

INCH Working Group
INTERNET-DRAFT
Expires in six months

J.J. Meijer
SURFnet bv
R. Danyliw
CERT Coordination Center
Y. Demchenko
TERENA
April 2002

Incident Object Description and Exchange Format
Data Model and Extensible Markup Language (XML)
Document Type Definition
<[draft-meijer-inch-iodef-00.txt](#)>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

Distribution of this memo is unlimited.

This Internet Draft expires October, 2002.

Copyright Notice

Copyright (C) The Internet Society (2001). All Rights Reserved.

Abstract

The purpose of the Incident Object Description and Exchange Format is to define a common data format for describing and exchanging incident

information between collaborating Computer Security Incident Response Teams (CSIRTs). The specific goals and requirements of the IODEF are described in [2]. One of the design principles in the IODEF is compatibility with the Intrusion Detection Message Exchange Format (IDMEF) [3] developed for intrusion detection systems. For this reason, IODEF is heavily based on the IDMEF and provides upward compatibility with it.

This document describes a data model for representing information produced by incident handling systems managing security incident data, and explains the rationale for using this model. An implementation of the data model in the Extensible Markup Language (XML) is presented, an XML Document Type Definition is developed, and examples are provided.

TABLE OF CONTENTS

1.	Conventions Used in This Document.....	4
2.	Introduction	4
2.1	About the IODEF Data Model	5
2.1.1	Problems Addressed by the Data Model	6
2.1.2	Data Model Design Goals	7
2.2	About the IODEF XML Implementation	7
2.3	Relation between IODEF and IDMEF	9
3.	Notational Conventions and Formatting Issues	9
3.1	UML Conventions used for Data Model Description	9
3.1.1	Relationships.....	10
3.1.2	Occurrence Indicators.....	11
3.2	XML Document Type Definitions	12
3.2.2	Element Declarations	12
3.2.2.1	Occurrence Indicators.....	13
3.2.2.2	Alternative Content and Grouping.....	13
3.2.2.3	Element Content.....	14
3.2.3	Attribute Declarations	15
3.2.3.1	Attribute Types.....	15
3.2.3.2	Attribute Content.....	16
3.2.4	Entity Declarations	16

3.3	XML Documents	17
3.3.1	The Document Prolog	17
3.3.1.1	XML Declaration	17
3.3.1.2	IODEF DTD Formal Public Identifier	18
3.3.1.3	IODEF DTD Document Type Declaration	18
3.3.2	Character Data Processing in XML and IODEF	19
3.3.2.1	Character Entity References.....	20
3.3.2.2	Character Code References.....	20
3.3.2.3	White Space Processing.....	21
3.3.3	Languages in XML and IODEF	21
3.3.4	Inheritance and Aggregation	22
3.4	IODEF Data Types	22
3.4.1	Integers	23
3.4.2	Real Numbers	23
3.4.3	Characters and Strings	23
3.4.4	Bytes	24

3.4.5	Enumerated Types	24
3.4.6	Date-Time Strings	24
3.4.7	NTP Timestamps	26
3.4.8	Port Lists	26
3.4.9	Unique Identifiers	27
3.4.10	Personal name.....	28
3.4.11	Organization name.....	28
3.4.12	Postal address.....	28
3.4.13	Telephone and Fax numbers.....	29
4.	The IODEF Data Model and XML DTD.....	29
4.1	Data Model Overview.....	29
4.2	The IODEF-Description Class.....	32
4.3	The Incident Class.....	33
4.4	The CorrelationIncident Class.....	36
4.5	The IncidentAlert Class.....	37
4.6	The Core Classes.....	38
4.6.1	The Attack Class.....	39
4.6.2	The Source Class.....	42
4.6.3	The Target Class.....	44
4.6.4	The Method Class.....	46
4.6.5	The Attacker Class.....	47
4.6.6	The Victim Class.....	48
4.6.7	The Evidence Class.....	50
4.6.8	The Assessment Class.....	51

