router Advertisement Options. It provides hosts with on-link prefixes and prefixes for Address Autoconfiguration. (type=3). RFC 4861.

Property	Type	Mult	Description
addr_config_flag	boolean	1..1	1-bit autonomous address-configuration flag. When set indicates that this prefix can be usd for stateless address configuration.
link_flag	boolean	1..1	1-bit on-link flag. When set, indicates that this prefix can be used for on-link determintation. When not set the advertisement makes no statement about

			on-link or off-link properties of the prefix.
Length	Common: IntegerObject AttributeType	0..1	Length characterizes the length of the option (the number of valid leading bits in the prefix), and is represented as a 32-bit integer.
Prefix_Length	Common: IntegerObject AttributeType	0..1	8-bit unsigned integer. The number of leading bits in the Prefix that are valid. The value ranges from 0 to 128. The prefix length field provides necessary information for on-link determination (when combined with the L flag in the prefix information option).
Valid_Lifetime	Common: IntegerObject AttributeType	0..1	32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that the prefix is valid for the purpose of on-link determination. A value of all one bits (0xffffffff) represents infinity.
Preferred_Lifetime	Common: IntegerObject AttributeType	0..1	32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that addresses generated from the prefix via stateless address autoconfiguration remain preferred.
Prefix	PacketObj:PrefixType	0..1	The Prefix is an IP address or a prefix of an IP address.

3.2.20.24 NDPRedirectedHeaderType

The redirected header option is used in redirect messages and contains all or part of the packet that is being redirected. (type=4)

Property	Type	Mult	Description
Length	Common: IntegerObject AttributeType	0..1	The length of the option (including the type and length fields) in units of 8 octets.
IPHeader_And_Data	Common: HexBinary ObjectAttributeType	0..1	As much as possible of the IP packet that triggered the sending of the redirect without making redirect packet larger than MTU.

3.2.20.25 NDPMTUType

The MTU option is used in Router Advertisement messages to ensure that all nodes on a link use the same MTU value in those cases where the link MTU is not well known. (type=5).

Property	Type	Mult	Description
Length	Common: IntegerObject AttributeType	0..1	The length of the MTU option type: length=1.
MTU	Common: IntegerObject AttributeType	0..1	The recommended MTU for the link. 32-bit unsigned integer.

3.2.20.26 InternetLayerType

The Internet layer is the group of methods, protocols, and specifications that are used to transport packets from the origiating host across network boundaries. Not all protocols are currently defined, just those most commonly used: IPv4, ICMPv4, IPv6, ICMPv6. Other protocols will be added as needed. (http://en.wikipedia.org/wiki/Internet_layer)

Property	Type	Mult	Description
IPv4	PacketObj: IPv4PacketType	0..1	Internet Protocol version 4 (IPv4) is a connectionless protocol for use on packet-switched link layer networks (e.g., Ethernet).
ICMPv4	PacketObj: ICMPv4PacketType	0..1	ICMP is chiefly used the the operating systems of networked computers to send error messages indicating, for example, that a requested serivce is not available or that a host or router could not be reached (http://en.wikipedia.org/wiki/Internet_Control_Message_Protocol; REF: http://www.networksorcery.com/enp/protocol/icmp.htm).
IPv6	PacketObj: IPv6PacketType	0..1	Internet Protocol version 6 (IPv6) is intended to succeed IPv4, and like IPv4 it is a connectionless protocol for use on packet-switched link layer networks.
ICMPv6	PacketObj: ICMPv6PacketType	0..1	ICMPv6 is the implementation of the ICMP for INv6. ICMPv6 performs error reporting and diagnostic functions.

3.2.20.27 IPv4PacketType

Internet Protocol version 4 (IPv4) is a connectionless protocol for use on packet-switched link layer networks (e.g., Ethernet). REF: RFC 791; http://en.wikipedia.org/wiki/IPv4.

Property	Type	Mult	Description
IPv4_Header	PacketObj:| IPv4HeaderType	0..1	The IPv4 header provides addressing, and internet modules use fields in the header to fragment and reassemble internet datagrams when necessary for transmission through small packet networks.
Data	Common: HexBinary ObjectAttributeType	0..1	The data portion of an IP packet is interpreted based on the value of the Protocol header field. Actual field values will probably be specified in the elements of the different network layers, but we provide an element here to capture any data as necessary.

3.2.20.28 IPv4HeaderType

The IPv4 header provides addressing, and internet modules use fields in the header to fragment and reassemble internet datagrams when necessary for transmission through small packet networks. REF: RFC 791.

Property	Type	Mult	Description
IP_Version	PacketObj:IPVersionType	0..1	The version field indicates the format of the internet header. For IP v4, the version is 4.
Header_Length	Common: IntegerObject AttributeType	0..1	The Internet Header Length specifies the length of IP packet header in 32 bit words. Min value = 5.
DSCP	Common: HexBinary ObjectAttributeType	0..1	Originally defined as the Type of Service field, the Differentiated Services Code Point (DSCP) field is now defined by RFC 2474 for Differentiated

			services (DiffServ). New technologies are emerging that require real-time data streaming and therefore make use of the DSCP field. An example is Voice over IP (VoIP), which is used for interactive data voice exchange (http://en.wikipedia.org/wiki/IPv4).
ECN	Common: HexBinary ObjectAttributeType	0..1	Explicit Congestion Notification: This field is defined in RFC 3168 and allows end-to-end notification of network congestion without dropping packets. ECN is an optional feature that is only used when both endpoints support it and are willing to use it. It is only effective when supported by the underlying network. (http://en.wikipedia.org/wiki/IPv4).
Total_Length	Common: HexBinary ObjectAttributeType	0..1	This 16-bit field defines the entire datagram size, including header and data, in bytes.
Identification	Common: PositiveInteger ObjectAttributeType	0..1	The Identification field is primarily used for uniquely identifying fragments of an original IP datagram. (http://en.wikipedia.org/wiki/IPv4)
Flags	PacketObj:IPv4FlagsType	0..1	This is a three-bit field used to control or identify fragments. An element has been defined for each bit with associated enumerated types.
Fragment_Offset	Common: HexBinary ObjectAttributeType	0..1	The fragment offset field is 13 bits long and specifies the offset of a particular fragment relative to the beinnng of the original unfragmented IP datagram. http://en.wikipedia.org/wiki/IPv4
TTL	Common: HexBinary ObjectAttributeType	0..1	This 8-bit field helps prevent datagrams from persisting on an internet (it limits a datagram's lifetime).
Protocol	PacketObj: IANAAssignedIPNumbersType	0..1	This field defines the protocol used in the data portion of the IP datagram. The type of this element is an enumerated list of IP protocol numbers as maintained by the Internet Assigned Numbers Authority.
Checksum	Common: HexBinary ObjectAttributeType	0..1	This field is a 16-bit checksum used for error-checking of the header.
Src_IPv4_Addr	AddressObj: AddressObjectType	0..1	This field is the IPv4 address of the sender of the packet.
Dest_IPv4_Addr	AddressObj: AddressObjectType	0..1	This field is the IPv4 address of the receiver of the packet.
Option	PacketObj: IPv4OptionType	0..∞	The IPv4 option field is variable in length with zero or more options. It is not often used. http://en.wikipedia.org/wiki/IPv4

3.2.20.29 IPv4FlagsType

These flag types are used to control or identify fragments in an IP packet. It is a three-bit field, each of the three bits are defined by an element with a string value that indicates the meaning of whether or not the bit is set.

Property	Type	Mult	Description
Reserved	Common: IntegerObject AttributeType	0..1	Bit 0: This bit value (0) is reserved and must be zero.
Do_Not_Fragment	PacketObj: DoNotFragmentType	0..1	Bit 1: This is the "don't fragment" bit. Values are specified in the DoNotFragmentType.
More_Fragments	PacketObj: MoreFragmentsType	0..1	Bit 2: This is the "more fragments" bit. Values are specified in the MoreFragmentsType.

3.2.20.30 DoNotFragmentType (restriction Common:BaseObjectAttributeType)

DoNotFragmentType specifies fragmenting options, via a union of the DoNotFragmentTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:DoNotFragmentTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.31 DoNotFragmentTypeEnum

This type enumerates the meaning of the Do Not Fragment bit used in IPv4 flags.

Restriction base: string

Enumeration Value	Description
fragementifnecessary(0)	Indicates that the router or other device should fragment the packet if necessary, especially if the packet size is bigger than the MTU of an outgoing interface.
donotfragment(1)	Indicates that the router or other device should NOT fragment the packet in any circumstance.

3.2.20.32 MoreFragmentsType (restriction Common:BaseObjectAttributeType)

MoreFragmentsType specifies whether there are more fragments, via a union of the MoreFragmentsTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:MoreFragmentsTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.33 MoreFragmentsTypeEnum

This type enumerates the meaning of the More Fragments bit used in IPv4 flags.

Restriction base: string

Enumeration Value	Description
lastfragment(0)	Indicates that the last fragment has been received. In other words, the "more fragments" flag is set to 0.
morefragmentstofollow(1)	Indicates that more fragments need to be received. In other words, the "more fragments" flag is set.

3.2.20.34 IPv4OptionType

The IPv4 option field is variable in length with zero or more options.

Property	Type	Mult	Description
Copy_Flag	PacketObj:IPv4CopyFlagType	0..1	The copied flag indicates that this option is copied into all fragments on fragmentation. 1 bit. They are represented in this element by a string which specifies their value.
Class	PacketObj:IPv4ClassType	0..1	The option class is represented by 2 bits where 0 = control; 1 = reserved for future use; 2 = debugging and measurement; 3 = reserved for future use. These enumerated values are defined for this element.
Option	PacketObj:IPv4OptionsType	0..1	The Internet Protocol has provision for optional header fields indeitified by an option type. These types are enumerated in the IPv4OptionsType.

3.2.20.35 IPv4CopyFlagType (restriction Common:BaseObjectAttributeType)

IPv4CopyFlagType specifies value of IPv4 copy flag, via a union of the IPv4CopyFlagTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:IPv4CopyFlagTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.36 IPv4CopyFlagTypeEnum

The copy flag indicates whether the option is copied into all fragments on fragmentation (0=not copied; 1=copied). This information is also captured in the IPv4OptionsTypeEnum which lists all options, which incorporates copy and class numbers.

Restriction base: string

Enumeration Value	Description
donotcopy(0)	Indicates that the options need NOT be copied into all fragments of a fragmented packet.
copy(1)	Indicates that the options need to be copied into all fragments of a fragmented packet.

3.2.20.37 IPv4ClassType (restriction Common:BaseObjectAttributeType)

IPv4ClassType specifies IPv4 class type, via a union of the IPv4ClassTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:IPv4ClassTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.38 IPv4ClassTypeEnum

The option class is represented by 2 bits. The explicit meanings are captured here in an enumerated list. This information is also captured in the IPv4OptionsTypeEnum which lists all options, which incorporates copy and class numbers.

Restriction base: string

Enumeration Value	Description
control(0)	Indicates the "control" options.
reserved(1)	Indicates a reserved value.
debuggingandmeasurement(2)	Indicates the debugging and measurement options.
reserved(3)	Indicates a reserved value.

3.2.20.39 IPv4OptionsType (restriction Common:BaseObjectAttributeType)

IPv4OptionsType specifies IPv4 options, via a union of the IPv4OptionsTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:IPv4OptionsTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.40 IPv4OptionsTypeEnum

The Internet Protocol (IP) has provision for optional header fields identified by an option type field. Options 0 and 1 are exactly one octet which is their type field. All other options have their one octet type field, followed by a one octet length field, followed by length-2 octets of option data. The option type field is sub-divided into a one bit copied flag, a two bit class field, and a five bit option number. These taken together form an eight bit value for the option type field. IP options are commonly refered to by this value. The IPv4OptionsEnum enumerates the options numbers that can be applied in IP. See http://www.iana.org/assignments/ip-parameters for more information.

Restriction base: string

Enumeration Value	Description
endofoptionslist(0)	Indicates the End of Options List option, or EOOL.
nop(1)	Indicates the No Operation option, or NOP.
security(2)	Indicates the Security option, or SEC.
loosesourceroute(3)	Indicates the Loose Source Route option, or LSR.
timestamp(4)	Indicates the Time Stamp option, or TS.
extendedsecurity(5)	Indicates the Extended Security option, or E-SEC.
commercialsecurity(6)	Indicates the Commercial Security option, or CIPSO.
recordroute(7)	Indicates the Record Route option, or RR.
streamidentifier(8)	Indicates the Stream ID option, or SID.
strictsourceroute(9)	Indicates the Strict Source Route option, or SSR.
experimentalmeasure(10)	Indicates the Experimental Measurement option, or ZSU.
mtuprobe(11)	Indicates the MTU probe option, or MTUP.
mtureply(12)	Indicates the MTU reply option, or MTUR.
experimentalflowcontrol(13)	Indicates the Experimental Flow Control option, or FINN.

experimentalaccesscontrol(14)	Indicates the Experimental Access Control option, or FINN.
encode(15)	
imitrafficdescriptor(16)	Indicates the IMI Traffic Descriptor option, or IMITD.
extendedip(17)	Indicates the Extended Internet Protocol option, or EIP.
traceroute(18)	Indicates the Trace Route option, or TR.
addressextension(19)	Indicates the Address Extension option, or ADDEXT.
routeralert(20)	Indicates a Router Alert option, or RTRALT.
selectivedirectedbroadcasemode(21)	Indicates a Selective Directed Broadcast option, or SDB.
dynamicepacketstate(23)	Indicates the Dynamic Packet State option, or DPS.
upstreammulticastpacket(24)	Indicates the Upstream Multicast Packet option, or UMP.
quickstart(25)	Indicates the Quick-Start option, or QS.
exp(30)	Indicates the RFC3692-style Experiment option, or EXP.

3.2.20.41 IPv6PacketType

Internet Protocol version 6 (IPv6) is intended to succeed IPv4, and like IPv4 it is a connectionless protocol for use on packet-switched link layer networks. RFC 3513, RFC 2460, http://en.wikipedia.org/wiki/IPv6.

Property	Type	Mult	Description
IPv6_Header	PacketObj:IPv6HeaderType	0..1	IPv6 headers is a simplification of the IPv4 header.
Ext_Headers	PacketObj:IPv6ExtHeaderType	0..∞	In IPv6, optional internet-layer information is encoded in separate headers that may be placed between the IPv6 header and the upper-layer header in a packet. http://tools.ietf.org/html/rfc2460

3.2.20.42 IPv6HeaderType

The IPv6 header is a simplification of the IPv4 header.

Property	Type	Mult	Description
IP_Version	PacketObj: IPVersionTypeEnum	0..1	4-bit Internet Protocol version number =6.
Traffic_Class	Common: HexBinary ObjectAttributeType	0..1	8-bit traffic class field. Available for use by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets. http://tools.ietf.org/html/rfc2460#section-7
Flow_Label	Common: HexBinary ObjectAttributeType	0..1	20-bit flow label. Used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as non-default quality of service. http://tools.ietf.org/html/rfc2460#section-6.
Payload_Length	Common: HexBinary ObjectAttributeType	0..1	16-bit unsigned integer. Length of the IPv6 payload (the rest of the packet following the IPv6 header) in octets. Any extension headers are considered part of the payload.
Next_Header	PacketObj: IANAAssigned IPNumbersType	0..1	8-bit selector. Identifies the type of header immediately following the IPv6 header. Uers the same values as the IPv4 protocol field.
TTL	Common:	0..1	TTL/hop limit specifies how many times a packet

	PositiveInteger ObjectAttributeType		can be forwarded. 8-bit unsigned integer.
Src_IPv6_Addr	AddressObj: AddressObjectType	0..1	128-bit address of the originator of the packet.
Dest_IPv6_Addr	AddressObj: AddressObjectType	0..1	128-bit address of the intended recipient of the packet.

3.2.20.43 IPv6ExtHeaderType

In IPv6, optional internet-layer information is encoded in separate headers that may be placed between the IPv6 header and the upper-layer header in a packet. An IPv6 packet may carry zero, one, or more extension headers, each identified by the Next Header field of the preceding header. http://tools.ietf.org/html/rfc2460

Property	Type	Mult	Description
Hop_by_Hop_Options	PacketObj: HopByHop OptionsType	0..1	The Hop-by-Hop Options header is used to carry optional information that must be examined by every node along a packet's delivery path. It carries a variable number of type-length-value (TLV) encoded options.
Routing	PacketObj: RoutingType	0..1	The Routing header is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination. http://tools.ietf.org/html/rfc2460
Fragment	PacketObj: FragmentType	0..1	The Fragment header is used by an IPv6 source to send a packet larger than would fit in the path MTU. A fragment packet begins with an unfragmentable part consisting of the IPv6 header plus all extension headers up to and including the routing header. We don't include it for this element because the data is already stored in other elements. We provide the elements necessary for the Fragmentable Part. http://tools.ietf.org/html/rfc2460
Destination_Options	PacketObj: DestinationOptions Type	0..2	The Destination Options header is used to carry optional information that needs to be examined only by a packet's destination node(s).
Authentication_Header	PacketObj: Authentication HeaderType	0..1	Follows RFC2402. The IP Authentication Header is used to provide connectionless integrity and data origin authentication for IP datagrams and to provide protection against replays. http://www.ietf.org/rfc/rfc2402.txt
Excapsulating_Security_ Payload	PacketObj: Excapsulating Security PayloadType	0..1	Follows RFC2406. ESP is used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity), and limited traffic flow confidentiality.

3.2.20.44 IPv6DoNotRecogActionType (restriction Common:BaseObjectAttributeType)

IPv6DoNotRecogActionType specifies possible actions when option is not recognized, via a union of the IPv6DoNotRecogActionTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:IPv6DoNotRecogActionTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.45 IPv6DoNotRecogActionTypeEnum

Enumerates possible actions when an option is not recognized.

Restriction base: string

Enumeration Value	Description
skipoption(00)	Indicates that the option should be skipped and the header should continue to be processed. See RFC 2460.
discardpacket(01)	Indicates that the packet should be discarded. See RFC 2460.
discardpacketsendicmpcode2(10)	Indicates that the packet should be discarded and regardless of whether or not the packet's Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type. See RFC 2460.
discardpacketsendicmpcode2nomulti(11)	Indicates that the packet should be discarded and only if the packet's Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type. See RFC 2460.

3.2.20.46 IPv6PacketChangeType (restriction Common:BaseObjectAttributeType)

IPV6PacketChangeType specifies whether a packet has changed, via a union of the IPv6PacketChangeTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:IPv6PacketChangeTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.47 IPv6PacketChangeTypeEnum

Enumerated list that specifies whether or not the Option Data of an option can change en-route to the packet's final destination.

Restriction base: string

Enumeration Value	Description
nochange(0)	Indicates that the packet does not change en-route. See RFC 2460.
change(1)	Indicates that the packet may change en-route. See RFC 2460.

3.2.20.48 IPv6OptionType

Specifies the meaning of each bit of the 8-bit IPv6OptionType type.

Property	Type	Mult	Description
Do_Not_Recogn_Action	PacketObj:IPv6DoNotRecogActionType	0..1	Action to be taken if the processing IPv6 nodes does not recognize the Option Type. This information is internally encoded in the Option Type identifier

			(higest-order two bits) such that their highest-order two bits specify the action that must be taken if the processing IPv6 node does not recognize the Option type. These possible actions are enumerated via IPv6DoNotRecogActionType.
Packet_Change	PacketObj: IPv6PacketChangeType	0..1	The third highest order bit of the Option Data specifies whether or not the Option Data of that option can change en-route to the packet's final destination.
Option_Byte	Common: HexBinary ObjectAttributeType	0..1	This field may be used to specify the actual Option Type byte, with no explicit meaning attached. Meaning/intrepretation provided by the Do_Not_Recogn_Action and Packet_Change fields.

3.2.20.49 IPVersionType (restriction Common:BaseObjectAttributeType)

IPVersionType specifies IP versions, via a union of the IPVersionTypeEnum type and the atomic xs:string type. See http://www.iana.org/assignments/version-numbers/version-numbers.xml for a complete list. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:IPVersionTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.50 IPVersionTypeEnum

Enumerates the different Internet Protocol versions. IPv4(4) and IPv6(6) are the most common.

Restriction base: string

Enumeration Value	Description
IPv4(4)	Indicates IP Version 4.
ST(5)	Indicates the IP version designating ST Datagram Mode.
IPv6(6)	Indicates IP Version 6.
TP/IX(7)	Indicates the IP version designating TP/IX: The Next Internet.
PIP(8)	Indicates the IP version designating PIP: The P Internet Protocol.
TUBA(9)	Indicates the IP version designating TUBA (TCP and UDP with Bigger Addresses, i.e. RFC 1347).

3.2.20.51 TransportLayerType

only UDP and TCP defined to begin. Other protocols will be defined as necessary.

Property	Type	Mult	Description
TCP	PacketObj:TCPType	0..1	TCP provides reliable, ordered delivery of a stream of bytes from a prograom on one computer to another program on another computer. http://en.wikipedia.org/wiki/Transmission_Control_Protocol
UDP	PacketObj:UDPType	0..1	UDP uses a simple transmission model without implicit handshaking dialogues for providing reliability, ordering, or data integrity. Thus, UDP

			provides an unreliable service and datagrams may arrive out of order, appear duplicated, or go missing without notice. http://en.wikipedia.org/wiki/User_Datagram_Protocol

3.2.20.52 TCPType

TCP provides reliable, ordered delivery of a stream of bytes from a prograom on one computer to another program on another computer. http://en.wikipedia.org/wiki/Transmission_Control_Protocol

Property	Type	Mult	Description
TCP_Header	PacketObj: TCPHeaderType	0..1	The TCP header contains 10 mandatory fields and an optional extension field. http://en.wikipedia.org/wiki/Transmission_Control_Protocol
Options	Common: HexBinary ObjectAttributeType	0..1	Options have up to three fields: Option-Kind (1 byte), Option-Length (1 byte), Option-Data (variable). This field will be further defined when required.
Data	Common:DataSegmentType	0..1	The Data element specifies the data payload of the TCP packet.

3.2.20.53 UDPType

UDP uses a simple transmission model without implicit handshaking dialogues for providing reliability, ordering, or data integrity. Thus, UDP provides an unreliable service and datagrams may arrive out of order, appear duplicated, or go missing without notice.
http://en.wikipedia.org/wiki/User_Datagram_Protocol

Property	Type	Mult	Description
UDP_Header	PacketObj: UDPHeaderType	0..1	The UDP header consists of four fields, which are defined here.
Data	Common:DataSegmentType	0..1	The Data element specifies the data payload of the UDP packet.

3.2.20.54 TCPHeaderType

The TCP header contains 10 mandatory fields and an optional extension field.
http://en.wikipedia.org/wiki/Transmission_Control_Protocol

Property	Type	Mult	Description
Src_Port	PortObj: PortObjectType	0..1	Identifies the sending port.
Dest_Port	PortObj: PortObjectType	0..1	Identifies the receiveing port.
Seq_Num	Common: HexBinary ObjectAttributeType	0..1	The Sequence number (32-bits) has a dual role: If the SYN flag is set, then this is the initial sequence numbers. If the SYN flag is clear (see Control Bits element), then this is the accumulated sequence number of the first data byte of this packet for the current session. http://en.wikipedia.org/wiki/Transmission_Control_

			Protocol
ACK_Num	Common: HexBinary ObjectAttributeType	0..1	If the ACK flag (see Control Bits element) is set then the value of this field is the next sequence number that the receiver is expecting.
Data_Offset	Common: HexBinary ObjectAttributeType	0..1	Specifies the size of the TCP header in 32-bit words.
Reserved	Common: HexBinary ObjectAttributeType	0..1	these 3 bits are reserved for future use and should be set to zero.
TCP_Flags	PacketObj: TCPFlagsType	0..1	The TCP header contains 9 flags (aka Control Bits).
Window	Common: HexBinary ObjectAttributeType	0..1	The size of the receive window, which specifies the number of bytes (beyond the sequence number in the acknowledgment field) that the sender of this segment is currently willing to receive. http://en.wikipedia.org/wiki/Transmission_Control_Protocol
Checksum	Common: HexBinary ObjectAttributeType	0..1	The 16-bit checksum field is used for error-checking of the header and data. http://en.wikipedia.org/wiki/Transmission_Control_Protocol
Urg_Ptr	Common: HexBinary ObjectAttributeType	0..1	If the URG flag is set, then this 16-bit field is an offset from the sequence number indicating the last urgent data byte. http://en.wikipedia.org/wiki/Transmission_Control_Protocol

3.2.20.55 TCPFlagsType

Defines the 9 different flags in the TCP header.

Property	Type	Mult	Description
ack	boolean	1..1	indicates that the Acknowledgment field is significant. All packets after the initial SYN packet sent by the client should have this flag set. http://en.wikipedia.org/wiki/Transmission_Control_Protocol
cwr	boolean	1..1	Congestion Window Reduced (CWR) flag is set by the sending host to indicate that it received a TCP segment with the ECE flag set and had responded in congestion control mechanism. http://en.wikipedia.org/wiki/Transmission_Control_Protocol
ece	boolean	1..1	ECN-Echo indicates: if the SYN flag is set, that the TCP peer is ECN capable; if the SYN flag is clear, that a packet with Congestion Experienced flag in IP header set is received during normal transmission. http://en.wikipedia.org/wiki/Transmission_Control_Protocol
fin	boolean	1..1	If this flag is set, it means there is no more data from sender.
ns	boolean	1..1	ECN-nonce concealment protection.
psh	boolean	1..1	Push functions. asks to push the buffered dtata to

			the receiving application. http://en.wikipedia.org/wiki/Transmission_Control_Protocol
rst	boolean	1..1	Reset the connection.
syn	boolean	1..1	Synchronize sequence numbers. Only the first packet sent from each end should have this flag set. http://en.wikipedia.org/wiki/Transmission_Control_Protocol
urg	boolean	1..1	Indicates that the Urgent point field is significant.

3.2.20.56 UDPHeaderType

The UDP header type defines the four fields in the UDP header.

Property	Type	Mult	Description
SrcPort	PortObj:PortObjectType	0..1	Identifies the sender's port.
DestPort	PortObj:PortObjectType	0..1	Identifies the receiver's port.
Length	Common:IntegerObjectAttributeType	0..1	Specifies the length in bytes of the entire datagram (header and data).
Checksum	Common:HexBinaryObjectAttributeType	0..1	The checksum is used for error-checking of the header and data.

3.2.20.57 IANAHardwareType (restriction Common:BaseObjectAttributeType)

IANAHardwareType specifies the type of hardware, via a union of the IANAHardwareTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:IANAHardwareTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.58 IANAHardwareTypeEnum

This enumerated type specifies Address Resolution Protocol (ARP) parameters. http://www.iana.org/assignments/arp-parameters/arp-parameters.xml

Restriction base: string

Enumeration Value	Description
Ethernet(1)	Indicates Ethernet hardware.
IEEE802(6)	Indicates IEEE 802 compliant hardware for networks carrying variable-size packets.
ARCNET(7)	Indicates the ARCNET LAN protocol.
FrameRelay(15)	Indicates the Frame Relay WAN technology.
ATM(16)	Indicates the ATM (Asynchronous Transfer Mode) networking standard.
HDLC(17)	Indicates the HDLC (High-Level Data Link Control) protocol.
FibreChannel(18)	Indicates the FibreChannel technology.
ATM(19)	Indicates the ATM (Asynchronous Transfer Mode) networking standard.
SerialLine(20)	Indicates the Serial Line protocol, or SLIP.

3.2.20.59 IANAEtherType (restriction Common:BaseObjectAttributeType)

EtherObjectType specifies "type" field of Ethernets, via a union of the IANAEtherTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:IANAEtherTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.60 IANAEtherTypeEnum

http://cavebear.com/archive/cavebear/Ethernet/type.html http://www.iana.org/assignments/ethernet-numbers http://standards.ieee.org/develop/regauth/ethertype/eth.txt http://en.wikipedia.org/wiki/EtherType

Restriction base: string

Enumeration Value	Description
IPv4(0x0800)	Indicates the IPv4 Ethernet type is specified.
ARP(0x0806)	Indicates the ARP Ethernet type is specified.
RARP(0x8035)	Indicates the RARP Ethernet type is specified.
IPX(0x8137)	Indicates the IPX Ethernet type is specified.
SNMP(0x814C)	Indicates the SNMP Ethernet type is specified.
IPv6(0x86DD)	Indicates the IPv6 Ethernet type is specified.

3.2.20.61 IANAAssignedIPNumbersType (restriction Common:BaseObjectAttributeType)

IANAAssignedIPNumbersType specifies Internet Protocol numbers, via a union of the IANAAssignedIPNumbersTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:IANAAssignedIPNumbersTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.62 IANAAssignedIPNumbersTypeEnum

Assigned Internet Protocol Numbers http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml List of protocol numbers used in the Protocol fields of the IPv4 header and the Next Header of the IPv6 header.

Restriction base: string

Enumeration Value	Description
IPv6hopbyhop(0)	Indicates the IPv6 Hop-By-Hop option protocol (HOPOPT).
ICMP(1)	Indicates the Internet Control Message protocol (HOPOPT).
IGMP(2)	Indicates the Internet Control Message protocol (HOPOPT).
GGP(3)	Indicates the Gateway-to-Gateway protocol (HOPOPT).
IPv4Encapsulation(4)	Indicates the IPv4 Encapsulation protocol (IPv4).
ST(5)	Indicates the Stream protocol (HOPOPT).
TCP(6)	Indicates the TCP protocol.

EGP(8)	Indicates the EGP (Exterior Gateway) protocol.
IGRP(9)	Indicates the IGP/IGRP (Cisco) protocol.
NVP(11)	Indicates the Network-Voice protocol.
PUP(12)	Indicates the PUP protocol.
ARGUS(13)	Indicates the ARGUS protocol.
EMCON(14)	Indicates the EMCON protocol.
XNET(15)	Indicates the Cross Net Debugger protocol.
UDP(17)	Indicates the UDP protocol.
IPv6Encapsulation(41)	Indicates the IPv6 protocol.
SDRP(42)	Indicates the Source Demand Routing protocol.
IPv6routingheader(43)	Indicates the routing header for IPv6.
IPv6fragmentheader(44)	Indicates the fragment header for IPv6.
RSVP(46)	Indicates the Reservation Protocol.
GRE(47)	Indicates the General Routing Encapsulation protocol number.
encapsultaesecuritypayload_ESP(50)	Indicates the Encapsulated Security Payload protocol number.
authenticationheader_AH(51)	Indicates the Authentication Header protocol number.
ICMPv6(58)	Indicates the ICMP for v6 protocol number.
IPv6nonextheader(59)	Indicates the No Next Header for IPv6 protocol number.
IPv6destinationoptions(60)	Indicates the Destination Options for IPv6 protocol number.
mobilityheader(135)	Indicates the Mobility Header protocol number.

3.2.20.63 IANAPortNumberRegistryType (restriction Common:BaseObjectAttributeType)

IANAPortNumberRegistryType specifies port numbers, via a union of the IANAPortNumberRegistryTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:IANAPortNumberRegistryTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.64 IANAPortNumberRegistryTypeEnum

http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml

Restriction base: string

Enumeration Value	Description
ftpdata(20)	Indicates the port for ftpdata.
ftp(21)	Indicates the port for ftp.
ssh(22)	Indicates the port for ssh.
telnet(23)	Indicates the port for telnet.
smtp(25)	Indicates the port for smtp.
domain(53)	Indicates the domain port.
tftp(69)	Indicates the port for tftp.
http(80)	Indicates the port for http.
ldap(389)	Indicates the port for ldap.
https(443)	Indicates the port for https.

3.2.20.65 ICMPv4PacketType

ICMP is used to send error messages (e.g., a datagram cannot reach its destination), informational messages (e.g., timestamp information), or a traceroute message. REF: http://www.networksorcery.com/enp/protocol/icmp.htm

Property	Type	Mult	Description
ICMPv4_Header	PacketObj: ICMPv4HeaderType	0..1	Actual header bytes are captured here. The message content of each type/code pair is also defined as part of the larger, complex "ICMPv4PacketType" type as either an error message, an informational message, or a traceroute message. The meaning of the type and code bytes is made explicit in the elements corresponding to each message type.
Error_Msg	PacketObj: ICMPv4ErrorMessageType	1..1	For ICMP error messages, boolean values are used in this element to explicitly interpret the type and code bytes appearing in the ICMP header. Additional fields and message content are also defined here.
Info_Msg	PacketObj: ICMPv4InfoMessageType	1..1	For ICMP informational messages, boolean values are used in this element to explicitly interpret the type and code bytes appearing in the ICMP header. Additional fields and message content are also defined here.
Traceroute	PacketObj: ICMPv4TracerouteType	1..1	For ICMP traceroute messages (type = 30), specifies related fields and ICMP code value. A boolean value is used to explicitly interpret the code byte appearing in the ICMP header. Additional fields and message content are also defined here.

3.2.20.66 ICMPv4HeaderType

Actual ICMP header bytes are defined, corresponding to the ICMP type, ICMP code, and to the checksum.

Property	Type	Mult	Description
Type	Common: HexBinary ObjectAttributeType	0..1	ICMP Type byte specifies the format of the ICMP message.
Code	Common: HexBinary ObjectAttributeType	0..1	ICMP Code byte further qualifies the ICMP message.
Checksum	Common: HexBinary ObjectAttributeType	0..1	ICMP Checksum (16 bits) covers the ICMP message.

3.2.20.67 ICMPv4ErrorMessageType

ICMP error messages include destination unreachable messages, source quench messages, redirect messages, and time exceeded messages.

Property	Type	Mult	Description
Destination_Unreachable	PacketObj: ICMPv4Destination	1..1	A destination unreachable message is an ICMP message which is generated by the host or its

	UnreachableType		inbound gateway to inform the client that the destiation is unreachable for some reason (http://en.wikipedia.org/wiki/ICMP_Destination_Unreachable).
Source_Quench	PacketObj: ICMPv4SourceQuenchType	1..1	A source quench message is an ICMP message that requests that the sender decrease the rate of messages sent to a router or host. This message may be generated if a router or host does not have sufficient buffer space to process the request or may occur if the router or host buffer is approaching its limit (http://en.wikipedia.org/wiki/ICMP_Source_Quench).
Redirect_Message	PacketObj: ICMPv4RedirectMessageType	1..1	A redirect message is used to send data packets on an alternative route. This ICMP redirect message informs a host to update its routing information.
Time_Exceeded	PacketObj: ICMPv4TimeExceededType	1..1	An ICMP time exceeded message is generated by a gateway to inform the source of a datagram that the datagram has been discarded due to the time to live field reaching zero. A time exceeded message may also be sent by a host if it fails to reassemble a fragmented datagram within its time limit (http://en.wikipedia.org/wiki/ICMP_Time_Exceeded).
Error_Msg_Content	PacketObj: ICMPv4ErrorMessageContentType	0..1	Message content common to all ICMP error messages are defined here. Fields that are specific to individual messages are defined separately under each message type.

3.2.20.68 ICMPv4ErrorMessageContentType

Elements associated with ICMPv4 error messages (as opposed to ICMP informational messages or ICMP traceroute message).

Property	Type	Mult	Description
IP_Header	PacketObj:IPv4HeaderType	0..1	IP header from the original datagram.
First_Eight_Bytes	Common: HexBinaryObjectAttributeType	0..1	First 8 bytes of the original datagram's data.

3.2.20.69 ICMPv4InfoMessageType

ICMP informational messages include echo request/reply, timestamp request/reply, and address mask request/reply.

Property	Type	Mult	Description
Echo_Reply	PacketObj: ICMPv4EchoReplyType	1..1	Echo reply/request messages are also known as "ping". The Info_Message_Content element contains an identifier and sequence number which together form the "quench" for echo reply and echo request. Fields specific to an echo reply message are given as elements to this echo reply element (type=0).

Echo_Request	PacketObj: ICMPv4EchoRequestType	1..1	Echo reply/request messages are also known as "ping". The Info_Message_Content element contains an identifier and sequence number which together form the "quench" for echo reply and echo request. Fields specific to an echo request message are given as elements to this echo request element (type=8).
Timestamp_Request	PacketObj: ICMPv4Timestamp RequestType	1..1	A timestamp request is an ICMP informational message used for time synchronization.
Timestamp_Reply	PacketObj: ICMPv4Timestamp ReplyType	1..1	A timestamp reply is an informational ICMP message which replies to a timestamp request message.
Address_Mask_Request	PacketObj: ICMPv4AddressMask RequestType	1..1	An address mask request is an ICMP informational message (query message) normally sent by a host to a router in order to obtain an appropriate subnet mask (type=17).
Address_Mask_Reply	PacketObj: ICMPv4AddressMask ReplyType	1..1	An address mask reply is an ICMP informational message, used to reply to an address mask request message with an appropriate subnet mask (type=18).
Info_Msg_Content	PacketObj: ICMPv4InfoMessage ContentType	0..1	Fields that are common to all ICMP informational messages are defined here. Fields that are specific to individual messages are defined separately under each message type.

3.2.20.70 ICMPv4InfoMessageContentType

Elements associated with ICMPv4 informational messages (as opposed to ICMP error messages or ICMP traceroute message).

Property	Type	Mult	Description
Identifier	Common: HexBinary ObjectAttributeType	0..1	16-bit identifier. Combined with the sequence number, called the "quench" for echo reply and echo request.
Sequence_Number	Common: HexBinary ObjectAttributeType	0..1	16-bit sequence number. The identifier and sequence number can be used by the client to match the reply with the request that caused the reply.

3.2.20.71 ICMPv4TracerouteType

Elements associated with ICMPv4 traceroute message (as opposed to ICMP error messages or ICMP informational messages); corresponds to ICMP type =30. (http://www.networksorcery.com/enp/protocol/icmp/msg30.htm)

Property	Type	Mult	Description
Outbound_Packet_Forward_ Success	boolean	1..1	One of two possible subtypes for an ICMP traceroute message. This subtype means that

			the outbound packet was successfully forwarded (code=0).
Outbound_Packet_no_Route	boolean	1..1	One of two possible subtypes for an ICMP traceroute message. This one means that there is no route for the outbound packet and the packet was discarded (code=1).
Identifier	Common: HexBinary ObjectAttributeType	0..1	16 bits. The ID number as copied from the ICMP traceroute option of the packet which caused this traceroute message to be sent (not related to the ID number in the IP header). (http://www.networksorcery.com/enp/protocol/icmp/msg30.htm)
Outbound_Hop_Count	Common: HexBinary ObjectAttributeType	0..1	16 bits. Outbound hop count as copied from the IP traceroute option of the packet which caused this traceroute message to be sent (http://www.networksorcery.com/enp/protocol/icmp/msg30.htm).
Return_Hop_Count	Common: HexBinary ObjectAttributeType	0..1	16 bits. Return hop count as copied from the IP traceroute options of the packet which caused this traceroute message to be sent. (http://www.networksorcery.com/enp/protocol/icmp/msg30.htm)
Output_Link_Speed	Common: HexBinary ObjectAttributeType	0..1	32 bits. The speed in bytes per second of the link over which the Outbound/Return Packet will be sent. If this value cannot be determined, the field should be set to zero. (http://www.networksorcery.com/enp/protocol/icmp/msg30.htm)
Output_Link_MTU	Common: HexBinary ObjectAttributeType	0..1	32 bits. The MTU in bytes of the link over which the Outbound/Return Packet will be sent. MTU refers to the data portion (includes IP header; excludes datalink header/trailer) of the packet. If this value cannot be determined, this field should be set to zero. (http://www.networksorcery.com/enp/protocol/icmp/msg30.htm)

3.2.20.72 ICMPv6PacketType

ICMP is used to send error messages (e.g., a datagram cannot reach its destination), informational messages (e.g., ping). Only the message types defined in RFC 4443 (ICMP v6) are included; additional message types will be defined as needed. REF: http://tools.ietf.org/html/rfc4443 and http://www.networksorcery.com/enp/protocol/icmpv6.htm and http://en.wikipedia.org/wiki/ICMPv6.

Property	Type	Mult	Description
ICMPv6_Header	PacketObj: ICMPv6HeaderType	0..1	Actual ICMP v6 header bytes are defined, corresponding to the ICMP type, ICMP code, and to the checksum.
Error_Msg	PacketObj: ICMPv6ErrorMessageType	1..1	For ICMP v6 error messages, boolean values are used in this element to explicitly interpret the type and code bytes appearing in the

			ICMP header. Additional fields and message content are also defined here. The type value indicates whether an ICMP message is an error message (type is 0 to 127) or an information message (type is 128 to 255).
Info_Msg	PacketObj: ICMPv6InfoMessageType	1..1	For ICMP v6 informational messages, boolean values are used in this element to explicitly interpret the type and code bytes appearing in the ICMP header. Additional fields and message content are also defined here. The type value indicates whether an ICMP message is an error message (type is 0 to 127) or an information message (type is 128 to 255).

3.2.20.73 ICMPv6HeaderType

Actual ICMP header bytes are defined, corresponding to the ICMP type, ICMP code, and to the checksum. Translation of each type and code byte are defined in text by using boolean values associated with corresponding elements in the informational and error message type elements.

Property	Type	Mult	Description
Type	Common: HexBinary ObjectAttributeType	0..1	The ICMP v6 type byte specifies the type of the message. Values range from 0 to 127 (high order bit is 0) indicate an error messages; values from 128 to 255 (high order bit is 1) indicate an informational message.
Code	Common: HexBinary ObjectAttributeType	0..1	The code byte value depends on the message type and provides an additional level of message granularity.
Checksum	Common: HexBinary ObjectAttributeType	0..1	Checksum characterizes the checksum information of an ICMPv6 header.

3.2.20.74 ICMPv6ErrorMessageType

ICMP v6 error messages include destination unreachable messages, packet too big messages, and time exceeded messages, and parameter problem messages, as defined in RFC 2463. Type values of ICMP v6 error messages range from 1 to 127.

Property	Type	Mult	Description
Destination_Unreachable	PacketObj: ICMPv6Destination UnreachableType	0..1	A destination unreachable message should be generated by a router, or by the IPv6 later in the originating node, in response to a packet that cannot be delivered to its destination address for reasons other than congestion. (http://tools.ietf.org/html/rfc4443)
Packet_Too_Big	PacketObj: ICMPv6Packet TooBigType	0..1	A packet too big message must be sent by a router in response to a packet that it cannot forward because the packet is larger than the MTU of the outgoing link.
Time_Exceeded	PacketObj: ICMPv6Time	0..1	A time exceeded message is send if either the hop limit is exceeded (hop limit = 0) or if fragment

	ExceededType		reassembly has timed out.
Parameter_Problem	PacketObj: ICMPv6Parameter ProblemType	0..1	If an IPv6 node processing a packet finds a problem with a field in the IPv6 header or extension headers and it cannot complete processing of the packet, it should send an ICMPv6 Parameter Problem message to the packet's source (http://tools.ietf.org/html/rfc4443).
Invoking_Packet	Common: HexBinary ObjectAttributeType	0..1	as much of invoking packet as possible without the ICMPv6 packet exceeding the minimum IPc6 MTU

3.2.20.75 ICMPv6InfoMessageType

ICMP v6 informational messages include echo request/reply; other informational message types will be added in the future as they are more commonly used (only echo request/reply are defined in RFC 4443).