4.6.8.1	The Impact Class.....	52
4.6.8.2	The Action Class.....	53
4.6.8.3	The Confidence Class.....	54
4.6.9	The Authority Class.....	56
4.6.10	The History Class.....	56
4.6.11	The AdditionalData Class.....	58
4.7	The Time Classes.....	59
4.7.1	The DetectTime Class.....	59
4.7.2	The StartTime Class.....	60
4.7.3	The EndTime Class.....	60
4.7.4	The DateTime Class.....	60
4.8	The Support Classes.....	61
4.8.1	The Node Class.....	61
4.8.1.1	The Address Class.....	63
4.8.1.2	The NodeRole Class.....	65
4.8.2	The User Class.....	66
4.8.2.1	The UserId Class.....	67
4.8.3	The Process Class.....	69
4.8.4	The Service Class.....	71
4.8.4.1	The WebService Class.....	72
4.8.4.2	The SNMPService Class.....	73
4.8.5	The Classification Class.....	74
4.8.6	The EvidenceData Class.....	76
4.8.6.1	The EvidenceDesc Class.....	77
4.8.6.2	The EventList Class.....	78

4.8.7	The Organization Class.....	79
4.8.8	The Contact Class.....	81
4.8.9	The Reported Class.....	82
4.8.10	The Received Class.....	83
4.8.11	The ActionList Class.....	85
4.8.12	The FileList Class.....	86
4.8.12.1	The File Class.....	86
4.8.12.2	The FileAccess Class.....	89
4.8.12.3	The Linkage Class.....	90
4.8.12.4	The Inode Class.....	92
4.8.13	The Analyzer Class.....	94
4.9	The Simple Classes.....	96
4.9.1	The Description Class.....	96
4.9.2	The IRTcontact Class.....	96
4.9.3	The EvidenceItem Class.....	97

4.9.4	The CorrEvidence Class.....	98
4.9.5	The Name Class.....	98
5.	Extending the IODEF	99
5.1	Extending the Data Model	99
5.2	Extending the XML DTD	99
6.	Special Considerations	101
6.1	XML Validity and Well-Formedness	102
6.2	Unrecognized XML Tags	102
6.3	Digital Signatures	103
7.	Examples	103
8.	The IODEF Document Type Definition	104
9.	References	119
10.	Security Considerations	120
11.	IANA Considerations	120
12.	Acknowledgements	120
13.	Authors' Addresses	121
14.	Full Copyright Statement	121

1. Conventions Used in This Document

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](#) [2].

Network and Computer Security related terminology used in this documents is of common use, however it contains some specific conventions described in [2] and [4].

2. Introduction

The Incident Object Description and Exchange Format (IODEF) is

intended to be a standard format for Computer Security Incident Response Teams (CSIRTs) to exchange operational and statistical incident information among themselves and their collaborators. It can also provide the basis for the development of interoperable tools and procedures for incident reporting.

By using IODEF in their workflow and incident handling system, a CSIRT can benefit from:

- + a single organizational data schema that can represent information from a variety of subordinate teams or CSIRTs;
- + a common incident data format that facilitates collaboration among affected members of the security community (e.g. users, vendors, response teams, law enforcement);
- + the simplification in building an incident correlation and statistics system that process incident reports from different CSIRTs.

One of the design principles of the IODEF is complete compatibility with the Intrusion Detection Message Exchange Format (IDMEF) [3] developed for intrusion detection systems. For this reason, IODEF is heavily based on the IDMEF and provides upward compatibility with it. IDMEF messages may be entirely encapsulated into an explicit IDMEF container provided in the IODEF data model. A goal of this version of the Internet Draft is to provide context to discuss compatibility issues between the IODEF and the IDMEF.

The IODEF description also intends to be capable of referencing relevant external computer security information (e.g., vulnerability and virus databases).

The computer security related terminology used in this document is described in [1] and [4]. Specific terminology, notation, and conventions related to the data model and XML DTD are presented in Sections 3 and 4. The data model is described in Section 5 with examples of its use in Section 8. Recognizing the potentially diverse user-base implementing IODEF, Section 6 discusses the ability to extend the model.

2.1 IODEF Data Model Design principles

The IODEF data model is an object-oriented representation of information reported and maintained by a CSIRT about a computer security incident.

2.1.1 Problems Addressed by the Data Model

The data model addresses several problems in representing incident description data:

- + Incident data is inherently heterogeneous. It may encompass many functional purposes such as a description of intruder behavior, a vulnerability report, or analysis results correlating related incidents. However, even in a single type of incident, seemingly disparate information from many sources may need to be represented.

This representation of the data is further complicated by the fact that incidents may consist of varying levels of detail depending on their stage in the lifecycle. For example, newly reported incidents may only contain a short description of the involved parties. On the other hand, closed incidents can contain a full description complete with the associated evidence and annotation of actions taken by the CSIRT. The data model that represents this information must be flexible to accommodate different needs.

An object-oriented model provides extensibility via aggregation and sub-classing while preserving the consistency of the model. If the data model required modification, it is extended with new classes. In implementations that do not recognize these extensions, the basic subset of the data model will still be understood.

In order to address the various types of incidents, the IODEF data model creates top-level classes for each of the different incident profiles. Just as another other extensions to the data model, creating new profiles is possible through sub-classing or aggregation based on the core and supportive classes.

- + From the purview of a CSIRT, incident information can originate from a number of sources.

The data model defines support classes that accommodate the differences in the incident reporter. This support includes various meta information to represent the reporter's identity as well as prescribe a confidence level to the submitted information.

- + Incidents may contain sensitive information. Such

during collaboration.

The data model allows for a highly granular level of element tagging to indication potential restrictions on the usage of the data. However, it is the role of the IHS to honor these labels.

2.1.2 Data Model Design Goals

The IODEF data model was designed to provide a standard representation of a computer security incident.

- + The design of the data model is content-driven. This design dictates that new objects are introduced to accommodate additional content, not semantic differences between incidents.
- + The data model must be unambiguous. Functionality similar incident data (e.g. attacking hostname) must populate the same elements of the schema. Likewise, the same incident described by different CSIRTs (potentially using different initial information) should be able to be identified as the same incident. This correlation should be possible in spite of descriptions having different levels of detail or the source and target being described from a different perspective.
- + In order to investigate an incident across multiple sites, aggregation of incident data from the responsible CSIRTs may be required. This data model provides the facility for logically groups related incidents. However, the methods and algorithms for performing this correlation are left to the IHS.
- + The data model was designed with the intention to be both human and machine-readable. This level of readability will allow IODEF to be incorporated into incident handling system used by CSIRTs.

- + The ability to seamlessly integration IDMEF documents was explicitly designed into the data model.

2.2 Using XML for IODEF Description

The current IODEF implementation is based on XML. As a meta-language, XML allows the definition of customized markup languages for different types of documents and different

Meijer, et al.

Expires October 2002

[page 7]

Internet Draft

[draft-ietf-inch-iodef-00.txt](#)

Apr 2002

applications. XML provides both the syntax for declaring document markup and structure (i.e., defining elements and attributes, specifying the order in which they appear, etc.) as well as, a syntax for using that markup in documents.

XML-based applications define their own XML DTD or Schema and register a specific XML namespace [6]. The IETF has a defined procedure for registering an application specific XML namespace [9].

NOTE: For clarity in this document, we will use the terms "XML" and "XML documents" when speaking in the general case about the Extensible Markup Language (XML). The terms "IODEF description", "IODEF markup" and "IODEF document" will be used to refer to specific elements (tags) and attributes of the IODEF DTD. Furthermore, the terms "class" and "subclass" are synonymous to an `element` in the XML DTD.

The implementation of the IODEF in XML has many benefits:

- + XML provides all the necessary features to define a specific markup language for describing security incidents. It also defines a standard way to extend this language, either for later revisions ("standard" extensions), or for vendor-specific use ("non-standard" extensions).
- + Software tools for processing XML documents are widely available in commercial and open source forms. Numerous tools and APIs for parsing and/or validating XML are available in a variety of languages, including Java, C, C++, Tcl, Perl, and Python.

Widespread access to tools will make the adoption of the IODEF by product developers easier, and hopefully, faster.

- + XML meets IODEF Requirement 4.1 that message formats support full internationalization and localization. The XML standard requires support for both the UTF-8 and UTF-16 encodings of ISO/IEC 10646 (Universal Multiple-Octet Coded Character Set, "UCS") and Unicode, making all XML applications (and therefore all IODEF-compliant applications) compatible with these common character encodings.

XML also provides support for specifying on a per-element basis, the language in which the element's content is written, making IODEF easy to adapt to local languages in which CSIRTs and their constituency work.