Property	Type	Mult	Description
Echo_Request	PacketObj: ICMPv6EchoRequestType	0..1	Echo request and reply messages are used for diagnostic purposes.
Echo_Reply	PacketObj: ICMPv6EchoReplyType	0..1	Echo request and reply messages are used for diagnostic purposes
Info_Msg_Content	PacketObj: ICMPv6Info MessageContentType	0..1	ields that are common to all ICMP v6 informational messages are defined here. Fields that are specific to individual messages are defined separately under each message type.

3.2.20.76 ICMPv6InfoMessageContentType

Elements associated with ICMPv6 informational messages (as opposed to ICMP v6 error messages).

Property	Type	Mult	Description
Identifier	Common: HexBinary ObjectAttributeType	0..1	16-bit identifier. Combined with the sequence number, called the "quench" for echo reply and echo request.
Sequence_Number	Common: HexBinary ObjectAttributeType	0..1	16-bit sequence number. The identifier and sequence number can be used by the client to match the reply with the request that caused the reply.

3.2.20.77 ICMPv4EchoReplyType

Echo reply v4 informational message (used to ping); ICMP type=0.

Property	Type	Mult	Description
Echo_Reply	boolean	1..1	Echo reply is the only subtype (code=0).
Data	Common: HexBinary ObjectAttributeType	0..1	This data is optional and is used for the different kind of answers given with an ICMP Echo Reply message. Can be arbitrary length (but less than the MTU of the network).

3.2.20.78 ICMPv4DestinationUnreachableType

Destination Unreachable error message; ICMP type=3.

Property	Type	Mult	Description

Destination_Network_ Unreachable	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; destination network unreachable (code=0).
Destination_Host_ Unreachable	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; destination host unreachable (code=1).
Destination_Protocol_ Unreachable	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; destination protocol unreachable (code=2).
Destination_Port_ Unreachable	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; destination port unreachable (code=3).
Fragmentation_Required	PacketObj: FragmentationRequiredType	0..1	One of 16 different subtypes of a destination unreachable ICMP message; fragmentation required (code=4). This element has an additional field (Next-Hop MTU), as well as a boolean value indicating this subtype.
Source_Route_Failed	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; source route failed (code=5).
Destination_Network_ Unknown	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; destination network unknown (code=6).
Destination_Host_ Unknown	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; destination host unknown (code=7).
Source_Host_Isolated	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; source host isolated (code=8).
Network_Administrativel_ Prohibited	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; host administratively prohibited (code=9).
Host_Administratively_ Prohibited	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; host administratively prohibited (code=10).
Network_Unreachable_ For_TOS	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; network unreachable for TOS (code=11).
Host_Unreachable_For_ TOS	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; host unreachable for TOS (code=12).
Communication_ Administratively_ Prohibited	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; communication administratively prohibited (code=13).
Host_Precedence_ Violation	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; host precedence violation (code=14).
Precedence_Cutoff_In_ Effect	boolean	0..1	One of 16 different subtypes of a destination unreachable ICMP message; precedence cutoff in effect (code=15).

3.2.20.79 FragmentationRequiredType

This further specifies an ICMP destination unreachable (type=3) message of code=4 (fragmentation required) message by providing a Next-Hop MTU field.

Property	Type	Mult	Description
Fragmentation_Required	boolean	0..1	Indicates that the subtype of the destination unreachable ICMP message is "fragmentation required".
Next_Hop_MTU	Common: HexBinaryObjectAttributeType	0..1	The Next-Hop MTU field contains the MTU of the next-hop network is a code 4 error (fragmentation required) occurs.

3.2.20.80 ICMPv4SourceQuenchType

Source Quench (congestion control) error message; ICMP type=4.

Property	Type	Mult	Description
Source_Quench	boolean	0..1	Source quench is the only subtype (code=0).

3.2.20.81 ICMPv4RedirectMessageType

Redirect Message error message; ICMP type=5.

Property	Type	Mult	Description
Network_Redirect	boolean	1..1	One of 4 different subtypes of a redirect ICMP message; redirect datagram for the network (code=0).
Host_Redirect	boolean	1..1	One of 4 different subtypes of a redirect ICMP message; redirect datagram for the host (code=1).
ToS_Network_Redirect	boolean	1..1	One of 4 different subtypes of a redirect ICMP message; redirect datagram for the TOS and network (code=2).
ToS_Host_Redirect	boolean	1..1	One of 4 different subtypes of a redirect ICMP message; redirect datagram for the TOS and host (code=3).
IP_Address	AddressObj: AddressObjectType	0..1	The IP address is the 32-bit address of the gateway to which the redirection should be sent.

3.2.20.82 ICMPv4EchoRequestType

Echo Request informational message (used to ping); ICMP type=8.

Property	Type	Mult	Description
Echo_Request	Boolean	1..1	Echo request is the only subtype (code=0).
Data	Common: HexBinaryObjectAttributeType	0..1	This data is optional and is used for the different kind of answers given with an ICMP Echo Request message. Can be arbitrary length (but less than the MTU of the network).

3.2.20.83 ICMPv4TimeExceededType

Time Exceeded error message; ICMP type=11.

Property	Type	Mult	Description
TTL_Exceeded_In_Transit	boolean	1..1	specifies that the time-to-live was exceeded in transit (code=0).
Frag_Reassembly_Time_ Exceeded	boolean	1..1	specifies that the fragment reassembly time was exceeded (code=1).

3.2.20.84 ICMPv4TimestampRequestType

Time Stamp Request informational message; ICMP type=13.

Property	Type	Mult	Description
Timestamp	boolean	1..1	This is the only subtype of a timestamp request message (code=0).
Originate_Timestamp	Common: NonNegativeInteger ObjectAttributeType	0..1	32-bits; number of ms since midnight UT. The originate timestamp is the time the sender last touched the message before senting it. If the time is not available in milliseconds or cannot be provided with respect to midnight UT, then any time can be inserted in a timestamp provided the high order bit of the timestamp is also set to indicate this non-standard value.

3.2.20.85 ICMPv4TimestampReplyType

Time Stamp Reply informational message; ICMP type=14.

Property	Type	Mult	Description
Timestamp_Reply	boolean	1..1	This is the only subtype of a timestamp reply message (code=0).
Originate_Timestamp	Common: NonNegativeInteger ObjectAttributeType	0..1	The originate timestamp is the time the sender last touched the message before senting it. If the time is not available in milliseconds or cannot be provided with respect to midnight UT, then any time can be inserted in a timestamp provided the high order bit of the timestamp is also set to indicate this non-standard value.
Receive_Timestamp	Common: NonNegativeInteger ObjectAttributeType	0..1	The receive timestamp is the time the echoer first touched the message on receipt. If the time is not available in milliseconds or cannot be provided with respect to midnight UT, then any time can be inserted in a timestamp provided the high order bit of the timestamp is also set to indicate this non-standard value.
Transmit_Timestamp	Common: NonNegativeInteger ObjectAttributeType	0..1	The transmit timestamp is the time the echoer last touched the message on sending it. If the time is not available in milliseconds or cannot be provided with respect to midnight UT, then any time can be inserted in a timestamp provided the high order bit of the timestamp is also set to indicate this non-standard value.

3.2.20.86 ICMPv4AddressMaskRequestType

Address Mask Request informational message; ICMP type=17.

Property	Type	Mult	Description
Address_Mask_Request	Boolean	1..1	This is the only possible subtype of an address mask request message (code=0).
Address_Mask	AddressObj: AddressObjectType	0..1	The address mask can be set to 0 in an address mask request message (as opposed to an address mask reply message, in which case it should be set to the subnet mask).

3.2.20.87 ICMPv4AddressMaskReplyType

Address Mask informational message; ICMP type=18.

Property	Type	Mult	Description
Address_Mask_Reply	Boolean	1..1	This is the only possible subtype of an address mask reply message (code=0).
Address_Mask	AddressObj: AddressObjectType	0..1	This address mask field should be set to the subnet mask.

3.2.20.88 ICMPv6DestinationUnreachableType

Destination unreachable error message; ICMP v6 type=1.

Property	Type	Mult	Description
No_Route	boolean	0..1	No route to destination (ICMP v6 code=0).
Comm_Prohibited	boolean	0..1	Communication with destination administratively prohibited (ICMP v6 code=1).
Beyond_Scope	boolean	0..1	Beyond scope of source address (ICMP v6 code =2).
Address_Unreachable	boolean	0..1	Address is unreachable (ICMP v6 code=3).
Port_Unreachable	boolean	0..1	Port is unreachable (ICMP v6 code=4).
Src_Addr_Failed_Policy	boolean	0..1	Source address failed ingress/egress policy (ICMP v6 code=5).
Reject_Route	boolean	0..1	Reject route to destination (ICMP v6 code=6).

3.2.20.89 ICMPv6PacketTooBigType

Packet too big error message; ICMP v6 type=2.

Property	Type	Mult	Description
Packet_Too_Big	boolean	0..1	Only one code value is defined and is set to 0 (zero) by the originator and ignored by the receiver.
MTU	Common: HexBinary ObjectAttributeType	0..1	Maximum Transmission Unit describes the size limit for any given physical network.

3.2.20.90 ICMPv6TimeExceededType

Time exceeded error message; ICMP v6 type=3.

Property	Type	Mult	Description
Hop_Limit_Exceeded	boolean	0..1	Hop limit exceeded in transit (ICMP v6 code=0).
Fragment_Reassem_Time_ Exceeded	boolean	0..1	Fragment reassembly time exceeded (ICMP v6 code=1).

3.2.20.91 ICMPv6ParameterProblemType

Parameter problem error message; ICMP v6 type=4.

Property	Type	Mult	Description
Erroneous_Header_Field	boolean	0..1	Erroneous header field encountered (ICMP v6 code=0).
Unrecognized_Next_Header_Type	boolean	0..1	Unrecognized next header type encountered (ICMP v6 code=1).
Unrecognized_IPv6_Option	boolean	0..1	Unrecognized IP v6 option encountered (ICMP v6 code=2).
Pointer	Common: HexBinary ObjectAttributeType	0..1	identifies octet offset within invoking packet where error was detected.

3.2.20.92 ICMPv6EchoRequestType

Echo request informational ICMP v6 message; type=128.

Property	Type	Mult	Description
Echo_Request	Boolean	0..1	Every node must implement an ICMP v6 Echo responder function that receives Echo Requests (ICMP v6 code=0).
Data	Common: HexBinary ObjectAttributeType	0..1	Zero or more octets of arbitrary data.

3.2.20.93 ICMPv6EchoReplyType

Echo reply informational ICMP v6 message; type=129.

Property	Type	Mult	Description
Echo_Reply	Boolean	0..1	Every node must implement an ICMP v6 Echo responder function that originates corresponding Echo Replies(ICMP v6 code=0).
Data	Common: HexBinary ObjectAttributeType	0..1	This is the data from the invoking echo request message.

3.2.20.94 PrefixType

Provides an IP address or a prefix of an IP address for NDP for IPv6.

Property	Type	Mult	Description
IPv6_Addr	AddressObj: AddressObjectType	1..1	IPv6 address
IP_Addr_Prefix	AddressObj: AddressObjectType	1..1	The initial bits of an IPv6 address (these are identical for all hosts in a network) form the network's prefix. http://ipv6.com/articles/general/IPv6-Addressing.htm

3.2.20.95 HopByHopOptionsType

Defines fields for the IPv6 Hop-by-Hop Options header which is used to carry optional information that must be examined by every node along a packet's delivery path.

Property	Type	Mult	Description
Next_Header	PacketObj: IANAAssignedIPNumbersType	0..1	Identifies the type of header immediately following the Hop-by-Hop Options header. Uses the same values as the IPv4 Protocol field.
Header_Ext_Len	Common: HexBinary ObjectAttributeType	0..1	Length of the Hop-by-Hop Options header in 8-octet units, not including the first 8 octets.
Option_Data	PacketObj: OptionDataType	0..∞	Variable-length field, of length such that the complete Hop-by-Hop Options header is an integer multiple of 8 octets long. Contains one or more type-length-value (TLV)-encoded options.

3.2.20.96 OptionDataType

Defines the variable-length fields associated with IPv6 extension headers (the Hop-by-Hop Options header and the Destination Options header). Contains one or more type-length-value (TLV)-encoded options.

Property	Type	Mult	Description
Option_Type	PacketObj:IPv6OptionType	0..1	Identifies the type of option. This 8-bit Option Type identifier is internally encoded such that different bits have different meanings. These meanings are further specified in the IPv6OptionType type.
Option_Data_Len	Common: HexBinary ObjectAttributeType	0..1	Length of the Option Data field of this option, in octets.
Pad1	PacketObj:Pad1Type	0..1	The Pad1 option is used to insert one octet of padding into the Options area of a header. The Pad1 option does not have length and value fields.
PadN	PacketObj:PadNType	0..1	The PadN option is used to insert two or more octets of paddings into the Options area of a header.

3.2.20.97 RoutingType

Specifies the fields of the Routing header, which is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination.
http://tools.ietf.org/html/rfc2460

Property	Type	Mult	Description
Next_Header	PacketObj: IANAAssigned IPNumbersType	0..1	Identifies the type of header immediately following the Routing header. Uses the same values as the IPv4 Protocol field.
Header_Ext_Len	Common: IntegerObject AttributeType	0..1	length of the Routing header in 8-octet units, not including the first 8 octets.
Routing_Type	Common: HexBinary ObjectAttributeType	0..1	8-bit identifiers of a particular Routing header variant. Further definition will be added as required.
Segments_Left	Common: IntegerObject	0..1	Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still

	AttributeType		to be visited before reaching the final destination. http://tools.ietf.org/html/rfc2460
Type_Specific_Data	Common: StringObject AttributeType	0..1	Variable length field, of format determined by the Routing Type.

3.2.20.98 FragmentType

Specifies the fields of the Fragment header, which is used by an IPv6 source to send a packet larger than would fit in the path MTU. http://tools.ietf.org/html/rfc2460

Property	Type	Mult	Description
Fragment_Header	PacketObj: FragmentHeaderType	0..1	Each fragment has a header containing next header information, the offset of the fragment, an M flag specifying whether or not it is the last fragment, and an identification value.
Fragment	Common: HexBinary ObjectAttributeType	0..1	The fragment of the packet that corresponds to the fragment header. The length of the fragment must fit with the MTU of the path to the packets' destination.

3.2.20.99 DestinationOptionsType

Defines fields for the IPv6 Destination Options header which is used to carry optional information that needs to be examined only by a packet's destination node(s).

Property	Type	Mult	Description
Next_Header	PacketObj: IANAAssigned IPNumbersType	0..1	Identifies the type of header immediately following the Destination_Options options header. Uses the same values as the IPv4 Protocol field.
Header_Ext_Len	Common: HexBinary ObjectAttributeType	0..1	Length of the Destination Options header in 8-octet units, not including the first 8 octets.
Option_Data	PacketObj: OptionDataType	0..∞	Variable-length field, of length such that the complete Destinations Options header is an integer multiple of 8 octets long. Contains one or more type-length-value (TLV)-encoded options.

3.2.20.100 AuthenticationHeaderType

The IP Authentication Header is used to provide connectionless integrity and data origin authentication for IP datagrams and to provide protection against replays. http://www.ietf.org/rfc/rfc2402.txt

Property	Type	Mult	Description
Next_Header	PacketObj: IANAAssigned IPNumbersType	0..1	Identifies the type of header immediately following the Authentication header. Uses the same values as the IPv4 Protocol field.
Header_Ext_Len	Common: HexBinary ObjectAttributeType	0..1	An 8-bit field specifying the length of the AH in 32-bit words.
Security_Parameters_Index	Common: HexBinary ObjectAttributeType	0..1	The SPI is an arbitrary 32-bit value that, in combination with the destination IP address and security protocol (AH), uniquely identifies the Security Association for this datagram. The set of

			SPI values in the range 1 through 255 are reserved by the Internet Assigned Numbers Authority (IANA) for future use. http://www.ietf.org/rfc/rfc2402.txt
Sequence_Number	Common: HexBinary ObjectAttributeType	0..1	This unsigned 32-bit field contains a monotonically increasing counter value (sequence number).
Authenication_Data	Common: HexBinary ObjectAttributeType	0..1	This is a variable-length field that contains the Integrity Check Value (ICV) for this packet. The field must be an integer multiple of 32 bits in length.

3.2.20.101 ExcapsulatingSecurityPayloadType

ESP is used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity), and limited traffic flow confidentiality. http://www.ietf.org/rfc/rfc2406.txt

Property	Type	Mult	Description
Security_Parameters_Index	Common: HexBinary ObjectAttributeType	0..1	The SPI is an arbitrary 32-bit value that, in combination with the destination IP address and security protocol (ESP), uniquely identifies the Security Association for this datagram. http://www.ietf.org/rfc/rfc2406.txt
Sequence_Number	Common: HexBinary ObjectAttributeType	0..1	This unsigned 32-bit field contains a monotonically increasing counter value (sequence number).
Payload_Data	Common: HexBinary ObjectAttributeType	0..1	Payload Data is a variable-length field containing data described by the Next Header field.
Padding	Common: HexBinary ObjectAttributeType	0..1	The padding field can be used for various reasons, such as to fill in the plaintext as required by an encryption algorithm or to conceal the actual length of the payload.
Padding_Len	Common: HexBinary ObjectAttributeType	0..1	The pad length indicates the number of pad bytes immediately preceding it. Range is 0-255, where a value of zero indicates that no padding bytes are present. http://www.ietf.org/rfc/rfc2406.txt
Next_Header	PacketObj: IANAAssigned IPNumbersType	0..1	Identifies the type data contained in the payload data field. Uses the same values as the IPv4 Protocol field.
Authenication_Data	Common: HexBinary ObjectAttributeType	0..1	The Authentication Data is a variable-length field containing an Integrity Check Value (ICV) computed over the ESP packet minus the Authentication Data. http://www.ietf.org/rfc/rfc2406.txt

3.2.20.102 Pad1Type

The Pad1 type specifies how one octet of padding is inserted into the Options area of a header. The Pad1 option type does not have length and value fields.

Property	Type	Mult	Description
Octet	Common: HexBinary ObjectAttributeType	1..1	The fixed 00 value specifies that the Pad1 option is used and also serves as the single octet of padding.

3.2.20.103 PadNType

The PadN type specifies how two or more octets of padding are inserted into the Options area of a header.

Property	Type	Mult	Description
Octet	Common: HexBinary ObjectAttributeType	0..1	Specifies the PandN option.
Option_Data_Length	Common: IntegerObject AttributeType	0..1	Length of the padding. For N octets of padding, the Option_Data_Length fields contains the value N-2.
Option_Data	Common: IntegerObject AttributeType	0..1	Actual padding; consists of N-2 zero-valued octets.

3.2.20.104 FragmentHeaderType

Each fragment has a header containing next header information, the offset of the fragment, an M flag specifying whether or not it is the last fragment, and an identification value.

Property	Type	Mult	Description
Next_Header	Common: HexBinary ObjectAttributeType	0..1	Identifies the type of header immediately following the Fragment header. Uses the same values as the IPv4 Protocol field.
Fragment_Offset	Common: HexBinary ObjectAttributeType	0..1	13-bit unsigned integer. The offset, in 8-octet units, of the data following this header, relative to the start of the Fragmentable Part or the original packet.
M_Flag	PacketObj:MFlagType	0..1	Indicates whether this is the last fragment or whether there are more fragments.
Identification	Common: HexBinary ObjectAttributeType	0..1	For every packet that is to be fragmented, the source node generates a 32-bit Identification value.

3.2.20.105 MFlagType (restriction Common:BaseObjectAttributeType)

MFlagType specifies whether there are more fragments, via a union of the MFlagTypeEnum type and the atomic xs:string type. Its base type is the BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PacketObj:MFlagTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.20.106 MFlagTypeEnum

Used by the IPv6 Fragment Header to indicate whether or not there are more fragments.

Restriction base: string

Enumeration Value	Description
lastfragment(0)	Fragment is the last fragment.
morefragments(1)	There are more fragments (current is not the last).

3.2.21 NetworkRouteEntryObjectType (extends Common:DefinedObjectType)

The NetworkRouteEntryObjectType type is intended to characterize generic system network routing table entries.

Property	Type	Mult	Description
is_autoconfigure_address	boolean	1..1	The is_autoconfigure_address attribute specifies whether the destination IP address for the route is automatically configured.
is_immortal	boolean	1..1	The is_immortal attribute specifies whether the lifetimes for the route prefixes are infinite.
is_ipv6	boolean	1..1	The isipv6 attribute specifies whether the route uses IPv6 addresses.
is_loopback	boolean	1..1	The is_loopback attribute specifies whether the route is the default for all packets sent to local network addresses.
is_publish	boolean	1..1	The is_publish attribute specifies whether the route is published.
Destination_Address	AddressObj: AddressObjectType	0..1	The Destination_Address element specifies the destination IP address of the network route. It imports and uses the AddressObjectType from the CybOX Address object.
Origin	AddressObj: AddressObjectType	0..1	The Origin element specifies the origin address of the network route. It imports and uses the AddressObjectType from the CybOX Address object.
Netmask	AddressObj: AddressObjectType	0..1	The Netmask element specifies the netmask for te destination network.
Gateway_Address	AddressObj: AddressObjectType	0..1	The Gateway_Address element specifies the IP address of the gateway through which all packets using this route will be gatewayed. It imports and uses the AddressObjectType from the CybOX Address object.
Metric	Common: UnsignedLong ObjectAttributeType	0..1	The Metric element specifies the distance to the target, in terms of hops.
Type	NetworkRouteEntryObj: RouteType	0..1	The Type element specifies the type of network route being characterized.
Protocol	Common: StringObject AttributeType	0..1	The Protocol element specifies the name of the routing protocol that the route was added with.
Interface	Common: StringObject AttributeType	0..1	The Interface element specifies the name of the network interface to which all packets for the route will be sent.
Preferred_Lifetime	Common:	0..1	The Preferred_Lifetime element specifies the

	DurationObject AttributeType		preferred lifetime of the route, in seconds.
Valid_Lifetime	Common: DurationObject AttributeType	0..1	The Valid_Lifetime element specifies the lifetime for which the route is valid, in seconds.
Route_Age	Common: DurationObject AttributeType	0..1	The Route_Age element specifies the number of seconds since the route was added or modified in the routing table.

3.2.21.1 RouteType (restriction Common:BaseObjectAttributeType)

RouteType specifies route types, via a union of the RouteTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: NetworkRouteEntryObj:RouteTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.21.2 RouteTypeEnum

The RouteTypeEnum type is an enumeration of network route types.

Restriction base: string

Enumeration Value	Description
Other	
Invalid	Indicates a route that is invalid.
Direct	Indicates routing from one machine directly to another, where both machines reside on the same physical network.
Indirect	Indicates routing that is not direct and must be set to a gateway.

3.2.22 NetRouteObjectType (extends Common:DefinedObjectType)

The NetRouteObjectType type is intended to characterize a specific network route.

Property	Type	Mult	Description
is_autoconfigure_address	boolean	1..1	The is_autoconfigure_address specifies if the IP address is autoconfigured.
is_immortal	boolean	1..1	The is_immortal attribute specifies if the route is immortal.
is_ipv6	boolean	1..1	The is_ipv6 attribute specifies whether or not the route uses IPv6 addresses.
is_loopback	boolean	1..1	The is_loopback attribute specifies if the route is a loopback route (the gateway is on the local host).
is_publish	boolean	1..1	The is_publish attribute specifies if the route is published.
Description	Common: StructuredTextType	0..1	The Description element is intended for use in providing a brief description of the network route.
Network_Route_Entries	NetworkRouteObj: NetworkRouteEntriesType	0..1	The Network_Route_Entries element is optional and characterizes a set of network route segment

			entries.
Preferred_Lifetime	Common: DurationObject AttributeType	0..1	The Preferred_Lifetime element is intended to specify the preferred time, in seconds, that the IP route entry is valid. A value of 0xffffffff is considered to be infinite.
Valid_Lifetime	Common: DurationObject AttributeType	0..1	The Valid_Lifetime element is intended to specify the maximum time, in seconds, that the IP route entry is valid. A value of 0xffffffff is considered to be infinite.
Route_Age	Common: DurationObject AttributeType	0..1	The Route_Age element is intended to characterize the number of seconds since the route was added or modified in the network routing table.

3.2.22.1 NetworkRouteEntriesType

The NetworkRouteEntriesType type is intended to characterize the set of network route segments for this route.

Property	Type	Mult	Description
Network_Route_Entry	NetRouteEntryObj: NetworkRouteEntry ObjectType	0..∞	The Network_Route element is optional and characterizes a single network route segment entry.

3.2.23 NetworkSubnetObjectType (extends Common:DefinedObjectType)

The NetworkSubnetObjectType type is intended to characterize a generic system network subnet.

Property	Type	Mult	Description
Name	Common: StringObject AttributeType	0..1	The Name element is intended to specify a name for the network subnet.
Description	Common: StructuredTextType	0..1	The Description element is intended to provide a description of the network subnet.
NumberOfIPAddresses	Common: IntegerObject AttributeType	0..1	The NumberOfIPAddresses element is intended to specify the number of valid IP addresses within the scope of the network subnet.
Routes	NetworkSubnetObj: RoutesType	0..1	The Routes element is intended to characterize a set of network routes.

3.2.23.1 RoutesType

The RoutesType is intended to characterize a set network routes.

Property	Type	Mult	Description
Route	NetworkRouteEntryObj: NetworkRouteEntry ObjectType	1..∞	The Route element is intended to characterize a single network route.

3.2.24 PipeObjectType (extends Common:DefinedObjectType)

The PipeObjectType type is intended to characterize generic system pipes.

Property	Type	Mult	Description
named	Boolean	1..1	The named attribute specifies whether the pipe is

			named.
Name	Common: StringObject AttributeType	0..1	The Name element specifies the name of the pipe, if applicable.

3.2.25 PortObjectType (extends Common:DefinedObjectType)

The PortObjectType type is intended to characterize networking ports.

Property	Type	Mult	Description
Port_Value	Common: PositiveInteger ObjectAttributeType	1..1	The required Port_Value element specifies the actual value of the port.
Layer4_Protocol	PortObj: Layer4ProtocolType	0..1	The Layer4_Protocol element specifies the Layer 4 Protocol (OSI Model) associated with the port.

3.2.25.1 Layer4ProtocolType (restriction Common:BaseObjectAttributeType)

Layer4ProtocolType specifies Layer 4 (OSI model) protocols, via a union of the Layer4ProtocolEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: PortObj:Layer4ProtocolEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.25.2 Layer4ProtocolEnum

The Layer4ProtocolEnum type is an enumeration of relevant Layer4 networking protocols.

Restriction base: string

Enumeration Value	Description
TCP	Indicates the Layer 4 (OSI model) TCP protocol.
UDP	Indicates the Layer 4 (OSI model) UDP protocol.

3.2.26 ProcessObjectType (extends Common:DefinedObjectType)

The ProcessObjectType type is intended to characterize system processes.

Property	Type	Mult	Description
is_hidden	boolean	1..1	The hidden attribute specifies whether the process is hidden or not.
PID	Common: UnsignedInteger ObjectAttributeType	0..1	The PID element specifies the Process ID, or PID, of the process.
Name	Common: StringObject AttributeType	0..1	The Name element specifies the name of the process.
Path	Common: StringObject AttributeType	0..1	The Path element specifies the path of the process.
Current_Working_Directory	Common: StringObject AttributeType	0..1	The Current_Working_Directory element specifies the directory dynamically associated with the process.

Creation_Time	Common: DateTimeObject AttributeType	0..1	The Creation_Time element specifies the local date/time at which the process was created.
Parent_PID	Common: UnsignedInteger ObjectAttributeType	0..1	The Parent_PID element specifies the process ID (PID) of the parent process (i.e. the process that spawned this one), if applicable.
Child_PID_List	ProcessObj: ChildPIDListType	0..1	The Child_PID_List element specifies any children spawned by the process being characterized, by way of a list of PIDs.
Argument_List	ProcessObj: ArgumentListType	0..1	The Argument_List element is optional and specifies a list of arguments utlized in intiating the process.
Environment_Variable_List	Common: EnvironmentVariable ListType	0..1	The Environment_Variable_List element specifies any environment variables associated with the process. This element imports and uses the EnvironmentVariableListType from the CybOX Common Types.
Image_Info	ProcessObj: ImageInfoType	0..1	The Image_Info element specifies information about the image associated with the process, such as its file name and path.
Kernel_Time	Common: DurationObject AttributeType	0..1	The Kernel_Time element specifies the duration of time that the process has executed in kernel mode.
Port_List	ProcessObj:PortListType	0..1	The Port_List element is optional and specifies a list of ports owned by the process.
Network_Connection_List	ProcessObj: NetworkConnection ListType	0..1	The Network_Connection_List element specifies information about any network connections opened or initiated by the process.
Start_Time	Common: DateTimeObject AttributeType	0..1	The Start_Time element specifies the local date/time at which the process was started.
Status	ProcessObj: ProcessStatusType	0..1	The Status element specifies the current status of the process. Since this is an operating system specific attribute, this is defined here as an abstract type which is then used as a base type in any OS-specific extensions.
String_List	Common: ExtractedStringsType	0..1	The String_List element specifies any strings found in the memory image of the process.
Username	Common: StringObject AttributeType	0..1	The Username element specifies the name of the user that created the process.
User_Time	Common: DurationObject AttributeType	0..1	The User_Time element specifies the duration of time that the process has executed in user mode.

3.2.26.1 NetworkConnectionType

The NetworkConnectionType type captures the critical information about a TCP or UDP network connection.

Property	Type	Mult	Description
Creation_Time	Common: DateTimeObject AttributeType	0..1	The Creation_Time element specifies the date/time the network connection was created.

Destination_IP_Address	AddressObj: AddressObjectType	0..1	The Destination_IP_Address element specifies the destination IP Address of the network connection.
Destination_Port	PortObj: PortObjectType	0..1	The Destination_Port element specifies the destination port of the network connection. It imports and uses the Port_Object type from the CybOX Port Object.
Source_IP_Address	AddressObj: AddressObjectType	0..1	The Source_IP_Address element specifies the source IP Address of the network connection.
Source_Port	PortObj:PortObjectType	0..1	The Source_Port element specifies the source port of the network connection. It imports and uses the Port_Object type from the CybOX Port Object.
TCP_State	ProcessObj: ConnectionStateType	0..1	The TCP_State element specifies the current state of the TCP network connection, if applicable.

3.2.26.2 NetworkConnectionListType

The NetworkConnectionListType type is a list of network connections.

Property	Type	Mult	Description
Network_Connection	ProcessObj: NetworkConnectionType	1..∞	The Network_Connection element specifies information about a single network connection opened or initiated by the process.

3.2.26.3 ImageInfoType

The ImageInfoType type captures information about the process image.

Property	Type	Mult	Description
Command_Line	Common: StringObject AttributeType	0..1	The Command_Line element specifies the complete command used to execute the process image.
Current_Directory	Common: StringObject AttributeType	0..1	The Current_Directory element specifies the current directory of the process image.
Path	Common: StringObject AttributeType	0..1	The Path element specifies the full path to the image file, including the file name.

3.2.26.4 ProcessStatusType (abstract)

The ProcessStatusType is used for specifying the status of a running or terminated process. Since this property is platform-specific, it is created here as an abstract type and then used in the platform-specific process CybOX objects.

3.2.26.5 ChildPIDListType

The ChildPIDListType type captures the PID's of the children of the process in a list format.

Property	Type	Mult	Description
Child_PID	Common: UnsignedInteger ObjectAttributeType	1..∞	The Child_PID element specifies the process ID of a single child process.

3.2.26.6 ConnectionStateType (restriction Common:BaseObjectAttributeType)

ConnectionStateType specifies connection states, via a union of the ConnectionStateEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: ProcessObj:ConnectionStateEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.26.7 ConnectionStateEnum

The ConnectionStateEnum type is an enumeration of TCP connection states.

Restriction base: string

Enumeration Value	Description
UNKNOWN	Indicates an unknown TCP connection state.
CLOSED	Indicates the closed TCP connection state--i.e. no connection state at all.
LISTENING	Indicates the listening TCP connection state.
SYN_SENT	Indicates the SYN sent TCP connection state--i.e. wait for a matching connection request after having sent a connection request.
SYN_RECEIVED	Indicates the SYN received TCP connection state--i.e. waiting for a confirming connection request acknowledgment after having both received and sent a connection request.
ESTABLISHED	Indicates the established TCP connection state--i.e. an open connection in which data received can be delivered to the user.
FIN_WAIT_1	Indicates the FIN-WAIT-1 TCP connection state--i.e. waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.
FIN_WAIT_2	Indicates the FIN-WAIT-2 TCP connection state--i.e. waiting for a connection termination request from the remote TCP.
CLOSE_WAIT	Indicates the CLOSE-WAIT TCP connection state--i.e. waiting for a connection termination request from the local user.
CLOSING	Indicates the CLOSING TCP connection state--i.e. waiting for a connection termination request acknowledgment from the remote TCP.
LAST_ACK	Indicates the LAST-ACK connection state--i.e. waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).
TIME_WAIT	Indicates the TIME-WAIT connection state--i.e. waiting for for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.
DELETING_TCB	Indicates the DELETE-TCB connection state--i.e. the Transmission Control Block (TCB) is being deleted.

3.2.26.8 ArgumentListType

The ArgumentListType is intended to specify a list of arguments utlized in intiating the process.

Property	Type	Mult	Description
Argument	Common: StringObject	0..1	The Argument element is optional and specifies a single argument utlized in intiating the process.

	AttributeType		

3.2.26.9 PortListType

The PortListType is intended to specify a list of network ports.

Property	Type	Mult	Description
Port	PortObj:PortObjectType	1..∞	The Port element is optional and specifies a single network port.

3.2.27 ProductObjectType (extends Common:DefinedObjectType)

The ProductObjectType type is intended to characterize software or hardware products.

Property	Type	Mult	Description
Edition	Common: StringObject AttributeType	0..1	The Edition element specifies the edition of the product, if applicable.
Language	Common: StringObject AttributeType	0..1	The Language element specifies the language of the product, if applicable.
Product	Common: StringObject AttributeType	1..1	The Product element specifies the name of the product. This element is required.
Update	Common: StringObject AttributeType	0..1	The Update element specifies the update/revision of the product, if applicable.
Vendor	Common: StringObject AttributeType	1..1	The Vendor element specifies the name of the product vendor. This element is required.
Version	Common: StringObject AttributeType	0..1	The Version element speciifes the version of the product, if applicable.

3.2.28 SemaphoreObjectType (extends Common:DefinedObjectType)

The SemaphoreObjectType type is intended to characterize generic semaphore objects.

Property	Type	Mult	Description
named	boolean	1..1	The named attribute specifies whether the Semaphore is named.
Current_Count	Common: UnsignedInteger ObjectAttributeType	0..1	The Current_Count element specifies the current count value for the semaphore.
Maximum_Count	Common: PositiveInteger ObjectAttributeType	0..1	The Maximum_Count element specifies the maximum count value for the semaphore.
Name	Common: StringObject AttributeType	0..1	The Name element specifies the name of the semaphore, if applicable.

3.2.29 SocketObjectType (extends Common:DefinedObjectType)

The SocketObjectType type is intended to characterize network sockets.

Property	Type	Mult	Description
is_blocking	boolean	1..1	The isblocking attribute specifies whether or not the socket is in blocking mode.
is_listening	boolean	1..1	The islistening attribute specifies whether or not the

			socket is in listening mode.
Address_Family	SocketObj: AddressFamilyType	0..1	The Address_Family element specifies the address family (AF_*) that the socket is configured for.
Domain	SocketObj: DomainFamilyType	0..1	The Domain element specifies the communication domain (PF_*) of the socket.
Local_Address	SocketObj: SocketAddressType	0..1	The Local_Address element specifies the IP address and port for the socket on the local machine.
Options	SocketObj: SocketOptionsType	0..1	The Options element specifies any particular options used by the socket.
Protocol	SocketObj: ProtocolType	0..1	The Protocol element specifies the type of IP layer protocol used by the socket.
Remote_Address	SocketObj: SocketAddressType	0..1	The Remote_Address element specifies the IP address and port for the socket on the remote machine.
Type	SocketObj:SocketType	0..1	The Type element specifies the type of socket being characterized.

3.2.29.1 SocketOptionsType

The SocketOptionsType type specifies any particular options used by the socket. If an options is supported only by specific address families or socket types, that's indicated in parentheses.

Property	Type	Mult	Description
IP_MULTICAST_IF	Common: StringObject AttributeType	0..1	Set the interface over which outgoing multicast datagrams should be sent (AF_INET / SOCK_DGRAM or SOCK_RAW).
IP_MULTICAST_IF2	Common: StringObject AttributeType	0..1	Set the interface over which outgoing multicast datagrams should be sent (AF_INET6 / SOCK_DGRAM or SOCK_RAW) .
IP_MULTICAST_LOOP	boolean	0..1	Specify that the sending host receives a copy of an outgoing multicast datagram (AF_INET / SOCK_DGRAM or SOCK_RAW).
IP_TOS	Common: StringObject AttributeType	0..1	Set Type of Service (TOS) and Precedence in the IP header (AF_INET).
SO_BROADCAST	boolean	0..1	Enable the socket for issuing messages to a broadcast address (AF_INET / SOCK_DGRAM or SOCK_RAW). (
SO_CONDITIONAL_ACCEPT	boolean	0..1	Allows an application to decide whether or not to accept an incoming connection on a listening socket (Windows only).
SO_KEEPALIVE	boolean	0..1	Keep the connection up by sending periodic transmissions (AF_INET or AF_INET6 / SOCK_STREAM).
SO_DONTROUTE	boolean	0..1	Bypass normal routing mechanisms (AF_INET or AF_INET6)
SO_LINGER	Common: UnsignedInteger ObjectAttributeType	0..1	Specfies if the system attempts delivery of or discards any buffered data when a close() is issued.
SO_DONTLINGER	boolean	0..1	Complement of SO_LINGER.
SO_OOBINLINE	boolean	0..1	Indicates whether out-of-band data is received inline with normal data (AF_INET or AF_INET6).

SO_RCVBUF	Common: UnsignedInteger ObjectAttributeType	0..1	Set size of the receive buffer.
SO_GROUP_PRIORITY	Common: UnsignedInteger ObjectAttributeType	0..1	Sets the relative priority for the socket in its group (Windows only).
SO_REUSEADDR	boolean	0..1	Indicates if the local socket address can be reused (AF_INET or AF_INET6 / SOCK_DGRAM or SOCK_RAW)
SO_DEBUG	boolean	0..1	Indicates if low-level debugging is active.
SO_RCVTIMEO	Common: UnsignedInteger ObjectAttributeType	0..1	Set the receive timeout value.
SO_SNDBUF	Common: UnsignedInteger ObjectAttributeType	0..1	Set size of the send buffer.
SO_SNDTIMEO	Common: UnsignedInteger ObjectAttributeType	0..1	Set the send timeout value.
SO_UPDATE_ACCEPT_CONTEXT	Common: UnsignedInteger ObjectAttributeType	0..1	Updates the properties of the socket which are inherited from the listening socket (Windows only).
SO_TIMEOUT	Common: UnsignedInteger ObjectAttributeType	0..1	Set the socket timeout.
TCP_NODELAY	boolean	0..1	When set, TCP will send data immediately instead of using the Nagle delay algorithm (AF_INET or AF_INET6 / SOCK_STREAM). (

3.2.29.2 SocketAddressType

The SocketAddressType type specifies an IP address/port pair.

Property	Type	Mult	Description
IP_Address	AddressObj: AddressObjectType	1..1	The IP_Address element specifies the IP address of the socket connection (remote or local).
Port	PortObj:PortObjectType	1..1	The Port element specifies the port number of the socket connection (remote or local).

3.2.29.3 AddressFamilyType (restriction Common:BaseObjectAttributeType)

AddressFamilyType specifies address family types, via a union of the AddressFamilyTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: SocketObj:AddressFamilyTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.29.4 DomainFamilyType (restriction Common:BaseObjectAttributeType)

DomainFamilyType specifies domain family types, via a union of the DomainTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: SocketObj:DomainTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.29.5 SocketType (restriction Common:BaseObjectAttributeType)

SocketType specifies socket types, via a union of the SocketTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: SocketObj:SocketTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.29.6 ProtocolType (restriction Common:BaseObjectAttributeType)

ProtocolType specifies protocol types, via a union of the ProtocolTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: SocketObj:ProtocolTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.29.7 AddressFamilyTypeEnum

The AddressFamilyTypeEnum type is an enumeration of address family (AF_*) types.

Restriction base: string

Enumeration Value	Description
AF_UNSPEC	Specifies an unspecified address family.
AF_INET	Specifies sockets using for the Internet when using Berkeley sockets.
AF_IPX	Specifies the IPX (Novell Internet Protocol) address family.
AF_APPLETALK	Specifies the APPLETALK DDP address family.
AF_NETBIOS	Specifies the NETBIOS address family.
AF_INET6	Specifies the IP version 6 address family.
AF_IRDA	Specifies IRDA sockets.
AF_BTH	Specifies BTH sockets.

3.2.29.8 DomainTypeEnum

The DomainTypeEnum type is an enumeration of communication domain (PF_*) types.

Restriction base: string

Enumeration Value	Description
PF_LOCAL	Specifies the communication domain from local to host.
PF_UNIX	Specifies the communication domain from UNIX to host.
PF_FILE	Specifies the communication domain from file to host.
PF_INET	Specifies the IP protocol family.

PF_AX25	Specifies the Amateur Radio AX.25 family.
PF_IPX	Specifies the Novell Internet Protocol family.
PF_INET6	Specifies the IP version 6 protocol family.
PF_APPLETALK	Specifies the Appletalk DDP protocol family.
PF_NETROM	Specifies the Amateur radio NetROM protocol family.
PF_BRIDGE	Specifies the Multiprotocol bridge protocol family.
PF_ATMPVC	Specifies the ATM PVCs protocol family.
PF_X25	Specifies the protocol family reserved for the X.25 project.
PF_ROSE	Specifies the PF_KEY key management API family.
PF_DECnet	Specifies the protocol family reserved for the DECnet project.
PF_NETBEUI	Specifies the protocol family reserved for the 802.2LLC project.
PF_SECURITY	Specifies the Security callback pseudo AF protocol family.
PF_KEY	Specifies the PF_KEY key management API protocol family.
PF_NETLINK	Specifies the netlink routing API family.
PF_ROUTE	Specifies the PF_ROUTE routing API family.
PF_PACKET	Specifies the packet family.
PF_ASH	Specifies the Ash family.
PF_ECONET	Specifies the Acorn Econet family.
PF_ATMSVC	Specifies the ATM SVCs protocol family.
PF_SNA	Specifies the Linux SNA Project protocol family.
PF_IRDA	Specifies IRDA sockets.
PF_PPPOX	Specifies PPPoX sockets.
PF_WANPIPE	Specifies Wanpipe API sockets.
PF_BLUETOOTH	Specifies Bluetooth sockets.

3.2.29.9 SocketTypeEnum

The SocketTypeEnum type is an enumeration of socket (SOCK_*) types.

Restriction base: string

Enumeration Value	Description
SOCK_STREAM	Specifies a pipe-like socket which operates over a connection with a particular remote socket, and transmits data reliably as a stream of bytes.
SOCK_DGRAM	Specifies a socket in which individually-addressed packets are sent (datagram).
SOCK_RAW	Specifies raw sockets which allow new IP protocls to be implemented in user space. A raw socket receives or sends the raw datagram not including link level headers.
SOCK_RDM	Specifies a socket indicating a reliably-delivered message..
SOCK_SEQPACKET	Specifies a datagram congestion control Protocol socket.

3.2.29.10 ProtocolTypeEnum

The ProtocolTypeEnum type is an enumeration of protocol types.

Restriction base: string

Enumeration Value	Description
IPPROTO_ICMP	Indicates the ICMP protocol.
IPPROTO_IGMP	Indicates the IGMP protocol.
BTHPROTO_RFCOMM	Indicates the Bluetooth protocol.
IPPROTO_TCP	Indicates the TCP protocol.
IPPROTO_UDP	Indicates the UDP protocol.
IPPROTO_ICMPV6	Indicates the ICMP v6 protocol.
IPPROTO_RM	Indicates the Reliable Multicating protocol.

3.2.30 SystemObjectType (extends Common:DefinedObjectType)

The SystemObjectType type is intended to characterize computer systems (as a combination of both software and hardware).

Property	Type	Mult	Description
Available_Physical_Memory	Common: UnsignedLong ObjectAttributeType	0..1	The Available_Physical_Memory element specifies the amount of physical memory available on the system, in bytes.
BIOS_Info	SystemObj:BIOSInfoType	0..1	The BIOS_Info element specifies information about the BIOS on the system.
Date	Common: DateObjectAttributeType	0..1	The Date element specifies the current date on the system.
Hostname	Common: StringObject AttributeType	0..1	The Hostname element specifies the hostname of the system.
Local_Time	Common: TimeObjectAttributeType	0..1	The Local_Time element specifies the local time on the system.
Network_Interface_List	SystemObj: NetworkInterfaceListType	0..1	The Network_Interface_List element specifies the list of network interfaces present on the system.
OS	SystemObj:OSType	0..1	The OS element specifies information about the operating system installed on the system.
Processor	Common: StringObject AttributeType	0..1	The Processor element specifies the name of the CPU used by the system.
Processor_Architecture	SystemObj: ProcessorArchType	0..1	The Processor_Architecture element specifies the specific architecture (e.g. x86) used by the CPU of the system.
System_Time	Common: TimeObjectAttributeType	0..1	The System_Time element specifies the system, or hardware, time on the system.
Timezone_DST	Common: StringObject AttributeType	0..1	The Timezone_DST element specifies the time zone used by the system, taking daylight savings time (DST) into account.
Timezone_Standard	Common: StringObject AttributeType	0..1	The Timezone_Standard element specifies the time zone used by the system, wihtout taking daylight savings time (DST) into account.
Total_Physical_Memory	Common: UnsignedLong ObjectAttributeType	0..1	The Total_Physical_Memory element specifies the total amount of physical memory present on the system, in bytes.
Uptime	Common: DurationObject AttributeType	0..1	The Uptime element specifies the duration that represents the current amount of time that the system has been up.
Username	Common: StringObject AttributeType	0..1	The Username element specifies the name of the user currently logged into the system.

3.2.30.1 BIOSInfoType

The BIOSInfoType type specifies information about a system's BIOS.

Property	Type	Mult	Description
BIOS_Date	Common: DateObjectAttributeType	0..1	The BIOS_Date element specifies the date of the bios (e.g. the datestamp of the BIOS

			revision).
BIOS_Version	Common: StringObject AttributeType	0..1	The BIOS_Version element specifies the version of the BIOS.
BIOS_Manufacturer	Common: StringObject AttributeType	0..1	The BIOS_Manufacturer element specifies the manufacturer of the BIOS.
BIOS_Release_Date	Common: DateObjectAttributeType	0..1	The BIOS_Release_Date element specifies the date the BIOS was released.
BIOS_Serial_Number	Common: StringObject AttributeType	0..1	The BIOS_Serial_Number element specifies the serial number of the BIOS.

3.2.30.2 NetworkInterfaceListType

The NetworkInterfaceListType type specifies information about the network interfaces present on the system.

Property	Type	Mult	Description
Network_Interface	SystemObj: NetworkInterfaceType	1..∞	The Network_Interface element specifies information about a network interface, such as its MAC address.

3.2.30.3 IPGatewayListType

The IPGatewayListType type specifies the IP Addresses of the gateways used by the system.

Property	Type	Mult	Description
IP_Gateway_Address	AddressObj: AddressObjectType	1..∞	The IP_Gateway_Address element specifies the IP Address of a gateway used by the system.

3.2.30.4 NetworkInterfaceType

The NetworkInterfaceType type specifies information about a network interface, such as its MAC address.

Property	Type	Mult	Description
Adapter	Common: StringObject AttributeType	0..1	The Adapter element specifies the name of the network adapter used by the network interface.
Description	Common: StringObject AttributeType	0..1	The Description element specifies the description of the network interface.
DHCP_Lease_Expires	Common: DateTimeObject AttributeType	0..1	The DHCP_Lease_Expires element specifies the date/time that the DHCP lease obtained on the network interface expires.
DHCP_Lease_Obtained	Common: DateTimeObject AttributeType	0..1	The DHCP_Lease_Obtained element specifies the date/time that the DHCP lease was obtained on the network interface.
DHCP_Server_List	SystemObj: DHCPServerListType	0..1	The DHCP_Server_List element specifies the list of DHCP servers used by the network interface.
IP_Gateway_List	SystemObj: IPGatewayListType	0..1	The IP_Gateway_List element specifies the list of IP Gateways used by the network interface.
IP_List	SystemObj: IPInfoListType	0..1	The IP_List element specifies the list of IP addresses used by the network interface.
MAC	Common:	0..1	The MAC element specifies the MAC or hardware

	StringObject AttributeType		address of the physical network card. Either a colon (':') or a dash ('-') may be used a separator between the octets.

3.2.30.5 IPInfoListType

The IPInfoListType type specifies a list of IP address/subnet mask pairs associated with a network interface.

Property	Type	Mult	Description
IP_Info	SystemObj:IPInfoType	1..∞	The IP_Info element specifies an IP Address/Subnet mask entry in the list.

3.2.30.6 IPInfoType

The IP_Info type specifies information about the IP address and its associated subnet mask used by a network interface.

Property	Type	Mult	Description
IP_Address	AddressObj: AddressObjectType	1..1	The IP_Address element specifies an IP address.
Subnet_Mask	AddressObj: AddressObjectType	0..1	The Subnet_Mask element specifies a subnet mask.

3.2.30.7 DHCPServerListType

The DHCPServerListType type specifies a list of DHCP Servers, via their IP addresses.

Property	Type	Mult	Description
DHCP_Server_Address	AddressObj: AddressObjectType	1..∞	The DHCP_Server_Address element specifies the IP address of a DHCP server.

3.2.30.8 OSType (extends Common:CPESpecificationType)

The OSType type specifies information about an operating system. It imports and extends the CPESpecificationType from the CybOX Common Types.

Property	Type	Mult	Description
Bitness	SystemObj:BitnessType	0..1	The Bitness element specifies the bitness of the operating system (i.e. 32 or 64).
Build_Number	Common: StringObject AttributeType	0..1	The Build_Number element specifies the build number of the operating system.
Environment_Variable_List	Common: EnvironmentVariable ListType	0..1	The EnvironmentVariableList element specifies a list of environment variables present on the operating system.
Install_Date	Common: DateObject AttributeType	0..1	The Install_Date element specifies the date the operating system was installed.
Patch_Level	Common: StringObject AttributeType	0..1	The Patch_Level element specifies the patch level of the operating system.

3.2.30.9 ProcessorArchType (restriction Common:BaseObjectAttributeType)

ProcessorArchType specifies CPU architecture types, via a union of the ProcessorArchEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: SystemObj:ProcessorArchEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.30.10 BitnessType (restriction Common:BaseObjectAttributeType)

BitnessType specifies CPU architecture bitness, via a union of the BitnessEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: SystemObj:BitnessEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.30.11 ProcessorArchEnum

The ProcessorArchEnum type is a (non-exhaustive) enumeration of computer processor architectures.

Restriction base: string

Enumeration Value	Description
x86-32	Specifies the 32-bit x86 architecture.
x86-64	Specifies the 64-bit x86 architecture.
IA-64	Specifies the 64-bit IA (Itanium) architecture.
PowerPC	Specifies the PowerPC IA (Itanium) architecture.
ARM	Specifies the ARM architecture.
Alpha	Specifies the Alpha architecture.
SPARC	Specifies the SPARC architecture.
z/Architecture	Specifies the z/architecture, used on IBM mainframes.
eSi-RISC	Specifies the eSi-RISC architecture.
MIPS	Specifies the MIPS architecture.
Motorola 68k	Specifies the Motorola 68k architecture.
Other	Specifies a processor architecture other than those defined in this enumeration.

3.2.30.12 BitnessEnum

The BitnessEnum type is an enumeration of word sizes that define classes of computer architectures.

Restriction base: string

Enumeration Value	Description
32	Specifies a 32-bit architecture.
64	Specifies a 64-bit architecture.

3.2.31 URIObjectType (extends Common:DefinedObjectType)

The URIObjectType type is intended to characterize Uniform Resource Identifiers (URI's).

Property	Type	Mult	Description
type	URIObj: URITypeEnum	1..1	The Type attribute specifies the type of URI that is being defined.
Value	Common: AnyURIObject AttributeType	1..1	The Value element specifies the value of the URI.

3.2.31.1 URITypeEnum

The URITypeEnum is an enumeration of types of URIs.

Restriction base: string

Enumeration Value	Description
URL	Specifies a URL type of URI.
General URN	Specifies a General URN type of URI.
Domain Name	Specifies a Domain Name type of URI.

3.2.32 UnixFileObjectType (extends FileObj:FileObjectType)

The UnixFileObjectType type is intended to characterize Unix files.

Property	Type	Mult	Description
Group_Owner	Common: StringObject AttributeType	0..1	The Group_Owner element specifies the name of the group which owns the file.
INode	Common: UnsignedLong ObjectAttributeType	0..1	The INode element specifies the inode, or index node, value of the file.
Type	UnixFileObj: UnixFileType	0..1	Specifies file type using the UnixFileTypeEnum enumeration.

3.2.32.1 UnixFilePermissionsType (extends FileObj:FilePermissionsType)

The UnixFilePermissionsType type specifies the specific permissions used by the Unix family of operating systems.

Property	Type	Mult	Description
gexec	boolean	1..1	The gexec attribute specifies whether or not the group owner of the file can execute it.
gread	boolean	1..1	The gread attribute specifies whether or not the group owner of the file can read its contents.
gwrite	boolean	1..1	The gwrite attribute specifies whether or not the group owner of the file can write to it.
oexec	boolean	1..1	The oexec attribute specifies whether or not all other users can execute the file.
oread	boolean	1..1	The oread attribute specifies whether or not all other users can read the contents of the file.
owrite	boolean	1..1	The owrite attribute specifies whether or not all other users can write to the file.
sgid	boolean	1..1	The sgid attribute specifies whether or not the file may be executed with the privileges of the file's

			group owner.
suid	boolean	1..1	The suid attribute specifies whether or not the file may be exectued with the privileges of the file's owner.
uexec	boolean	1..1	The uexec attribute specifies whether or not the owner of the file can execute it.
uread	boolean	1..1	The uread attribute specifies whether or not the owner of the file can read its contents.
uwrite	boolean	1..1	The uwrite attribute specifies whether or not the owner of the file can write to it.

3.2.32.2 UnixFileType (restriction Common:BaseObjectAttributeType)

UnixFileType specifies Unix file types, via a union of the UnixFileTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: UnixFileObj:UnixFileTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.32.3 UnixFileTypeEnum

The UnixFileTypeEnum type is an enumeration of file types used by the Unix family of operating systems. These file types can be determined via the output of the ls and stat commands.

Restriction base: string

Enumeration Value	Description
regularfile	Specifies a regular file, denoted in UNIX by the first dash (-) in a file with permissions -rw-r--r--.
directory	Specifies a directory, denoted in UNIX by the d in a file with permissions drw-r--r--.
socket	Specifies a socket, denoted in UNIX by the s in a file with permissions srw-r--r--.
symboliclink	Specifies a symbolic link, denoted in UNIX by the l in a file with permissions lrw-r--r--.
blockspecialfile	Specifies a block device, such as /dev/sda, denoted in UNIX by the b in a file with permissions brw-rw----.
characterspecialfile	Specifies a character device, such as /dev/null, denoted in UNIX by the c in a file with permissions crw-------.

3.2.33 UnixNetworkRouteEntryObjectType (extends NetworkRouteEntryObj:NetworkRouteEntryObjectType)

The UnixNetworkRouteEntryObjectType type is intended to characterize entries in the network routing table of a Unix system.

Property	Type	Mult	Description
Flags	Common:StringObjectAttributeType	0..1	The Flags element specifies any flags used for the network route, such as G (use gateway).
MSS	Common:UnsignedIntegerObjectAttributeType	0..1	The MSS element specifies the maximum segment size for TCP connections over this network route, in bytes.
Ref	Common:	0..1	The Ref element specifies the number of

	UnsignedLong ObjectAttributeType		references to this network route.
Use	Common: UnsignedLong ObjectAttributeType	0..1	The Use element specifies the number of lookups that were performed for this network route.
Window	Common: UnsignedIntegerObject AttributeType	0..1	The Window element specifies the default window size for TCP connections over this network route, in bytes.

3.2.34 UnixPipeObjectType (extends PipeObj:PipeObjectType)

The UnixPipeObjectType type is intended to characterize Unix pipes.

Property	Type	Mult	Description
Permission_Mode	Common: StringObject AttributeType	0..1	The Permission_Mode element specifies the Unix permission mode for the pipe.

3.2.35 UnixProcessObjectType (extends ProcessObj:ProcessObjectType)

The UnixProcessObjectType type is intended to characterize Unix processes.

Property	Type	Mult	Description
Open_File_Descriptor_List	UnixProcessObj: FileDescriptorListType	0..1	The Open_File_Descriptor_List element specifies a listing of the current file descriptors used by the Unix process.
Priority	Common: NonNegativeInteger ObjectAttributeType	0..1	The Priority element specifies the priority of the Unix process.
RUID	Common: NonNegativeInteger ObjectAttributeType	0..1	The RUID element specifies the real user ID, which represents the Unix user who created the process.
Session_ID	Common: NonNegativeInteger ObjectAttributeType	0..1	The Session_ID element specifies the Unix Sesssion ID of the process.

3.2.35.1 UnixProcessStatusType (extends ProcessObj:ProcessStatusType)

The UnixProcessStatusType element specifies the current status of the running Unix process. It extends the abstract ProcessStatusType from the CybOX Process Object.

Property	Type	Mult	Description
Current_Status	ProcessObj: ProcessStatusType	0..1	Specifies the current state of the Unix process, using the ProcessStatusEnum enumeration.
Timestamp	Common: DateTimeObject AttributeType	0..1	Specifies when the process started up.

3.2.35.2 FileDescriptorListType

The FileDescriptorListType type specifies a list of Unix file descriptors.

Property	Type	Mult	Description
File_Descriptor	Common: UnsignedInteger ObjectAttributeType	1..∞	The File_Descriptor element specifies a particular Unix File Descriptor.

3.2.35.3 ProcessStatusType (restriction Common:BaseObjectAttributeType)

ProcessStatusType specifies Unix process states, via a union of the ProcessStatusEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications. See "man ps" for more information.

Data restrictions: UnixProcessObj:ProcessStatusEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.35.4 ProcessStatusEnum

The ProcessStatusEnum is an enumeration of Unix process states.

Restriction base: string

Enumeration Value	Description
Running	Specifies a running process or runnable [on run queue] (R).
UninterruptibleSleep	Specifies a process in uninterruptable sleep [usually IO] (D).
InterruptibleSleep	Specifies a process in interruptable sleep [waiting for an event to complete] (S).
Stopped	Specifies a stopped process, either by a job control signal or because it is being traced (T).
Paging	Specifies a paging process [not valid since the 2.6.xx kernel] (W).
Dead	Specifies a dead process [should never be seen] (X).
Zombie	Specifies a defunct, zombie process [terminated but not reaped by its parent] (Z).

3.2.36 UnixUserAccountObjectType (extends UserAccountObj:UserAccountObjectType)

The UnixUserAccountType type is intended to characterize Unix user accounts.

Property	Type	Mult	Description
Group_ID	Common:UnsignedIntegerObjectAttributeType	0..1	The Group_ID element specifies the ID of the primary group to which the Unix user account belongs.
User_ID	Common:UnsignedIntegerObjectAttributeType	0..1	The User_ID element specifies the ID of the Unix user account.
Login_Shell	Common:StringObjectAttributeType	0..1	The Login_Shell element specifies the name of the default login shell used by the Unix user account.

3.2.36.1 UnixGroupType (extends UserAccountObj:GroupType)

The UnixGroupType type is used for specifying Unix groups. It extends the abstract GroupType from the Cybox UserAccount element.

Property	Type	Mult	Description
Group_ID	Common:NonNegativeIntegerObjectAttributeType	1..1	The Group_ID element specifies the Unix ID of the group.

3.2.36.2 UnixPrivilegeType (extends UserAccountObj:PrivilegeType)

The UnixPrivilegeType type is used to specify Unix privileges. It extends the abstract PrivilegeType from the CybOX UserAccount object.

Property	Type	Mult	Description
Permissions_Mask	Common: StringObject AttributeType	1..1	The Permissions_Mask element specifies the Unix permissions mask for the privilege.

3.2.37 UnixVolumeObjectType (extends VolumeObj:VolumeObjectType)

The UnixVolumeObjectType type is intended to characterize Unix disk volumes.

Property	Type	Mult	Description
Mount_Point	Common: StringObject AttributeType	0..1	The Mount_Point element specifies the specific mounting point for the Unix volume.
Options	Common: StringObject AttributeType	0..1	The Options element specifies any options used when mounting the volume.

3.2.38 UserAccountObjectType (extends Account:AccountObjectType)

The UserAccountObjectType type is intended to characterize generic user accounts.

Property	Type	Mult	Description
password_required	boolean	1..1	The passwordrequired attribute specifies whether a password is required for this user account.
User_ID	Common: IntegerObject AttributeType	0..1	The User_ID element specifies the user id of the user for with this account was created
Full_Name	Common: StringObject AttributeType	0..1	The Full_Name element specifies the full name of the user for which this account was created.
Group_List	UserAccountObj: GroupListType	0..1	The Group_List element specifies the list of groups to which the user account belongs to.
Home_Directory	Common: StringObject AttributeType	0..1	The Home_Directory element specifies the fully-qualified path to the home directory of the user account.
Last_Login	Common: DateTimeObject AttributeType	0..1	The Last_Login element specifies the date/time that the user account was last logged into.
Privilege_List	UserAccountObj: PrivilegeListType	0..1	The Privilege_List element specifies the privileges that the user account has.
Script_Path	Common: StringObject AttributeType	0..1	The Script_Path element specifies the fully-qualified path to the directory where the logon script for the user account resides.
Username	Common: StringObject AttributeType	0..1	The Username element specifies the particular username of the user account.
User_Password_Age	Common: DurationObject AttributeType	0..1	The User_Password_Age element specifies the current age of the user account's password.

3.2.38.1 PrivilegeListType

The PrivilegeListType type specifies the list of privileges that the user account has.

Property	Type	Mult	Description
Privilege	UserAccountObj:PrivilegeType	1..∞	The Privilege element specifies a specific privilege that a user has. This is an abstract type since user privileges are operating-

			system specific, and is extended as needed in the derived CybOX object schemas.

3.2.38.2 PrivilegeType (abstract)

The PrivilegeType type specifies a specific privilege that a user has. This is an abstract type since user privileges are operating-system specific, and is extended as needed in the derived CybOX object schemas.

3.2.38.3 GroupListType

The GroupListType type specifies the groups that the user account belongs to.

Property	Type	Mult	Description
Group	UserAccountObj:GroupType	1..∞	The Group element specifies a group that a user account belongs to. This is an abstract type since group IDs are operating-system specific, and is extended as needed in the derived CybOX object schemas.

3.2.38.4 GroupType (abstract)

The GroupType type specifies a group that a user account belongs to. This is an abstract type since group IDs are operating-system specific, and is extended as needed in the derived CybOX object schemas.

3.2.39 UserSessionObjectType (extends Common:DefinedObjectType)

The UserSessionObjectType type is intended to characterize user sessions.

Property	Type	Mult	Description
Effective_Group	Common:StringObjectAttributeType	0..1	The Effective_Group element specifies the name of the effective group used in the user session.
Effective_Group_ID	Common:StringObjectAttributeType	0..1	The Effective_Group_ID element specifies the effective group ID of the group used in the user session.
Effective_User	Common:StringObjectAttributeType	0..1	The Effective_User element specifies the effective username used in the user session.
Effective_User_ID	Common:StringObjectAttributeType	0..1	The Effective_Group element specifies the effective user ID of the user used in the user session.
Login_Time	Common:DateTimeObjectAttributeType	0..1	The Login_Time element specifies the date/time of the login for the user session.
Logout_Time	Common:DateTimeObjectAttributeType	0..1	The Logout_Time element specifies the date/time of the logout for the user session.

3.2.40 VolumeObjectType (extends Common:DefinedObjectType)

The VolumeObjectType type is intended to characterize generic drive volumes.

Property	Type	Mult	Description
is_mounted	boolean	1..1	The ismounted attribute specifies whether the volume is mounted.

Name	Common: StringObject AttributeType	0..1	The Name element specifies the name of the volume.
Device_Path	Common: StringObject AttributeType	0..1	The Device_Path specifies the full path to the volume, including the device on which it resides.
File_System_Type	Common: StringObject AttributeType	0..1	The File_System_Type element specifies the name of the file system which is used on the volume.
Total_Allocation_Units	Common: UnsignedLong ObjectAttributeType	0..1	The Total_Allocation_Units element specifies the total number of allocation units available on the volume.
Sectors_Per_Allocation_Unit	Common: UnsignedInteger ObjectAttributeType	0..1	The Sectors_Per_Allocation_Unit element speciifes the number of disk sectors used for each allocation unit on the volume.
Bytes_Per_Sector	Common: PositiveInteger ObjectAttributeType	0..1	The Bytes_Per_Sector element specifies the number of bytes allocated for each sector of the volume.
Actual_Available_Allocation_Units	Common: UnsignedLong ObjectAttributeType	0..1	The Actual_Available_Allocation_Units element specifies the number of allocation units, or clusters, available on the volume.
Creation_Time	Common: DateTimeObject AttributeType	0..1	The Creation_Time element specifies the date/time that the volume was created.
File_System_Flag_List	VolumeObj: FileSystemFlagListType	0..1	The File_System_Flag_List element specifies the particular flags set for the volume by the file system which is used on the volume.
Serial_Number	Common: StringObject AttributeType	0..1	The Serial_Number element specifies the serial number of the volume.

3.2.40.1 VolumeOptionsType (abstract)

The VolumeOptionsType type specifies the particular options set for the volume. This is an abstract type since volume options are OS-specific, and is extended by the related OS-specific CybOX volume objects.

3.2.40.2 FileSystemFlagListType

The FileSystemFlagListType is a listing of the flags specified for the volume by the file system.

Property	Type	Mult	Description
File_System_Flag	VolumeObj: VolumeFileSystem FlagType	1..20	The File_System_Flag element specifies a particular flag used on the volume by the file system.

3.2.40.3 VolumeFileSystemFlagType (restriction Common:BaseObjectAttributeType)

VolumeFileSystemFlagType specifies file system flags, via a union of the VolumeFileSystemFlagEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: VolumeObj:VolumeFileSystemFlagEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.40.4 VolumeFileSystemFlagEnum

The FileSystemFlagEnum type is an enumeration of flags used by file systems on volumes, especially those on Windows Operating Systems. See http://msdn.microsoft.com/en-us/library/windows/desktop/aa364993(v=vs.85).aspx and http://msdn.microsoft.com/en-us/library/cc232101(v=prot.13).aspx for more information.

Enumeration Value	Description
FILE_CASE_SENSITIVE_SEARCH	Indicates that the specified volume supports case-sensitive file names. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00000001.
FILE_CASE_PRESERVED_NAMES	Indicates that the specified volume supports preserved case of file names when it places a name on disk. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00000002.
FILE_UNICODE_ON_DISK	Indicates that the specified volume supports preserved case of file names when it places a name on disk. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00000004.
FILE_PERSISTENT_ACLS	Indicates that the specified volume preserves and enforces access control lists (ACL). For example, the NTFS file system preserves and enforces ACLs, and the FAT file system does not. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00000008.
FILE_FILE_COMPRESSION	Indicates that the specified volume supports file-based compression. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00000010.
FILE_VOLUME_QUOTAS	Indicates that the specified volume supports disk quotas. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00000020.
FILE_SUPPORTS_SPARSE_FILES	Indicates that the specified volume supports sparse files. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00000040.
FILE_SUPPORTS_REPARSE_POINTS	Indicates that the specified volume supports re-parse points. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00000080.
FILE_SUPPORTS_REMOTE_STORAGE	Indicates that the specified volume supports remote storage. This is not listed with a lpFileSystemFlags value in documentation, but corresponds to the FileSystemAttributes value 0x00000100.
FILE_VOLUME_IS_COMPRESSED	Indicates that the specified volume is a compressed volume, for example, a DoubleSpace volume. This flag is incompatible with the FILE_FILE_COMPRESSION flag. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00008000.
FILE_SUPPORTS_OBJECT_IDS	Indicates that the specified volume supports object identifiers. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00010000.
FILE_SUPPORTS_ENCRYPTION	Indicates that the specified volume supports encryption. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00020000.
FILE_NAMED_STREAMS	Indicates that the specified volume supports named streams. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00040000.
FILE_READ_ONLY_VOLUME	Indicates that the specified volume is read-only. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00080000.
FILE_SEQUENTIAL_WRITE_ONCE	Indicates that the specified volume supports a single sequential write. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00100000.

FILE_SUPPORTS_TRANSACTIONS	Indicates that the specified volume supports transactions. For more information about transactions, see http://msdn.microsoft.com/en-us/library/windows/desktop/aa365993(v=vs.85).aspx. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00200000.
FILE_SUPPORTS_HARD_LINKS	Indicates that the specified volume supports hard links. For more information about hard links, see http://msdn.microsoft.com/en-us/library/windows/desktop/aa365006(v=vs.85).aspx. Note that hard links are DIFFERENT from symbolic links. This value is ONLY supported for Windows Server 2008 R2 and Windows 7 and later. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00400000.
FILE_SUPPORTS_EXTENDED_ ATTRIBUTES	Indicates that the specified volume supports extended attributes. An extended attribute is a piece of application-specific metadata that an application can associate with a file and is not part of the file's data. This value is ONLY supported for Windows Server 2008 R2 and Windows 7 and later. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x00800000.
FILE_SUPPORTS_OPEN_BY_FILE_ID	Indicates that the specified volume supports open by FileID. For more information about open by FileID, see http://msdn.microsoft.com/en-us/library/windows/desktop/aa364226(v=vs.85).aspx. This value is ONLY supported for Windows Server 2008 R2 and Windows 7 and later. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x01000000.
FILE_SUPPORTS_USN_JOURNAL	Indicates that the specified volume supports unique service number (USN) journals. For more information about USN journals, see http://msdn.microsoft.com/en-us/library/windows/desktop/aa363803(v=vs.85).aspx. This value is ONLY supported for Windows Server 2008 R2 and Windows 7 and later. This corresponds to the lpFileSystemFlags and FileSystemAttributes value 0x02000000.
FILE_SUPPORTS_INTEGRITY_ STREAMS	Indicates that the specified volume supports integrity streams. Currently, this value is ONLY available for ReFS and Windows 8 Beta. This corresponds to the FileSystemAttributes value 0x04000000.

3.2.41 WinComputerAccountObjectType (extends Account:AccountObjectType)

The WinComputerAccountObject type is intended to characterize Windows computer accounts.

Property	Type	Mult	Description
Fully_Qualified_Name	WinComputerAccountObj: FullyQualifiedNameType	0..1	The Fully_Qualified_Name element refers to the fully qualified name(s) of the Windows computer account.
Kerberos	WinComputerAccountObj: KerberosType	0..1	The Kerberos element specifies the Kerberos authentication protocol specific attributes for the Windows computer account.
Security_ID	Common: StringObject AttributeType	0..1	The Security_ID element specifies the Security ID (SID) value assigned to the Windows computer account.
Security_Type	Common:SIDType	0..1	The Security_Type element specifies the type of Security ID (SID) assigned to the Windows computer account.
Type	Common: StringObject AttributeType	0..1	The Type element specifies the type of the Windows computer account.

3.2.41.1 FullyQualifiedNameType

The FullyQualifiedNameType type refers to the fully qualified name(s) of the Windows computer account.

Property	Type	Mult	Description
NetBEUI_Name	Common: StringObject AttributeType	0..1	The NetBEUI_Name element specifies the NETBEUI name of the Windows computer account.
Full_Name	Common: StringObject AttributeType	0..1	The Full_Name element specifies the full name of the Windows computer account.

3.2.41.2 KerberosType

The KerberosType type specifies the Kerberos authentication protocol specific attributes for the Windows computer account.

Property	Type	Mult	Description
Delegation	WinComputerAccountObj: KerberosDelegationType	1..1	The Delegation element specifies the Kerberos delegation used for the Windows computer account.
Ticket	Common: UnsignedLong ObjectAttributeType	0..1	The Ticket element specifies the ID of the Kerberos ticket assigned to the Windows computer account.

3.2.41.3 KerberosDelegationType

The Delegation element specifies the Kerberos delegation used for the Windows computer account.

Property	Type	Mult	Description
Bitmask	Common: HexBinary ObjectAttributeType	1..1	The Bitmask element specifies the bitmask used in the Kerberos delegation for the Windows computer account.
Service	WinComputerAccountObj: KerberosServiceType	1..1	The Service element specifies the attributes of the Kerberos delegation service for the Windows computer account.

3.2.41.4 KerberosServiceType

The KerberosServiceType specifies the attributes of the Kerberos delegation service for the Windows computer account.

Property	Type	Mult	Description
Computer	Common: StringObject AttributeType	0..1	The Computer element specifies the computer name for the Kerberos service.
Name	Common: StringObject AttributeType	0..1	The Name element specifies the name of the Kerberos service.
Port	PortObj:PortObjectType	0..1	The Port element specifies the port for the Kerberos service.
User	Common: StringObject AttributeType	0..1	The User element specifies the username for the Kerberos service.

3.2.42 WinCriticalSectionObjectType (extends Common:DefinedObjectType)

The WinCriticalSectionObjectType type is intended to characterize Windows Critical Section objects.

Property	Type	Mult	Description
Address	Common: HexBinary ObjectAttributeType	0..1	The Address element specifies the address of the code that crated the critical section object.
Spin_Count	Common: NonNegativeInteger ObjectAttributeType	0..1	The Spin_Count element specifies the spin count value for the critical section object.

3.2.43 WindowsDriverObjectType (extends Common:DefinedObjectType)

The WindowsDriverObject type is intended to characterize Windows device drivers.

Property	Type	Mult	Description
Device_Object_List	WinDriverObj: DeviceObjectListType	0..1	The Device_Object_List element specifies the device objects that were created by the driver.
Driver_Init	Common: UnsignedLong ObjectAttributeType	0..1	The Driver_Init element specifies the entry point for the driver's DriverEntry routine. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff544174(v=vs.85).aspx
Driver_Name	Common: StringObject AttributeType	0..1	The Driver_Name element specifies the name of the driver.
Driver_Object_Address	Common: HexBinary ObjectAttributeType	0..1	The Driver_Object_Address element specifies the address to the driver's driver object, which contains the storage for the entry point to many of the driver's standard routines. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff548034(v=vs.85).aspx
Driver_Start_IO	Common: HexBinary ObjectAttributeType	0..1	The Driver_Start_IO element specifies the entry point for the driver's StartIO routine. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff544174(v=vs.85).aspx
Driver_Unload	Common: HexBinary ObjectAttributeType	0..1	The Driver_Unload element specifies the entry point for the driver's unload routine. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff544174(v=vs.85).aspx
Image_Base	Common: HexBinary ObjectAttributeType	0..1	The Image_Base element specifies the preferred address of the first byte of the driver's image when it is loaded into memory.
Image_Size	Common: HexBinary ObjectAttributeType	0..1	The Image_Size element specifies the size of the driver's image, in bytes.
IRP_MJ_CLEANUP	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_CLEANUP element represents a count of the number of times the CLEANUP function code was processed by the driver.
IRP_MJ_CLOSE	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_CLOSE element represents a count of the number of times the CLOSE

			function code was processed by the driver.
IRP_MJ_CREATE	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_CREATE element represents a count of the number of times the CREATE function code was processed by the driver.
IRP_MJ_CREATE_MAILSLOT	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_CREATE_MAILSLOT element represents a count of the number of times the CREATE_MAILSLOT function code was processed by the driver.
IRP_MJ_CREATE_NAMED_ PIPE	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_CREATE_NAMED_PIPE element represents a count of the number of times the CREATE_NAMED_PIPE function code was processed by the driver.
IRP_MJ_DEVICE_CHANGE	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_DEVICE_CHANGE element represents a count of the number of times the DEVICE_CHANGE function code was processed by the driver.
IRP_MJ_DEVICE_CONTROL	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_DEVICE_CONTROL element represents a count of the number of times the DEVICE_CONTROL function code was processed by the driver.
IRP_MJ_DIRECTORY_ CONTROL	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_DIRECTORY_CONTROL element represents a count of the number of times the DIRECTORY_CONTROL function code was processed by the driver.
IRP_MJ_FILE_SYSTEM_ CONTROL	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_FILE_SYSTEM_CONTROL element represents a count of the number of times the FILE_SYSTEM_CONTROL function code was processed by the driver.
IRP_MJ_FLUSH_BUFFERS	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_FLUSH_BUFFERS element represents a count of the number of times the FLUSH_BUFFERS function code was processed by the driver.
IRP_MJ_INTERNAL_DEVICE_ CONTROL	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_INTERNAL_DEVICE_CONTROL element represents a count of the number of times the INTERNAL_DEVICE_CONTROL function code was processed by the driver.
IRP_MJ_LOCK_CONTROL	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_LOCK_CONROL element represents a count of the number of times the LOCK_CONROL function code was processed by the driver.
IRP_MJ_PNP	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_PNP element represents a count of the number of times the PNP function code was processed by the driver.
IRP_MJ_POWER	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_POWER element represents a count of the number of times the POWER function code was processed by the driver.
IRP_MJ_READ	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_READ element represents a count of the number of times the READ function code was processed by the driver.
IRP_MJ_QUERY_EA	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_QUERY_EA element represents a count of the number of times the QUERY_EA function code was processed by the driver.

IRP_MJ_QUERY_ INFORMATION	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_QUERY_INFORMATION element represents a count of the number of times the QUERY_INFORMATION function code was processed by the driver.
IRP_MJ_QUERY_SECURITY	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_QUERY_SECURITY element represents a count of the number of times the QUERY_SECURITY function code was processed by the driver.
IRP_MJ_QUERY_QUOTA	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_QUERY_QUOTA element represents a count of the number of times the QUERY_QUOTA function code was processed by the driver.
IRP_MJ_QUERY_VOLUME_ INFORMATION	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_QUERY_VOLUME_INFORMATION element represents a count of the number of times the QUERY_VOLUME_INFORMATION function code was processed by the driver.
IRP_MJ_SET_EA	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_SET_EA element represents a count of the number of times the SET_EA function code was processed by the driver.
IRP_MJ_SET_INFORMATION	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_SET_INFORMATION element represents a count of the number of times the SET_INFORMATION function code was processed by the driver.
IRP_MJ_SET_SECURITY	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_SET_SECURITY element represents a count of the number of times the SET_SECURITY function code was processed by the driver.
IRP_MJ_SET_QUOTA	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_SET_QUOTA element represents a count of the number of times the SET_QUOTA function code was processed by the driver.
IRP_MJ_SET_VOLUME_ INFORMATION	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_SET_VOLUME_INFORMATION element represents a count of the number of times the SET_VOLUME_INFORMATION function code was processed by the driver.
IRP_MJ_SHUTDOWN	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_SHUTDOWN element represents a count of the number of times the SHUTDOWN function code was processed by the driver.
IRP_MJ_SYSTEM_CONTROL	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_SYSTEM_CONTROL element represents a count of the number of times the SYSTEM_CONTROL function code was processed by the driver.
IRP_MJ_WRITE	Common: UnsignedLong ObjectAttributeType	0..1	The IRP_MJ_WRITE element represents a count of the number of times the WRITE function code was processed by the driver.

3.2.43.1 DeviceObjectType

The DeviceObjectType type specifies the attributes of a device object. In this context, a device object represents a logical, virtual, or physical device for which a driver handles I/O requests. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff543147(v=vs.85).aspx

Property	Type	Mult	Description
Attached_Device_Name	Common: StringObject AttributeType	0..1	The Attached_Device_Name element specifies the name of another device object that was attached to this one. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff543147(v=vs.85).aspx
Attached_Device_Object	Common: UnsignedLong ObjectAttributeType	0..1	The Attached_Device_Object element specifies a pointer to another device object that was attached to this one. Typically this is a filter driver. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff543147(v=vs.85).aspx
Attached_To_Device_Name	Common: StringObject AttributeType	0..1	The Attached_To_Device_Name element specifies the name of another device object that this one was attached to.
Attached_To_Device_Object	Common: UnsignedLong ObjectAttributeType	0..1	The Attached_To_Device_Object element specifies a pointer to another device object that this one was attached to.
Attached_To_Driver_Object	Common: UnsignedLong ObjectAttributeType	0..1	The Attached_To_Driver_Object element specifies a pointer to the driver to which this device object was attached.
Attached_To_Driver_Name	Common: StringObject AttributeType	0..1	The Attached_To_Driver_Name element specifies the name of the driver to which this device object was attached.
Device_Name	Common: StringObject AttributeType	0..1	The Device_Name element specifies the name of the device object.
Device_Object	Common: UnsignedLong ObjectAttributeType	0..1	The Device_Object element specifies a pointer to the driver object for the caller.

3.2.43.2 DeviceObjectListType

The DeviceObjectListType specifies a list of device objects.

Property	Type	Mult	Description
Device_Object	WinDriverObj: DeviceObjectType	1..∞	The Device_Object element specifies a single device object.

3.2.44 WindowsEventLogObjectType (extends Common:DefinedObjectType)

The WindowsEventLogObjectType type is intended to characterize entries in the Windows event log.

Property	Type	Mult	Description
EID	Common: LongObject AttributeType	0..1	The EID element specifies the ID of the event for which the event log entry was created.
Type	Common: StringObject AttributeType	0..1	The event type associated with the entry in the event log, e.g., warning, information, error.
Log	Common: StringObject AttributeType	0..1	The name of the log.
Message	Common: StringObject	0..1	The rendered message string for the event.

	AttributeType		
Category_Num	Common: LongObject AttributeType	0..1	The event entry's category number, as defined by the source.
Category	Common: StringObject AttributeType	0..1	The text associated with Category_Num.
Generation_Time	Common: DateTime ObjectAttributeType	0..1	The Generation_Time element specifies the date/time the event was generated.
Source	Common: StringObject AttributeType	0..1	What logged the event, typically the name of an application or sub-component.
Machine	Common: StringObject AttributeType	0..1	The name of the computer on which the event log entry was generated.
User	Common: StringObject AttributeType	0..1	The name of the user (the security ID) responsible for the event.
Blob	Common: Base64Binary ObjectAttributeType	0..1	The event data as a binary blob.
Correlation_Activity_ID	Common: StringObject AttributeType	0..1	A globally unique identifier that identifies the current activity.
Correlation_Related_ Activity_ID	Common: StringObject AttributeType	0..1	A globally unique identifier that identifies the activity to which control was transferred to.
Execution_Process_ID	Common: StringObject AttributeType	0..1	The Execution_Process_ID element specifies the Process ID (PID) of the process which created the event.
Execution_Thread_ID	Common: StringObject AttributeType	0..1	The Execution_Thread_ID element specifies the Thread ID (TID) of the thread which created the event.
Index	Common: LongObjectAttribute Type	0..1	The index of the event entry in the log.
Reserved	Common: LongObjectAttribute Type	0..1	A DWORD value that is always set to ELF_LOG_SIGNATURE (the value 0x654c664c), which is ASCII for eLfL.
Unformatted_Message_List	WinEventLogObj: UnformattedMessage ListType	0..1	List of unformatted messages in the event log entry.
Write_Time	Common: DateTime ObjectAttributeType	0..1	The Write_Time element specifies the date/time that the entry was written into the event log.

3.2.44.1 UnformattedMessageListType

The UnformattedMessageListType type is a list of unformatted messages in the event log entry.

Property	Type	Mult	Description
Unformatted_Message	Common: StringObject AttributeType	1..∞	A single unformatted message in the event log entry.

3.2.45 WindowsEventObjectType (extends Common:DefinedObjectType)

The WindowsEventObjectType type is intended to characterize Windows event (synchronization) objects.