- + XML meets IODEF Requirement 4.2 that message formats must support modularity, filtering and aggregation. XML's integration with XSL, a style language, allows messages to be

combined, discarded, and rearranged.

- + XML is free (no license, license fees or royalties).

2.3 Relation between IODEF and IDMEF

The IODEF is heavily based on the IDMEF reusing most of its data model. The data model has upward compatibility (i.e. IDMEF messages can be directly encapsulated in an IODEF document) with the IDMEF whereby ensuring the inheritance of all IDMEF data structures. IDMEF documents provide a description of an incident from the perspective of a single intrusion detection system. IODEF will be able to further annotate this information with incident handling data.

Due to the close relationship between them, it is recommended that IH systems understand both the IODEF and IDMEF formats. For the most part, given a system that uses IODEF, adding IDMEF support should be trivial since IODEF duplicated the IDMEF namespace.

3. Notational Conventions and Formatting Issues

This document uses three notations: Unified Modeling Language (UML) to describe the data model, Extensible Markup Language (XML) to define the markup of the IODEF, and IODEF markup to represent the documents themselves.

This section describes these notations in sufficient detail that readers unfamiliar with them can understand the document. Note,

however, that these descriptions are not comprehensive; they only cover the components of the notations used by the data model and document format.

This section also explains several issues that apply to XML and IODEF documents such as the format of various data types, special characters, whitespace processing, character sets and languages.

3.1 Unified Modeling Language conventions used for IODEF Data Model description

The IODEF data model is described using the Unified Modeling Language (UML) [10]. UML provides a simple framework to represent entities and their relationships. UML defines entities as classes. In this document, we have identified several classes and their

associated attributes. The symbols used in this document to represent classes and attributes are shown in Figure 3.1.

```
+-----+
| Class Name |    <----- Name of class
+-----+
| Attribute 1 |    <----- Name of first attribute
| ...         |
| Attribute N |    <----- Name of nth attribute
+-----+
```

Figure 3.1 - Symbols representing classes and attributes

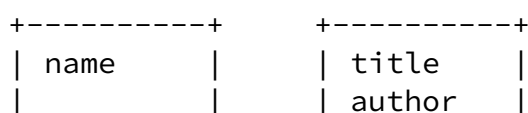
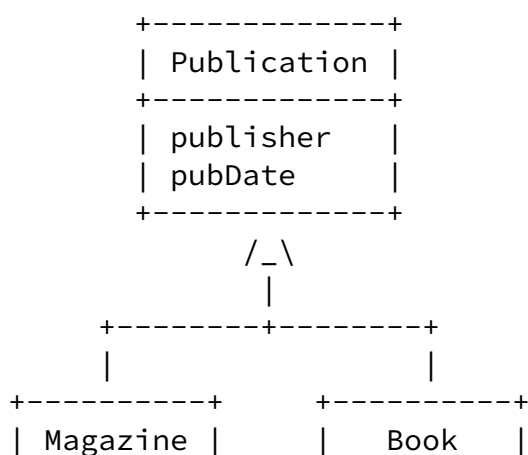
Note that the associated attributes for a class may not appear in all diagrams in which the class is used.

3.1.1 Relationships

The IODEF model currently uses only two of the relationship types defined by UML: inheritance and aggregation.

Inheritance denotes a superclass/subclass type of relationship where the subclass inherits all the attributes, operations, and relationships of the superclass. This type of relationship is also called a "is-a" or "kind-of" relationship. Subclasses may have additional attributes or operations that apply only to the subclass, and not to the superclass.

In this document, inheritance is denoted by the /_\
| symbol. In Figure 3.2, we are showing that Book and Magazine are two types of Publication. Book inherits all the attributes of Publication, plus all of its own attributes (thus, it has four attributes in total); as does Magazine (giving it three attributes in total).



```
+-----+ +-----+
```

Figure 3.2 - Inheritance relationships

Aggregation is a form of association in which the whole is related to its parts. This type of relationship is also referred to as a "part-of" relationship. In this case, the aggregate class contains all of its own attributes and as many of the attributes associated with its parts as required and specified by the occurrence indicators (see [Section 4.1.2](#)).

In this document, the symbol <> is used to indicate aggregation. It is placed at the end of the association line closest to the aggregate (whole) class. In Figure 4.3, we are showing that a Book is made up of pieces called Preface, Chapter, Appendix, Bibliography, and Index.