Property	Type	Mult	Description
Handle	WinHandleObj: WindowsHandleObjectType	0..1	The Handle element specifies the handle to the Windows event object. It imports and uses the WindowsHandleObjectType type from the CybOX Windows Handle object.
Name	Common: StringObject AttributeType	0..1	The Name element specifies the name of the Windows event object.
Type	WinEventObj:EventType	0..1	The Type element specifies the type of the Windows event.

3.2.45.1 EventType (restriction Common:BaseObjectAttributeType)

EventType specifies Windows event types, via a union of the EventTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinEventObj:EventTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.45.2 EventTypeEnum

The EventTypeEnum type is an enumeration of Windows synchronization event types. These are described in detail in http://msdn.microsoft.com/en-us/library/windows/desktop/ms682655(v=vs.85).aspx.

Restriction base: string

Enumeration Value	Description
ManualReset	Indicates an event object whose state remains signaled until it is explicitly reset to nonsignaled by the ResetEvent function. While it is signaled, any number of waiting threads, or threads that subsequently specify the same event object in one of the wait functions, can be released.
AutoReset	Indicates an event object whose state remains signaled until a single waiting thread is released, at which time the system automatically sets the state to nonsignaled. If no threads are waiting, the event object's state remains signaled. If more than one thread is waiting, a waiting thread is selected. Do not assume a first-in, first-out (FIFO) order. External events such as kernel-mode APCs can change the wait order.

3.2.46 WindowsExecutableFileObjectType (extends WinFileObj:WindowsFileObjectType)

The WindowsExecutableFileObjectType type is intended to characterize Windows PE (Portable Executable) files.

Property	Type	Mult	Description
Peak_Code_Entropy	WinExecutableFileObj: EntropyType	0..1	The Peak_Code_Entropy element specifies the maximum entropy of the code sections found in the

			file.
PE_Attributes	WinExecutableFileObj: PEAttributesType	0..1	The PE_Attributes element specifies the various PE-format specifics attributes for the file, such as the sections and headers.

3.2.46.1 PEAttributesType

The PEAttributesType specifies the attributes of a file in the Portable Executable (PE) format, such as execuatbles or dynamically loaded libraries.

Property	Type	Mult	Description
Base_Address	Common: HexBinaryObjectAttributeType	0..1	The Base_Address element specifies the base memory address of the PE binary.
Detected_EntryPoint_Signatures	WinExecutableFileObj: EntryPointSignatureListType	0..1	The Detected_Entrypoint_Signatures element specifies the entrypoint signatures that were detected for the PE binary.
Digital_Signature	Common: DigitalSignatureInfoType	0..1	The Digital_Signature element specifies the information about the digital signature used to sign the PE binary.
EP_Jump_Codes	WinExecutableFileObj: EPJumpCodeType	0..1	The EP_Jump_Codes element characterizes the entry point jump codes of the PE binary.
Exports	WinExecutableFileObj: PEExportsType	0..1	The Exports element characterizes the PE Export table of the PE Binary.
Extraneous_Bytes	Common: IntegerObjectAttributeType	0..1	The Extraneous_Bytes element specifies the number of extraneous bytes contained in the PE binary.
Headers	WinExecutableFileObj: PEHeadersType	0..1	The Headers element contains fields for characterizing attributes of the various types of PE headers.
Imports	WinExecutableFileObj: PEImportListType	0..1	The Imports element characterizes the PE Import Table of the binary.
PE_Checksum	WinExecutableFileObj: PEChecksumType	0..1	The PE_Checksum element specifies the checksum of the PE file.
PE_Timestamp	Common: DateTimeObjectAttributeType	0..1	The PE_Timestamp specifies the PE date/time stamp for the binary.
Resources	WinExecutableFileObj: PEResourceListType	0..1	The Resources element characterizes the PE Resources of the binary.
Sections	WinExecutableFileObj: PESectionListType	0..1	The Sections element characterizes the PE Sections of the binary.
Strings	WinExecutableFileObj: StringListType	0..1	The Strings element contains fields for characterizing any strings extracted from a PE file.
Subsystem	WinExecutableFileObj: SubsystemType	0..1	The Subsystem element specifies the type of subsystem that the PE binary was compiled for.
Type	WinExecutableFileObj: PEType	0..1	The Type specifies the particular type of the PE binary, e.g. Executable.

3.2.46.2 PEChecksumType

The PECheckSumType records the checksum of the PE file, both as found in the file and computed.

Property	Type	Mult	Description
PE_Computed_API	Common:	0..1	PE_Computed_API specifies a checksum computed

	LongObject AttributeType		by an external algorithm.
PE_File_API	Common: LongObject AttributeType	0..1	PE_File_API specifed the checksum computed by IMAGHELP.DLL.
PE_File_Raw	Common: LongObject AttributeType	0..1	PE_File_Raw specifies the checksum found in the PE file (in the Optional Header).

3.2.46.3 PEExportsType

PEExportsType specifies the PE File exports data section. The exports data section contains information about symbols exported by the PE File (a DLL) which can be dynamically loaded by other executables. This type abstracts, and its components, abstract the Windows structures.

Property	Type	Mult	Description
Exported_Functions	WinExecutableFileObj: PEExportedFunctionsType	0..1	A list of the exported functions in this section.
Exports_Time_Stamp	Common: DateTime ObjectAttributeType	0..1	The date and time the export data was created.
Number_Of_Addresses	Common: LongObject AttributeType	0..1	The number of addresses in the export data section's address table.
Number_Of_Names	Common: LongObject AttributeType	0..1	The number of names in the export data section's name table.

3.2.46.4 PEExportedFunctionsType

PEExportedFunctionsType specifies a list of PE exported functions

Property	Type	Mult	Description
Exported_Function	WinExecutableFileObj: PEExportedFunctionType	1..∞	Specifies a single element in the list of exported functions.

3.2.46.5 StringListType

StringListType specifies a list of strings contained in the PE File.

Property	Type	Mult	Description
String	WinExecutableFileObj: PEStringType	1..∞	Specifies a single element in the list of strings.

3.2.46.6 EPJumpCodeType

Property	Type	Mult	Description
Depth	Common: IntegerObject AttributeType	1..1	
Opcodes	Common: StringObject AttributeType	0..1	

3.2.46.7 EntryPointSignatureType

Specifies an entry point signature.

Property	Type	Mult	Description

Name	Common: StringObject AttributeType	0..1	Specifies the signature name.
Type	WinExecutableFileObj: DetectedType	1..1	Specifies the type of entry point detected (e.g., packer, compiled file).

3.2.46.8 EntryPointSignatureListType

Species a list of entry point signatures.

Property	Type	Mult	Description
Entry_Point_Signature	WinExecutableFileObj: EntryPoint SignatureType	1..∞	Specifies a single element in a list of entry point signatures.

3.2.46.9 PESectionListType

Specifies a list of sections that appear in the PE file.

Property	Type	Mult	Description
Section	WinExecutableFileObj: PESectionType	1..∞	Specifies an element of a list of PE file sections.

3.2.46.10 EntropyType

Specifies the result of an entropy computation.

Property	Type	Mult	Description
Value	Common: FloatObjectAttributeType	0..1	Specifies the computed entropy value.
Min	Common: FloatObjectAttributeType	0..1	Specifies the smallest possible value for the entropy computation.
Max	Common: FloatObjectAttributeType	0..1	Specifies the largest possible value for the entropy computation (eg., this would be 8 if the entropy computations is based on bits of information).

3.2.46.11 PEStringType

The PEStringType type is intended as container for strings extracted from PE binaries.

Property	Type	Mult	Description
Encoding	WinExecutableFileObj: CharacterEncodingEnum	0..1	This field refers to the encoding method used for the string extracted from the PE binary.
Address	Common: HexBinary ObjectAttributeType	0..1	The Address element specifies the location of the specified string in the PE binary.
Hashes	Common:HashListType	0..1	The Hashes element is used to include any hash values computed using the string extracted from the PE binary as input.
Language	Common: StringObject AttributeType	0..1	The Language element specifies the language the string is written in, e.g. English.
Length	Common: PositiveInteger ObjectAttributeType	0..1	The Length element specifies the length, in characters, of the string extracted from the PE binary.
String_Value	Common: StringObject	0..1	The String_Value element specifies the actual value

	AttributeType		of the string extracted from the PE binary.
Translation	Common: StringObject AttributeType	0..1	The Translation element specifies the English translation of the string, if it is not written in English.

3.2.46.12 PEImportType

The PEImportType type is intended as container for the attributes relevant to PE binary imports.

Property	Type	Mult	Description
delay_load	boolean	1..1	The delay_load attribute is a boolean value that is intended to describe whether a PE binary import is delay-load or not.
initially_visible	boolean	1..1	The initially_visible attribute refers to whether the import is initially visible, with regards to being initially visible or hidden in relation to PE binary packing. A packed binary will typically have few initially visible imports, and thus it is necessary to make the distinction between those that are visible initially or only after the binary is unpacked.
File_Name	Common: StringObject AttributeType	0..1	The File_Name element specifies the name of the library (file) that the PE binary imports.
Imported_Functions	WinExecutableFileObj: PEImportedFunctionsType	0..1	The Imported_Functions element is used to enumerate any functions imported from a particular library.
Virtual_Address	Common: HexBinary ObjectAttributeType	0..1	The Virtual_Address element specifies the relative virtual address (RVA) of the PE binary library import.

3.2.46.13 PEImportedFunctionsType

A list of PE imported functions

Property	Type	Mult	Description
Imported_Function	WinExecutableFileObj: PEImportedFunctionType	1..∞	Specifies a single element in a list of imported functions.

3.2.46.14 PEResourceType

The PEResourceType type is intended as container for the attributes relevant to PE binary resources.

Property	Type	Mult	Description
Type	WinExecutableFileObj: PEResourceTypeEnum	0..1	This field refers to the type of data referred to by this resource.
Name	Common: StringObject AttributeType	0..1	The Name element specifies the name of the resource used by the PE binary.
Hashes	Common:HashListType	0..1	The Hashes element is used to include any hash values computed using the specified PE binary resource as input.

3.2.46.15 PEExportedFunctionType

PEExportType sepcifies the type describing exported functions.

Property	Type	Mult	Description
Function_Name	Common: StringObject AttributeType	0..1	The Function_Name element specifies the name of the function exported by the PE binary.
Entry_Point	Common: HexBinary ObjectAttributeType	0..1	The Entry_Point element specifies the entry point of the function exported by the PE binary.
Ordinal	Common: NonNegativeInteger ObjectAttributeType	0..1	The Ordinal element specifies the ordinal value (index) of the function exported by the PE binary.

3.2.46.16 PEResourceListType

PEResourceListType specifies a list of resources found in the PE file.

Property	Type	Mult	Description
Resource	WinExecutableFileObj: PEResourceType	1..∞	Specifies an element of a list of resources.

3.2.46.17 PEImportedFunctionType

PEImportedFunctionType specifies the type describing imported functions.

Property	Type	Mult	Description
Function_Name	Common: StringObject AttributeType	0..1	The Function_Name element specifies the name of the function from the specified library that the PE binary imports.
Hint	Common: HexBinary ObjectAttributeType	0..1	The Hint element specifies the index into the export table of the library that the function is found in.
Ordinal	Common: NonNegativeInteger ObjectAttributeType	0..1	The Ordinal element specifies the ordinal value (index) of the function in the library that is found in.
Bound	Common: HexBinary ObjectAttributeType	0..1	The Bound element specifies the precomputed address if the imported function is bound.
Virtual_Address	Common: HexBinary ObjectAttributeType	0..1	The Virtual_Address element specifies the relative virtual address (RVA) of the PE binary library imported function.

3.2.46.18 PEImportListType

PEImportListType specifies a list of functions in an import data section.

Property	Type	Mult	Description
Import	WinExecutableFileObj: PEImportType	1..∞	Specifies a single element in a list of imported functions.

3.2.46.19 PESectionType

The PESectionType type is intended as container for the attributes relevant to PE binary sections. A PE Section consists of a header and data. The PESectionType contains attributes that describe the Section Header and metadata computed about the section (e.g., hashes, entropy).

Property	Type	Mult	Description
Section_Header	WinExecutableFileObj: PESectionHeader StructType	0..1	The Section_Header element contains the attributes of the section's section header structure.

Data_Hashes	Common:HashListType	0..1	The Data_Hashes element is used to include any hash values computed using the data contained in the specified PE binary section as input.
Entropy	WinExecutableFileObj:EntropyType	0..1	The Entropy element specifies the calculated entropy of the PE binary section.
Header_Hashes	Common:HashListType	0..1	The Header_Hashes element is used to include any hash values computed using the header of the specified PE binary section as input.
Type	WinExecutableFileObj:SectionType	0..1	Specifies the type of the section.

3.2.46.20 PEDataDirectoryStructType

The PEDataDirectoryStruct type is intended as container for the attributes present in a PE binary's data directory structure.

Property	Type	Mult	Description
Virtual_Address	Common:HexBinaryObjectAttributeType	0..1	The Virtual_Address element specifies the relative virtual address (RVA) of the data structure.
Size	Common:NonNegativeIntegerObjectAttributeType	0..1	The size element specifies the size of the data structure, in bytes.

3.2.46.21 PESectionHeaderStructType

The PESectionHeaderStruct type is intended as container for the attributes present in a PE binary's section header structure.

Property	Type	Mult	Description
Name	Common:StringObjectAttributeType	0..1	The Name element specifies the name of the PE binary section.
Physical_Address	Common:HexBinaryObjectAttributeType	0..1	The Physical_Address element specifies the physical address of the PE binary section.
Virtual_Address	Common:HexBinaryObjectAttributeType	0..1	The Virtual_Address element specifies the relative virtual address (RVA) of the PE binary section.
Size_Of_Raw_Data	Common:HexBinaryObjectAttributeType	0..1	The Size_Of_Raw_Data element specifies the size of the data contained in the PE binary section.
Pointer_To_Raw_Data	Common:HexBinaryObjectAttributeType	0..1	The Pointer_To_Raw_Data element specifies the file offset of the beginning of the PE binary section.
Pointer_To_Relocations	Common:HexBinaryObjectAttributeType	0..1	The Pointer_To_Relocations element specifies the offset of the PE binary section relocations, if applicable.
Pointer_To_Linenumbers	Common:HexBinaryObjectAttributeType	0..1	Specifies the beginning of line-number entries for the section. Should be 0.
Number_Of_Relocations	Common:NonNegativeIntegerObjectAttributeType	0..1	The Number_Of_Relocations element specifies the number of relocations defined for the specified PE binary section.
Number_Of_Linenumbers	Common:NonNegativeIntegerObjectAttributeType	0..1	Specifies the number of line number entreis for the section. Should be 0.

Characteristics	Common: HexBinary ObjectAttributeType	0..1	The Characteristics element specifies any flags defined for the specified PE binary section.

3.2.46.22 DOSHeaderType

The DOSHeaderType type is a container for the attributes present in the _IMAGE_DOS_HEADER structure, which can be found in Winnt.h and pe.h. See http://www.csn.ul.ie/~caolan/pub/winresdump/winresdump/doc/pefile.html for more information about the winnt.h file, and http://www.tavi.co.uk/phobos/exeformat.html for even more clarification.

Property	Type	Mult	Description
e_magic	Common: HexBinary ObjectAttributeType	0..1	Specifies the magic number, specifically the Windows OS signature value, which can either take on MZ for DOS (which is, for all intensive purposes, MOST Windows executables), NE for OS2, LE for OS2 LE, or PE00 for NT.
e_cblp	Common: HexBinary ObjectAttributeType	0..1	Specifies the number of bytes actually used in the last page, with the special case of a full page being represented by a value of zero (since the last page is never empty). For example, assuming a page size of 512 bytes, this value would be 0x0000 for a 1024 byte file, and 0x0001 for a 1025 byte file (since it only contains one valid byte).
e_cp	Common: HexBinary ObjectAttributeType	0..1	Specifies the the number of pages required to hold the file. For example, if the file contains 1024 bytes, and we assume the file has pages of a size of 512 bytes, this word would contain 0x0002; if the file contains 1025 bytes, this word would contain 0x0003.
e_crlc	Common: HexBinary ObjectAttributeType	0..1	Specifies the number of relocation items, i.e. the number of entries that exist in the relocation pointer table. If there are no relocation entries, this value is zero.
e_cparhdr	Common: HexBinary ObjectAttributeType	0..1	Specifies the size of the executable header in terms of paragraphs (16 byte chunks). It indicates the offset of the program's compiled/assembled and linked image (the load module) within the executable file. The size of the load module can be deduced by subtracting this value (converted to bytes) from the overall file size derived from combining the e_cp (number of file pages) and e_cblp (number of bytes in last page) values. The header always spans an even number of paragraphs.
e_minalloc	Common: HexBinary ObjectAttributeType	0..1	Specifies the minimum number of extra paragraphs needed to be allocated to begin execution. This is IN ADDITION to the memory required to hold the load module. This value normally represents the total size of any uninitialised data and/or stack segments that are linked at the end of a program. This space is not directly included in the load module, since there are no particular initializing values and it would simply waste disk space.

e_maxalloc	Common: HexBinary ObjectAttributeType	0..1	Specifies the maximum number of extra paragraphs needed to be allocated by the program before it begins execution. This indicates ADDITIONAL memory over and above that required by the load module and the value specified by MINALLOC. If the request cannot be satisfied, the program is allocated as much memory as is available.
e_ss	Common: HexBinary ObjectAttributeType	0..1	Specifies the initial SS value, which is the paragraph address of the stack segment relative to the start of the load module. At load time, this value is relocated by adding the address of the start segment of the program to it, and the resulting value is placed in the SS register before the program is started. In DOS, the start segment of the program is the first segment boundary in memory after the PSP.
e_sp	Common: HexBinary ObjectAttributeType	0..1	Specifies the initial SP value, which is the absolute value that must be loaded into the SP register before the program is given control. Since the actual stack segment is determined by the loader, and this is merely a value within that segment, it does not need to be relocated.
e_csum	Common: HexBinary ObjectAttributeType	0..1	Specifies the checksum of the contents of the executable file. It is used to ensure the integrity of the data within the file. For full details on how this checksum is calculated, see http://www.tavi.co.uk/phobos/exeformat.html#checksum.
e_ip	Common: HexBinary ObjectAttributeType	0..1	Specifies the initial IP value, which is the absolute value that should be loaded into the IP register in order to transfer control to the program. Since the actual code segment is determined by the loader, and this is merely a value within that segment, it does not need to be relocated.
e_cs	Common: HexBinary ObjectAttributeType	0..1	Specifies the pre-relocated initial CS value, relative to the start of the load module, that should be placed in the CS register in order to transfer control to the program. At load time, this value is relocated by adding the address of the start segment of the program to it, and the resulting value is placed in the CS register when control is transferred.
e_lfarlc	Common: HexBinary ObjectAttributeType	0..1	Specifies the file address of the relocation table, or more specifically, the offset from the start of the file to the relocation pointer table. This value must be used to locate the relocation pointer table (rather than assuming a fixed location) because variable-length information pertaining to program overlays can occur before this table, causing its position to vary. A value of 0x40 in this field generally indicates a different kind of executable file, not a DOS 'MZ' type.
e_ovro	Common:	0..1	Specifies the overlay number, which is normally set

	HexBinaryObjectAttributeType		to 0x0000, because few programs actually have overlays. It changes only in files containing programs that use overlays. See http://www.tavi.co.uk/phobos/exeformat.html#overlaynote for more information about overlays.
reserved1	Common:HexBinaryObjectAttributeType	0..4	Specifies reserved words for the program (known in winnt.h as e_res[4]), usually set to zero by the linker. In this case, just use a single reserved1 set to zero; if not zero create four reserved1 with the correct value.
e_oemid	Common:HexBinaryObjectAttributeType	0..1	Specifies the identifier for the OEM for e_oeminfo.
e_oeminfo	Common:HexBinaryObjectAttributeType	0..1	Specifies the OEM information for a specific value of e_oeminfo.
reserved2	Common:HexBinaryObjectAttributeType	0..1	Specifies reserved words for the program (known in winnt.h as e_res[10]), usually set to zero by the linker. In this case, just use a single reserved1 set to zero; if not zero create ten reserved1 with the correct value.
e_lfanew	Common:HexBinaryObjectAttributeType	0..1	Specifies the file adress of the of the new exe header. In particular, it is a 4-byte offset into the file where the PE file header is located. It is necessary to use this offset to locate the PE header in the file.
Hashes	Common:HashListType	0..1	The Hashes element is used to include any hash values computed using the specified PE binary MS-DOS header as input.

3.2.46.23 PEHeadersType

PEHeaderType specifies the headers found in PE and COFF files.

Property	Type	Mult	Description
DOS_Header	WinExecutableFileObj:DOSHeaderType	0..1	The DOS_Header element refers to the MS-DOS PE header and its associated attributes.
Signature	Common:HexBinaryObjectAttributeType	0..1	The Signature element specifies the 4-bytes sugnature that identifies the file as a PE file..
File_Header	WinExecutableFileObj:PEFileHeaderType	0..1	The File_Header element refers to the PE file header (somtimes referred to as the COFF header) and its associated attributes.
Optional_Header	WinExecutableFileObj:PEOptionalHeaderType	0..1	The Optional_Header element refers to the PE optional header and its associated attributes. The Optional Header is required for executable (PE) files, but optional for object (COFF) files.
Entropy	WinExecutableFileObj:EntropyType	0..1	The Entropy element specifies the calculated entropy of the PE file header.
Hashes	Common:HashListType	0..1	The Hashes element is used to include any hash values computed using the specified PE binary file header as input.

3.2.46.24 PEFileHeaderType

The PEFileHeaderType type refers to the PE file header (somtimes referred to as the COFF header) and its associated attributes.

Property	Type	Mult	Description
Machine	Common: HexBinary ObjectAttributeType	0..1	Specifies the type of target machine.
Number_Of_Sections	Common: NonNegative IntegerObject AttributeType	0..1	Specifies the number of sections in the file.
Time_Date_Stamp	Common: HexBinary ObjectAttributeType	0..1	Specifies the time when the file was created (the low 32 bits of the number of seconds since epoch).
Pointer_To_Symbol_Table	Common: HexBinary ObjectAttributeType	0..1	Specifies the file offset of the COFF symbol table (should be 0).
Number_Of_Symbols	Common: NonNegativeIntegerObject AttributeType	0..1	Specifies the number of entries in the symbol table. Should be 0.
Size_Of_Optional_Header	Common: HexBinary ObjectAttributeType	0..1	Specifies the size of the optional header. Should be 0 for object files and non-zero for executables.
Characteristics	Common: HexBinary ObjectAttributeType	0..1	Specifies the flags that indicate the file's attributes.
Hashes	Common:HashListType	0..1	Any hashes computed for the Optional Header.

3.2.46.25 SubsystemType (restriction Common:BaseObjectAttributeType)

SubsystemTypes specifies subsystem types via a union of the SubsystemTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinExecutableFileObj:SubsystemTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.46.26 DetectedType (restriction Common:BaseObjectAttributeType)

DetectedType specifies the type of entrypoint that was detected via a union of the DetectedTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinExecutableFileObj:DetectedTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.46.27 PEType (restriction Common:BaseObjectAttributeType)

PEType specifies PE file types via a union of the PETypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinExecutableFileObj:PETypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.46.28 SectionType (restriction Common:BaseObjectAttributeType)

SectionTypes specifies PE section types via a union of the SectionTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinExecutableFileObj:SectionTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.46.29 SectionTypeEnum

SectionTypeEnum enumerates the types of PE sections in an executable. See http://www.silurian.com/inspect/peformat.htm for more information. These sections can be viewed in a Disassembler, such as IDA and more specifically in the freeware CFF Explorer.

Restriction base: string

Enumeration Value	Description
Text	Denoted by .text, this specifies the main program code--usually execute and read access only.
Data	Denoted by .data, this specifies main initialized data code that is used by the program.
Resource	Denoted by .rsrc, this specifies Windows Resource data.
ReadonlyData	Denoted by .rdata, this specifies read only data.
Relocations	Denoted by .reloc, this specifies base relocations.
Debug	Denoted by .debug, this specifies debug information.
IData	Denoted by .idata, this specifies imported function data.
TLS	Denoted by .tls, this specifies Thread Local Storage. Data is private to each thread.
CRT	Denoted by .CRT, this specifies data reserved for the C Run-Time library.

3.2.46.30 SubsystemTypeEnum

SubsystemTypeEnum enumerates the types of subsystems in Windows an executable can be compatible for, according to winnt.h and more specifically, the Subsystem value of the IMAGE_OPTIONAL_HEADER structure. See http://source.winehq.org/source/include/winnt.h and http://msdn.microsoft.com/en-us/library/windows/desktop/ms680339(v=vs.85).aspx for more information.

Restriction base: string

Enumeration Value	Description
Unknown	Specifies an unknown subsystem.
Native	Specifies that no subsystem is required to run the image (i.e. only device drivers and

	native system processes are needed).
Windows_GUI	Specifies the Windows Graphical user interface (GUI) subsystem.
Windows_CUI	Specifies the Windows character-mode user interface (CUI) subsystem.
OS2_CUI	Specifies the OS/2 CUI subsystem.
POSIX_CUI	Specifies the POSIX CUI subsystem.
Native_Win9x_Driver	Specifies the Native Windows 9x drivers. This is denoted by the value IMAGE_SUBSYSTEM_NATIVE_WINDOWS or 0x8.
Windows_CE_GUI	Specifies the Windows CE system with a GUI.
EFI_Application	Specifies the Extensible Firmware Interface (EFI) application.
EFI_Boot_Service_Driver	Specifies the Extensible Firmware Interface (EFI) driver with boot services.
EFI_Runtime_Driver	Specifies the Extensible Firmware Interface (EFI) driver with run-time services.
EFI_ROM	Specifies the Extensible Firmware Interface (EFI) image.
XBOX	Specifies the XBOX system.
Windows_Boot_Application	Specifies the Windows Boot application.

3.2.46.31 DetectedTypeEnum

Restriction base: string

Enumeration Value	Description
None	Specifies a type other than those listed.
Compiler	Specifies an executable that acts as a compiler.
Packer	Specifies an executable that acts as a packer.
Installer	Specifies an executable that acts as an installer.

3.2.46.32 PETypeEnum

PETypeEnum enumerates the characteristics flags for the executable file in question. These are detailed in winnt.h.

Restriction base: string

Enumeration Value	Description
Executable	Specifies an executable image (not an OBJ or LIB).
Dll	Specifies a dynamic link library, not a program.
Invalid	Specifies an invalid executable file (i.e. not one of the listed types).

3.2.46.33 CharacterEncodingEnum

Restriction base: string

Enumeration Value	Description
ANSI	Indicates an ANSI-encoded string extracted from the PE binary.
Unicode	Indicates a Unicode-encoded string extracted from the PE binary.
Other	Indicates a differently encoded string extracted from the PE binary from those listed.

3.2.46.34 PEResourceTypeEnum

Restriction base: string

Enumeration Value	Description
Cursor	The resource specified is a cursor or animated cursor defined by naming it and specifying the name of the file that contains it. (To use a particular cursor, the application requests it by name.)

Bitmap	The resource specified is a bitmap defined by naming it and specifying the name of the file that contains it. (To use a particular cursor, the application requests it by name.)
Icon	The resource specified is an icon or animated icon by naming it and specifying the name of the file that contains it. (To use a particular icon, the application requests it by name.)
Menu	The resource specified defines the appearance and function of a menu. Does not define help or regular identifiers, nor uses the MFT_* type and MFS_* state flags.
MenuEX	The resource specified defines the appearance and function of a menu, which can also utilize help or regular identifiers, as well as the MFT_* type and MFS_* state flags.
Popup	The resource specified defines a menu item that can contain menu items and submenus.
Dialog	The resource specified defines a template that an application can use to create dialog boxes. This type is considered obsolete in Windows and newer applications use the DIALOGEX resource.
DialogEX	The resource specified defines a template that newer applications can use to create dialog boxes.
String	
StringTable	The resource specified defines string resources. String resources are Unicode or ASCII strings that can be loaded from the executable file.
Fontdir	
Font	The resource specified defines the name of a file that contains a font.
Accelerators	The resource specified defines menu accelerator keys.
RCData	The resource specified defines data resources. Data resources let you include binary data in the executable file.
MessageTable	The resource specified defines a message table by naming it and specifying the name of the file that contains it. The file is a binary resource file generated by the message compiler.
GroupCursor	
GroupIcon	
VersionInfo	The resource specified defines version-information. Vontains information such as the version number, intended operating system, and so on.
DLGInclude	
PlugPlay	This resource is obsolete and included for completeness.
TextInclude	This is a special resource that is interpreted by Visual C++. For more information see http://go.microsoft.com/FWLink/?LinkId=83951.
TypeLib	This is a special resource that is used with /TLBID and /TLBOUT linker options. For more information see http://go.microsoft.com/FWLink/?LinkId=83960 (for /TLBID) and http://go.microsoft.com/FWLink/?LinkId=83947 (for /TLBOUT).
Vxd	This resource is obsolete and included for completeness.
AniCursor	
AniIcon	
HTML	The resource specified defines an HTML file.
Manifest	
Other	The resource specified defines a different object than those listed. This resource type can also be considered User-Defined, i.e. defines a resource that contains application-specific data, as noted in MSDN.

3.2.46.35 PEOptionalHeaderType

The PEOptionalHeaderType type describes the PE Optional Header structure. Additional computed metadata, e.g., hashes of the header, are also included.

Property	Type	Mult	Description
Magic	Common: HexBinaryObjectAttributeType	0..1	Specifies the unsigned integer that indicates the type of executable file.
Major_Linker_Version	Common: HexBinaryObjectAttributeType	0..1	Specifies the linker major version number.
Minor_Linker_Version	Common: HexBinaryObjectAttributeType	0..1	Specifies the linker minor version number.
Size_Of_Code	Common: HexBinaryObjectAttributeType	0..1	Specifies the size of the code (text) section. If there are multiple sections, size is the sum of the sizes if each.
Size_Of_Initialized_Data	Common: HexBinaryObjectAttributeType	0..1	Specifies the size of the initialized data section. If there are multiple sections, size is the sum of the sizes if each.
Size_Of_Uninitialized_Data	Common: HexBinaryObjectAttributeType	0..1	Specifies the size of the uninitialized (bss) data section. If there are multiple sections, size is the sum of the sizes if each.
Address_Of_Entry_Point	Common: HexBinaryObjectAttributeType	0..1	Specifies the address of the entry point relative to the image base when the executable is loaded into memory. When there is no entry point (e.g., optional for DLLs), the value should be 0.
Base_Of_Code	Common: HexBinaryObjectAttributeType	0..1	Specifies the address that is relative to the image base of the beginning-of-code section when it is loaded into memory.
Base_Of_Data	Common: HexBinaryObjectAttributeType	0..1	Specifies the address that is relative to the image base of the beginning-of-data section when it is loaded into memory.
Image_Base	Common: HexBinaryObjectAttributeType	0..1	Specifies the preferred address of the first byte of image when loaded into memory; must be a multiple of 64 K.
Section_Alignment	Common: HexBinaryObjectAttributeType	0..1	Specifies the alignment (in bytes) of sections when they are loaded into memory.
File_Alignment	Common: HexBinaryObjectAttributeType	0..1	Specifies the factor (in bytes) that is used to align the raw data of sections in the image file.
Major_OS_Version	Common: HexBinaryObjectAttributeType	0..1	Specifies the major version number of the required operating system.
Minor_OS_Version	Common: HexBinaryObjectAttributeType	0..1	Specifies the minor version number of the required operating system.
Major_Image_Version	Common: HexBinaryObjectAttributeType	0..1	Specifies the major version number of the image.
Minor_Image_Version	Common: HexBinaryObjectAttributeType	0..1	Specifies the minor version number of the image.
Major_Subsystem_Version	Common: HexBinaryObjectAttributeType	0..1	Specifies the major version number of the subsystem.
Minor_Subsystem_Version	Common: HexBinaryObjectAttributeType	0..1	Specifies the minor version number of the subsystem.
Win32_Version_Value	Common:	0..1	Reserved; must be 0.

	HexBinaryObjectAttributeType		
Size_Of_Image	Common:HexBinaryObjectAttributeType	0..1	Specifies the size (in bytes) of the image, including all headers, as the image is loaded in memory.
Size_Of_Headers	Common:HexBinaryObjectAttributeType	0..1	Specifies the combined size of the MS DOS header, PE header, and section headers rounded up to a multiple of FileAlignment.
Checksum	Common:HexBinaryObjectAttributeType	0..1	Specifies the checksum of the PE file.
Subsystem	Common:HexBinaryObjectAttributeType	0..1	Specifies the subsystem (e.g., GUI, device driver) that is required to run this image.
DLL_Characteristics	Common:HexBinaryObjectAttributeType	0..1	Specifies flags that characterize the PE file.
Size_Of_Stack_Reserve	Common:HexBinaryObjectAttributeType	0..1	Specifies the size of the stack to reserve.
Size_Of_Stack_Commit	Common:HexBinaryObjectAttributeType	0..1	Specifies the size of the stack to commit.
Size_Of_Heap_Reserve	Common:HexBinaryObjectAttributeType	0..1	Specifies the size of the local heap space to reserve.
Size_Of_Heap_Commit	Common:HexBinaryObjectAttributeType	0..1	Specifies the size of the local heap space to commit.
Loader_Flags	Common:HexBinaryObjectAttributeType	0..1	Reserved; must be 0.
Number_Of_Rva_And_Sizes	Common:HexBinaryObjectAttributeType	0..1	Specifieshe number of data-directory entries in the remainder of the optional header.
Data_Directory	WinExecutableFileObj:DataDirectoryType	0..1	Specifies the data directories in the remainder in the optional header. This field will be repeated for each data directory.
Hashes	Common:HashListType	0..1	The Hashes element is used to include any hash values computed using the specified PE binary optional header as input.

3.2.46.36 DataDirectoryType

The DataDirectoryType specifies the data directories that can appear in the PE file's optional header. The data directories, except the Certificate Table, are loaded into memory so they can be used at runtime.

Property	Type	Mult	Description
Export_Table	WinExecutableFileObj:PEDataDirectoryStructType	0..1	Specifies the export table data directory.
Import_Table	WinExecutableFileObj:PEDataDirectoryStructType	0..1	Specifies the import table data directory.
Resource_Table	WinExecutableFileObj:PEDataDirectoryStructType	0..1	Specifies the resource table data directory.
Exception_Table	WinExecutableFileObj:PEDataDirectory	0..1	Specifies the exception table data directory.

	StructType		
Certificate_Table	WinExecutableFileObj: PEDataDirectory StructType	0..1	Specifies the certifcate table data directory. The table of attribute certificates is in a file which the data directory points to.
Base_Relocation_Table	WinExecutableFileObj: PEDataDirectory StructType	0..1	Specifies the base relocation table data directory.
Debug	WinExecutableFileObj: PEDataDirectory StructType	0..1	Specifies the debug data directory.
Architecture	WinExecutableFileObj: PEDataDirectory StructType	0..1	Reserved, must be 0.
Global_Ptr	WinExecutableFileObj: PEDataDirectory StructType	0..1	Specifies the RVA of the value to be stored in the global pointer register.
TLS_Table	WinExecutableFileObj: PEDataDirectory StructType	0..1	Specifies the thread local storage (TLS) table data directory.
Load_Config_Table	WinExecutableFileObj: PEDataDirectory StructType	0..1	Specifies the load configuration table data directory.
Bound_Import	WinExecutableFileObj: PEDataDirectory StructType	0..1	Specifies the bound import table data directory.
Import_Address_Table	WinExecutableFileObj: PEDataDirectory StructType	0..1	Specifies the import address table (IAT) data directory.
Delay_Import_Descriptor	WinExecutableFileObj: PEDataDirectory StructType	0..1	Specifies the delay import descriptor data directory.
CLR_Runtime_Header	WinExecutableFileObj: PEDataDirectory StructType	0..1	Specifies the Common Language Runtime (CLR) header data directory.
Reserved	WinExecutableFileObj: PEDataDirectory StructType	0..1	Reserved; must be 0.

3.2.47 WindowsFileObjectType (extends FileObj:FileObjectType)

The WindowsFileObjectType type is intended to characterize Windows files.

Property	Type	Mult	Description
Filename_Accessed_Time	Common: DateTime ObjectAttributeType	0..1	The Filename_Accessed_Time element specifies the date/time the filename of the Windows file was last accessed.
Filename_Created_Time	Common: DateTime ObjectAttributeType	0..1	The Filename_Created_Time element specifies the date/time the filename of the Windows file was created.
Filename_Modified_Time	Common: DateTime ObjectAttributeType	0..1	The Filename_Modified_Time element specifies the date/time the filename of the Windows file was last modified.
Drive	Common: StringObject AttributeType	0..1	The Drive element specifies the drive letter of the drive that the file resides on.
Security_ID	Common: StringObject AttributeType	0..1	The Security_ID element specifies the Security ID (SID) value assigned to the file.
Security_Type	Common:SIDType	0..1	The Security_Type element specifies the type of

			Security ID (SID) assigned to the file.
Stream_List	WinFileObj:StreamListType	0..1	The Stream_List element specifies any alternate data streams contained within the file.

3.2.47.1 StreamObjectType (extends Common:HashListType)

The StreamObjectType type is intended to characterize NTFS alternate data streams.

Property	Type	Mult	Description
Name	Common:StringObjectAttributeType	0..1	The Name element specifies the name of the alternate data stream.
Size_In_Bytes	Common:UnsignedLongObjectAttributeType	1..1	The Size_In_Bytes element specifies the size of the alternate data stream, in bytes.

3.2.47.2 StreamListType

The StreamListType type specifies a list of NTFS alternate data streams.

Property	Type	Mult	Description
Stream	WinFileObj:StreamObjectType	1..∞	The Stream element characterizes a single NTFS alternate data stream.

3.2.47.3 WindowsFileAttributesType (extends FileObj:FileAttributeType)

The WindowsFileAttributesType type specifies Windows file attributes. It imports and extends the FileAttributeType from the CybOX File Object.

Property	Type	Mult	Description
Attribute	WinFileObj:WindowsFileAttributeType	1..∞	The WindowsFileAttributeType specifies a single Windows file attribute.

3.2.47.4 WindowsFileAttributeType (restriction Common:BaseObjectAttributeType)

WindowsFileAttributeType specifies Windows file attributes via a union of the FileAttributesEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinFileObj:FileAttributesEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.47.5 WindowsFilePermissionsType (extends FileObj:FilePermissionsType)

The WindowsFilePermissionsType type specifies Windows file permissions. It imports and extends the FilePermissionsType from the CybOX File Object.

Property	Type	Mult	Description
Full_Control	boolean	0..1	The Full_Control element specifies whether reading, writing, changing and deleting of the file is perfmitted.
Modify	boolean	0..1	The Modify element specifies whether reading and writing or deletion of the file is permitted.

Read	boolean	0..1	The Read element specifies whether viewing or accessing of the file's contents is permitted.
Read_And_Execute	boolean	0..1	The Read_And_Execute element specifies whether viewing and accessing of the file's contents as well as executing of the file is permitted.
Write	boolean	0..1	The Write element specifies whether writing to the file is permitted.

3.2.47.6 FileAttributesEnum

The FileAttributesEnum type is an enumeration of Windows file attributes. These refer to the constants specified in http://msdn.microsoft.com/en-us/library/gg258117(v=vs.85).aspx.

Enumeration Value	Description
ReadOnly	Specifies a file is read only, as denoted by the constant value, 0x1. Applications can read the file, but cannot write to it or delete it. This attribute is not honored on directories. For more information as to why, see http://go.microsoft.com/FWLink/?LinkId=125896.
Hidden	Specifies a file or directory is hidden, as denoted by the constant value, 0x2. It is not included in an ordinary directory listing.
System	Specifies a file or directory that the operating system uses a part of, or uses exclusively, as denoted by the constant value, 0x4.
Directory	Specifies a directory, as denoted by the constant value, 0x10.
Archive	Specifies a file or directory that is an archive file or directory, as denoted by the constant value, 0x20. Applications typically use this attribute to mark files for backup or removal.
Device	Specifies a reserved system value, as denoted by the constant value, 0x40.
Normal	Specifies a file that has no other attributes set, and is only valid when this attribute is used alone, as denoted by the constant value, 0x80.
Temporary	Specifies a file being used for temporary storage, as denoted by the constant value, 0x100.
SparseFile	Specifies a sparse file, as denoted by the constant value, 0x200.
ReparsePoint	Specifies a file or directory that has an associated reparse point, or a file that is a symbolic link, as denoted by the constant value, 0x400.
Compressed	Specifies a file or directory that is compressed, as denoted by the constant value, 0x800. For a file, all of the data in the file is compressed. For a directory, compression is the default for newly created files and subdirectories.
Offline	Specifies that the data of a file is not available immediately, as denoted by the constant value, 0x1000. This attribute indicates that the file data is physically moved to offline storage. This attribute is used by Remote Storage, which is the hierarchical storage management software. Applications should not arbitrarily change this attribute.
NotContentIndexed	Specifies that a file is not to be indexed by the content indexing service, as denoted by the constant value, 0x2000.
Encrypted	Specifies a file or directory that is encrypted, as denoted by the constant value, 0x4000. For a file, all data streams in the file are encrypted. For a directory, encryption is the default for newly created files and subdirectories.
Deleted	Specifies a file or directory that is marked as deleted.
IntegrityStream	Specifies the directory or user data stream is configured with integrity (only supported on ReFS volumes), as denoted by the constant value, 0x8000. It is not included in an ordinary directory listing. The integrity setting persists with the file if it's renamed. If a file is copied the destination file will have integrity set if either the source file or destination directory have integrity set. NOTE: This flag is supported ONLY for Windows Server 8 Beta and later.
Virtual	Specifies a reserved system value, as denoted by the constant value, 0x10000.
NoScrubData	The user data stream not to be read by the background data integrity scanner (AKA

	scrubber), as denoted by the constant value, 0x20000. When set on a directory it only provides inheritance. This flag is only supported on Storage Spaces and ReFS volumes in Windows 8 and Windows Server 8 Beta and later. It is not included in an ordinary directory listing.

3.2.48 WindowsHandleObjectType (extends Common:DefinedObjectType)

The WindowsHandleObjectType type is intended to characterize Windows handles.

Property	Type	Mult	Description
ID	Common: UnsignedInteger ObjectAttributeType	0..1	The ID element refers to the unique number used to identify the handle.
Name	Common: StringObject AttributeType	0..1	The Name element specifies the name of the handle.
Type	WinHandleObj: HandleType	0..1	The Type element specifies the handle type, which is equivalent to the type of Windows object that the handle refers to.
Object_Address	Common: UnsignedLong ObjectAttributeType	0..1	The Object_Address element specifies the address of the Windows object that the handle refers to.
Access_Mask	Common: UnsignedLong ObjectAttributeType	0..1	The Access_Mask element specifies the access bitmask of the handle.
Pointer_Count	Common: UnsignedLong ObjectAttributeType	0..1	The Pointer_Count element specifies the count of pointer references to the Windows object that the handle refers to.

WindowsHandleListType

The WindowsHandleListType type specifies a list of Windows handles, for re-use in other objects.

Property	Type	Mult	Description
Handle	WinHandleObj: WindowsHandle ObjectType	1..∞	The Handle element characterizes a single Windows handle.

3.2.48.1 HandleType (restriction Common:BaseObjectAttributeType)

HandleType specifies Windows handle types via a union of the HandleTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinHandleObj:HandleTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.48.2 HandleTypeEnum

The WindowsHandleType is a non-exhaustive enumeration of Windows handle types.

Restriction base: string

Enumeration Value	Description
AccessToken	Specifies an access token handle.
Event	Specifies an event handle.
File	Specifies a file handle.

FileMapping	Specifies a file mapping handle.
Job	Specifies a job handle.
IOCompletionPort	Specifies an IO completion port handle.
Mailslot	Specifies a mailslot handle.
Mutex	Specifies a mutex handle.
NamedPipe	Specifies a named pipe handle.
Pipe	Specifies a pipe handle.
Process	Specifies a process handle.
Semaphore	Specifies a semiphore handle.
Thread	Specifies a thread handle.
Transaction	Specifies a transaction handle.
WaitableTimer	Specifies a waitable timer handle.
RegistryKey	Specifies a registry key handle.
Window	Specifies a window handle.
ServiceControlManager	Specifies a service control manager handle.

3.2.49 WindowsKernelHookObjectType (extends Common:DefinedObjectType)

The WindowsKernelHookObjectType type is intended to characterize Windows kernel function hooks.

Property	Type	Mult	Description
Digital_Signature_Hooking	Common: DigitalSignature InfoType	0..1	The Digital_Signature_Hooked element is optional and specifies the digital signature of the hooking code.
Digital_Signature_Hooked	Common: DigitalSignature InfoType	0..1	The Digital_Signature_Hooked element is optional and specifies the digital signature of the hooked code.
Hooking_Address	Common: UnsignedLong ObjectAttributeType	0..1	The Hooking_Address element is optional and specifies the address from where the hooking occurs.
Hook_Description	Common: StringObject AttributeType	0..1	The Hook_Description element is optional and provides a description of the nature of the hook.
Hooked_Function	Common: StringObject AttributeType	0..1	The Hooked_Function element specifies the name of the function that is hooked.
Hooked_Module	Common: StringObject AttributeType	0..1	The Hooked_Module element specifies the name of the module that is hooked.
Hooking_Module	Common: StringObject AttributeType	0..1	The Hooking_Module element specifies the name of the module that is doing the hooking.
Type	WinKernelHookObj: KernelHookType	0..1	The Type element specifies the type of hook being characterized.

3.2.49.1 KernelHookType (restriction Common:BaseObjectAttributeType)

KernelHookType specifies Windows kernel hook types via a union of the KernelHookTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinKernelHookObj:KernelHookTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected

			type for the value of the specified element.

3.2.49.2 KernelHookTypeEnum

The KernelHookTypeEnum type is a non-exhaustive enumeration of Windows kernel hook types.

Restriction base: string

Enumeration Value	Description
IAT_API	Specifies a kernel hook type of IAT_API.
Inline_Function	Specifies an inline function type of kernel hook.
Instruction_Hooking	Specifies an instruction hooking type of kernel hook.

3.2.50 WindowsKernelObjectType (extends Common:DefinedObjectType)

The WindowsKernelObjectType type is intended to characterize Windows Kernel structures.

Property	Type	Mult	Description
IDT	WinKernelObj: IDTEntryListType	0..1	The IDT element characterizes the Windows Interrupt Descriptor Table (IDT).
SSDT	WinKernelObj: SSDTEntryListType	0..1	The SSDT element characterizes the Windows System Service Descriptor Table (SSDT). The SSDT is a structure that kernel uses to dispatch functions. KeServiceDescriptorTable is a table exported by the kernel that contains pointers to four SSDTs, one for the native API, one for user/GDI support, one of IIS SPUD (in Windows 2000), and one unused.See http://www.honeynet.org/node/438; Sven Boris Schreiber, Undocumented Windows 2000 Secrets (http://undocumented.rawol.com/sbs-w2k-2-the-windows-2000-native-api.pdf); Greg Hoglund and James Butler, Rootkits: Subverting the WIndows kernel

3.2.50.1 SSDTEntryListType

The SSDTEntryListType type specifies a listing of the entries in the System Service Descriptor Table (SSDT).

Property	Type	Mult	Description
SSDT_Entry	WinKernelObj:SSDTEntryType	1..∞	Specifies an entry in the System Service Descriptor Table.

3.2.50.2 SSDTEntryType

The SSDTEntryType type specifies a single entry in the System Service Descriptor Table (SSDT).

Property	Type	Mult	Description
hooked	boolean	1..1	The hooked attribute specifies whether the SSDT entry is hooked.
Service_Table_Base	Common: HexBinary ObjectAttributeType	0..1	Pointer to the system service dispatch table, an array of function addresses which is indexed by the system call number.
Service_Counter_Table_Base	Common: HexBinary ObjectAttributeType	0..1	Pointer to an array of usage counters.

Number_Of_Services	Common: NonNegative IntegerObject AttributeType	0..1	Number of entries in the system service dispatch table.
Argument_Table_Base	Common: HexBinary ObjectAttributeType	0..1	Pointer to an array of bytes, which indicate the number of bytes used by the function's arguments.

3.2.50.3 IDTEntryListType

The IDTEntryListType type specifies a listing of the entries in the Interrupt Descriptor Table (IDT). The IDT is specific to the I386 architecture, indicating where the Prtoetcted mode Interrupt Service Routines (ISR) are located. See http://wiki.osdev.org/Interrupt_Descriptor_Table

Property	Type	Mult	Description
IDT_Entry	WinKernelObj:IDTEntryType	1..∞	Specifies an entry in the Interrupt Descriptor Table.

3.2.50.4 IDTEntryType

The IDTEntryType type specifies a single entry in the Interrupt Descriptor Table (IDT). Entries can be interrupt gates, task gates, and trap gates.

Property	Type	Mult	Description
Type_Attr	Common: HexBinary ObjectAttributeType	0..1	A byte that encodes the gate type and interrupt attributes (e.g., the Descriptor Privilege Level).
Offset_High	Common: HexBinary ObjectAttributeType	0..1	Higher part of the interrupt function's offset address bits 16-31 in 32-bit, bits 32-63 in 64-bit)
Offset_Low	Common: HexBinary ObjectAttributeType	0..1	Lower part of the interrupt function's offset address (bits 0-15)
Offset_Middle	Common: HexBinary ObjectAttributeType	0..1	In 64-bit architectures, middle part of the interrupt function's offset address (bits 16-31)
Selector	Common: HexBinary ObjectAttributeType	0..1	A 16-bit value that points to a code segment selector in the Global Descriptot Table.

3.2.51 WindowsMailslotObjectType (extends Common:DefinedObjectType)

The WindowsMailslotObjectType is intended to characterize Windows mailslot objects.

Property	Type	Mult	Description
Handle	WinHandleObj: WindowsHandleListType	0..1	The Handle element specifies the open Windows handle to the mailslot. It imports and uses the WindowsHandleObjectType from the CybOX Windows Handle Object.
Max_Message_Size	Common: NonNegative IntegerObject AttributeType	0..1	The Max_Message_Size element specifies the maximum message size for the mailslot, in bytes.
Name	Common: StringObject AttributeType	0..1	The Name element specifies the name of the mailslot.
Read_Timeout	Common: NonNegative	0..1	The Read_Timeout element specifies the amount of time, in milliseconds, a read operation can wait for a

	IntegerObjectAttributeType		message to be written to the mailslot before a time-out occurs.
Security_Attributes	Common:StringObjectAttributeType	0..1	The Security_Attributes element specifies the Windows security attributes for the mailslot.

3.2.52 WindowsMutexObjectType (extends MutexObj:MutexObjectType)

The WindowsMutexObjectType type is intended to characterize Windows mutual exclusion (mutex) objects.

Property	Type	Mult	Description
Handle	WinHandleObj:WindowsHandleObjectType	0..1	The Handle element specifies the open Windows handle to the mutex. It imports and uses the WindowsHandleObjectType from the CybOX Windows Handle Object.
Security_Attributes	Common:StringObjectAttributeType	0..1	The Security_Attributes element specifies the Windows security attributes for the mutex.

3.2.53 WindowsNetworkRouteEntryObjectType (extends NetworkRouteEntryObj:NetworkRouteEntryObjectType)

The WindowsNetworkRouteEntryObjectType type is intended to characterize Windows network routing table entries.

Property	Type	Mult	Description
NL_Route_Origin	WinNetworkRouteEntryObj:NLRouteOriginType	0..1	The NLRouteOrigin is a route origination point, as detailed in the NL_ROUTE_ORIGIN enumertaion in the MIB_IPFORWARD_ROW2 structure. For more information, see http://msdn.microsoft.com/en-us/library/windows/desktop/aa814494(v=vs.85).aspx for the MIB_IPFORWARD_ROW2 structure and http://msdn.microsoft.com/en-us/library/windows/hardware/ff568764(v=vs.85).aspx for the NL_ROUTE_ORIGIN enumeration.

3.2.53.1 NLRouteOriginType (restriction Common:BaseObjectAttributeType)

NLRouteOriginType specifies Windows-centric network route origination values via a union of the RouteOriginEnum type and the atomic xs:string type. Its base type is the CybOX BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinNetworkRouteEntryObj:NLRouteOriginEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.53.2 NLRouteOriginEnum

The NLRouteOriginEnum type is a enumeration of network route origination points, as detailed in the NL_ROUTE_ORIGIN enumertaion in the MIB_IPFORWARD_ROW2 structure. For more information, see http://msdn.microsoft.com/en-us/library/windows/desktop/aa814494(v=vs.85).aspx for the

MIB_IPFORWARD_ROW2 structure and http://msdn.microsoft.com/en-us/library/windows/hardware/ff568764(v=vs.85).aspx for the NL_ROUTE_ORIGIN enumeration.

Restriction base: string

Enumeration Value	Description
NlroManual	Specifies that the origin was determined as a result of manual configuration.
NlroWellKnown	Specifies that the route is well-known.
NlroDHCP	Specifies that the origin was determined as a result of DHCP configuration.
NlroRouterAdvertisement	Specifies that the origin was determined as a result of router advertisement.
Nlro6to4	Specifies that the origin was determined as a result of 6to4 tunneling.

3.2.54 WindowsNetworkShareObjectType (extends Common:DefinedObjectType)

he WindowsNetworkShareObjectType type is intended to characterize Windows network shares.

Property	Type	Mult	Description
Current_Uses	Common: NonNegative IntegerObject AttributeType	0..1	The Current_Uses element specifies the current number of uses of the network share.
Local_Path	Common: StringObject AttributeType	0..1	The Local_Path element specifies the fully-qualified path on the local system to the network share.
Max_Uses	Common: NonNegative IntegerObject AttributeType	0..1	The Max_Uses element specifies the maximum number of concurrent connections to the network share.
Netname	Common: StringObject AttributeType	1..1	The Netname element specifies the network name of the network share.
Type	WinNetworkShareObj: SharedResourceType	0..1	The Type element specifies the type of the network share.

3.2.54.1 SharedResourceType (restriction Common:BaseObjectAttributeType)

SharedResourceType specifies Windows shared resource types via a union of the SharedResourceTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinNetworkShareObj:SharedResourceTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.54.2 SharedResourceTypeEnum

The SharedResourceTypeEnum type is an enumeration of Windows that specifies shared resource types for shared devices. These can be checked via the NetShareCheck function. See http://msdn.microsoft.com/en-us/library/windows/desktop/bb525385(v=vs.85).aspx for more information.

Restriction base: string

Enumeration Value	Description
STYPE_DISKTREE	Specifies that the shared device is a disk drive.

STYPE_DISKTREE_SPECIAL	Specifies that the shared device is a disk drive with special share reserved for interprocess communcation (IPC$) or remote administration of the server (ADMIN$). Can also refer to administrative shares such as C$, D$, E$, and so forth. For more information, see http://msdn.microsoft.com/en-us/library/windows/desktop/bb525391(v=vs.85).aspx.
STYPE_DISKTREE_TEMPORARY	Specifies that the shared device is a disk drive and serves as a temporary share.
STYPE_DISKTREE_SPECIAL_TEMPORARY	Specifies that the shared device is a disk drive with special share reserved for interprocess communcation (IPC$) or remote administration of the server (ADMIN$) and serves a temporary share. Can also refer to administrative shares such as C$, D$, E$, and so forth. For more information, see http://msdn.microsoft.com/en-us/library/windows/desktop/bb525391(v=vs.85).aspx.
STYPE_PRINTQ	Specifies that the shared device is a print queue.
STYPE_PRINTQ_SPECIAL	Specifies that the shared device is a disk drive with special share reserved for interprocess communcation (IPC$) or remote administration of the server (ADMIN$). Can also refer to administrative shares such as C$, D$, E$, and so forth. For more information, see http://msdn.microsoft.com/en-us/library/windows/desktop/bb525391(v=vs.85).aspx.
STYPE_PRINTQ_TEMPORARY	Specifies that the shared device is a print queue and serves as a temporary share.
STYPE_PRINTQ_SPECIAL_TEMPORARY	Specifies that the shared device is a print queue with special share reserved for interprocess communcation (IPC$) or remote administration of the server (ADMIN$) and serves a temporary share. Can also refer to administrative shares such as C$, D$, E$, and so forth. For more information, see http://msdn.microsoft.com/en-us/library/windows/desktop/bb525391(v=vs.85).aspx.
STYPE_DEVICE	Specifies that the shared device is a communications device.
STYPE_DEVICE_SPECIAL	Specifies that the shared device is a communications device with special share reserved for interprocess communcation (IPC$) or remote administration of the server (ADMIN$). Can also refer to administrative shares such as C$, D$, E$, and so forth. For more information, see http://msdn.microsoft.com/en-us/library/windows/desktop/bb525391(v=vs.85).aspx.
STYPE_DEVICE_TEMPORARY	Specifies that the shared device is a communications device and serves as a temporary share.
STYPE_DEVICE_SPECIAL_TEMPORARY	Specifies that the shared device is a communications device with special share reserved for interprocess communcation (IPC$) or remote administration of the server (ADMIN$) and serves a temporary share. Can also refer to administrative shares such as C$, D$, E$, and so forth. For more information, see http://msdn.microsoft.com/en-us/library/windows/desktop/bb525391(v=vs.85).aspx.
STYPE_IPC	Specifies that the shared device is an Interprocess Communication (IPC) device.
STYPE_IPC_SPECIAL	Specifies that the shared device is an Interprocess Communication (IPC) device with special share reserved for interprocess communcation (IPC$) or remote administration of the server (ADMIN$). Can also refer to administrative shares such as C$, D$, E$, and so forth. For more information, see http://msdn.microsoft.com/en-us/library/windows/desktop/bb525391(v=vs.85).aspx.

STYPE_IPC_TEMPORARY	Specifies that the shared device is an Interprocess Communication (IPC) device and serves as a temporary share.
STYPE_IPC_SPECIAL_TEMPORARY	Specifies that the shared device is an Interprocess Communication (IPC) device with special share reserved for interprocess communcation (IPC$) or remote administration of the server (ADMIN$) and serves a temporary share. Can also refer to administrative shares such as C$, D$, E$, and so forth. For more information, see http://msdn.microsoft.com/en-us/library/windows/desktop/bb525391(v=vs.85).aspx.

3.2.54.3 AccessPermissionsGroup

The accesspermissions group specifies the various permissions for Windows network shares.

Property	Type	Mult	Description
ACCESS_ALL	boolean	1..1	The ACCESS_ALL field specifies the permission to read, write, create, execute, and delete resources, and to modify their attributes and permissions.
ACCESS_ATRIB	boolean	1..1	The ACCESS_ATRIB field specifies the permission to modify the resource's attributes (such as the date and time when a file was last modified).
ACCESS_CREATE	boolean	1..1	The ACCESS_CREATE field specifies the permission to create an instance of the resource (such as a file); data can be written to the resource as the resource is created.
ACCESS_DELETE	boolean	1..1	The ACCESS_DELETE field specifies the permission to delete the resource.
ACCESS_EXEC	boolean	1..1	The ACCESS_EXEC field specifies the permission to execute the resource.
ACCESS_PERM	boolean	1..1	The ACCESS_PERM field specifies the permission to modify the permissions (read, write, create, execute, and delete) assigned to a resource for a user or application.
ACCESS_READ	boolean	1..1	The ACCESS_READ field specifies the permission to read data from a resource and, by default, to execute the resource.
ACCESS_WRITE	boolean	1..1	The ACCESS_WRITE field specifies the permission to write data to the resource.

3.2.55 WindowsPipeObjectType (extends PipeObj:PipeObjectType)

The WindowsPipeObjectType type is intended to characterize Windows pipes.

Property	Type	Mult	Description
Default_Time_Out	Common: NonNegativeIntegerObjectAttributeType	0..1	The Default_Time_Out element specifies the default time-out value for the pipe, in milliseconds.
Handle	WinHandleObj: WindowsHandleObjectType	0..1	The Handle element specifies the open Windows handle to the pipe. It imports and uses the WindowsHandleObjectType from the CybOX Windows Handle Object.
In_Buffer_Size	Common: NonNegative	0..1	The In_Buffer_Size element specifies the number of bytes to reserve for the input buffer of the pipe.

	Common:IntegerObjectAttributeType		
Max_Instances	Common:NonNegativeIntegerObjectAttributeType	0..1	The Max_Instances element specifies the maximum number of instances that can be created for this pipe.
Open_Mode	Common:HexBinaryObjectAttributeType	0..1	The Open_Mode element specifies the open mode used for the pipe.
Out_Buffer_Size	Common:NonNegativeIntegerObjectAttributeType	0..1	The Out_Buffer_Size element specifies the number of bytes to reserve for the output buffer of the pipe.
Pipe_Mode	Common:HexBinaryObjectAttributeType	0..1	The Pipe_Mode element specifies the mode used for the pipe.
Security_Attributes	Common:StringObjectAttributeType	0..1	The Security_Attributes element specifies the Windows security attributes for the pipe.

3.2.56 WindowsPrefetchObjectType (extends Common:DefinedObjectType)

The WindowsPrefetchObjectType type is intended to characterize entries in the Windows prefetch files. Starting with Windows XP, prefetching was introduced to speed up application startup. The prefetch object draws upon the descriptions and XML sample at http://www.forensicswiki.org/wiki/Prefetch_XML

Property	Type	Mult	Description
Application_File_Name	Common:StringObjectAttributeType	0..1	Name of the executable of the prefetch file.
Prefetch_Hash	Common:StringObjectAttributeType	0..1	An eight character hash of the location from which the application was run.
Times_Executed	Common:LongObjectAttributeType	0..1	The number of times the prefetch application has executed.
First_Run	Common:DateTimeObjectAttributeType	0..1	Timestamp of when the prefetch application was first run.
Last_Run	Common:DateTimeObjectAttributeType	0..1	Timestamp of when the prefetch application was last run.
Volume	WinPrefetchObj:VolumeType	0..1	The volume from which the prefetch application was run. If the applicatin was run from multiple volumes, there will be a separate prefetch file for each.
Accessed_File_List	WinPrefetchObj:AccessedFileListType	0..1	Files (e.g., DLLs and other support files) used by the application during startup.
Accessed_Directory_List	WinPrefetchObj:AccessedDirectoryListType	0..1	Directories accessed by the prefetch application during starup.

3.2.56.1 AccessedFileListType

The AccessedFileListType specifies a list of files accessed by a prefetch application.

Property	Type	Mult	Description
Accessed_Filename	Common:StringObject	1..∞	Specifies the filename of the accessed file.

	AttributeType		

3.2.56.2 AccessedDirectoryListType

The AccessedDirectoryListType specifies a list of directories accessed by a prefetch application.

Property	Type	Mult	Description
Accessed_Directory	Common: StringObject AttributeType	1..∞	Specifies the pathname of the accessed directory.

3.2.56.3 VolumeType

VolumeType characterizes the volume information in the Windows prefetch file.

Property	Type	Mult	Description
VolumeItem	WinVolumeObj: WindowsVolumeObjectType	1..∞	The volume that the prefetch application was run from. The only item in the prefecth file is the volume name.
DeviceItem	DeviceObj:DeviceObjectType	1..∞	The device that the prefetch application was run from. The only item in the prefetch file is the device serial number.

3.2.57 WindowsProcessObjectType (extends ProcessObj:ProcessObjectType)

The WindowsProcessObjectType type is intended to characterize Windows processes.

Property	Type	Mult	Description
aslr_enabled	boolean	1..1	The aslr_enabled attribute specifies whether Address Space Layout Randomization (ASLR) is enabled for the process.
dep_enabled	boolean	1..1	The dep_enabled attribute specifies whether Data Execution Prevention (DEP) is enabled for the process.
Handle_List	WinHandleObj: WindowsHandleListType	0..1	The Handle_List element specifies a list of Windows Handles opened or used by the process.
Priority	Common: StringObject AttributeType	0..1	The Priority element speciifes the current priority of the process in Windows.
Section_List	WinProcessObj: MemorySectionListType	0..1	The Section_List element specifies the memory sections used by the process.
Security_ID	Common: StringObject AttributeType	0..1	The Security_ID element specifies the Security ID (SID) value assigned to the process.
Startup_Info	WinProcessObj: StartupInfoType	0..1	The Startup_Info element specifies the STARTUP_INFO struct used by the process.
Security_Type	Common:SIDType	0..1	The Security_Type element specifies the type of Security ID (SID) assigned to the process.
Window_Title	Common: StringObject AttributeType	0..1	The Window_Title element specifies the title of the main window of the process.

3.2.57.1 MemorySectionListType

The MemorySectionListType type specifies a list of memory sections used by the process.

Property	Type	Mult	Description

Memory_Section	MemoryObj: MemoryObjectType	1..∞	The Memory_Section element specifies a memory section used by the process. It imports and uses the MemoryObjectType from the CybOX Memory Object.

3.2.57.2 StartupInfoType

The StartupInfoType type encapsulates the information contained in the STARTUPINFO struct for the process.

Property	Type	Mult	Description
lpDesktop	Common: StringObject AttributeType	0..1	The lpDesktop element specifies the name of the desktop, or the name of both the desktop and window station for this process.
lpTitle	Common: StringObject AttributeType	0..1	The lpTitle element specifies the title displayed in the title bar if a new console window is created.
dwX	Common: IntegerObject AttributeType	0..1	The dwX element specifies the x offset of the upper left corner of a window if a new window is created, in pixels.
dwY	Common: IntegerObject AttributeType	0..1	The dwY element specifies the y offset of the upper left corner of a window if a new window is created, in pixels.
dwXSize	Common: PositiveInteger ObjectAttributeType	0..1	The dwXSize element specifies the width of the window if a new window is created, in pixels.
dwYSize	Common: PositiveInteger ObjectAttributeType	0..1	The dwYSize element specifies the height of the window if a new window is created, in pixels.
dwXCountChars	Common: PositiveInteger ObjectAttributeType	0..1	The dwXCountChars element specifies the screen buffer width, in character columns.
dwYCountChars	Common: PositiveInteger ObjectAttributeType	0..1	The dwYCountChars element specifies the screen buffer height, in character rows.
dwFillAttribute	Common: IntegerObject AttributeType	0..1	The dwFillAttribute element specifies the initial text and background colors if a new console window is created in a console application.
dwFlags	Common: IntegerObject AttributeType	0..1	The dwFlags element specifies a bitfield that determines whether certain STARTUPINFO members are used when the process creates a window.
wShowWindow	Common: IntegerObject AttributeType	0..1	The wShowWindow element specifies STARTF_USESHOWWINDOW, this member can be any of the values that can be specified in the nCmdShow parameter for the ShowWindow function, except for SW_SHOWDEFAULT.
hStdInput	WinHandleObj: WindowsHandleObjectType	0..1	The hStdInput element specifies the standard input handle for the process.
hStdOutput	WinHandleObj: WindowsHandleObjectType	0..1	The hStdOutput element specifies the standard output handle for the process.
hStdError	WinHandleObj: WindowsHandleObjectType	0..1	The hStdError element specifies the standard error handle for the process.

3.2.58 WindowsRegistryKeyObjectType (extends Common:DefinedObjectType)

The WindowsRegistryObjectType type is intended to characterize Windows registry objects, including Keys and Key/Value pairs.

Property	Type	Mult	Description
Key	Common: StringObject AttributeType	0..1	The Key element specifies the full key to the Windows registry object, not including the hive.
Hive	WinRegistryKeyObj: RegistryHiveType	0..1	The Hive element specifies the Windows registry hive to which the registry object belongs to.
Number_Values	Common: UnsignedInteger ObjectAttributeType	0..1	The Number_Values element specifies the number of values found in the registry key.
Values	WinRegistryKeyObj: RegistryValuesType	0..1	The Values element specifies the values (with their name/data pairs) held within the registry key.
Modified_Time	Common: DateTime ObjectAttributeType	0..1	The Modified_Time element specifies the last date/time that the registry object was modified.
Creator_Username	Common: StringObject AttributeType	0..1	The Creator_Username element specifies the name of the user who created the registry object.
Handle_List	WinHandleObj: WindowsHandleListType	0..1	The Handle_List element specifies a list of open Handles for this registry object.
Number_Subkeys	Common: UnsignedInteger ObjectAttributeType	0..1	The Number_Subkeys element specifies the number of subkeys contained under the registry key.
Subkeys	WinRegistryKeyObj: RegistrySubkeysType	0..1	The Subkeys element specifies the set of subkeys contained under the registry key.
Byte_Runs	Common:ByteRunsType	0..1	The Byte_Runs element contains a list of byte runs from the raw registry.

3.2.58.1 RegistryValueType

The RegistryValueType type is intended to characterize Windows registry Value name/data pairs.

Property	Type	Mult	Description
Name	Common: StringObject AttributeType	0..1	The Name element specifies the name of the registry value.
Data	Common: StringObject AttributeType	0..1	The Data element specifies the data contained in the registry value.
Datatype	WinRegistryKeyObj: RegistryDatatypeType	0..1	The Datatype element specifies the registry (REG_*) datatype used in the registry value.
Byte_Runs	Common:ByteRunsType	0..1	The Byte_Runs element contains a list of byte runs from the raw registry key entry.

3.2.58.2 RegistryDatatypeType (restriction Common:BaseObjectAttributeType)

Registry_Datatype specifies Windows registry datatypes via a union of the RegistryDataTypesEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinRegistryKeyObj:RegistryDataTypesEnum, string

Property	Type	Mult	Description

datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.58.3 RegistryHiveType (restriction Common:BaseObjectAttributeType)

RegistryHiveType specifies Windows registry hive types via a union of the RegistryHiveEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinRegistryKeyObj:RegistryHiveEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.58.4 RegistryDataTypesEnum

The RegistryDataTypesEnum type is an enumeration of Windows registry datatypes (REG_*). See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms724884(v=vs.85).aspx See also: http://pubs.logicalexpressions.com/Pub0009/LPMArticle.asp?ID=361

Restriction base: string

Enumeration Value	Description
REG_NONE	No defined value type.
REG_SZ	A null-terminated string. This will be either a Unicode or an ANSI string, depending on whether you use the Unicode or ANSI functions.
REG_EXPAND_SZ	A null-terminated string that contains unexpanded references to environment variables (for example, "%PATH%"). It will be a Unicode or ANSI string depending on whether you use the Unicode or ANSI functions.
REG_BINARY	Binary data in any form.
REG_DWORD	A 32-bit number.
REG_DWORD_BIG_ENDIAN	A 32-bit number in big-endian format. Some UNIX systems support big-endian architectures.
REG_LINK	A null-terminated Unicode string that contains the target path of a symbolic link.
REG_MULTI_SZ	A sequence of null-terminated strings, terminated by an empty string (\0).
REG_RESOURCE_LIST	A series of nested arrays designed to store a resource list used by a hardware device driver or one of the physical devices it controls. This data is detected and written into the ResourceMap tree by the system and is displayed in Registry Editor in hexadecimal format as a Binary Value.
REG_FULL_RESOURCE_DESCRIPTOR	A series of nested arrays designed to store a resource list used by a physical hardware device. This data is detected and written into the HardwareDescription tree by the system and is displayed in Registry Editor in hexadecimal format as a Binary Value.
REG_RESOURCE_REQUIREMENTS_LIST	Device driver list of hardware resource requirements in Resource Map tree. See http://www.mdgx.com/reg.htm
REG_QWORD	A 64-bit number.
REG_INVALID_TYPE	Specifies an invalid key.

3.2.58.5 RegistryHiveEnum

The RegistryHiveEnum type is an enumeration of Windows registry hives (HKEY_*). See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms724836(v=vs.85).aspx

Restriction base: string

Enumeration Value	Description
HKEY_CLASSES_ROOT	Registry entries subordinate to this key define types (or classes) of documents and the properties associated with those types. Shell and COM applications use the information stored under this key.
HKEY_CURRENT_CONFIG	Contains information about the current hardware profile of the local computer system. The information under HKEY_CURRENT_CONFIG describes only the differences between the current hardware configuration and the standard configuration.
HKEY_CURRENT_USER	Registry entries subordinate to this key define the preferences of the current user. These preferences include the settings of environment variables, data about program groups, colors, printers, network connections, and application preferences. This key makes it easier to establish the current user's settings; the key maps to the current user's branch in HKEY_USERS.
HKEY_LOCAL_MACHINE	Registry entries subordinate to this key define the physical state of the computer, including data about the bus type, system memory, and installed hardware and software.
HKEY_USERS	Registry entries subordinate to this key define the default user configuration for new users on the local computer and the user configuration for the current user.
HKEY_CURRENT_USER_LOCAL_SETTINGS	Registry entries subordinate to this key define preferences of the current user that are local to the machine. These entries are not included in the per-user registry portion of a roaming user profile.
HKEY_PERFORMANCE_DATA	Registry entries subordinate to this key allow you to access performance data. The data is not actually stored in the registry; the registry functions cause the system to collect the data from its source.
HKEY_PERFORMANCE_NLSTEXT	Registry entries subordinate to this key reference the text strings that describe counters in the local language of the area in which the computer system is running. These entries are not available to Regedit.exe and Regedt32.exe.
HKEY_PERFORMANCE_TEXT	Registry entries subordinate to this key reference the text strings that describe counters in US English. These entries are not available to Regedit.exe and Regedt32.exe.

3.2.58.6 RegistryValuesType

The RegistryValuesType type specifies the values (with their name/data pairs) held within the registry key.

Property	Type	Mult	Description
Value	WinRegistryKeyObj: RegistryValueType	1..∞	The Value element specifies the value (with name/data pair) held within the registry key.

3.2.58.7 RegistrySubkeysType

The RegistrySubkeysType specifies the set of subkeys contained under the registry key.

Property	Type	Mult	Description
Subkey	WinRegistryKeyObj: WindowsRegistryKeyObjectType	1..∞	The Subkey element specifies a single subkey contained under the registry key.

3.2.59 WindowsSemaphoreObjectType (extends SemaphoreObj:SemaphoreObjectType)

The WindowsSemaphoreObjectType is intended to characterize Windows semaphore (synchronization) objects.

Property	Type	Mult	Description
Handle	WinHandleObj: WindowsHandleObjectType	0..1	The Handle element specifies the open Windows handle to the semaphore. It imports and uses the WindowsHandleObjectType from the CybOX Windows Handle Object.
Security_Attributes	Common: StringObject AttributeType	0..1	The Security_Attributes element specifies the Windows security attributes for the semaphore.

3.2.60 WindowsServiceObjectType (extends WinProcessObj:WindowsProcessObjectType)

The WindowsServiceObjectType type is intended to characterize Windows services.

Property	Type	Mult	Description
service_dll_signature_ exists	boolean	1..1	Indicates whether or not the DLL is signed.
service_dll_signature_ verified	boolean	1..1	Indicates whether or not the DLL's signature was verified.
Description_List	WinServiceObj: ServiceDescriptionList Type	0..1	A list of description items for this service.
Display_Name	Common: StringObject AttributeType	0..1	The Display_Name element specifies the displayed name of the service in Windows GUI controls. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms683228(v=vs.85).aspx
Service_Name	Common: StringObject AttributeType	0..1	The Name element specifies the name of the service. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms683229(v=vs.85).aspx
Service_DLL	Common: StringObject AttributeType	0..1	The Service_DLL element specifies name of the DLL instantiated in the service.
Service_DLL_Certificate_ Issuer	Common: StringObject AttributeType	0..1	The Certificate Authority (CA) that issued the certificate used to sign the service DLL.
Service_DLL_Certificate_ Subject	Common: StringObject AttributeType	0..1	The subject of the certifcate (the entity being authenticated).
Service_DLL_Hashes	Common:HashListType	0..1	Hashes for the Service DLL file.
Service_DLL_Signature_ Description	Common: StringObject AttributeType	0..1	The Service_DLL_Signature_Description element provides a description of the digital signature for the service DLL.
Startup_Command_Line	Common: StringObject AttributeType	0..1	The Startup_Command_Line element specifies the full command line used to start the service.
Startup_Type	WinServiceObj: ServiceModeType	0..1	Service start options. See http://msdn.microsoft.com/en-us/library/windows/desktop/ms682450(v=vs.85).aspx
Service_Status	WinServiceObj:	0..1	Status information for a service. See also:

	ServiceStatusType		http://msdn.microsoft.com/en-us/library/windows/desktop/ms685996(v=vs.85).aspx
Service_Type	WinServiceObj:ServiceType	0..1	The Type element specifies the type of the service.
Started_As	Common:StringObjectAttributeType	0..1	The Started_As element specifies the name of the account under which the service was started.

3.2.60.1 ServiceDescriptionListType

A collection of service descriptions.

Property	Type	Mult	Description
Description	Common:StringObjectAttributeType	1..∞	A description of the service. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms685156(v=vs.85).aspx

3.2.60.2 ServiceModeType (restriction Common:BaseObjectAttributeType)

ServiceModeType specifies Windows service modes via a union of the ServiceModeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinServiceObj:ServiceModeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.60.3 ServiceStatusType (restriction Common:BaseObjectAttributeType)

ServiceModeType specifies Windows service states via a union of the ServiceStatusEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinServiceObj:ServiceStatusEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.60.4 ServiceType (restriction Common:BaseObjectAttributeType)

ServiceType specifies Windows service types via a union of the ServiceTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinServiceObj:ServiceTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.60.5 ServiceModeEnum

The ServiceModeEnum type is an enumeration of service modes. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms682450(v=vs.85).aspx

Enumeration Value	Description
SERVICE_AUTO_START	A service started automatically by the service control manager during system startup.
SERVICE_BOOT_START	A device driver started by the system loader. This value is valid only for driver services.
SERVICE_DEMAND_START	A service started by the service control manager when a process calls the StartService function.
SERVICE_DISABLED	A service that cannot be started. Attempts to start the service result in the error code ERROR_SERVICE_DISABLED.
SERVICE_SYSTEM_START	A device driver started by the IoInitSystem function. This value is valid only for driver services.

3.2.60.6 ServiceStatusEnum

The ServiceStatusEnum type is an enumeration of potential service states. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms685996(v=vs.85).aspx

Enumeration Value	Description
SERVICE_CONTINUE_PENDING	The service continue is pending.
SERVICE_PAUSE_PENDING	The service pause is pending.
SERVICE_PAUSED	The service is paused.
SERVICE_RUNNING	The service is running.
SERVICE_START_PENDING	The service is starting.
SERVICE_STOP_PENDING	The service is stopping.
SERVICE_STOPPED	The service is not running.

3.2.60.7 ServiceTypeEnum

The ServiceTypeEnum type is an enumeration of service types. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms685996(v=vs.85).aspx

Enumeration Value	Description
SERVICE_KERNEL_DRIVER	The service is a device driver.
SERVICE_FILE_SYSTEM_DRIVER	The service is a file system driver.
SERVICE_WIN32_OWN_PROCESS	The service runs in its own process.
SERVICE_WIN32_SHARE_PROCESS	The service shares a process with other services.

3.2.61 WindowsSystemObjectType (extends SystemObj:SystemObjectType)

The WindowsSystemObjectType type is intended to characterize Windows systems.

Property	Type	Mult	Description
Domain	Common: StringObject AttributeType	0..∞	The domain that the system belongs to.
Global_Flag_List	WinSystemObj: GlobalFlagListType	0..1	A list of global flags. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff549557(v=vs.85).aspx
NetBIOS_Name	Common: StringObject	0..1	The NetBIOS_Name element specifies the NetBIOS (Network Basic Input/Output System) name of the

	AttributeType		Windows system. This is not the same as the host name.
Open_Handle_List	WinHandleObj:Windows HandleListType	0..1	The Open_Handle_List element specifies the list of open handles for the Windows system.
Product_ID	Common: StringObject AttributeType	0..1	The Product ID. See also: http://support.microsoft.com/gp/pidwin
Product_Name	Common: StringObject AttributeType	0..1	The ProductName of the current installation of Windows. This is typically found in HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion!ProductName
Registered_Organization	Common: StringObject AttributeType	0..1	The organization that this copy of Windows is registered to.
Registered_Owner	Common: StringObject AttributeType	0..1	The person or organization that is the registered owner of this copy of Windows.
Windows_Directory	Common: StringObject AttributeType	0..1	The Windows_Directory element specifies the fully-qualified path to the Windows install directory.
Windows_System_ Directory	Common: StringObject AttributeType	0..1	The Windows_System_Directory element specifies the fully-qualified path to the Windows system directory.
Windows_Temp_ Directory	Common: StringObject AttributeType	0..1	The Windows_Temp_Directory element specifies the fully-qualified path to the Windows temporary files directory.

3.2.61.1 GlobalFlagListType

The GlobalFlagListType type is a listing of all Windows global flags.

Property	Type	Mult	Description
Global_Flag	WinSystemObj:GlobalFlagType	1..∞	This characterizes Windows global flags. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff549557(v=vs.85).aspx

3.2.61.2 GlobalFlagType

The GlobalFlagType type is intended to characterize Windows global flags.

Property	Type	Mult	Description
Abbreviation	Common: StringObject AttributeType	0..1	The abbreviation of a global flag. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff549646(v=vs.85).aspx
Destination	Common: StringObject AttributeType	0..1	The destination of a global flag. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff549646(v=vs.85).aspx
Hexadecimal_Value	Common: HexBinary ObjectAttributeType	0..1	The hexadecimal value of a global flag. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff549646(v=vs.85).aspx
Symbolic_Name	Common: StringObject AttributeType	0..1	The symbolic name of a global flag. See also: http://msdn.microsoft.com/en-us/library/windows/hardware/ff549646(v=vs.85).aspx

3.2.62 WindowsSystemRestoreObjectType (extends Common:DefinedObjectType)

The WindowsSystemRestoreObjectType is intended to characterize Windows system restore points.

Property	Type	Mult	Description
Restore_Point_Description	Common: StringObject AttributeType	0..1	The description of this restore point
Restore_Point_Full_Path	Common: StringObject AttributeType	0..1	The full path to the restore point
Restore_Point_Name	Common: StringObject AttributeType	0..1	The name associated with this restore point.
Restore_Point_Type	Common: StringObject AttributeType	0..1	The type of restore point. (ex: "Checkpoint")
ACL_Change_SID	Common: StringObject AttributeType	0..1	The SID associated with a restore point change log event. This usually appears when the event flag includes "ACL Info".
ACL_Change_Username	Common: StringObject AttributeType	0..1	The username associated with a restore point change log event. It usually appears when the event flag includes "ACL Info"
Backup_File_Name	Common: StringObject AttributeType	0..1	The backup file name associated with a particular restore point change log event
Change_Event	WinSystemRestoreObj: ChangeLogEntryTypeType	0..1	The change event associated with this restore point object (ex: "System Checkpoint", "Software Installation", etc.)
ChangeLog_Entry_Flags	Common: StringObject AttributeType	0..1	The flags associated with a restore point change log entry (ex: "ACL Info, "Short Name", etc.)
ChangeLog_Entry_Sequence_Number	Common: LongObject AttributeType	0..1	The change log sequence number associated with this restore point object
ChangeLog_Entry_Type	WinSystemRestoreObj: ChangeLogEntryTypeType	0..1	The changelog entry type associated with this restore point object.
Change_Log_File_Name	Common: StringObject AttributeType	0..1	The changelog file associated with the restore point
Created	Common: DateTimeObject AttributeType	0..1	The created date of the system restore point.
File_Attributes	Common: StringObject AttributeType	0..1	Attributes of the file associated with this restore point object (ex: "Directory")
New_File_Name	Common: StringObject AttributeType	0..1	The new filename of the file associated with this restore point object.
Original_File_Name	Common: StringObject AttributeType	0..1	The original filename associated with this restore point change log event
Original_Short_File_Name	Common: StringObject AttributeType	0..1	The original Short filename (SFN) of the file associated with this restore point object
Process_Name	Common: StringObject AttributeType	0..1	The process name associated with this restore point object.