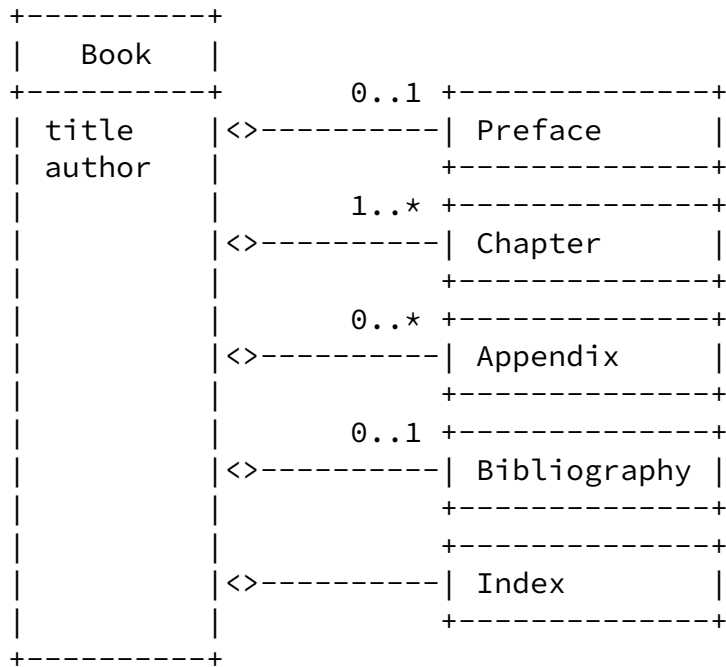


Figure 3.3 - Aggregation relationships

3.1.2 Occurrence Indicators

Occurrence indicators show the number of objects within a class that are linked to one another by an aggregation relationship. They are placed at the end of the association line closest to

the part they refer to. Occurrence indicators, as used in this document, are:

n	exactly "n" (left blank if n=1)
0..*	zero or more
1..*	one or more
0..1	zero or one (i.e., "optional")
n..m	between "n" and "m" (inclusive)

In Figure 3.3, the Book:

- + may have no Preface or one Preface;
- + must have at least one Chapter, but may have more;
- + may have any number of Appendixes; and
- + must have exactly one Index.

3.2 XML Document Type Definitions

XML Document Type Definitions (DTDs) are used to declare the markup for a document. This includes the different pieces of information the document will contain (the elements), characteristics of that information (the attributes), and the relationship between the pieces (the content model).

[Section 9](#) of this document contains the complete IODEF DTD.

3.2.2 Element Declarations

Elements are the main part of a document's markup; they define the names of the pieces of the document, and the content model for those pieces.

```
<!ELEMENT Book (  
    Preface, Chapter, Appendix, Bibliography, Index  
)>
```

In this example, the "Book" element is defined to consist of exactly one Preface, one Chapter, one Appendix, one Bibliography, and one Index. Furthermore, these parts must appear in this order (e.g., the Index cannot come before the Bibliography).

The XML document associated with this DTD might look like this:

```
<Book>
```

```
<Preface>
...
</Preface>
```

```
<Chapter>
...
</Chapter>
<Appendix>
...
</Appendix>
<Index>
...
</Index>
</Book>
```

NOTE: XML is for the most part a free-format language; the line breaks and indentation used in the examples are for the purpose of improving readability only.

3.2.2.1 Occurrence Indicators

In the example above, Book must contain exactly one of each part -- it cannot have more than one Chapter, the Preface is not optional, and so on. This is not a very good representation of real-life books.

XML provides occurrence indicators to make it possible to represent more complex content models. The occurrence indicators are:

?	the content may appear either once or not at all
*	the content may appear one or more times or not at all
+	the content must appear at least once, and may appear more than once
[none]	the content must appear exactly once

Occurrence indicators allow us to revise our Book content model

```
<!ELEMENT Book (
```

Preface?, Chapter+, Appendix*, Bibliography?, Index
)>

Now a Book may contain an optional Preface, one or more Chapters, any number of Appendixes, an optional Bibliography, and an Index. The parts must still occur in this order.

3.2.2.2 Alternative Content and Grouping

To allow the creation of arbitrarily complex content models,

Meijer, et al.

Expires October 2002

[page 13]

Internet Draft

[draft-ietf-inch-iodef-00.txt](#)

Apr 2002

XML also provides:

- + alternatives, specified with the '|' character
- + parentheses, to permit grouping of elements
- + occurrence indicators may also be used on parenthesized groups

For example:

```
<!ELEMENT x (a, (b | c | d), e)* >
```

would allow all of the following:

<x>	<x>	<x>	<x>	<x>
<a/>	<a/>	<a/>	<a/>	</x>
	<d/>	<c/>		
<e/>	<e/>	<e/>	<e/>	
</x>	</x>	<a/>	<a/>	
		<c/>	<c/>	
		<e/>	<e/>	
		</x>	<a/>	
			<d/>	
			<e/>	
			</x>	

The example above also introduces the "<tag/>" notation; this is used in XML to denote empty content. It is more or less equivalent to "<tag></tag>" (the differences are

beyond the scope of this document).

3.2.2.3 Element Content

An XML document has a tree structure. One element at the top is the parent of all other elements (e.g., Book); there is some number of other elements all with parents and children; and then at the bottom of the tree, there is some number of elements that have no children. These are the elements that contain the document content.

XML DTDs do not support data types such as integer, real, string, and so on (more on this later). However, they do require some indication of the type(s) of content that an element will contain. There are several types available, but only two are used in the IODEF:

PCDATA

An XML processor will find only text (parsed character data) in this element, no tags or entity references (see Section

3.2.4). This is the content type for all but one of the elements at the bottom of the IODEF document tree.

ANY

The element may contain anything -- text, other tags, entity references, etc. This is the content type for the AdditionalData element (see [Section 4.2.4.5](#)).

In the case where declaring the data is essential, future implementations of the IODEF should use an XML Schema definition instead of currently used XML DTD.

3.2.3 Attribute Declarations

Attributes allow data to be associated with an element. The decision to put data in an attribute or a child element is mostly one of style, although consideration should be given to the type and quantity of data as well. Attributes are,

generally, used for small, atomic data and elements are used for large or composite data.

Attributes are declared with their name, their content type, and their attribute type, as shown below:

```
<!ATTLIST Book
    title          CDATA          #REQUIRED
    author         CDATA          #REQUIRED
>
```

The declaration above defines two attributes of the Book element, title and author. Both may contain character data, and both are required. These might be given as follows in an XML document:

```
<Book title="The Cat in the Hat" author="Dr. Seuss">
```

3.2.3.1 Attribute Types

There are four attribute types:

#REQUIRED

The attribute is required, and has no default value. The XML document must specify a value for it.

#IMPLIED

The attribute is optional, and has no default value.

#FIXED [[value](#)]

The attribute must always have the default value "[[value](#)]." It is an error to specify the attribute with any other value. When an XML processor encounters an omitted attribute, it will behave as though it were present with the declared default value.

[[value](#)]

The attribute is optional, and has a default value of "[[value](#)]." When an XML processor encounters an omitted

attribute, it will behave as though it were present with the default value.

3.2.3.2 Attribute Content

There are a variety of attribute content types defined, but only two are used in the IODEF:

CDATA

An attribute of this type contains character data (text). Tags and entity references (see [Section 4.2.4](#)) are not processed.

[values]

An attribute may also be declared with a list of acceptable values; this functions somewhat like an enumerated type. For example:

```
<!ATTLIST Person
  gender      "unknown | male | female"  "unknown"
>
```

The gender attribute may have one of three values; if a Person tag appears without a gender attribute, the XML processor will behave as though it did have one, with value "unknown."

3.2.4 Entity Declarations

Entities allow symbols to be defined that will be replaced with other text when processed. There are two types of entities, "general" and "parameter." General entities are for use within XML document content; for example:

```
<!ENTITY IODEF "Intrusion Detection Message Exchange Format">
```

'&' and ';' -- whenever "&IODEF;" appears in the XML document from the example above, it will be replaced with the text "Intrusion Detection Message Exchange Format". General entities (and a special case of them called character references) are used extensively in handling special characters (see Sections [4.3.2.1](#) and [4.3.2.2](#)).