Registry_Hive_List	WinSystemRestoreObj: HiveListType	0..1	The registry hives associated with this restore point

3.2.62.1 HiveListType

Property	Type	Mult	Description
Hive	Common: StringObject AttributeType	1..∞	

3.2.62.2 ChangeLogEntryTypeType (restriction Common:BaseObjectAttributeType)

ChangeLogEntryTypeType types, via a union of the ChangeLogEntryTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinSystemRestoreObj:ChangeLogEntryTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.62.3 ChangeLogEntryTypeEnum

The change types found in a Restore Point changelog>

Restriction base: string

Enumeration Value	Description
UPDATE_ACL	Represents a changelog entry descriptor for updating an ACL. (0x00000001)
UPDATE_ATTRIBUTES	Represents a changelog entry descriptor for updating attributes. (0x00000002)
DELETE_FILE	Represents a changelog entry descriptor for deleting a file. (0x00000004)
CREATE_FILE	Represents a changelog entry descriptor for creating a file. (0x00000010)
RENAME_FILE	Represents a changelog entry descriptor for renaming a file. (0x00000020)
CREATE_DIRECTORY	Represents a changelog entry descriptor for creating a directory. (0x00000040)
RENAME_DIRECTORY	Represents a changelog entry descriptor for renaming a directory. (0x00000080)
DELETE_DIRECTORY	Represents a changelog entry descriptor for deleting a directory. (0x00000100)
MNT_CREATE	Related to filesystem attachment points. (0x00000200)

3.2.63 WindowsTaskObjectType (extends Common:DefinedObjectType)

The WindowsTaskObjectType type is intended to characterize Windows task scheduler tasks. See Also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381216(v=vs.85).aspx

Property	Type	Mult	Description
Status	WinTaskObj: TaskStatusType	0..1	The Status element specifies the current status of the scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381263(v=vs.85).aspx
Priority	WinTaskObj: TaskPriorityType	0..1	The Priority element specifies the priority of the scheduled task. This can either be a free-form string or one the values in the TaskPriorityEnum enumeration. See also: http://msdn.microsoft.com/en-

			us/library/windows/desktop/aa381876(v=vs.85).aspx
Name	Common: StringObject AttributeType	0..1	The Name element specifies the image name for the task.
Application_Name	Common: StringObject AttributeType	0..1	The Application_Name specifies the application name associated with the task.
Parameters	Common: StringObject AttributeType	0..1	The Parameters element specifies the command line parameters used to launch the scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381875(v=vs.85).aspx
Flags	WinTaskObj: TaskFlagType	0..1	The Flags element specifies any flags that modify the behavior of the scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381248(v=vs.85).aspx
Account_Name	Common: StringObject AttributeType	0..1	The Account_Name element specifies the name of the account used to run the scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381228(v=vs.85).aspx
Account_Run_Level	Common: StringObject AttributeType	0..1	The Account_Run_Level element specifies the permission level of the account that the task will be run at.
Account_Logon_Type	Common: StringObject AttributeType	0..1	The Account_Logon_Type element specifies the security logon method required to run the tasks associated with the account. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa383013(v=vs.85).aspx
Creator	Common: StringObject AttributeType	0..1	The Creator element specifies the name of the creator of the scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381235(v=vs.85).aspx
Creation_Date	Common: DateTimeObject AttributeType	0..1	The Creation_Date element specifies the date and time that the task was registered. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa382623(v=vs.85).aspx
Most_Recent_Run_Time	Common: DateTimeObject AttributeType	0..1	The Most_Recent_Run_Time element specifies the most recent run date/time of this scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381254(v=vs.85).aspx
Exit_Code	Common: LongObject AttributeType	0..1	The Exit_Code element specifies the last exit code of the scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381245(v=vs.85).aspx
Max_Run_Time	Common:	0..1	The Max_Run_Time element specifies the

	UnsignedLong ObjectAttributeType		maximum run time of the scheduled task before terminating, in milliseconds. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381874(v=vs.85).aspx
Next_Run_Time	Common: DateTimeObject AttributeType	0..1	The Next_Run_Time element specifies the next run date/time of the scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381257(v=vs.85).aspx
Action_List	WinTaskObj: TaskActionListType	0..1	The Action_List element specifies a list of actions to be performed by the scheduled task.
Trigger_List	WinTaskObj: TriggerListType	0..1	The Trigger_List element specifies a set of triggers used by the scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa383264(v=vs.85).aspx
Comment	Common: StringObject AttributeType	0..1	The Comment element specifies a comment for the scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381232(v=vs.85).aspx
Working_Directory	Common: StringObject AttributeType	0..1	The Working_Directory element specifies the working directory for the scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381878(v=vs.85).aspx
Work_Item_Data	Common: Base64Binary ObjectAttributeType	0..1	The Work_Item_Data element specifies application defined data associated with the scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381271(v=vs.85).aspx

3.2.63.1 TriggerListType

The TriggerListType type specifies a set of triggers associated with the scheduled task.

Property	Type	Mult	Description
Trigger	WinTaskObj:TriggerType	1..∞	A trigger associated with this scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381264(v=vs.85).aspx

3.2.63.2 TriggerType

The TriggerType type characterizes task triggers. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa383868(v=vs.85).aspx

Property	Type	Mult	Description
enabled	boolean	1..1	The enabled attribute specifies whether the trigger is enabled.
Trigger_Begin	Common: DateTime ObjectAttributeType	0..1	The Trigger_Begin_Element specifies the date/time that the trigger is activated.
Trigger_Delay	Common:	0..1	The Trigger_Delay element specifies the delay that

	DurationObjectAttributeType		takes place between when the task is registered and when the task is started.
Trigger_End	Common:DateTimeObjectAttributeType	0..1	The Trigger_End element specifies the date/time that the trigger is deactivated.
Trigger_Frequency	WinTaskObj:TaskTriggerFrequencyType	0..1	The Trigger_Frequency element specifies the frequency at which the trigger repeats.
Trigger_Max_Run_Time	Common:DurationObjectAttributeType	0..1	The maximum amount of time that the task launched by the trigger is allowed to run. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa383868(v=vs.85).aspx
Trigger_Session_Change_Type	Common:StringObjectAttributeType	0..1	The Trigger_Session_Change_Type element specifies the type of Terminal Server session change that would trigger a task launch. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381298(v=vs.85).aspx
Trigger_Type	WinTaskObj:TriggerType	0..1	The Trigger_Type specifies the type of the task trigger.

3.2.63.3 TaskActionListType

The TaskActionListType type specifies a list of task actions.

Property	Type	Mult	Description
Action	WinTaskObj:TaskActionType	1..∞	The work items performed by a task are called actions. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa383549(v=vs.85).aspx

3.2.63.4 TaskActionType

The TaskActionType type characterizes scheduled task actions.

Property	Type	Mult	Description
Action_Type	WinTaskObj:ActionType	0..1	The Action_Type element specifies the type of the action. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa380596(v=vs.85).aspx
Action_ID	Common:StringObjectAttributeType	0..1	The Action_ID element specifies the user-defined identifier for the action. This identifier is used by the Task Scheduler for logging purposes. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa380590(v=vs.85).aspx
IEmailAction	EmailMessageObj:EmailMessageObjectType	0..1	The IEmail_Action element specifies an action that sends an e-mail, which in this context refers to actual email message sent. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa380693(v=vs.85).aspx
IComHandlerAction	WinTaskObj:IComHandlerActionType	0..1	The IComHandlerAction element specifies an action that fires a handler.
IExecAction	WinTaskObj:IExecActionType	0..1	The IExecAction element specifies an action that executes a command-line operation. See also:

			http://msdn.microsoft.com/en-us/library/windows/desktop/aa380715(v=vs.85).aspx
IShowMessageAction	WinTaskObj:IShowMessageActionType	0..1	The IShowMessageAction element specifies an action that shows a message box when a task is activated. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381302(v=vs.85).aspx

3.2.63.5 ActionType (restriction Common:BaseObjectAttributeType)

The action type characterizes the specific types of task actions.

Data restrictions: WinTaskObj:TaskActionTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.63.6 IComHandlerActionType

The IComHandlerActionType type characterizes IComHandler actions.

Property	Type	Mult	Description
COM_Data	Common:StringObjectAttributeType	0..1	The COM_Data element specifies the data associated with the COM handler. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa380613(v=vs.85).aspx
COM_Class_ID	Common:StringObjectAttributeType	0..1	The COM_Class_ID element specifies the ID of the COM action. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa380613(v=vs.85).aspx

3.2.63.7 IExecActionType

The IExecActionType type characterizes IExec actions.

Property	Type	Mult	Description
Exec_Arguments	Common:StringObjectAttributeType	0..1	The Exec_Arguments element specifies the arguments associated with the command-line operation launched by the action. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa380715(v=vs.85).aspx
Exec_Program_Path	Common:StringObjectAttributeType	0..1	The Exec_Program_Path element specifies the path to the executable file launched by the action. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa380715(v=vs.85).aspx
Exec_Working_Directory	Common:StringObjectAttributeType	0..1	The Exec_Working_Directory element specifies the directory that contains either the executable file or the files that are used by the executable file launched by the action. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa380715(v=vs.85).aspx
Exec_Program_Hashes	Common:HashListType	0..1	The Exec_Program_Element specifies the hashes of the executable file launched by the action.

3.2.63.8 IShowMessageActionType

The IShowMessageActionType type characterizes IShowMessage actions.

Property	Type	Mult	Description
Show_Message_Body	Common: StringObject AttributeType	0..1	The Show_Message_Body element specifies the message text that is displayed in the body of the message box by the action. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381302(v=vs.85).aspx
Show_Message_Title	Common: StringObject AttributeType	0..1	The Show_Message_Title element specifies the title of the message box shown by the action. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381302(v=vs.85).aspx

3.2.63.9 TaskFlagType (restriction Common:BaseObjectAttributeType)

The TaskFlagType type specifies Windows Task flag types via a union of the TaskFlagEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinTaskObj:TaskFlagEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.63.10 TaskPriorityType (restriction Common:BaseObjectAttributeType)

The TaskPriorityType type specifies Windows Task priority types via a union of the TaskPriorityEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinTaskObj:TaskPriorityEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.63.11 TaskTriggerFrequencyType (restriction Common:BaseObjectAttributeType)

The TaskTriggerFrequencyType type specifies Windows Task trigger frequency types via a union of the TriggerFrequencyEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinTaskObj:TriggerFrequencyEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.63.12 TaskTriggerType (restriction Common:BaseObjectAttributeType)

The TaskTriggerType type specifies Windows Task trigger types via a union of the TriggerTypeEnum enumeration and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinTaskObj:TriggerTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.63.13 TaskStatusType (restriction Common:BaseObjectAttributeType)

The TaskStatusType type specifies Windows Task states via a union of the TaskStatusEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinTaskObj:TaskStatusEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.63.14 TaskActionTypeEnum

An enumeration of action types. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa380596(v=vs.85).aspx

Restriction base: string

Enumeration Value	Description
TASK_ACTION_EXEC	This action performs a command-line operation. For example, the action could run a script, launch an executable, or, if the name of a document is provided, find its associated application and launch the application with the document.
TASK_ACTION_COM_HANDLER	This action fires a handler.
TASK_ACTION_SEND_EMAIL	This action sends an e-mail.
TASK_ACTION_SHOW_MESSAGE	This action shows a message box.

3.2.63.15 TaskFlagEnum

The TaskFlagEnum enumeration specifies the run flags for a task scheduler task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381283(v=vs.85).aspx See Also: http://msdn.microsoft.com/en-us/library/microsoft.office.excel.server.addins.computecluster.taskscheduler.taskflags(v=office.12).aspx

Enumeration Value	Description
TASK_FLAG_ZERO	
TASK_FLAG_INTERACTIVE	This flag is used when converting Windows NT AT service jobs into work items. The Windows NT AT service job refers to At.exe, the Windows NT command-line utility used for creating jobs for the Windows NT Schedule service. The Task Scheduler service replaces the Schedule service and is backwards compatible with it. The conversion occurs when the Task Scheduler is installed on Windows NT/Windows 2000— for example, if you install Internet Explorer 4.0, or upgrade to Windows 2000. During the setup process, the Task Scheduler installation code searches the registry for jobs created for the AT service and creates work items that will accomplish the same operation. For such converted jobs, the interactive flag is set if the work item is intended to be displayed to the user.

	When this flag is not set, no work items are displayed in the Tasks folder, and no user interface associated with the work item is presented to the user when the work item is executed.
TASK_FLAG_DELETE_WHEN_DONE	The work item will be deleted when there are no more scheduled run times.
TASK_FLAG_DISABLED	The work item is disabled. This is useful to temporarily prevent a work item from running at the scheduled time(s).
TASK_FLAG_HIDDEN	The work item created will be hidden.
TASK_FLAG_RUN_ONLY_IF_LOGGED_ON	The work item runs only if the user specified in IScheduledWorkItem::SetAccountInformation is logged on interactively. This flag has no effect on the work items that are set to run in the local account.
TASK_FLAG_START_ONLY_IF_IDLE	The work item begins only if the computer is not in use at the scheduled start time.
TASK_FLAG_SYSTEM_REQUIRED	The work item causes the system to be resumed, or awakened, if the system is running on battery power. This flag is supported only on systems that support resume timers.
TASK_FLAG_KILL_ON_IDLE_END	The work item terminates if the computer makes an idle to non-idle transition while the work item is running. The computer is not considered idle until the IdleWait triggers' time elapses with no user input. For information regarding idle triggers, see Idle Trigger.
TASK_FLAG_RESTART_ON_IDLE_RESUME	The work item starts again if the computer makes a non-idle to idle transition before all the work item's task_triggers elapse. (Use this flag in conjunction with TASK_FLAG_KILL_ON_IDLE_END.)
TASK_FLAG_DONT_START_IF_ON_BATTERIES	The work item does not start if its target computer is running on battery power.
TASK_FLAG_KILL_IF_GOING_ON_BATTERIES	The work item ends, and the associated application quits if the work item's target computer switches to battery power.
TASK_FLAG_RUN_IF_CONNECTED_TO_INTERNET	The work item runs only if there is currently a valid Internet connection.

3.2.63.16 TaskPriorityEnum

The TaskPriorityEnum enumeration specifies the priority levels of task scheduler tasks. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa383512(v=vs.85).aspx

Restriction base: string

Enumeration Value	Description
HIGH_PRIORITY_CLASS	A priority class of high (1)
NORMAL_PRIORITY_CLASS	A priority class of normal (4-6)
IDLE_PRIORITY_CLASS	A priority class of idle (9-10)
REALTIME_PRIORITY_CLASS	A priority class of realtime (0)
ABOVE_NORMAL_PRIORITY_CLASS	A priority class of above normal (2-3)
BELOW_NORMAL_PRIORITY_CLASS	A priority class of below normal (7-8)

3.2.63.17 TriggerFrequencyEnum

The TriggerFrequencyEnum enumeration defines the frequency types that a trigger may use. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa383620(v=vs.85).aspx and http://msdn.microsoft.com/en-us/library/windows/desktop/aa383987(v=vs.85).aspx

Restriction base: string

Enumeration Value	Description
TASK_TIME_TRIGGER_ONCE	Trigger is set to run the task a single time.
TASK_EVENT_TRIGGER_ON_IDLE	Trigger is set to run the task if the system remains idle for the amount of time specified by the idle wait time of the task.
TASK_EVENT_TRIGGER_AT_SYSTEMSTART	Trigger is set to run the task at system startup.
TASK_EVENT_TRIGGER_AT_LOGON	Trigger is set to run the task when a user logs on.
TASK_TIME_TRIGGER_DAILY	Trigger is set to run the task on a daily interval.
TASK_TIME_TRIGGER_WEEKLY	Trigger is set to run the work item on specific days of a specific week of a specific month.
TASK_TIME_TRIGGER_MONTHLYDATE	Trigger is set to run the task on a specific day(s) of the month.
TASK_TIME_TRIGGER_MONTHLYDOW	Trigger is set to run the task on specific days, weeks, and months.

3.2.63.18 TriggerTypeEnum

The TriggerFrequencyEnum enumeration defines the types of triggers associated with a task.

Restriction base: string

Enumeration Value	Description
TASK_TRIGGER_EVENT	Triggers the task when a specific system event occurs.
TASK_TRIGGER_TIME	Triggers the task at a specific date and time.
TASK_TRIGGER_IDLE	Triggers the task when the computer enters an idle state.
TASK_TRIGGER_REGISTRATION	Triggers the task when the task is registered or updated.
TASK_TRIGGER_BOOT	Triggers the task when the system is booted.
TASK_TRIGGER_LOGON	Triggers the task when a user logs on.
TASK_TRIGGER_SESSION_STATE_CHANGE	Triggers the task when a Terminal Server session changes state.

3.2.63.19 TaskStatusEnum

The TaskStatusEnum enumeration specifies the possible statuses of a scheduled task. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa383604(v=vs.85).aspx See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381263(v=vs.85).aspx See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381833(v=vs.85).aspx See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa383617(v=vs.85).aspx

Restriction base: string

Enumeration Value	Description
SCHED_S_TASK_READY	The task is ready to run at its next scheduled time.
SCHED_S_TASK_RUNNING	The task is currently running.
SCHED_S_TASK_NOT_SCHEDULED	One or more of the properties that are needed to run this task on a schedule have not been set.
SCHED_E_SERVICE_NOT_RUNNING	The Task Scheduler service is not running.
SCHED_E_UNSUPPORTED_ACCOUNT_OPTION	The task has been configured with an unsupported combination of account settings and run time options.
SCHED_E_UNKNOWN_OBJECT_VERSION	The task object version is either unsupported or invalid.
SCHED_E_NO_SECURITY_SERVICES	Task Scheduler security services are available only on Windows NT.
SCHED_E_ACCOUNT_DBASE_CORRUPT	Corruption was detected in the Task Scheduler security database; the database has been reset.
SCHED_E_ACCOUNT_NAME_NOT_FOUND	Unable to establish existence of the account specified.
SCHED_E_ACCOUNT_INFORMATION_NOT_SET	No account information could be found in the Task Scheduler

	security database for the task indicated.
SCHED_E_INVALID_TASK	The object is either an invalid task object or is not a task object.
SCHED_E_CANNOT_OPEN_TASK	The task object could not be opened.
SCHED_E_SERVICE_NOT_INSTALLED	The Task Scheduler service is not installed on this computer.
SCHED_E_TASK_NOT_RUNNING	There is no running instance of the task.
SCHED_E_TASK_NOT_READY	One or more of the properties required to run this task have not been set.
SCHED_E_TRIGGER_NOT_FOUND	A task's trigger is not found.
SCHED_S_EVENT_TRIGGER	Event triggers do not have set run times.
SCHED_S_TASK_NO_VALID_TRIGGERS	Either the task has no triggers or the existing triggers are disabled or not set.
SCHED_S_TASK_TERMINATED	The last run of the task was terminated by the user.
SCHED_S_TASK_NO_MORE_RUNS	There are no more runs scheduled for this task.
SCHED_S_TASK_HAS_NOT_RUN	The task has not been run. This value is returned whenever the task has not been run, even if the task is ready to be run at the next scheduled time or the task is a recurring task.
SCHED_S_TASK_DISABLED	The task will not run at the scheduled times because it has been disabled.
TASK_STATE_UNKNOWN	The state of the task is unknown.
TASK_STATE_QUEUED	Instances of the task are queued.

3.2.64 WindowsThreadObjectType (extends Common:DefinedObjectType)

The Windows_ThreadObjectType is intended to characterize Windows process threads. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms684852(v=vs.85).aspx

Property	Type	Mult	Description
Thread_ID	Common: NonNegative IntegerObject AttributeType	0..1	Represents the identifier of this thread. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms683183(v=vs.85).aspx
Handle	WinHandleObj: WindowsHandleObjectType	0..1	Handle represents the handle of a specific thread. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms682453(v=vs.85).aspx
Running_Status	WinThreadObj: ThreadRunningStatusType	0..1	Running Status represents the running state that the thread is in.
Context	Common: StringObject AttributeType	0..1	The Context element specifies the thread context structure, which contains processor-specific register data.
Priority	Common: UnsignedInteger ObjectAttributeType	0..1	Represents the priority of the thread. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms685100(v=vs.85).aspx
Creation_Flags	Common: HexBinary ObjectAttributeType	0..1	The Creation flags element represents the creation flags that a thread may be launched with. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms684863(v=vs.85).aspx
Creation_Time	Common: DateTimeObject AttributeType	0..1	Creation time represents the creation time of the thread.
Start_Address	Common: HexBinary ObjectAttributeType	0..1	Start address represents the start address of this thread, representing the memory address where this thread should start. See Also: http://msdn.microsoft.com/en-

			us/library/windows/desktop/ms682453(v=vs.85).aspx
Parameter_Address	Common: HexBinary ObjectAttributeType	0..1	
Security_Attributes	Common: StringObject AttributeType	0..1	Security attributes represents the security attributes for the thread. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/aa379560(v=vs.85).aspx
Stack_Size	Common: NonNegative IntegerObject AttributeType	0..1	Represents the stack size of the thread. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms686774(v=vs.85).aspx

3.2.64.1 ThreadRunningStatusType (restriction Common:BaseObjectAttributeType)

ThreadRunningStatusType specifies Windows thread running states via a union of the ThreadRunningStatusEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinThreadObj:ThreadRunningStatusEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.64.2 ThreadRunningStatusEnum

Thread running status enumerates the various states that a thread may be in before, during, or after execution. See http://msdn.microsoft.com/en-us/library/system.diagnostics.threadstate(v=vs.110).aspx

Restriction base: string

Enumeration Value	Description
Initialized	A state that indicates the thread has been initialized, but has not yet started.
Ready	A state that indicates the thread is waiting to use a processor because no processor is free. The thread is preapared to run on the next available processor.
Running	A state that indicates the thread is currently using a prcoessor.
Standby	A state that indicates the thread is about to use a processor. Only one thread can be in this state at a time.
Terminated	A state that indicates the thread has finished executing and has exited.
Waiting	A state that indicates the thread is not ready to use the processor because it is waiting for a peripheral operation to complete or a resource to become free. When the thread is ready, it will be rescheduled.
Transition	A state that indicates the thread is waiting for a resource, other than the processor, before it can execute.
Unknown	The thread of the thread is unknown.

3.2.65 WindowsUserAccountObjectType (extends UserAccountObj:UserAccountObjectType)

The WinUserAccountObjectType type is intended to characterize Windows user accounts.

Property	Type	Mult	Description
Security_ID	Common: StringObject AttributeType	0..1	Security ID represents the Security ID (SID) of a windows user.
Security_Type	Common:SIDType	0..1	Security Type represents the type of the Security ID

			(SID).

3.2.65.1 WindowsGroupType (extends UserAccountObj:GroupType)

Windows Group represents a single windows group.

Property	Type	Mult	Description
Name	Common: StringObject AttributeType	1..1	Identifies the name of the windows group.

3.2.65.2 WindowsPrivilegeType (extends UserAccountObj:PrivilegeType)

Windows Privilege represents a single privilege that a user may have within Windows.

Property	Type	Mult	Description
User_Right	Common: StringObject AttributeType	1..1	User Right represents one right that a user may have.

3.2.66 WindowsVolumeObjectType (extends VolumeObj:VolumeObjectType)

The WindowsVolumeObjectType type is intended to characterize Windows disk volumes.

Property	Type	Mult	Description
Attributes_List	WinVolumeObj: WindowsVolume AttributesListType	0..1	Represents the attributes of this windows volume object.
Drive_Letter	Common: StringObject AttributeType	0..1	Represents the drive letter of this windows volume object.
Drive_Type	WinVolumeObj: WindowsDriveType	0..1	Represents the drive type of this windows volume object.

3.2.66.1 WindowsVolumeAttributesListType

A list of attributes describing this windows volume.

Property	Type	Mult	Description
Attribute	WinVolumeObj: WindowsVolumeAttributeType	1..4	Each attribute element represents a single attribute in the windows volume attribute list.

3.2.66.2 WindowsDriveType (restriction Common:BaseObjectAttributeType)

WindowsDriveType specifies Windows drive types via a union of the WindowsDriveTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinVolumeObj:WindowsDriveTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.66.3 WindowsVolumeAttributeType (restriction Common:BaseObjectAttributeType)

WindowsVolumeAttributeType specifies Windows volume attributes via a union of the WindowsVolumeAttributeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinVolumeObj:WindowsVolumeAttributeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.66.4 WindowsDriveTypeEnum

This enumeration contains possible drive types, as enumerated by the WINAPI GetDriveType function: http://msdn.microsoft.com/en-us/library/windows/desktop/aa364939(v=vs.85).aspx

Restriction base: string

Enumeration Value	Description
DRIVE_UNKNOWN	The drive type cannot be determined.
DRIVE_NO_ROOT_DIR	The root path is invalid; for example, there is no volume mounted at the specified path.
DRIVE_REMOVABLE	The drive has removable media; for example, a floppy drive, thumb drive, or flash card reader.
DRIVE_FIXED	The drive has fixed media; for example, a hard disk drive or flash drive.
DRIVE_REMOTE	The drive is a remote (network) drive.
DRIVE_CDROM	The drive is a CD-ROM drive.
DRIVE_RAMDISK	The drive is a RAM disk.

3.2.66.5 WindowsVolumeAttributeEnum

This enumeration is a list of attributes that may be returned by the diskpart attributes command: http://technet.microsoft.com/en-us/library/cc766465(v=ws.10).aspx

Restriction base: string

Enumeration Value	Description
ReadOnly	Specifies that the volume is read-only.
Hidden	Specifies that the volume is hidden.
NoDefaultDriveLetter	Specifies that the volume does not receive a drive letter by default.
ShadowCopy	Specifies that the volume is a shadow copy volume.

3.2.67 WindowsWaitableTimerObjectType (extends Common:DefinedObjectType)

The WindowsWaitableTimerObjectType is intended to characterize Windows waitable timer (synchronization) objects.

Property	Type	Mult	Description
type	WinWaitableTimerObj: WaitableTimer TypeEnum	1..1	The type attributes specifies the type of the windows waitable timer object.
Handle	WinHandleObj: WindowsHandle ObjectType	0..1	The Handle element specifies the handle to the Windows waitable timer object. It imports and uses the WindowsHandleObjectType type from the CybOX Windows Handle object.
Name	Common: StringObject	0..1	The Name element specifies the name of the

	AttributeType		Windows waitable timer object.
Security_Attributes	Common: StringObject AttributeType	0..1	The Security_Attributes element specifies the security attributes for the Windows waitable timer object.
Type	WinWaitableTimerObj: WaitableTimerType	0..1	The Type element specifies the type of the windows waitable timer object.

3.2.67.1 WaitableTimerType (restriction Common:BaseObjectAttributeType)

WaitableTimerType specifies Windows waitable timer types via a union of the WaitableTimerTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectAttributeType, for permitting complex (i.e. regular-expression based) specifications.

Data restrictions: WinWaitableTimerObj:WaitableTimerTypeEnum, string

Property	Type	Mult	Description
datatype	Common:DatatypeEnum	1..1	This attribute is optional and specifies the expected type for the value of the specified element.

3.2.67.2 WaitableTimerTypeEnum

The WaitableTimerTypeEnum type is an enumeration of Windows waitable timer types.

Restriction base: string

Enumeration Value	Description
ManualReset	A timer whose state remains signaled until SetWaitableTimer is called to establish a new due time. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms687012(v=vs.85).aspx
Synchronization	A timer whose state remains signaled until a thread completes a wait operation on the timer object. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms687012(v=vs.85).aspx
Periodic	A timer that is reactivated each time the specified period expires, until the timer is reset or canceled. A periodic timer is either a periodic manual-reset timer or a periodic synchronization timer. See also: http://msdn.microsoft.com/en-us/library/windows/desktop/ms687012(v=vs.85).aspx

3.2.68 X509CertificateObjectType (extends Common:DefinedObjectType)

The X509CertificateObjectType type is intended to characterize X.509 certificates.

Property	Type	Mult	Description
Certificate	X509CertificateObj: X509CertificateType	0..1	Certificate represents the contents of an X.509 certificate, including items such as issuer, subject, and others.
Certificate_Signature	X509CertificateObj: X509Certificate SignatureType	0..1	Certificate Signature contains the signature and signature algorithm of this X.509 certificate.

3.2.68.1 X509CertificateType

The X509CertificateType type represents the contents of an X.509 certificate, including items such as issuer, subject, and others.

Property	Type	Mult	Description
Issuer	Common: StringObject	0..1	The issuer is the Certificate Authority who issued

	AttributeType		the X.509 certificate.
Serial_Number	Common: IntegerObject AttributeType	0..1	The serial number is a unique identifier for each X.509 certificate issued by a specific Certificate Authority.
Signature_Algorithm	Common: StringObject AttributeType	0..1	The signature algorithm is the algorithm used to sign the X.509 certificate.
Subject	Common: StringObject AttributeType	0..1	The subject identifies the entity associated with the public key stored in the subject public key field of the X.509 certificate.
Subject_Public_Key	X509CertificateObj: SubjectPublicKeyType	0..1	The Subject Public Key is used to carry the public key and identify the algorithm with which the key is used.
Validity	X509CertificateObj: ValidityType	0..1	Validity is the time interval during which the issuer warrants that it will maintain information about the status of the certificate.
Version	Common: IntegerObject AttributeType	0..1	Version describes the version of the encoded certificate.

3.2.68.2 X509CertificateSignatureType

The X509CertificateSignatureType contains the signature and signature algorithm of this X.509 certificate.

Property	Type	Mult	Description
Signature	Common: StringObject AttributeType	1..1	Signature contains a digital signature computed upon this X.509 certificate.
Signature_Algorithm	Common: StringObject AttributeType	0..1	Signature Algorithm contains the algorithm identifier for the algorithm used by the Certificate Authority to compute the signature.

3.2.68.3 SubjectPublicKeyType

The SubjectPublicKeyType is used to carry the public key and identify the algorithm with which the key is used.

Property	Type	Mult	Description
Public_Key_Algorithm	Common: StringObject AttributeType	0..1	Public Key Algorithm is the algorithm with which to encrypt data being sent to the subject.
RSA_Public_Key	element	0..1	RSA Public Key is the public key contained in this X.509 certificate.
Modulus	Common: HexBinary ObjectAttributeType	1..1	Modulus is the modulus portion of a public key.
Exponent	Common: HexBinary ObjectAttributeType	0..1	Exponent is the exponent portion of a public key.

3.2.68.4 ValidityType

The ValidityType type is the time interval during which the issuer warrants that it will maintain information about the status of the certificate.

Property	Type	Mult	Description
Not_Before	Common: DateTimeObject AttributeType	0..1	Not before is the date on which the certificate validity period begins.
Not_After	Common: DateTimeObject AttributeType	0..1	Not after is the date on which the certificate validity period ends.

4 Language Representations & Example Content

4.1 XML

The XML Representation for the CybOX Language Data Model is documented via a series of XML Schemas.[4] These schemas describe how the information presented in this Specification is formatted and represented as XML Documents. Please refer to the appropriate Schema for more information about a specific XML representation.

CybOX Core Schema

http://cybox.mitre.org/XMLSchema/cybox_core_v1.0(draft).xsd

CybOX Common_Types Schema

http://cybox.mitre.org/XMLSchema/cybox_common_types_v1.0(draft).xsd

Defined Objects

Account Object Schema
http://cybox.mitre.org/XMLSchema/objects/Account/Account_Object_1.1.xsd
Address Object Schema
http://cybox.mitre.org/XMLSchema/objects/Address/Address_Object_1.1.xsd
API Object Schema
http://cybox.mitre.org/XMLSchema/objects/API/API_Object_1.0.xsd
Code Object Schema
http://cybox.mitre.org/XMLSchema/objects/Code/Code_Object_1.0.xsd
Device Object Schema
http://cybox.mitre.org/XMLSchema/objects/Device/Device_Object_1.0.xsd
Disk Object Schema
http://cybox.mitre.org/XMLSchema/objects/Disk/Disk_Object_1.2.xsd
Disk_Partition Object Schema
http://cybox.mitre.org/XMLSchema/objects/Disk_Partition/Disk_Partition_Object_1.2.xsd
DNS_Cache Object Schema
http://cybox.mitre.org/XMLSchema/objects/DNS_Cache/DNS_Cache_Object_1.2.xsd
DNS_Record Object Schema
http://cybox.mitre.org/XMLSchema/objects/DNS_Record/DNS_Record_Object_1.0.xsd
Email_Message Object Schema

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

http://cybox.mitre.org/XMLSchema/objects/Email_Message/Email_Message_Object_1.1.xsd

File Object Schema

http://cybox.mitre.org/XMLSchema/objects/File/File_Object_1.2.xsd

GUI_Dialogbox Object Schema

http://cybox.mitre.org/XMLSchema/objects/GUI_Dialogbox/GUI_Dialogbox_Object_1.1.xsd

GUI Object Schema

http://cybox.mitre.org/XMLSchema/objects/GUI/GUI_Object_1.1.xsd

GUI_Window Object Schema

http://cybox.mitre.org/XMLSchema/objects/GUI_Window/GUI_Window_Object_1.1.xsd

Library Object Schema

http://cybox.mitre.org/XMLSchema/objects/Library/Library_Object_1.2.xsd

Linux_Package Object Schema

http://cybox.mitre.org/XMLSchema/objects/Linux_Package/Linux_Package_Object_1.2.xsd

Memory Object Schema

http://cybox.mitre.org/XMLSchema/objects/Memory/Memory_Object_1.1.xsd

Mutex Object Schema

http://cybox.mitre.org/XMLSchema/objects/Mutex/Mutex_Object_1.2.xsd

Network_Flow Object Schema

http://cybox.mitre.org/XMLSchema/objects/Network_Flow/Network_Flow_Object_1.0.xsd

Network_Packet Object Schema

http://cybox.mitre.org/XMLSchema/objects/Network_Packet/Network_Packet_Object_1.0.xsd

Network_Route_Entry Object Schema

http://cybox.mitre.org/XMLSchema/objects/Network_Route_Entry/Network_Route_Entry_Object_1.0.xsd

Network_Route Object Schema

http://cybox.mitre.org/XMLSchema/objects/Network_Route/Network_Route_Object_1.1.xsd

Network_Subnet Object Schema

http://cybox.mitre.org/XMLSchema/objects/Network_Subnet/Network_Subnet_Object_1.0.xsd

Pipe Object Schema

http://cybox.mitre.org/XMLSchema/objects/Pipe/Pipe_Object_1.2.xsd

Port Object Schema

http://cybox.mitre.org/XMLSchema/objects/Port/Port_Object_1.2.xsd

Process Object Schema

http://cybox.mitre.org/XMLSchema/objects/Process/Process_Object_1.2.xsd

Product Object Schema

http://cybox.mitre.org/XMLSchema/objects/Product/Product_Object_Type_1.1.xsd

Semaphore Object Schema

http://cybox.mitre.org/XMLSchema/objects/Semaphore/Semaphore_Object_1.2.xsd

Socket Object Schema

http://cybox.mitre.org/XMLSchema/objects/Socket/Socket_Object_1.3.xsd

System Object Schema
http://cybox.mitre.org/XMLSchema/objects/System/System_Object_1.2.xsd
Unix_File Object Schema
http://cybox.mitre.org/XMLSchema/objects/Unix_File/Unix_File_Object_1.2.xsd
Unix_Network_Route_Entry Object Schema
http://cybox.mitre.org/XMLSchema/objects/Unix_Network_Route_Entry/Unix_Network_Route_Entry_Object_1.0.xsd
Unix_Pipe Object Schema
http://cybox.mitre.org/XMLSchema/objects/Unix_Pipe/Unix_Pipe_Object_1.1.xsd
Unix_Process Object Schema
http://cybox.mitre.org/XMLSchema/objects/Unix_Process/Unix_Process_Object_1.2.xsd
Unix_User_Account Object Schema
http://cybox.mitre.org/XMLSchema/objects/Unix_User_Account/Unix_User_Account_Object_1.1.xsd
Unix_Volume Object Schema
http://cybox.mitre.org/XMLSchema/objects/Unix_Volume/Unix_Volume_Object_1.1.xsd
URI Object Schema
http://cybox.mitre.org/XMLSchema/objects/URI/URI_Object_1.1.xsd
User_Account Object Schema
http://cybox.mitre.org/XMLSchema/objects/User_Account/User_Account_Object_1.1.xsd
User_Session Object Schema
http://cybox.mitre.org/XMLSchema/objects/User_Session/User_Session_Object_1.1.xsd
Volume Object Schema
http://cybox.mitre.org/XMLSchema/objects/Volume/Volume_Object_1.2.xsd
Win_Computer_Account Object Schema
http://cybox.mitre.org/XMLSchema/objects/Win_Computer_Account/Win_Computer_Account_Object_1.2.xsd
Win_Critical_Section Object Schema
http://cybox.mitre.org/XMLSchema/objects/Win_Critical_Section/Win_Critical_Section_Object_1.1.xsd
Win_Driver Object Schema
http://cybox.mitre.org/XMLSchema/objects/Win_Driver/Win_Driver_Object_1.1.xsd
Win_Event_Log Object Schema
http://cybox.mitre.org/XMLSchema/objects/Win_Event_Log/Win_Event_Log_Object_1.1.xsd
Win_Event Object Schema
http://cybox.mitre.org/XMLSchema/objects/Win_Event/Win_Event_Object_1.2.xsd
Win_Executable_File Object Schema
http://cybox.mitre.org/XMLSchema/objects/Win_Executable_File/Win_Executable_File_Object_1.2.xsd
Win_File Object Schema
http://cybox.mitre.org/XMLSchema/objects/Win_File/Win_File_Object_1.2.xsd
Win_Handle Object Schema
http://cybox.mitre.org/XMLSchema/objects/Win_Handle/Win_Handle_Object_1.2.xsd
Win_Kernel_Hook Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Kernel_Hook/Win_Kernel_Hook_Object_1.2.xsd

Win_Kernel Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Kernel/Win_Kernel_Object_1.1.xsd

Win_Mailslot Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Mailslot/Win_Mailslot_Object_1.1.xsd

Win_Mutex Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Mutex/Win_Mutex_Object_1.1.xsd

Win_Network_Route_Entry Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Network_Route_Entry/Win_Network_Route_Entry_Object_1.2.xsd

Win_Network_Share Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Network_Share/Win_Network_Share_Object_1.2.xsd

Win_Pipe Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Pipe/Win_Pipe_Object_1.1.xsd

Win_Prefetch Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Prefetch/Win_Prefetch_Object_1.1.xsd

Win_Process Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Process/Win_Process_Object_1.2.xsd

Win_Registry_Key Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Registry_Key/Win_Registry_Key_Object_1.2.xsd

Win_Semaphore Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Semaphore/Win_Semaphore_Object_1.1.xsd

Win_Service Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Service/Win_Service_Object_1.2.xsd

Win_System Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_System/Win_System_Object_1.1.xsd

Win_System_Restore Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_System_Restore/Win_System_Restore_Object_1.1.xsd

Win_Task Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Task/Win_Task_Object_1.2.xsd

Win_Thread Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Thread/Win_Thread_Object_1.2.xsd

Win_User_Account v

http://cybox.mitre.org/XMLSchema/objects/Win_User_Account/Win_User_Account_Object_1.2.xsd

Win_Volume Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Volume/Win_Volume_Object_1.2.xsd

Win_Waitable_Timer Object Schema

http://cybox.mitre.org/XMLSchema/objects/Win_Waitable_Timer/Win_Waitable_Timer_Object_1.2.xsd

X509_Ce*rtificate* ***Object Schema***

http://cybox.mitre.org/XMLSchema/objects/X509_Certificate/X509_Certificate_Object_1.1.xsd

4.2 Validation Requirements

All XML content written against the XML Schema implementation of the CybOX Language MUST be XML Schema valid as defined in the XML Schemas associated with the XML Schema implementation of the CybOX Language.