Parameter entities are for use within DTDs (they are not recognized in document content), and are declared and referenced in a slightly different way. The declaration includes a '%' symbol before the entity name, and they are referenced by bracketing them with the characters '%' (instead of '&') and ';'. For example, attributes that must appear on every element are declared in a parameter entity:

```
<!ENTITY % attlist.global      "  
    xmlns          CDATA      #FIXED      'urn:iana:xml:ns:IODEF'  
    xmlns:IODEF    CDATA      #FIXED      'urn:iana:xml:ns:IODEF'  
    xml:space       (default | preserve) 'default'  
    xml:lang        NMTOKEN    #IMPLIED  
>
```

and then referenced in each attribute list declaration:

```
<!ATTLIST IODEF-Description  
    %attlist.global;  
>  
<!ATTLIST Alert  
    %attlist.global;  
>
```

3.3 XML Documents

This section describes a number of XML document formatting rules; these rules apply to IODEF documents as well.

3.3.1 The Document Prolog

The "prolog" of an XML document, that part that precedes anything else, consists of the XML declaration and the document type declaration.

3.3.1.1 XML Declaration

Every XML document (and therefore every IODEF document)

starts with an XML declaration. The XML declaration specifies the version of XML being used; it may also specify the character encoding being used.

The XML declaration looks like:

```
<?xml version="1.0" ?>
```

If a character encoding is specified, the declaration looks like:

```
<?xml version="1.0" encoding="charset" ?>
```

where "charset" is the name of the character encoding in use (see [Section 3.3.2](#)). If no encoding is specified, UTF-8 is assumed. IODEF documents being exchanged between IODEF-compliant applications MUST begin with an XML declaration, and MUST specify the XML version in use. Specification of the encoding in use is RECOMMENDED.

IODEF-compliant applications MAY choose to omit the XML declaration internally to conserve space, adding it only when the message is sent to another destination (e.g., a web browser). This practice is NOT RECOMMENDED unless it can be accomplished without loss of each message's version and encoding information.

3.3.1.2 IODEF DTD Formal Public Identifier

The formal public identifier (FPI) for the IODEF Document Type Definition described in this document is:

```
"-//IETF//DTD RFCxxxx IODEF v0.0//EN"
```

NOTE: The "RFCxxxx" text in the FPI value will be replaced with the actual RFC number, if this document is published as an RFC.

This FPI MUST be used in the document type declaration within an XML document referencing the IODEF DTD defined by this document, as shown in the following section.

3.3.1.3 IODEF DTD Document Type Declaration

The document type declaration for an XML document referencing the IODEF DTD will usually be specified in the following ways:

Meijer, et al.

Expires October 2002

[page 18]

Internet Draft

[draft-ietf-inch-iodef-00.txt](#)

Apr 2002

```
<!DOCTYPE IODEF-Description PUBLIC
  "-//IETF//DTD RFCxxxx IODEF v0.0//EN">
```

The last component of the document type declaration is the FPI specified in the previous section.

```
<!DOCTYPE IODEF-Description SYSTEM
  "/some/path/to/the/IODEF-Description.dtd">
```

The last component of the document type declaration is a URI that points to a copy of the Document Type Definition.

In order to be valid (see [Section 7.1](#)), an XML document must contain a document type declaration. However, this requirement imposes a significant overhead on an IODEF-compliant application in bandwidth consumption and computation for the DTD may need to be downloaded and parsed before use by the XML parser.

Implementers MAY decide to have analyzers and managers agree out-of-band on the particular document type definition they will be using to exchange messages (the standard one as defined here, or one with extensions), and then omit the document type declaration from IODEF descriptions. The method for negotiating this agreement is outside the scope of this document. Note that great care must be taken in negotiating any such agreements, as the manager may have to accept messages from many different analyzers, each using a DTD with a different set of extensions.

3.3.2 Character Data Processing in XML and IODEF

A document's XML declaration (see [Section 4.3.1.1](#)) specifies the character encoding to be used in the document, as follows:

```
<?xml version="1.0" encoding="charset" ?>
```

where "charset" is the name of the character encoding, as registered with the Internet Assigned Numbers Authority (IANA), see [\[11\]](#).