4.3 Example Content

4.3.1 Simple Examples

4.3.1.1 Single URL

```
<?xml version="1.0" encoding="UTF-8"?>
<cybox:Observables
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cybox="http://cybox.mitre.org/cybox_v1"
        xmlns:common="http://cybox.mitre.org/Common_v1"
        xmlns:URIObj="http://cybox.mitre.org/objects#URIObject"
        xsi:schemaLocation="http://cybox.mitre.org/cybox_v1
                        http://cybox.mitre.org/XMLSchema/cybox_core_v1.0(draft).xsd
                        http://cybox.mitre.org/objects#URIObject
                        http://cybox.mitre.org/XMLSchema/objects/URI/URI_Object_1.1.xsd"
        cybox_major_version="1" cybox_minor_version="0(draft)">
        <cybox:Observable>
                <!-- Observable for a single URL -->
                <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:A1" type="URI">
                                <cybox:Defined_Object xsi:type="URIObj:URIObjectType">
                                        <URIObj:Value condition="Equals"
datatype="AnyURI">www.sample1.com/index.html</URIObj:Value>
                                </cybox:Defined_Object>
                        </cybox:Object>
                </cybox:Stateful_Measure>
        </cybox:Observable>
</cybox:Observables>
```

4.3.1.2 Observable pattern for a file with one of a set of three MD5 hashes

```
<?xml version="1.0" encoding="UTF-8"?>
<cybox:Observables
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cybox="http://cybox.mitre.org/cybox_v1"
        xmlns:common="http://cybox.mitre.org/Common_v1"
        xmlns:FileObj="http://cybox.mitre.org/objects#FileObject"
        xsi:schemaLocation="http://cybox.mitre.org/XMLSchema/cybox_v1
                        http://cybox.mitre.org/XMLSchema/cybox_core_v1.0(draft).xsd
                        http://cybox.mitre.org/objects#FileObject
                        http://cybox.mitre.org/XMLSchema/objects/File/File_Object_1.2.xsd"
        cybox_major_version="1" cybox_minor_version="0(draft)">
        <cybox:Observable>
```

```
                <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:A1" type="File">
                                <cybox:Defined_Object xsi:type="FileObj:FileObjectType">
                                        <FileObj:Hashes>
                                                <common:Hash>
                                                        <common:Type
datatype="String">MD5</common:Type>
                                                        <common:Simple_Hash_Value
condition="IsInSet" value_set="4EC0027BEF4D7E1786A04D021FA8A67F,
21F0027ACF4D9017861B1D021FA8CF76,2B4D027BEF4D7E1786A04D021FA8CC01"
datatype="hexBinary"></common:Simple_Hash_Value>
                                                </common:Hash>
                                        </FileObj:Hashes>
                                </cybox:Defined_Object>
                        </cybox:Object>
                </cybox:Stateful_Measure>
        </cybox:Observable>
</cybox:Observables>
```

4.3.1.3 File with basic information including multiple hashes

```
<?xml version="1.0" encoding="UTF-8"?>
<cybox:Observables
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cybox="http://cybox.mitre.org/cybox_v1"
        xmlns:common="http://cybox.mitre.org/Common_v1"
        xmlns:FileObj="http://cybox.mitre.org/objects#FileObject"
        xsi:schemaLocation="http://cybox.mitre.org/XMLSchema/cybox_v1

        http://cybox.mitre.org/XMLSchema/cybox_core_v1.0(draft).xsd
                                                        http://cybox.mitre.org/objects#FileObject

        http://cybox.mitre.org/XMLSchema/objects/File/File_Object_1.2.xsd"
        cybox_major_version="1" cybox_minor_version="0(draft)">
        <cybox:Observable>
                <!-- Observable for a file with name, path, MD5 hash, SHA1 hash, SHA256 hash and size
in bytes utilizing the base File_Object-->
                <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:A1" type="File">
                                <cybox:Defined_Object xsi:type="FileObj:FileObjectType">
                                        <FileObj:File_Name
datatype="String">notepad.exe</FileObj:File_Name>
                                        <FileObj:File_Path
datatype="String">C:\Temp</FileObj:File_Path>
                                        <FileObj:Size_In_Bytes
datatype="UnsignedLong">273845</FileObj:Size_In_Bytes>
                                        <FileObj:Hashes>
                                                <common:Hash>
                                                        <common:Type
datatype="String">MD5</common:Type>
                                                        <common:Simple_Hash_Value
condition="Equals"
datatype="hexBinary">59a7078444ee3c862e4c08b601ed7e01</common:Simple_Hash_Value>
                                                </common:Hash>
                                                <common:Hash>
```

```
                                                <common:Type
datatype="String">SHA1</common:Type>
                                                <common:Simple_Hash_Value
condition="Equals"
datatype="hexBinary">98e969b49ff2aedf66b94eb82c54b916f1a634cd</common:Simple_Hash_Value>
                                        </common:Hash>
                                        <common:Hash>
                                                <common:Type
datatype="String">SHA256</common:Type>
                                                <common:Simple_Hash_Value
condition="Equals"
datatype="hexBinary">1706c7cd14a5c9bbf674b21f9c4f873ac04b7a6f1f2202cd0c5977c48968d188</com
mon:Simple_Hash_Value>
                                        </common:Hash>
                                </FileObj:Hashes>
                        </cybox:Defined_Object>
                </cybox:Object>
        </cybox:Stateful_Measure>
</cybox:Observable>
</cybox:Observables>
```

4.3.1.4 Create File Action

```
<?xml version="1.0" encoding="UTF-8"?>
<cybox:Observables
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cybox="http://cybox.mitre.org/cybox_v1"
        xmlns:common="http://cybox.mitre.org/Common_v1"
        xmlns:FileObj="http://cybox.mitre.org/objects#FileObject"
        xsi:schemaLocation="http://cybox.mitre.org/XMLSchema/cybox_v1

        http://cybox.mitre.org/XMLSchema/cybox_core_v1.0(draft).xsd
                                                        http://cybox.mitre.org/objects#FileObject

        http://cybox.mitre.org/XMLSchema/objects/File/File_Object_1.2.xsd"
cybox_major_version="1" cybox_minor_version="0(draft)">
        <cybox:Observable>
                <cybox:Event>
                        <cybox:Actions>
                                <cybox:Action id="cybox:Action_1" type="Create"
action_status="Success" context="Host" timestamp="09:22:00.0Z">
                                        <cybox:Action_Name>
                                                <cybox:Defined_Name>Create
File</cybox:Defined_Name>
                                        </cybox:Action_Name>
                                        <cybox:Associated_Objects>
                                                <cybox:Associated_Object id="cybox:Object_1"
type="File" object_state="Exists" association_type="Affected">
                                                        <cybox:Defined_Object
xsi:type="FileObj:FileObjectType">

        <FileObj:File_Name>foobar.dll</FileObj:File_Name>

        <FileObj:File_Path>C:\Windows\system32</FileObj:File_Path>
                                                                <FileObj:Hashes>
                                                                        <common:Hash>
```

```
                                                                    <common:Type
datatype="String">MD5</common:Type>

            <common:Simple_Hash_Value
datatype="hexBinary">6E48C348D742A931EC2CE90ABD7DAC6A</common:Simple_Hash_Value>
                                                                        </common:Hash>
                                                                    </FileObj:Hashes>
                                                                </cybox:Defined_Object>
                                                            </cybox:Associated_Object>
                                                        </cybox:Associated_Objects>
                                                    </cybox:Action>
                                                </cybox:Actions>
                                            </cybox:Event>
                                        </cybox:Observable>
</cybox:Observables>
```

4.3.1.5 Simple Email

```
<?xml version="1.0" encoding="UTF-8"?>
<cybox:Observables xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:cybox="http://cybox.mitre.org/cybox_v1"
            xmlns:common="http://cybox.mitre.org/Common_v1"
            xmlns:AddrObj="http://cybox.mitre.org/objects#AddressObject"
            xmlns:URIObj="http://cybox.mitre.org/objects#URIObject"
            xmlns:FileObj="http://cybox.mitre.org/objects#FileObject"
            xmlns:EmailMessageObj="http://cybox.mitre.org/XMLSchema/objects#EmailMessageObject"
            xsi:schemaLocation="http://cybox.mitre.org/Common_v1

            http://cybox.mitre.org/XMLSchema/cybox_core_v1.0(draft).xsd
                                                                    http://cybox.mitre.org/objects#FileObject

            http://cybox.mitre.org/XMLSchema/objects/File/File_Object_1.2.xsd

            http://cybox.mitre.org/objects#EmailMessageObject

            http://cybox.mitre.org/XMLSchema/objects/Email_Message/Email_Message_Object_1.1.xsd"
            cybox_major_version="1" cybox_minor_version="0(draft)">
            <cybox:Observable>
                        <cybox:Stateful_Measure>
                                    <cybox:Object id="cybox:A1" type="Email Message">
                                                <cybox:Defined_Object
xsi:type="EmailMessageObj:EmailMessageObjectType">
                                                            <EmailMessageObj:Header>
                                                                        <EmailMessageObj:To>
                                                                                    <EmailMessageObj:Recipient category="e-
mail"><AddrObj:Address_Value
datatype="String">victim1@target.com</AddrObj:Address_Value></EmailMessageObj:Recipient>
                                                                                    <EmailMessageObj:Recipient category="e-
mail"><AddrObj:Address_Value
datatype="String">victim2@target.com</AddrObj:Address_Value></EmailMessageObj:Recipient>
                                                                        </EmailMessageObj:To>
                                                                        <EmailMessageObj:From category="e-mail">
                                                                                    <AddrObj:Address_Value
datatype="String">attacker@example.com</AddrObj:Address_Value>
                                                                        </EmailMessageObj:From>
```

```
                                                <EmailMessageObj:Subject datatype="String">New
modifications to the specification</EmailMessageObj:Subject>
                                        </EmailMessageObj:Header>
                                </cybox:Defined_Object>
                                <cybox:Related_Objects>
                                        <cybox:Related_Object idref="cybox:A2"
relationship="Received_From"/>
                                </cybox:Related_Objects>
                        </cybox:Object>
                </cybox:Stateful_Measure>
        </cybox:Observable>
        <cybox:Observable>
                <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:A2" type="IP Address">
                                <cybox:Defined_Object xsi:type="AddrObj:AddressObjectType"
category="ipv4-addr" is_source="true">
                                        <AddrObj:Address_Value
datatype="String">192.168.1.1</AddrObj:Address_Value>
                                </cybox:Defined_Object>
                        </cybox:Object>
                </cybox:Stateful_Measure>
        </cybox:Observable>
</cybox:Observables>
```

4.3.1.6 Simple email with simple file attachment

```
<?xml version="1.0" encoding="UTF-8"?>
<cybox:Observables xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cybox="http://cybox.mitre.org/cybox_v1"
        xmlns:common="http://cybox.mitre.org/Common_v1"
        xmlns:AddrObj="http://cybox.mitre.org/objects#AddressObject"
        xmlns:URIObj="http://cybox.mitre.org/objects#URIObject"
        xmlns:FileObj="http://cybox.mitre.org/objects#FileObject"
        xmlns:EmailMessageObj="http://cybox.mitre.org/XMLSchema/objects#EmailMessageObject"
        xsi:schemaLocation="http://cybox.mitre.org/Common_v1

        http://cybox.mitre.org/XMLSchema/cybox_core_v1.0(draft).xsd
                                                                http://cybox.mitre.org/objects#FileObject

        http://cybox.mitre.org/XMLSchema/objects/File/File_Object_1.2.xsd

        http://cybox.mitre.org/objects#EmailMessageObject

        http://cybox.mitre.org/XMLSchema/objects/Email_Message/Email_Message_Object_1.1.xsd"
        cybox_major_version="1" cybox_minor_version="0(draft)">
        <cybox:Observable>
                <cybox:Stateful_Measure>
                        <cybox:Object id="A1" type="Email Message">
                                <cybox:Defined_Object
xsi:type="EmailMessageObj:EmailMessageObjectType">
                                        <EmailMessageObj:Attachments>
                                                <EmailMessageObj:File>
                                                        <FileObj:File_Name datatype="String">FooBar
Specification (critical revision).doc</FileObj:File_Name>
                                                        <FileObj:Hashes>
```

```
                                                                <common:Hash>
                                                                        <common:Simple_Hash_Value
datatype="hexBinary">4EC0027BEF4D7E1786A04D021FA8A67F</common:Simple_Hash_Value>
                                                                </common:Hash>
                                                        </FileObj:Hashes>
                                                </EmailMessageObj:File>
                                        </EmailMessageObj:Attachments>
                                        <EmailMessageObj:Header>
                                                <EmailMessageObj:To>
                                                        <EmailMessageObj:Recipient category="e-
mail"><AddrObj:Address_Value
datatype="String">victim1@target.com</AddrObj:Address_Value></EmailMessageObj:Recipient>
                                                        <EmailMessageObj:Recipient category="e-
mail"><AddrObj:Address_Value
datatype="String">victim2@target.com</AddrObj:Address_Value></EmailMessageObj:Recipient>
                                                </EmailMessageObj:To>
                                                <EmailMessageObj:From category="e-mail">
                                                        <AddrObj:Address_Value
datatype="String">attacker@example.com</AddrObj:Address_Value>
                                                </EmailMessageObj:From>
                                                <EmailMessageObj:Subject datatype="String">New
modifications to the specification</EmailMessageObj:Subject>
                                        </EmailMessageObj:Header>
                                </cybox:Defined_Object>
                        </cybox:Object>
                </cybox:Stateful_Measure>
        </cybox:Observable>
</cybox:Observables>
```

4.3.1.7 Observable pattern for a URL matching one of three values utilizing IsInSet

```
<?xml version="1.0" encoding="UTF-8"?>
<cybox:Observables
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cybox="http://cybox.mitre.org/cybox_v1"
        xmlns:common="http://cybox.mitre.org/Common_v1"
        xmlns:URIObj="http://cybox.mitre.org/objects#URIObject"
        xsi:schemaLocation="http://cybox.mitre.org/cybox_v1

        http://cybox.mitre.org/XMLSchema/cybox_core_v1.0(draft).xsd
                                                                http://cybox.mitre.org/objects#URIObject

        http://cybox.mitre.org/XMLSchema/objects/URI/URI_Object_1.1.xsd"
        cybox_major_version="1" cybox_minor_version="0(draft)">
        <cybox:Observable id="cybox:Obs1">
                <!-- Observable for any single URL matching one of three URLs utilizing IsInSet-->
                <cybox:Stateful_Measure>
                        <cybox:Object id="A1" type="URI">
                                <cybox:Defined_Object xsi:type="URIObj:URIObjectType">
                                        <URIObj:Value condition="IsInSet"
value_set="www.sample1.com/index.html, sample2.com/login.html, dev.sample3.com/index/kb.html"
datatype="AnyURI"/>
                                </cybox:Defined_Object>
                        </cybox:Object>
                </cybox:Stateful_Measure>
```

```
        </cybox:Observable>
</cybox:Observables>
```

4.3.1.8 Observable pattern for a URL matching one of three values utilizing logical OR composition

```
<?xml version="1.0" encoding="UTF-8"?>
<cybox:Observables
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cybox="http://cybox.mitre.org/cybox_v1"
        xmlns:common="http://cybox.mitre.org/Common_v1"
        xmlns:URIObj="http://cybox.mitre.org/objects#URIObject"
        xsi:schemaLocation="http://cybox.mitre.org/cybox_v1

        http://cybox.mitre.org/XMLSchema/cybox_core_v1.0(draft).xsd
                                                        http://cybox.mitre.org/objects#URIObject

        http://cybox.mitre.org/XMLSchema/objects/URI/URI_Object_1.1.xsd"
        cybox_major_version="1" cybox_minor_version="0(draft)">
        <cybox:Observable id="cybox:Obs4">
                <!-- Observable for any single URL matching one of three URLs utilizing logical
composition -->
                <cybox:Observable_Composition operator="OR">
                        <cybox:Observable id="cybox:Obs1">
                                <cybox:Stateful_Measure>
                                        <cybox:Object id="cybox:A1" type="URI">
                                                <cybox:Defined_Object
xsi:type="URIObj:URIObjectType">
                                                        <URIObj:Value condition="Equals"
datatype="AnyURI">www.sample1.com/index.html</URIObj:Value>
                                                </cybox:Defined_Object>
                                        </cybox:Object>
                                </cybox:Stateful_Measure>
                        </cybox:Observable>
                        <cybox:Observable id="cybox:Obs2">
                                <cybox:Stateful_Measure>
                                        <cybox:Object id="cybox:A2" type="URI">
                                                <cybox:Defined_Object
xsi:type="URIObj:URIObjectType">
                                                        <URIObj:Value condition="Equals"
datatype="AnyURI">sample2.com/login.html</URIObj:Value>
                                                </cybox:Defined_Object>
                                        </cybox:Object>
                                </cybox:Stateful_Measure>
                        </cybox:Observable>
                        <cybox:Observable id="cybox:Obs3">
                                <cybox:Stateful_Measure>
                                        <cybox:Object id="cybox:A3" type="URI">
                                                <cybox:Defined_Object
xsi:type="URIObj:URIObjectType">
                                                        <URIObj:Value condition="Equals"
datatype="AnyURI">dev.sample3.com/index/kb.html</URIObj:Value>
                                                </cybox:Defined_Object>
                                        </cybox:Object>
                                </cybox:Stateful_Measure>
                        </cybox:Observable>
                </cybox:Observable_Composition>
        </cybox:Observable>
```

```
</cybox:Observables>
```

4.3.1.9 Observable pattern for a URL matching one of three values utilizing logical OR composition and Object pooling

```
<?xml version="1.0" encoding="UTF-8"?>
<cybox:Observables
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cybox="http://cybox.mitre.org/cybox_v1"
        xmlns:common="http://cybox.mitre.org/Common_v1"
        xmlns:URIObj="http://cybox.mitre.org/objects#URIObject"
        xsi:schemaLocation="http://cybox.mitre.org/cybox_v1

        http://cybox.mitre.org/XMLSchema/cybox_core_v1.0(draft).xsd
                                                        http://cybox.mitre.org/objects#URIObject

        http://cybox.mitre.org/XMLSchema/objects/URI/URI_Object_1.1.xsd"
        cybox_major_version="1" cybox_minor_version="0(draft)">
        <cybox:Observable id="cybox:Obs4">
                <!-- Observable for any single URL matching one of three URLs utilizing logical
composition and Object Pools-->
                <cybox:Observable_Composition operator="OR">
                        <cybox:Observable id="cybox:Obs1">
                                <cybox:Stateful_Measure>
                                        <cybox:Object idref="cybox:A1" type="URI"/>
                                </cybox:Stateful_Measure>
                        </cybox:Observable>
                        <cybox:Observable id="cybox:Obs2">
                                <cybox:Stateful_Measure>
                                        <cybox:Object idref="cybox:A2" type="URI"/>
                                </cybox:Stateful_Measure>
                        </cybox:Observable>
                        <cybox:Observable id="cybox:Obs3">
                                <cybox:Stateful_Measure>
                                        <cybox:Object idref="cybox:A3" type="URI"/>
                                </cybox:Stateful_Measure>
                        </cybox:Observable>
                </cybox:Observable_Composition>
        </cybox:Observable>
        <cybox:Pools>
                <cybox:Object_Pool>
                        <cybox:Object id="cybox:A1" type="URI">
                                <cybox:Defined_Object xsi:type="URIObj:URIObjectType">
                                        <URIObj:Value condition="Equals"
datatype="AnyURI">www.sample1.com/index.html</URIObj:Value>
                                </cybox:Defined_Object>
                        </cybox:Object>
                        <cybox:Object id="cybox:A2" type="URI">
                                <cybox:Defined_Object xsi:type="URIObj:URIObjectType">
                                        <URIObj:Value condition="Equals"
datatype="AnyURI">sample2.com/login.html</URIObj:Value>
                                </cybox:Defined_Object>
                        </cybox:Object>
                        <cybox:Object id="cybox:A3" type="URI">
                                <cybox:Defined_Object xsi:type="URIObj:URIObjectType">
```

```
                                        <URIObj:Value condition="Equals"
datatype="AnyURI">dev.sample3.com/index/kb.html</URIObj:Value>
                                    </cybox:Defined_Object>
                                </cybox:Object>
                            </cybox:Object_Pool>
                        </cybox:Pools>
</cybox:Observables>
```

4.3.2 Complex Example

The following complex example is derived from observable data from a real-world attack campaign observed in the wild during March, 2012. This campaign is known by many names but Iran-Oil is likely its most common name of reference.

4.3.2.1 Iran-Oil example as only static observable Stateful Measures

```
<?xml version="1.0" encoding="UTF-8"?>
<cybox:Observables
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cybox="http://cybox.mitre.org/cybox_v1"
        xmlns:common="http://cybox.mitre.org/Common_v1"
        xmlns:AddrObj="http://cybox.mitre.org/objects#AddressObject"
        xmlns:URIObj="http://cybox.mitre.org/objects#URIObject"
        xmlns:FileObj="http://cybox.mitre.org/objects#FileObject"
        xmlns:EmailMessageObj="http://cybox.mitre.org/XMLSchema/objects#EmailMessageObject"
        xsi:schemaLocation="http://cybox.mitre.org/Common_v1

        http://cybox.mitre.org/XMLSchema/cybox_core_v1.0(draft).xsd
                                                        http://cybox.mitre.org/objects#URIObject

        http://cybox.mitre.org/XMLSchema/objects/URI/URI_Object_1.1.xsd
                                                        http://cybox.mitre.org/objects#FileObject

        http://cybox.mitre.org/XMLSchema/objects/File/File_Object_1.2.xsd

        http://cybox.mitre.org/objects#EmailMessageObject

        http://cybox.mitre.org/XMLSchema/objects/Email_Message/Email_Message_Object_1.1.xsd"
        cybox_major_version="1" cybox_minor_version="0(draft)">

<!-- This collection of observables were observed as part of the widespread "Iran-Oil" (among many other
names used) attack campaign in March 2012 -->

        <cybox:Observable id="cybox:guid-guid-1a937ec2-90ab-4e0e-a37c-db9b2e66a58e">
                <!-- "Iran-Oil" attack campaign email message with raw header-->
                <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:guid-51359587-f201-4383-b032-5a64522fcd7d"
type="Email Message">
                                        <cybox:Defined_Object
xsi:type="EmailMessageObj:EmailMessageObjectType">
                                        <EmailMessageObj:Attachments><EmailMessageObj:File
object_reference="cybox:guid-49d31c13-8d7b-4528-b8d6-
ce8ed0d43ad7"/></EmailMessageObj:Attachments>
                                        <EmailMessageObj:Header>
```

<EmailMessageObj:To><EmailMessageObj:Recipient category="e-mail"><AddrObj:Address_Value datatype="String">william.abnett@gmail.com</AddrObj:Address_Value></EmailMessageObj:Recipient></EmailMessageObj:To>
<EmailMessageObj:From category="e-mail"><AddrObj:Address_Value datatype="String">wmorrison89@gmail.com</AddrObj:Address_Value></EmailMessageObj:From>
<EmailMessageObj:Subject datatype="String">Iran's Oil and Nuclear Situation</EmailMessageObj:Subject>
<EmailMessageObj:Date datatype="DateTime">2012-03-02T07:42:24Z</EmailMessageObj:Date>
</EmailMessageObj:Header>
<EmailMessageObj:Raw_Header datatype="String"><![CDATA[
Return-Path: <wmorrison89@gmail.com>
Received-SPF: pass (google.com: domain of wmorrison89@gmail.com designates 10.236.185.4 as permitted sender) client-ip=10.236.185.4;
Authentication-Results: mr.google.com; spf=pass (google.com: domain of wmorrison89@gmail.com designates 10.236.185.4 as permitted sender) smtp.mail=wmorrison89@gmail.com; dkim=pass header.i=wmorrison89@gmail.com
Received: from mr.google.com ([10.236.185.4]) by 10.236.185.4 with SMTP id t4mr5301660yhm.129.1330692273662 (num_hops = 1); Fri, 02 Mar 2012 04:44:33 -0800 (PST)
MIME-Version: 1.0
Received: by 10.236.185.4 with SMTP id t4mr4236541yhm.129.1330692265380; Fri,
02 Mar 2012 04:44:25 -0800 (PST)
Received: by 10.147.35.14 with HTTP; Fri, 2 Mar 2012 04:44:24 -0800 (PST)
In-Reply-To: <CADY6HTa-jmaqmtVyyT-nLz6reztNjcs-617wL4bt9YBOGu+h4w@mail.gmail.com>
References: <CADY6HTa-jmaqmtVyyT-nLz6reztNjcs-617wL4bt9YBOGu+h4w@mail.gmail.com>
Date: Fri, 2 Mar 2012 07:44:24 -0500
Message-ID: <CADY6HTZ6oopY5v6WkYU81YcSQw3X124CK_Fx4jhnhUiU3Y9z6A@mail.gmail.com>
Subject: Iran's Oil and Nuclear Situation
From: william abnett <wmorrison89@gmail.com>
To: william.abnett <william.abnett@gmail.com>
Content-Type: multipart/mixed; boundary="20cf303f67fac8928804ba41efd5"
]]></EmailMessageObj:Raw_Header>
</cybox:Defined_Object>
</cybox:Object>
</cybox:Stateful_Measure>
</cybox:Observable>

<cybox:Observable id="cybox:guid-cybox:35f04c28-5fd2-4d72-8aae-2ad04ee1811f">
<!-- Iran-Oil corrupted .doc file-->
<cybox:Stateful_Measure>
<cybox:Object id="cybox:guid-49d31c13-8d7b-4528-b8d6-ce8ed0d43ad7" type="File">
<cybox:Description><common:Text>The word document contains flash, which downloads a corrupted mp4 file. The mp4 file itself is not anything special but an 0C filled (22kb) mp4 file with a valid mp4 header.</common:Text></cybox:Description>
<cybox:Defined_Object xsi:type="FileObj:FileObjectType">
<FileObj:File_Name datatype="String">Iran's Oil and Nuclear Situation.doc</FileObj:File_Name>

```
                                        <FileObj:Size_In_Bytes
datatype="UnsignedLong">106604</FileObj:Size_In_Bytes>
                                        <FileObj:Hashes><common:Hash><common:Type
datatype="String">MD5</common:Type><common:Simple_Hash_Value condition="Equals"
datatype="hexBinary">E92A4FC283EB2802AD6D0E24C7FCC857</common:Simple_Hash_Value></co
mmon:Hash></FileObj:Hashes>
                                    </cybox:Defined_Object>
                                </cybox:Object>
                            </cybox:Stateful_Measure>
                        </cybox:Observable>

                        <cybox:Observable id="cybox:guid-f005fbc6-7427-43ea-8e1e-9a341836f76b">
                            <!-- Iran-Oil invalid .mp4 downloader file-->
                            <cybox:Stateful_Measure>
                                <cybox:Object id="cybox:guid-8b463e0d-cc16-4036-950e-5eeb09bc51aa"
type="File">
                                    <cybox:Description><common:Text>This mp4 file causes memory
corruption and code execution via heap-spraying code injection.</common:Text></cybox:Description>
                                    <cybox:Defined_Object xsi:type="FileObj:FileObjectType">
                                        <FileObj:File_Name
datatype="String">test.mp4</FileObj:File_Name>
                                        <FileObj:Size_In_Bytes
datatype="UnsignedLong">22384</FileObj:Size_In_Bytes>
                                        <FileObj:Hashes><common:Hash><common:Type
datatype="String">MD5</common:Type><common:Simple_Hash_Value condition="Equals"
datatype="hexBinary">8933598C8B1FA5E493497B11C48DA4F2</common:Simple_Hash_Value></com
mon:Hash></FileObj:Hashes>
                                    </cybox:Defined_Object>
                                    <cybox:Related_Objects>
                                        <cybox:Related_Object idref="cybox:guid-49d31c13-8d7b-4528-
b8d6-ce8ed0d43ad7" type="File" relationship="Downloaded_By"/>
                                        <cybox:Related_Object idref="cybox:guid-61041b8b-0c15-48a0-
ac5f-b49488788010" type="URI" relationship="Downloaded_From"/>
                                    </cybox:Related_Objects>
                                </cybox:Object>
                            </cybox:Stateful_Measure>
                        </cybox:Observable>

                        <cybox:Observable id="cybox:guid-b63c8bd4-e9c6-4e5a-b012-040f81dcc699">
                            <!-- URL from which malicious .mp4 file was downloaded-->
                            <cybox:Stateful_Measure>
                                <cybox:Object id="cybox:guid-61041b8b-0c15-48a0-ac5f-b49488788010"
type="URI">
                                    <cybox:Defined_Object xsi:type="URIObj:URIObjectType" type="URL">
                                        <URIObj:Value datatype="AnyURI"
condition="Equals">http://208.115.230.76/test.mp4</URIObj:Value>
                                    </cybox:Defined_Object>
                                </cybox:Object>
                            </cybox:Stateful_Measure>
                        </cybox:Observable>

                        <cybox:Observable id="cybox:guid-210f18f3-3874-4f9a-861d-71b328be90c6">
                            <!-- Iran-Oil .exe Trojan file-->
                            <cybox:Stateful_Measure>
                                <cybox:Object id="cybox:guid-b7e0bc39-f519-4878-8fb0-5902554efe1c"
type="File">
```

<cybox:Description><common:Text>The file (us.exe MD5: FD1BE09E499E8E380424B3835FC973A8 4861440 bytes) is created in the logged in user %Temp% directory. The size of the embedded file is 22.5 KB (23040 bytes) and the size of the created us.exe is 4.63MB. It is an odd discrepancy until you look at the file and it looks like the code is repeated over and over - 211 times. The file resource section indicates the file is meant to look like a java updater, which is always larger than 22.5KB and that would explain all this padding, which is done at the time when the file is being written to the disk.</common:Text></cybox:Description>
<cybox:Defined_Object xsi:type="FileObj:FileObjectType">
<FileObj:File_Name datatype="String">us.exe</FileObj:File_Name>
<FileObj:File_Path datatype="String">%Temp%</FileObj:File_Path>
<FileObj:Size_In_Bytes datatype="UnsignedLong">4861440</FileObj:Size_In_Bytes>
<FileObj:Hashes><common:Hash><common:Type datatype="String">MD5</common:Type><common:Simple_Hash_Value condition="Equals" datatype="hexBinary">FD1BE09E499E8E380424B3835FC973A8</common:Simple_Hash_Value></common:Hash></FileObj:Hashes>
</cybox:Defined_Object>
<cybox:Related_Objects>
<cybox:Related_Object idref="cybox:guid-8b463e0d-cc16-4036-950e-5eeb09bc51aa" type="File" relationship="Created_By"/>
<!-- The trojan connects to the following set of URLs/IPs for C&C -->
<cybox:Related_Object idref="cybox:guid-41b220d8-4c45-48de-9d08-30d661b2dc8e" type="URI" relationship="Connected_To"/>
<cybox:Related_Object idref="cybox:guid-61aa225b-90ef-415c-8bbd-a17282e457c9" type="IP Address" relationship="Connected_To"/>
<cybox:Related_Object idref="cybox:guid-568db11e-39ee-43d7-83d8-032bdec3801a" type="URI" relationship="Connected_To"/>
<cybox:Related_Object idref="cybox:guid-80bea4d1-0e70-4a03-a54f-e40373bf94f1" type="IP Address" relationship="Connected_To"/>
<cybox:Related_Object idref="cybox:guid-af7cb3b6-d70b-4b3b-b24f-7cfad739710f" type="URI" relationship="Connected_To"/>
<cybox:Related_Object idref="cybox:guid-5ceb9d54-24e2-4627-948d-6b92ac81962a" type="IP Address" relationship="Connected_To"/>
</cybox:Related_Objects>
</cybox:Object>
</cybox:Stateful_Measure>
</cybox:Observable>

<cybox:Observable id="cybox:guid-dee72b3e-82fb-4319-bfcc-007e3cf930e8">
<!-- Iran-Oil core embedded .exe Trojan file-->
<cybox:Stateful_Measure>
<cybox:Object id="cybox:guid-bed1ff22-08e8-4e04-b7ac-908b5271176f" type="File">
<cybox:Defined_Object xsi:type="FileObj:FileObjectType">
<FileObj:File_Name datatype="String">us-embedded.exe</FileObj:File_Name>
<FileObj:Size_In_Bytes datatype="UnsignedLong">23040</FileObj:Size_In_Bytes>
<FileObj:Hashes><common:Hash><common:Type datatype="String">MD5</common:Type><common:Simple_Hash_Value condition="Equals" datatype="hexBinary">CB3DCDE34FD9FF0E19381D99B02F9692</common:Simple_Hash_Value></common:Hash></FileObj:Hashes>
</cybox:Defined_Object>

```
                                        <cybox:Related_Objects>
                                                <cybox:Related_Object idref="cybox:guid-b7e0bc39-f519-4878-
8fb0-5902554efe1c" type="File" relationship="Contained_Within"/>
                                        </cybox:Related_Objects>
                                </cybox:Object>
                        </cybox:Stateful_Measure>
                </cybox:Observable>

                <!-- The next six Observables represent the 3 different URL/IP pairs of C&C servers that the
trojan communicates with-->
                <cybox:Observable id="cybox:guid-066cef51-c886-432e-9a22-a17f57f3f3f2">
                        <!-- One of three Command and Control URLs-->
                        <cybox:Stateful_Measure>
                                <cybox:Object id="cybox:guid-41b220d8-4c45-48de-9d08-30d661b2dc8e"
type="URI">
                                        <cybox:Defined_Object xsi:type="URIObj:URIObjectType" type="URL">
                                                <URIObj:Value datatype="AnyURI"
condition="Equals">www.documents.myPicture.info</URIObj:Value>
                                        </cybox:Defined_Object>
                                        <cybox:Related_Objects>
                                                <cybox:Related_Object  idref="cybox:guid-61aa225b-90ef-415c-
8bbd-a17282e457c9" type="IP Address" relationship="Resolved_To"/>
                                        </cybox:Related_Objects>
                                </cybox:Object>
                        </cybox:Stateful_Measure>
                </cybox:Observable>
                <cybox:Observable id="cybox:guid-4e05804c-f552-44e1-9793-ff4bb7f88f9c">
                        <!-- One of three Command and Control IPs-->
                        <cybox:Stateful_Measure>
                                <cybox:Object id="cybox:guid-61aa225b-90ef-415c-8bbd-a17282e457c9"
type="IP Address">
                                        <cybox:Defined_Object xsi:type="AddrObj:AddressObjectType"
category="ipv4-addr">
                                                <AddrObj:Address_Value datatype="String"
condition="Equals">199.192.156.134</AddrObj:Address_Value>
                                        </cybox:Defined_Object>
                                </cybox:Object>
                        </cybox:Stateful_Measure>
                </cybox:Observable>

                <cybox:Observable id="cybox:guid-75ce59ad-1f01-4eae-9ecc-0b22c4c24ce7">
                        <!-- One of three Command and Control URLs-->
                        <cybox:Stateful_Measure>
                                <cybox:Object id="cybox:guid-568db11e-39ee-43d7-83d8-032bdec3801a"
type="URI">
                                        <cybox:Defined_Object xsi:type="URIObj:URIObjectType" type="URL">
                                                <URIObj:Value datatype="AnyURI"
condition="Equals">documents.myPicture.info</URIObj:Value>
                                        </cybox:Defined_Object>
                                        <cybox:Related_Objects>
                                                <cybox:Related_Object idref="cybox:guid-80bea4d1-0e70-4a03-
a54f-e40373bf94f1" type="IP Address" relationship="Resolved_To"/>
                                        </cybox:Related_Objects>
                                </cybox:Object>
                        </cybox:Stateful_Measure>
                </cybox:Observable>
```

```
        <cybox:Observable id="cybox:guid-1ea53b14-8fe9-467b-a298-62d9684e797d">
                <!-- One of three Command and Control IPs-->
                <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:guid-80bea4d1-0e70-4a03-a54f-e40373bf94f1"
type="IP Address">
                                <cybox:Defined_Object xsi:type="AddrObj:AddressObjectType"
category="ipv4-addr">
                                        <AddrObj:Address_Value datatype="String"
condition="Equals">199.192.156.134</AddrObj:Address_Value>
                                </cybox:Defined_Object>
                        </cybox:Object>
                </cybox:Stateful_Measure>
        </cybox:Observable>

        <cybox:Observable id="cybox:guid-f6c8ee75-ee7e-4490-bd5d-0661d0db7264">
                <!-- One of three Command and Control URLs-->
                <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:guid-af7cb3b6-d70b-4b3b-b24f-7cfad739710f"
type="URI">
                                <cybox:Defined_Object xsi:type="URIObj:URIObjectType" type="URL">
                                        <URIObj:Value datatype="AnyURI"
condition="Equals">ftp.documents.myPicture.info</URIObj:Value>
                                </cybox:Defined_Object>
                                <cybox:Related_Objects>
                                        <cybox:Related_Object idref="cybox:guid-5ceb9d54-24e2-4627-
948d-6b92ac81962a" type="IP Address" relationship="Resolved_To"/>
                                </cybox:Related_Objects>
                        </cybox:Object>
                </cybox:Stateful_Measure>
        </cybox:Observable>
        <cybox:Observable id="cybox:guid-c78c0a83-6d14-45f8-827f-f758f0cd11ea">
                <!-- One of three Command and Control IPs-->
                <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:guid-5ceb9d54-24e2-4627-948d-6b92ac81962a"
type="IP Address">
                                <cybox:Defined_Object xsi:type="AddrObj:AddressObjectType"
category="ipv4-addr">
                                        <AddrObj:Address_Value datatype="String"
condition="Equals">199.192.156.134</AddrObj:Address_Value>
                                </cybox:Defined_Object>
                        </cybox:Object>
                </cybox:Stateful_Measure>
        </cybox:Observable>

        <cybox:Observable id="cybox:guid-47d6a950-884d-46b5-9938-ac5555065a81">
                <!-- This composed observable defines a pattern that is true if the observed email exists
AND the malicious .doc file exists AND the downloader .mp4 file exists AND the trojan .exe exists AND all
three of the C&C IP addresses are seen-->
                <!-- This yields a very tight filter that will have very low false positives but could miss
almost any variation of the attack elements-->
                <cybox:Observable_Composition operator="AND">
                        <!-- "Iran-Oil" attack campaign email message with raw header-->
                        <cybox:Observable idref="cybox:guid-1a937ec2-90ab-4e0e-a37c-
db9b2e66a58e"/>
                        <!-- Iran-Oil corrupted .doc file-->
```

```
                <cybox:Observable idref="cybox:guid-35f04c28-5fd2-4d72-8aae-
2ad04ee1811f"/>
                <!-- Iran-Oil invalid .mp4 downloader file-->
                <cybox:Observable idref="cybox:guid-f005fbc6-7427-43ea-8e1e-
9a341836f76b"/>
                <!-- Iran-Oil .exe Trojan file-->
                <cybox:Observable idref="cybox:guid-210f18f3-3874-4f9a-861d-
71b328be90c6"/>
                <!-- The three Command and Control IPs-->
                <cybox:Observable idref="cybox:guid-4e05804c-f552-44e1-9793-ff4bb7f88f9c"/>
                <cybox:Observable idref="cybox:guid-1ea53b14-8fe9-467b-a298-
62d9684e797d"/>
                <cybox:Observable idref="cybox:guid-c78c0a83-6d14-45f8-827f-f758f0cd11ea"/>
            </cybox:Observable_Composition>
        </cybox:Observable>

        <cybox:Observable id="cybox:guid-94b0aa45-065e-486f-acaf-2d8e793f525e">
            <!-- This composed observable defines a pattern that is true if the observed email exists
OR the malicious .doc file exists OR the downloader .mp4 file exists OR the trojan .exe exists OR any of
the three C&C IP addresses are seen-->
            <!-- This yields a very loose filter that could have false positives but could catch
numerous potential variations of the attack elements-->
            <cybox:Observable_Composition operator="OR">
                <!-- "Iran-Oil" attack campaign email message with raw header-->
                <cybox:Observable idref="cybox:guid-1a937ec2-90ab-4e0e-a37c-
db9b2e66a58e"/>
                <!-- Iran-Oil corrupted .doc file-->
                <cybox:Observable idref="cybox:guid-35f04c28-5fd2-4d72-8aae-
2ad04ee1811f"/>
                <!-- Iran-Oil invalid .mp4 downloader file-->
                <cybox:Observable idref="cybox:guid-f005fbc6-7427-43ea-8e1e-
9a341836f76b"/>
                <!-- Iran-Oil .exe Trojan file-->
                <cybox:Observable idref="cybox:guid-210f18f3-3874-4f9a-861d-
71b328be90c6"/>
                <!-- The three Command and Control IPs-->
                <cybox:Observable idref="cybox:guid-4e05804c-f552-44e1-9793-ff4bb7f88f9c"/>
                <cybox:Observable idref="cybox:guid-1ea53b14-8fe9-467b-a298-
62d9684e797d"/>
                <cybox:Observable idref="cybox:guid-c78c0a83-6d14-45f8-827f-f758f0cd11ea"/>
            </cybox:Observable_Composition>
        </cybox:Observable>

        <!-- CybOX enables a wide myriad of other potential observable pattern variations at the logical
composition level or utilizing patterns at the Object attribute level including Regex all of which allow the
user to define an almost infinitely variable set of patterns and filters -->

</cybox:Observables>
```

4.3.2.2 Iran-Oil example as dynamic observable Events

```
<?xml version="1.0" encoding="UTF-8"?>
<cybox:Observables
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cybox="http://cybox.mitre.org/cybox_v1"
```

```
        xmlns:common="http://cybox.mitre.org/Common_v1"
        xmlns:AddrObj="http://cybox.mitre.org/objects#AddressObject"
        xmlns:URIObj="http://cybox.mitre.org/objects#URIObject"
        xmlns:FileObj="http://cybox.mitre.org/objects#FileObject"
        xmlns:EmailMessageObj="http://cybox.mitre.org/XMLSchema/objects#EmailMessageObject"
        xsi:schemaLocation="http://cybox.mitre.org/Common_v1

        http://cybox.mitre.org/XMLSchema/cybox_core_v1.0(draft).xsd
                                                                http://cybox.mitre.org/objects#URIObject

        http://cybox.mitre.org/XMLSchema/objects/URI/URI_Object_1.1.xsd
                                                                http://cybox.mitre.org/objects#FileObject

        http://cybox.mitre.org/XMLSchema/objects/File/File_Object_1.2.xsd

        http://cybox.mitre.org/objects#EmailMessageObject

        http://cybox.mitre.org/XMLSchema/objects/Email_Message/Email_Message_Object_1.1.xsd"
        cybox_major_version="1" cybox_minor_version="0(draft)">
        <!-- This collection of observables were observed as part of the widespread "Iran-Oil" (among
many other names used) attack campaign in March 2012 -->
        <cybox:Observable id="cybox:guid-1a937ec2-90ab-4e0e-a37c-db9b2e66a58e">
                <!-- Receive "Iran-Oil" attack campaign email message -->
                <cybox:Event type="Email Ops">
                        <cybox:Description>
                                <common:Text>Receive "Iran-Oil" attack campaign email
message.</common:Text>
                        </cybox:Description>
                        <cybox:Actions>
                                <cybox:Action type="Receive">
                                        <cybox:Associated_Objects>
                                                <cybox:Associated_Object id="cybox:guid-51359587-
f201-4383-b032-5a64522fcd7d" type="Email Message" association_type="Returned">
                                                        <cybox:Defined_Object
xsi:type="EmailMessageObj:EmailMessageObjectType">
                                                                <EmailMessageObj:Attachments>
                                                                        <EmailMessageObj:File
object_reference="cybox:guid-49d31c13-8d7b-4528-b8d6-ce8ed0d43ad7"/>
                                                                </EmailMessageObj:Attachments>
                                                                <EmailMessageObj:Header>

        <EmailMessageObj:To><EmailMessageObj:Recipient category="e-mail">

        <AddrObj:Address_Value
datatype="String">william.abnett@gmail.com</AddrObj:Address_Value>

        </EmailMessageObj:Recipient></EmailMessageObj:To>
                                                                        <EmailMessageObj:From
category="e-mail">

        <AddrObj:Address_Value
datatype="String">wmorrison89@gmail.com</AddrObj:Address_Value>
                                                                        </EmailMessageObj:From>
                                                                        <EmailMessageObj:Subject
datatype="String">Iran's Oil and Nuclear Situation</EmailMessageObj:Subject>
```

<EmailMessageObj:Date datatype="DateTime">2012-03-02T07:42:24Z</EmailMessageObj:Date>
</EmailMessageObj:Header>
<EmailMessageObj:Raw_Header datatype="String"><![CDATA[
Return-Path: <wmorrison89@gmail.com>
Received-SPF: pass (google.com: domain of wmorrison89@gmail.com designates 10.236.185.4 as permitted sender) client-ip=10.236.185.4;
Authentication-Results: mr.google.com; spf=pass (google.com: domain of wmorrison89@gmail.com designates 10.236.185.4 as permitted sender) smtp.mail=wmorrison89@gmail.com; dkim=pass header.i=wmorrison89@gmail.com
Received: from mr.google.com ([10.236.185.4]) by 10.236.185.4 with SMTP id t4mr5301660yhm.129.1330692273662 (num_hops = 1); Fri, 02 Mar 2012 04:44:33 -0800 (PST)
MIME-Version: 1.0
Received: by 10.236.185.4 with SMTP id t4mr4236541yhm.129.1330692265380; Fri,
02 Mar 2012 04:44:25 -0800 (PST)
Received: by 10.147.35.14 with HTTP; Fri, 2 Mar 2012 04:44:24 -0800 (PST)
In-Reply-To: <CADY6HTa-jmaqmtVyyT-nLz6reztNjcs-617wL4bt9YBOGu+h4w@mail.gmail.com>
References: <CADY6HTa-jmaqmtVyyT-nLz6reztNjcs-617wL4bt9YBOGu+h4w@mail.gmail.com>
Date: Fri, 2 Mar 2012 07:44:24 -0500
Message-ID: <CADY6HTZ6oopY5v6WkYU81YcSQw3X124CK_Fx4jhnhUiU3Y9z6A@mail.gmail.com>
Subject: Iran's Oil and Nuclear Situation
From: william abnett <wmorrison89@gmail.com>
To: william.abnett <william.abnett@gmail.com>
Content-Type: multipart/mixed; boundary="20cf303f67fac8928804ba41efd5"
]]></EmailMessageObj:Raw_Header>
</cybox:Defined_Object>
</cybox:Associated_Object>
</cybox:Associated_Objects>
</cybox:Action>
</cybox:Actions>
</cybox:Event>
</cybox:Observable>

<cybox:Observable id="cybox:guid-35f04c28-5fd2-4d72-8aae-2ad04ee1811f">
<!-- Open Iran-Oil corrupted .doc file-->
<cybox:Event type="File Ops (CRUD)">
<cybox:Description>
<common:Text>Open Iran-Oil corrupted .doc file.</common:Text>
</cybox:Description>
<cybox:Actions>
<cybox:Action type="Open">
<cybox:Associated_Objects>
<cybox:Associated_Object id="cybox:guid-49d31c13-8d7b-4528-b8d6-ce8ed0d43ad7" type="File" association_type="Affected">
<cybox:Description>
<common:Text>The word document contains flash, which downloads a corrupted mp4 file. The mp4 file itself is not anything special but an 0C filled (22kb) mp4 file with a valid mp4 header.</common:Text>
</cybox:Description>

```
                                                <cybox:Defined_Object
xsi:type="FileObj:FileObjectType">
                                                        <FileObj:File_Name
datatype="String">Iran's Oil and Nuclear Situation.doc</FileObj:File_Name>
                                                        <FileObj:Size_In_Bytes
datatype="UnsignedLong">106604</FileObj:Size_In_Bytes>
                                                        <FileObj:Hashes>
                                                                <common:Hash>
                                                                        <common:Type
datatype="String">MD5</common:Type>

        <common:Simple_Hash_Value condition="Equals"
datatype="hexBinary">E92A4FC283EB2802AD6D0E24C7FCC857</common:Simple_Hash_Value>
                                                                </common:Hash>
                                                        </FileObj:Hashes>
                                                </cybox:Defined_Object>
                                        </cybox:Associated_Object>
                                </cybox:Associated_Objects>
                        </cybox:Action>
                </cybox:Actions>
        </cybox:Event>
</cybox:Observable>

<cybox:Observable id="cybox:guid-f005fbc6-7427-43ea-8e1e-9a341836f76b">
        <!-- Download Iran-Oil invalid .mp4 downloader file-->
        <cybox:Event type="File Ops (CRUD)">
                <cybox:Description>
                        <common:Text>Download Iran-Oil invalid .mp4 downloader
file.</common:Text>
                </cybox:Description>
                <cybox:Actions>
                        <cybox:Action type="Download">
                                <cybox:Associated_Objects>
                                        <cybox:Associated_Object idref="cybox:guid-49d31c13-
8d7b-4528-b8d6-ce8ed0d43ad7" type="File" association_type="Initiating"/>
                                        <cybox:Associated_Object id="cybox:guid-8b463e0d-
cc16-4036-950e-5eeb09bc51aa" type="File" association_type="Affected">
                                                <!-- Iran-Oil invalid .mp4 downloader file-->
                                                <cybox:Description>
                                                        <common:Text>This mp4 file causes
memory corruption and code execution via heap-spraying code injection.</common:Text>
                                                </cybox:Description>
                                                <cybox:Defined_Object
xsi:type="FileObj:FileObjectType">
                                                        <FileObj:File_Name
datatype="String">test.mp4</FileObj:File_Name>
                                                        <FileObj:Size_In_Bytes
datatype="UnsignedLong">22384</FileObj:Size_In_Bytes>
                                                        <FileObj:Hashes>
                                                                <common:Hash>
                                                                        <common:Type
datatype="String">MD5</common:Type>

        <common:Simple_Hash_Value condition="Equals"
datatype="hexBinary">8933598C8B1FA5E493497B11C48DA4F2</common:Simple_Hash_Value>
                                                                </common:Hash>
```

```
                                                    </FileObj:Hashes>
                                                </cybox:Defined_Object>
                                                <cybox:Related_Objects>
                                                    <cybox:Related_Object
idref="cybox:guid-49d31c13-8d7b-4528-b8d6-ce8ed0d43ad7" type="File"
relationship="Downloaded_By"/>
                                                    <cybox:Related_Object
idref="cybox:guid-61041b8b-0c15-48a0-ac5f-b49488788010" type="URI"
relationship="Downloaded_From"/>
                                                </cybox:Related_Objects>
                                            </cybox:Associated_Object>
                                            <cybox:Associated_Object id="cybox:guid-61041b8b-
0c15-48a0-ac5f-b49488788010" type="URI" association_type="Utilized">
                                                <!-- URL from which malicious .mp4 file was
downloaded-->
                                                <cybox:Defined_Object
xsi:type="URIObj:URIObjectType" type="URL">
                                                    <URIObj:Value datatype="AnyURI"
condition="Equals">http://208.115.230.76/test.mp4</URIObj:Value>
                                                </cybox:Defined_Object>
                                            </cybox:Associated_Object>
                                        </cybox:Associated_Objects>
                                    </cybox:Action>
                                </cybox:Actions>
                            </cybox:Event>
                        </cybox:Observable>

                        <cybox:Observable id="cybox:guid-210f18f3-3874-4f9a-861d-71b328be90c6">
                            <!-- Create Iran-Oil .exe Trojan file-->
                            <cybox:Event type="File Ops (CRUD)">
                                <cybox:Description>
                                    <common:Text_Title>Create Iran-Oil .exe Trojan
file.</common:Text_Title>
                                </cybox:Description>
                                <cybox:Actions>
                                    <cybox:Action type="Create">
                                        <cybox:Associated_Objects>
                                            <cybox:Associated_Object idref="cybox:guid-8b463e0d-
cc16-4036-950e-5eeb09bc51aa" type="File" association_type="Initiating"/>
                                            <cybox:Associated_Object id="cybox:guid-b7e0bc39-
f519-4878-8fb0-5902554efe1c" type="File" association_type="Affected">
                                                <cybox:Description>
                                                    <common:Text>The file (us.exe MD5:
FD1BE09E499E8E380424B3835FC973A8 4861440 bytes) is created in the logged in user %Temp%
directory. The size of the embedded file is 22.5 KB (23040 bytes) and the size of the created us.exe is
4.63MB. It is an odd discrepancy until you look at the file and it looks like the code is repeated over and
over - 211 times. The file resource section indicates the file is meant to look like a java updater, which is
always larger than 22.5KB and that would explain all this padding, which is done at the time when the file
is being written to the disk.</common:Text>
                                                </cybox:Description>
                                                <cybox:Defined_Object
xsi:type="FileObj:FileObjectType">
                                                    <FileObj:File_Name
datatype="String">us.exe</FileObj:File_Name>
                                                    <FileObj:File_Path
datatype="String">%Temp%</FileObj:File_Path>
```

```
                                                <FileObj:Size_In_Bytes
datatype="UnsignedLong">4861440</FileObj:Size_In_Bytes>
                                                <FileObj:Hashes>
                                                        <common:Hash>
                                                                <common:Type
datatype="String">MD5</common:Type>

                <common:Simple_Hash_Value condition="Equals"
datatype="hexBinary">FD1BE09E499E8E380424B3835FC973A8</common:Simple_Hash_Value>
                                                        </common:Hash>
                                                </FileObj:Hashes>
                                        </cybox:Defined_Object>
                                        <cybox:Related_Objects>
                                                <cybox:Related_Object
idref="cybox:guid-8b463e0d-cc16-4036-950e-5eeb09bc51aa" type="File" relationship="Created_By"/>
                                                <!-- The trojan connects to the following
set of URLs/IPs for C&C -->
                                                <cybox:Related_Object
idref="cybox:guid-41b220d8-4c45-48de-9d08-30d661b2dc8e" type="URI" relationship="Connected_To"/>
                                                <cybox:Related_Object
idref="cybox:guid-61aa225b-90ef-415c-8bbd-a17282e457c9" type="IP Address"
relationship="Connected_To"/>
                                                <cybox:Related_Object
idref="cybox:guid-568db11e-39ee-43d7-83d8-032bdec3801a" type="URI" relationship="Connected_To"/>
                                                <cybox:Related_Object
idref="cybox:guid-80bea4d1-0e70-4a03-a54f-e40373bf94f1" type="IP Address"
relationship="Connected_To"/>
                                                <cybox:Related_Object
idref="cybox:guid-af7cb3b6-d70b-4b3b-b24f-7cfad739710f" type="URI" relationship="Connected_To"/>
                                                <cybox:Related_Object
idref="cybox:guid-5ceb9d54-24e2-4627-948d-6b92ac81962a" type="IP Address"
relationship="Connected_To"/>
                                        </cybox:Related_Objects>
                                </cybox:Associated_Object>
                        </cybox:Associated_Objects>
                </cybox:Action>
        </cybox:Actions>
</cybox:Event>
</cybox:Observable>

<cybox:Observable id="cybox:guid-b650c988-aac7-45ff-967d-9f1e5fc66161">
        <!-- Execute Iran-Oil .exe Trojan file-->
        <cybox:Event type="File Ops (CRUD)">
                <cybox:Description>
                        <common:Text>Execute Iran-Oil .exe Trojan file.</common:Text>
                </cybox:Description>
                <cybox:Actions>
                        <cybox:Action type="Execute">
                                <cybox:Associated_Objects>
                                        <cybox:Associated_Object idref="cybox:guid-8b463e0d-
cc16-4036-950e-5eeb09bc51aa" type="File" association_type="Initiating"/>
                                        <cybox:Associated_Object idref="cybox:guid-b7e0bc39-
f519-4878-8fb0-5902554efe1c" type="File" association_type="Affected"/>
                                </cybox:Associated_Objects>
                        </cybox:Action>
                </cybox:Actions>
```

```
                </cybox:Event>
        </cybox:Observable>


        <cybox:Observable id="cybox:guid-dee72b3e-82fb-4319-bfcc-007e3cf930e8">
                <!-- Iran-Oil core embedded .exe Trojan file-->
                <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:guid-bed1ff22-08e8-4e04-b7ac-908b5271176f"
type="File">
                                <cybox:Defined_Object xsi:type="FileObj:FileObjectType">
                                        <FileObj:File_Name datatype="String">us-
embedded.exe</FileObj:File_Name>
                                        <FileObj:Size_In_Bytes
datatype="UnsignedLong">23040</FileObj:Size_In_Bytes>
                                        <FileObj:Hashes>
                                                <common:Hash>
                                                        <common:Type
datatype="String">MD5</common:Type>
                                                        <common:Simple_Hash_Value
condition="Equals"
datatype="hexBinary">CB3DCDE34FD9FF0E19381D99B02F9692</common:Simple_Hash_Value>
                                                </common:Hash>
                                        </FileObj:Hashes>
                                </cybox:Defined_Object>
                                <cybox:Related_Objects>
                                        <cybox:Related_Object idref="cybox:guid-b7e0bc39-f519-4878-
8fb0-5902554efe1c" type="File" relationship="Contained_Within"/>
                                </cybox:Related_Objects>
                        </cybox:Object>
                </cybox:Stateful_Measure>
        </cybox:Observable>

        <cybox:Observable id="cybox:guid-a24ff8bc-b534-4616-838b-8bbe260a8e8f">
                <!-- Trojan .exe file connects out to C&C URLs/IPs-->
                <cybox:Event type="App Layer Traffic">
                        <cybox:Description>
                                <common:Text>Trojan .exe file connects out to C2
URLs/IPs.</common:Text>
                        </cybox:Description>
                        <cybox:Actions>
                                <cybox:Action type="Connect">
                                        <cybox:Associated_Objects>
                                                <cybox:Associated_Object idref="cybox:guid-b7e0bc39-
f519-4878-8fb0-5902554efe1c" type="File" association_type="Initiating"/>
                                                <cybox:Associated_Object idref="cybox:guid-41b220d8-
4c45-48de-9d08-30d661b2dc8e" type="URI" association_type="Utilized"/>
                                                <cybox:Associated_Object idref="cybox:guid-61aa225b-
90ef-415c-8bbd-a17282e457c9" type="IP Address" association_type="Utilized"/>
                                                <cybox:Associated_Object idref="cybox:guid-568db11e-
39ee-43d7-83d8-032bdec3801a" type="URI" association_type="Utilized"/>
                                                <cybox:Associated_Object idref="cybox:guid-80bea4d1-
0e70-4a03-a54f-e40373bf94f1" type="IP Address" association_type="Utilized"/>
                                                <cybox:Associated_Object idref="cybox:guid-af7cb3b6-
d70b-4b3b-b24f-7cfad739710f" type="URI" association_type="Utilized"/>
                                                <cybox:Associated_Object idref="cybox:guid-5ceb9d54-
24e2-4627-948d-6b92ac81962a" type="IP Address" association_type="Utilized"/>
```

```
                                        </cybox:Associated_Objects>
                                </cybox:Action>
                        </cybox:Actions>
                </cybox:Event>
        </cybox:Observable>

        <!-- The next six Observables represent the 3 different URL/IP pairs of C&C servers that the
trojan communicates with-->
        <cybox:Observable id="cybox:guid-066cef51-c886-432e-9a22-a17f57f3f3f2">
                <!-- One of three Command and Control URLs-->
                <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:guid-41b220d8-4c45-48de-9d08-30d661b2dc8e"
type="URI">
                                <cybox:Defined_Object xsi:type="URIObj:URIObjectType" type="URL">
                                        <URIObj:Value datatype="AnyURI"
condition="Equals">www.documents.myPicture.info</URIObj:Value>
                                </cybox:Defined_Object>
                                <cybox:Related_Objects>
                                        <cybox:Related_Object idref="cybox:guid-61aa225b-90ef-415c-
8bbd-a17282e457c9" type="IP Address" relationship="Resolved_To"/>
                                </cybox:Related_Objects>
                        </cybox:Object>
                </cybox:Stateful_Measure>
        </cybox:Observable>
        <cybox:Observable id="cybox:guid-4e05804c-f552-44e1-9793-ff4bb7f88f9c">
                <!-- One of three Command and Control IPs-->
                <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:guid-61aa225b-90ef-415c-8bbd-a17282e457c9"
type="IP Address">
                                <cybox:Defined_Object xsi:type="AddrObj:AddressObjectType"
category="ipv4-addr">
                                        <AddrObj:Address_Value datatype="String"
condition="Equals">199.192.156.134</AddrObj:Address_Value>
                                </cybox:Defined_Object>
                        </cybox:Object>
                </cybox:Stateful_Measure>
        </cybox:Observable>
        <cybox:Observable id="cybox:guid-75ce59ad-1f01-4eae-9ecc-0b22c4c24ce7">
                <!-- One of three Command and Control URLs-->
                <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:guid-568db11e-39ee-43d7-83d8-032bdec3801a"
type="URI">
                                <cybox:Defined_Object xsi:type="URIObj:URIObjectType" type="URL">
                                        <URIObj:Value datatype="AnyURI"
condition="Equals">documents.myPicture.info</URIObj:Value>
                                </cybox:Defined_Object>
                                <cybox:Related_Objects>
                                        <cybox:Related_Object idref="cybox:guid-80bea4d1-0e70-4a03-
a54f-e40373bf94f1" type="IP Address" relationship="Resolved_To"/>
                                </cybox:Related_Objects>
                        </cybox:Object>
                </cybox:Stateful_Measure>
        </cybox:Observable>
        <cybox:Observable id="cybox:guid-1ea53b14-8fe9-467b-a298-62d9684e797d">
                <!-- One of three Command and Control IPs-->
                <cybox:Stateful_Measure>
```

```
                        <cybox:Object id="cybox:guid-80bea4d1-0e70-4a03-a54f-e40373bf94f1"
type="IP Address">
                            <cybox:Defined_Object xsi:type="AddrObj:AddressObjectType"
category="ipv4-addr">
                                <AddrObj:Address_Value datatype="String"
condition="Equals">199.192.156.134</AddrObj:Address_Value>
                            </cybox:Defined_Object>
                        </cybox:Object>
                    </cybox:Stateful_Measure>
                </cybox:Observable>
                <cybox:Observable id="cybox:guid-f6c8ee75-ee7e-4490-bd5d-0661d0db7264">
                    <!-- One of three Command and Control URLs-->
                    <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:guid-af7cb3b6-d70b-4b3b-b24f-7cfad739710f"
type="URI">
                            <cybox:Defined_Object xsi:type="URIObj:URIObjectType" type="URL">
                                <URIObj:Value datatype="AnyURI"
condition="Equals">ftp.documents.myPicture.info</URIObj:Value>
                            </cybox:Defined_Object>
                            <cybox:Related_Objects>
                                <cybox:Related_Object idref="cybox:guid-5ceb9d54-24e2-4627-
948d-6b92ac81962a" type="IP Address" relationship="Resolved_To"/>
                            </cybox:Related_Objects>
                        </cybox:Object>
                    </cybox:Stateful_Measure>
                </cybox:Observable>
                <cybox:Observable id="cybox:guid-c78c0a83-6d14-45f8-827f-f758f0cd11ea">
                    <!-- One of three Command and Control IPs-->
                    <cybox:Stateful_Measure>
                        <cybox:Object id="cybox:guid-5ceb9d54-24e2-4627-948d-6b92ac81962a"
type="IP Address">
                            <cybox:Defined_Object xsi:type="AddrObj:AddressObjectType"
category="ipv4-addr">
                                <AddrObj:Address_Value datatype="String"
condition="Equals">199.192.156.134</AddrObj:Address_Value>
                            </cybox:Defined_Object>
                        </cybox:Object>
                    </cybox:Stateful_Measure>
                </cybox:Observable>


                <cybox:Observable id="cybox:guid-47d6a950-884d-46b5-9938-ac5555065a81">
                    <!-- This composed observable defines a pattern that is true if the receive email event
occurs AND the create malicious .doc file event occurs AND the download the downloader .mp4 file event
occurs AND the create trojan .exe file event occurs AND the execute trojan .exe file event occurs AND
the connect to all three of the C&C URLs/IPs event occurs-->
                    <!-- This yields a very tight filter that will have very low false positives but could miss
almost any variation of the attack elements-->
                    <cybox:Observable_Composition operator="AND">
                        <!-- Receive "Iran-Oil" attack campaign email message -->
                        <cybox:Observable idref="cybox:guid-1a937ec2-90ab-4e0e-a37c-
db9b2e66a58e"/>
                        <!-- Open Iran-Oil corrupted .doc file-->
                        <cybox:Observable idref="cybox:guid-35f04c28-5fd2-4d72-8aae-
2ad04ee1811f"/>
                        <!-- Download Iran-Oil invalid .mp4 downloader file-->
```

```
                    <cybox:Observable idref="cybox:guid-f005fbc6-7427-43ea-8e1e-
9a341836f76b"/>
                    <!-- Create Iran-Oil .exe Trojan file-->
                    <cybox:Observable idref="cybox:guid-210f18f3-3874-4f9a-861d-
71b328be90c6"/>
                    <!-- Execute Iran-Oil .exe Trojan file-->
                    <cybox:Observable idref="cybox:guid-b650c988-aac7-45ff-967d-9f1e5fc66161"/>
                    <!-- Trojan .exe file connects out to C&C URLs/IPs-->
                    <cybox:Observable idref="cybox:guid-a24ff8bc-b534-4616-838b-
8bbe260a8e8f"/>
                </cybox:Observable_Composition>
            </cybox:Observable>

            <cybox:Observable id="cybox:guid-80594430-7567-4402-88a4-05d556b21884">
                <!-- This composed observable defines a pattern that is true if the receive email event
occurs OR the create malicious .doc file event occurs OR the download the downloader .mp4 file event
occurs OR the create trojan .exe file event occurs OR the execute trojan .exe file event occurs OR the
connect to all three of the C&C URLs/IPs event occurs-->
                <!-- This yields a very loose filter that could have false positives but could catch
numerous potential variations of the attack elements-->
                <cybox:Observable_Composition operator="OR">
                    <!-- Receive "Iran-Oil" attack campaign email message -->
                    <cybox:Observable idref="cybox:guid-1a937ec2-90ab-4e0e-a37c-
db9b2e66a58e"/>
                    <!-- Open Iran-Oil corrupted .doc file-->
                    <cybox:Observable idref="cybox:guid-35f04c28-5fd2-4d72-8aae-
2ad04ee1811f"/>
                    <!-- Download Iran-Oil invalid .mp4 downloader file-->
                    <cybox:Observable idref="cybox:guid-f005fbc6-7427-43ea-8e1e-
9a341836f76b"/>
                    <!-- Create Iran-Oil .exe Trojan file-->
                    <cybox:Observable idref="cybox:guid-210f18f3-3874-4f9a-861d-
71b328be90c6"/>
                    <!-- Execute Iran-Oil .exe Trojan file-->
                    <cybox:Observable idref="cybox:guid-b650c988-aac7-45ff-967d-9f1e5fc66161"/>
                    <!-- Trojan .exe file connects out to C&C URLs/IPs-->
                    <cybox:Observable idref="cybox:guid-a24ff8bc-b534-4616-838b-
8bbe260a8e8f"/>
                </cybox:Observable_Composition>
            </cybox:Observable>

            <cybox:Observable id="cybox:guid-7d932074-fded-4056-870e-dd51980501d4">
                <!-- This composed observable defines a pattern that is true if (the receive email event
occurs AND the create malicious .doc file event occurs) OR (the download the downloader .mp4 file event
occurs AND the create trojan .exe file event occurs AND the execute trojan .exe file event occurs) OR the
connect to all three of the C&C URLs/IPs event occurs-->
                <cybox:Observable_Composition operator="OR">
                    <cybox:Observable><cybox:Observable_Composition operator="AND">
                        <!-- Receive "Iran-Oil" attack campaign email message -->
                        <cybox:Observable idref="cybox:guid-1a937ec2-90ab-4e0e-a37c-
db9b2e66a58e"/>
                        <!-- Open Iran-Oil corrupted .doc file-->
                        <cybox:Observable idref="cybox:guid-35f04c28-5fd2-4d72-8aae-
2ad04ee1811f"/>
                    </cybox:Observable_Composition></cybox:Observable>
                    <cybox:Observable><cybox:Observable_Composition operator="AND">
```

```
                        <!-- Download Iran-Oil invalid .mp4 downloader file-->
                        <cybox:Observable idref="cybox:guid-f005fbc6-7427-43ea-8e1e-
9a341836f76b"/>
                        <!-- Create Iran-Oil .exe Trojan file-->
                        <cybox:Observable idref="cybox:guid-210f18f3-3874-4f9a-861d-
71b328be90c6"/>
                        <!-- Execute Iran-Oil .exe Trojan file-->
                        <cybox:Observable idref="cybox:guid-b650c988-aac7-45ff-967d-
9f1e5fc66161"/>
                    </cybox:Observable_Composition></cybox:Observable>
                    <!-- Trojan .exe file connects out to C&C URLs/IPs-->
                    <cybox:Observable idref="cybox:guid-a24ff8bc-b534-4616-838b-
8bbe260a8e8f"/>
                </cybox:Observable_Composition>
        </cybox:Observable>

        <!-- CybOX enables a wide myriad of other potential observable pattern variations at the logical
composition level or utilizing patterns at the Object attribute level including Regex all of which allow the
user to define an almost infinitely variable set of patterns and filters -->
</cybox:Observables>
```

Appendix A. Leveraging the CybOX Language Data Model

There are two primary modes for leveraging the CybOX language to define cyber observable content: directly and indirectly.

- Directly leveraging the CybOX language involves simply leveraging a schematic implementation of the language to capture and utilize content.
- Indirectly leveraging the CybOX language involves leveraging a domain-specific language, standard, process or tool which within its own structure imports or includes elements of the CybOX language. Any domain-specific language, standard, process or tool is free to incorporate any relevant portions of the CybOX language via importing or including the appropriate data model types as instantiated in a schematic implementation of the language (e.g. using XML Schema).

  For example:

  - The Common Attack Pattern Enumeration and Classification (CAPEC) can import the entire CybOX language XML Schema implementation from the ObservablesType on down.
  - The Malware Attribute Enumeration and Characterization (MAEC) can import just the CybOX ActionType & ObjectType (along with portions of the CybOX library of common defined objects) to utlize as the foundation of its malware characterization.
  - The Common Event Expression (CEE) can align with and import the CybOX EventType to serve as its broad scope structure for characterizing cyber events.

Appendix B. Extending the CybOX Language Data Model

The CybOX Language Data Model defines a set of core capabilities, as described within this Specification document and the accompanying CybOX Language Defined Objects Specification, with numerous extension points. This appendix highlights the opportunities for extension within the CybOX Language. It is particulary important to understand the role of CybOX Defined Object Models within the CybOX Language, as they form a large basis of cyber observable expression and allow CybOX to easily expand to cover new object types or new levels of detailed characterization of existing object types. Additionally, this appendix will raise awareness of some other extension points that have been built into the CybOX Language.

CybOX Defined Object Models

The primary foundation of the cyber observables construct lies in the set of observable objects that exist as stateful measures or are involved in observable actions and events. As such, any language like CybOX providing a practical solution for characterizing cyber observables must include the capability to describe a set of commonly observed objects utilizing a common set of attributes for any given object type. The diversity of the cyber domain however makes such a set of potential objects very large with new objects coming into play over time and differing use cases requiring different objects. To provide effective capability to a diverse set of use cases a cyber observable expression language like CybOX must provide a common library of defined object models for unrestricted use but also must incorporate them into the language in a way that makes it easy for new objects to be added. It must support the addition of new objects by domain-specific use cases independent of CybOX as well as the addition of new objects to the CybOX common library without affecting the rest of the CybOX language as the defined object portions of the language are the likely to experience the highest rate of change over time.

In the CybOX language, these defined object structures are defined in their own Models as described in the Data Model section of the accompanying CybOX Language Defined Objects Specification. The CybOX Defined Object Data Models each provide the necessary constructs for characterizing a comprehensive set of commonly leveraged attributes for any given defined object type. Where possible and appropriate the structure and syntax of these models or portions thereof adhere to relevant existing normative specifications. Due to the nature of uniquely comprehensive coverage of the CybOX language and its targeted support of a broad range of use cases, there exist several instances where the CybOX data models diverge from existing normative specifications through extension, aggregation, restriction or abstraction.

To ensure flexibility and extensibility all defined objects are incorporated into the CybOX language as extensions of the abstract DefinedObjectType which acts as a generalized placeholder in the language for context-specific structures and syntax of the various potential defined object types. Through this mechanism new defined object types can be created or existing types modified with no effect on the core CyboX language or any other non-dependent defined object type. Similarly, any domain-specific use case could create their own new defined object types as extensions of the abstract DefinedObjectType and use them in localized content. Sharing this data with any entities outside their scope may result in a limited ability to parse of validate content for that object type (unless the

appropriate model is also shared) but all other portions of the CybOX language should work without issue. Over time, independently created defined object models will be reviewed and, if appropriate, incorporated into the CybOX common defined object library.

The CybOX library of defined object models is designed in an intentionally architected and modular fashion such that more complex or specialized objects can leverage and incorporate existing objects where appropriate. The two most common situations for this sort of incorporation are:

1. Defined objects which require attributes that are themselves more atomic-level defined objects.

   For example, the DNSRecordObjectType could make use of the AddressObjectType and the URIObjectType to describe its associated IP_Address and Domain_Name attributes.

2. Defined objects that are specializations sharing a significant basis with other defined objects.

   For example, the WindowsExecutableFileObjectType could be an extension of the WindowsFileObjectType adding PE-specific attributes and the WindowsFileObjectType could further be an extension of a basic FileObjectType adding Windows specific attributes to the general set of attributes shared by all files.

Other Abstract Types

The same abstract type approach described above for the DefinedObjectType is also leveraged by the CybOX language to enable other points of generalized extension. A short list of these other extension points includes:

- BaseObjectAttributeType

  The BaseObjectAttributeType is an abstract type that acts as a basis for all atomic-level object attribute types and provides the basic capabilities for pattern characterization for a given object attribute. There are a range of extensions of this abstract type provided in the CybOX language for a variety of primitive data types. All leaf attributes for CybOX objects should be of types defined using extensions from the BaseObjectAttributeType.

- DomainSpecificObjectAttributeType

  The DomainSpecificObjectAttributesType is an abstract type placeholder within the CybOX language enabling the inclusion of domain-specific metadata for an object through the use of a custom type defined as an extension of this base abstract type. This enables domains utilizing CybOX such as malware analysis or forensics to incorporate non-generalized object metadata from their domains into CybOX objects.

- DefinedEffectType

  The DefinedEffectType is an abstract placeholder for various predefined Object Effect types (e.g. DataReadEffect, ValuesEnumeratedEffect or StateChangeEffect) that can be instantiated in its place through extension of the DefinedEffectType. This mechanism enables the specification of a broad range of types of potential complex action effects on Objects. The

set of Defined Effect types (extending the DefinedEffectType) are maintained as part of the core CybOX language.

- PersonnelType

  The PersonnelType is an abstracted data type to standardize the description of sets of personnel.

- ToolSpecificDataType

  The ToolSpecificDataType is an abstract type placeholder within the CybOX language enabling the inclusion of metadata for a specific type of tool through the use of a custom type defined as an extension of this base abstract type.

- IndicatorType

  The IndicatorType is an abstract type placeholder within the CybOX language enabling the inclusion of varying specifications for indicators contributing to this cyber observation. Externally defined indicator structures can be defined through the use of a custom type defined as an extension of this base abstract type.

- **FileObjectType**
  - FileAttributeType

    The FileAttributeType type specifies a native attribute of a file. Since native attributes are platform-specific, it is defined here as an abstract type.

  - FilePermissionsType

    The FilePermissionsType specifies the native permissions of a file. Since this is a platform-specific attribute, it is defined here as an abstract type and then implemented in any platform-specific derived CybOX file objects.

- **ProcessObjectType**
  - ProcessStatusType

    The ProcessStatusType is used for specifying the status of a running or terminated process. Since this property is platform-specific, it is created here as an abstract type and then used in the platform-specific process CybOX objects.

- **UserAccountObjectType**
  - PrivilegeType

    The PrivilegeType specifies a specific privilege that a user has. This is an abstract type since user privileges are OS-specific, and is extended as needed in the derived CybOX objects.

  - GroupType

The GroupType specifies a group that a user account belongs to. This is an abstract type since group IDs are OS-specific, and is extended as needed in the derived CybOX objects.

- **VolumeObjectType**
  - VolumeOptionsType

    The VolumeOptionsType specifies the particular options set for the volume. This is an abstract type since volume options are OS-specific, and is extended by the related OS-specific CybOX volume objects.

Generalized Extension Mechanisms

To support domain-specific attribute adornment on key components, CybOX provides an open attribute wildcard extension mechanism as part of DefinedObjectType, ActionType and ObjectType.

CybOX provides a generalized data structure named MetadataType that can be used to capture any sort of custom metadata structure via a field/value tuple and recursion.

Fundamental Extension

The most basic, simple and broadly applicable extension mechanism is via domain-specific extension of any of the modular and layered set of native CybOX types.

Appendix C. Normative References

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

[2] W3C Recommendation for Base64-Encoded Binary Data
http://www.w3.org/TR/xmlschema-2/#base64Binary

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

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

[5] W3C Recommendation for Unsigned Integer Data
http://www.w3.org/TR/xmlschema-2/#unsignedInt

[6] W3C Recommendation for Non-Negative Integer Data
http://www.w3.org/TR/xmlschema-2/#nonNegativeInteger

[7] W3C Recommendation for Positive Integer Data
http://www.w3.org/TR/xmlschema-2/#positiveInteger

[8] W3C Recommendation for Long Data
http://www.w3.org/TR/xmlschema-2/#long

[9] W3C Recommendation for Unsigned Long Data
http://www.w3.org/TR/xmlschema-2/#unsignedLong

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

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

[12] W3C Recommendation for Time Data
http://www.w3.org/TR/xmlschema-2/#time

[13] W3C Recommendation for Date Data
http://www.w3.org/TR/xmlschema-2/#date

[14] W3C Recommendation for DateTime Data
http://www.w3.org/TR/xmlschema-2/#dateTime

[15] W3C Recommendation for Duration Data
http://www.w3.org/TR/xmlschema-2/#duration

[16] W3C Recommendation for String Data

http://www.w3.org/TR/xmlSchema-2/#string

[17] W3C Recommendation for QName Data
http://www.w3.org/TR/xmlschema-2/#QName

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

RFC 2461: Neighbor Discovery for IP Version 6 (IPv6)
http://www.ietf.org/rfc/rfc2461.txt

RFC 4861: Neighbor Discovery for IP Version 6 (IPv6)
http://www.ietf.org/rfc/rfc4861.txt

RFC 791: Internet Protocol
http://www.ietf.org/rfc/rfc791.txt

RFC 2474: Differentiated Services Field
http://www.ietf.org/rfc/rfc2474.txt

RFC 3168: Explicit Congestion Notification
http://www.ietf.org/rfc/rfc3168.txt

RFC 3692: Experimental and Testing Numbers
http://www.ietf.org/rfc/rfc3692.txt

RFC 3513: Internet Protocol Version 6 (IPv6) Addressing Architecture
http://www.ietf.org/rfc/rfc3513.txt

RFC 2460: Internet Protocol Version 6 (IPv6) Specification
http://www.ietf.org/rfc/rfc2460.txt

RFC 2402: IP Authentication Header
http://www.ietf.org/rfc/rfc2402.txt

RFC 2406: IP Encapsulating Security Payload (ESP)
http://www.ietf.org/rfc/rfc2406.txt

RFC 1347: TCP and UDP with Bigger Addresses (TUBA)
http://www.ietf.org/rfc/rfc1347.txt

RFC 4443: Internet Control Message Protocol (ICMPv6) for IPv6 Specification
http://www.ietf.org/rfc/rfc4443.txt

RFC 2463: Internet Control Message Protocol (ICMPv6) for IPv6 Specification
http://www.ietf.org/rfc/rfc2463.txt

RFC 768: User Datagram Protocol
http://www.ietf.org/rfc/rfc768.txt

RFC 791: Internet Protocol

http://www.ietf.org/rfc/rfc791.txt

RFC 792: Internet Control Message Protocol (ICMP)
http://www.ietf.org/rfc/rfc792.txt

RFC 793: Transmission Control Protocol
http://www.ietf.org/rfc/rfc793.txt

RFC 826: Ethernet Address Resolution Protocol
http://tools.ietf.org/html/rfc826

RFC 903: A Reverse Address Resolution Protocol
http://www.ietf.org/rfc/rfc903.txt

RFC 1219: On the Assignment of Subnet Numbers
http://www.ietf.org/rfc/rfc1219.txt

RFC 1349: Type of Service in the Internet Protocol Suite
http://www.ietf.org/rfc/rfc1349.txt

RFC 5101: Specification of the IPFIX Protocol
http://www.ietf.org/rfc/rfc5101.txt

RFC 5102: Information Model for IP Flow Information Export
http://www.ietf.org/rfc/rfc5102.txt

RFC 3954: Cisco Systems Netflow Services Export Version 9
http://www.ietf.org/rfc/rfc3954.txt

RFC 821: Simple Mail Transfer Protocol
http://www.ietf.org/rfc/rfc821.txt

RFC 2076:Common Internet Message Headers
http://www.ietf.org/rfc/rfc2076.txt

RFC 822: Standard for the Format of Arpa Internet Text Messages
http://www.ietf.org/rfc/rfc822.txt

RFC 2822: Internet Message Format
http://www.ietf.org/rfc/rfc2822.txt

RFC 1035: Domain Names – Implementation and Specification
http://www.ietf.org/rfc/rfc1035.txt

RFC 3597: Handling of Unknown DNS Resource Record (RR) Types
http://www.ietf.org/rfc/rfc3597.txt

RFC 1058: Routing Information Protocol
http://www.ietf.org/rfc/rfc1058.txt

RFC 147: The Definition of a Socket

http://www.ietf.org/rfc/rfc147.txt

RFC 1122: Requirements for Internet Hosts –Communication Layers
http://www.ietf.org/rfc/rfc1122.txt

RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax
http://www.ietf.org/rfc/rfc2396.txt

RFC 1518: An Architecture for IP Address Allocation with CIDR
http://www.ietf.org/rfc/rfc1518.txt

RFC 5070: The Incident Object Description Exchange Format
http://www.ietf.org/rfc/rfc5070.txt

RFC 5901: Extensions to the IODEF-Document Class for Reporting Phishing
http://www.ietf.org/rfc/rfc5901.txt

X.509
http://www.itu.int/rec/T-REC-X.509/en

<Semaphore.h>
http://pubs.opengroup.org/onlinepubs/007904975/basedefs/semaphore.h.html

<Socket.h>
http://pubs.opengroup.org/onlinepubs/009695399/basedefs/sys/socket.h.html

route(8) – Linux man page
http://linux.die.net/man/8/route

mount(8) – Linux man page
http://linux.die.net/man/8/mount

Event Schema
http://msdn.microsoft.com/en-us/library/aa385201(v=vs.85).aspx

Event Objects
http://msdn.microsoft.com/en-us/library/windows/desktop/ms682655(v=vs.85).aspx

Mutex Objects
http://msdn.microsoft.com/en-us/library/windows/desktop/ms684266(v=vs.85).aspx

Semaphore Objects
http://msdn.microsoft.com/en-us/library/windows/desktop/ms685129(v=vs.85).aspx

Waitable Timer Objects
http://msdn.microsoft.com/en-us/library/windows/desktop/ms687012(v=vs.85).aspx

An In-depth Look into the Win32 Portable Executable File Format
http://msdn.microsoft.com/en-us/magazine/cc301805.aspx

File Management Reference

http://msdn.microsoft.com/en-us/library/aa364233(v=vs.85).aspx

Mailslots
http://msdn.microsoft.com/en-us/library/windows/desktop/aa365576(v=vs.85).aspx

Network Share Functions
http://msdn.microsoft.com/en-us/library/windows/desktop/bb525391(v=vs.85).aspx

Named Pipes
http://msdn.microsoft.com/en-us/library/windows/desktop/aa365590(v=vs.85).aspx

Registry
http://msdn.microsoft.com/en-us/library/windows/desktop/ms724871(v=vs.85).aspx

Services
http://msdn.microsoft.com/en-us/library/windows/desktop/ms685141(v=vs.85).aspx

GFlags
http://msdn.microsoft.com/en-us/library/windows/hardware/ff549557(v=vs.85).aspx

System Restore
http://msdn.microsoft.com/en-us/library/windows/desktop/dd408121(v=vs.85).aspx

Task Scheduler
http://msdn.microsoft.com/en-us/library/windows/desktop/aa383614(v=vs.85).aspx

Process and Thread Reference
http://msdn.microsoft.com/en-us/library/windows/desktop/ms684852(v=vs.85).aspx

Volume Object
http://msdn.microsoft.com/en-us/library/windows/desktop/aa383970(v=vs.85).aspx

Memory Management
http://msdn.microsoft.com/en-us/library/windows/desktop/aa366779(v=vs.85).aspx

Task Scheduler
http://msdn.microsoft.com/en-us/library/windows/desktop/aa383614(v=vs.85).aspx

Windows Driver Development
http://msdn.microsoft.com/en-us/library/windows/hardware/ff557573(v=vs.85).aspx

Appendix D. Changelog

Appendix E. Acronyms

ACL	Access Control List
API	Application Programming Interface
APC	Asynchronous Procedure Calls
ARE	Advanced Regular Expression
ARP	Address Resolution Protocol
ARM	Acorn RISC Machine
API	Application Programming Interface
AS	Autonomous System
ASN	Autonomous System Number
ASP	Active Server Pages
ATM	Asynchronous Transfer Mode
BIOS	Basic Input/Output System
BCC	Blind Carbon Copy
BRE	Basic Regular Expression
CAPEC	Common Attack Pattern Enumeration and Classification
CC	Carbon Copy
CCE	Common Configuration Enumeration
CDS	Content Delivery System
CIDR	Classless Inter-Domain Routing
CLR	Common Language Runtime
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
CybOX	Cyber Observable eXpression
DBMS	Database Management System
DHCP	Dynamic Host Configuration Protocol
DHS	Department of Homeland Security
DLL	Dynamically Linked Library
DNS	Domain Name System
DST	Daylight Savings Time
ECMA	European Computer Manufacturers Association
EP	Entry Point
ERE	Extended Regular Expression
EVR	Epoch, version, and release
FTP	File Transfer Protocol
FQDN	Fully Qualified Domain Name
FQN	Fully Qualified Name
GNU	GNU's Not Unix!
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HIDS	Host Intrusion Detection System
HIPS	Host Intrusion Prevention System
HTML	HyperText Markup Language
HTTP	Hyper Text Transfer Protocol
IAVM	Information Assurance Vulnerability Management

ICMP	Internet Control Message Protocol
ID	Identifier
IDT	Interrupt Descriptor Table
IETF	Internet Engineering Task Force
IMAP	Internet Message Access Protocol
INODE	Index Node
IP	Internet Protocol
IPFIX	Internet Protocol Flow Information Export
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IRP	Interrupt Request Packet
IPC	Inter-Process Communication
JSP	Java Server Pages
KVM	Keyboard Video Mouse
MAC	Media Access Control
MIB	Management Information Base
MIPS	Microprocessor without Interlocked Pipeline Stages
MUTEX	MUTual Exclusion
MSS	Maximum Segment Size
NAC	Network Access Control
NDP	Network Discovery Protocol
NETBEUI	NetBIOS Extended User Interface
NETBIOS	Network Basic Input/Ouput System
NIDS	Network Intrusion Detection System
NIPS	Network Intrustion Prevention System
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OS	Operating System
PCRE	Perl-Compatible Regular Expression
PE	Portable Executable
PEID	Portable Executable Identifier
PID	Process Identifier
POP	Post Office Protocol
POSIX	Portable Operating System Interface
PHP	PHP HyperText Processor
PPC	PowerPC
RARP	Reverse Address Resolution Protocol
RDF	Resource Description Framework
RFC	Request For Comment
RISC	Reduced Instruction Set
RSA	Ron Rivest, Adi Shamir, and Leonard Adleman
RUID	Real User ID
RVA	Relative Virtual Address
SID	Security Identifier
SIM	Security Information Management
SMB	Server Message Block
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol

SO	Socket Option
SOAP	Simple Object Access Protocol
SPARC	Scalable Processor ARChitecture
SSDT	System Service Dispatch Table
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Thread Local Storage
TOS	Type of Service
TTL	Time To Live
UDP	User Datagram Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URN	Uniform Resource Name
USB	Universal Serial Bus
UUID	Universally Unique Identifier
VLAN	Virtual Lan
VM	Virtual Machine
W3C	World Wide Web Consortium
XOR	Exclusive OR
XML	eXtensible Markup Language
XSD	XML Schema Document