The XML standard requires that XML processors support the UTF-8 and UTF-16 encodings of ISO/IEC 10646 (UCS) and Unicode, making all XML applications (and therefore, all IODEF-compliant applications) compatible with these common character encodings. The XML standard also permits other character encodings to be used (e.g., UTF-7, UTF-8, UTF-32). However, support for these

encodings is not guaranteed to be present in all XML applications.

For portability reasons, IODEF-compliant applications SHOULD NOT use, and IODEF descriptions SHOULD NOT be encoded in, character encodings other than UTF-8 and UTF-16. Consistent with the XML standard, if no encoding is specified for an IODEF description, UTF-8 is assumed.

NOTE: The ASCII character set is a subset of the UTF-8 encoding, and therefore may be used to encode IODEF descriptions.

Per the XML standard, IODEF documents encoded in UTF-16 MUST begin with the Byte Order Mark described by ISO/IEC 10646 Annex E and Unicode [Appendix B](#) (the "ZERO WIDTH NO-BREAK SPACE" character, #xFEFF).

3.3.2.1 Character Entity References

Within XML documents, certain characters have special meanings in some contexts. To include the actual character itself in one of these contexts, a special escape sequence,

called an entity reference, must be used.

The characters that sometimes need to be escaped, and their entity references, are:

Character	Entity Reference

&	&
<	<
>	>
"	"
'	'

It is RECOMMENDED that IODEF-compliant applications use the entity reference form whenever writing these characters in data, to avoid any possibility of misinterpretation.

3.3.2.2 Character Code References

Any character defined by the ISO/IEC 10646 and Unicode standards may be included in an XML document by the use of a character reference. A character reference is started with the characters '&' and '#', and ended with the character ';'. Between these characters, the character code for the

character inserted.

If the character code is preceded by an 'x' it is interpreted in hexadecimal (base 16), otherwise, it is interpreted in decimal (base 10). For instance, the ampersand (&) is encoded as & or & and the less-than sign (<) is encoded as < or <.

Any one-, two-, or four-byte character specified in the ISO/IEC 10646 and Unicode standards can be included in a document using this technique.

3.3.2.3 White Space Processing

XML preserves white space by default. The XML processor passes all white space characters to the application unchanged. This behavior is much different from HTML (and SGML) in which the presence (or lack of) spaces is meaningful, but one space is interpreted the same as multiple spaces.

XML allows elements to identify the importance of white space in their content by using the "xml:space" attribute:

```
<tag xml:space="action">
```

where "action" is either "default" or "preserve."

If "action" is "preserve," the application MUST treat all white space in the element's content as significant. If "action" is "default," the application is free to do whatever it normally would with white space in the element's content.

The intent declared with the "xml:space" attribute is considered to apply to all attributes and content of the element where it is specified (including sub-elements), unless overridden with an instance of "xml:space" on another element within that content.

All IODEF elements support the "xml:space" attribute.

3.3.3 Languages in XML and IODEF

XML allows elements to identify the language their content is written in by using the "xml:lang" attribute:

```
<tag xml:lang="langcode">
```

where "langcode" is a language tag as described in [RFC 3066](#) [12].

The intent declared with the "xml:lang" attribute is considered

to apply to all attributes and content of the element where it is specified (including sub-elements), unless overridden with an instance of "xml:lang" on another element within that content.

IODEF-compliant applications SHOULD specify the language in which their contents are encoded. In general, the language can be specified with the "xml:lang" attribute in the top-level element and letting all other elements "inherit" that definition.

If no language is specified for an IODEF description, English SHALL be assumed.

All IODEF tags support the "xml:lang" attribute.

3.3.4 Inheritance and Aggregation

XML DTDs do not support inheritance as used by the IODEF data model (i.e., there is no support for "kind-of" relationships). This limitation does not present a major problem in practice because aggregation relationships can be used instead with little loss of functionality.

As a note of interest, XML Schemas, recently approved by the W3C, will provide support for inheritance, stronger data typing and other useful features [7]. Future versions of the IODEF will probably use XML Schemas instead of DTDs. It was recognized that in the initial stage of the design of a new application, an XML DTD was useful since it provides a better human readable format for document and element descriptions. However, with further the development of applications and integration into IH systems a more detailed definition of data types and elements relations as provided by XML Schemas may be required.

3.4 IODEF Data Types

Within an XML IODEF description, all data will be expressed as "text" (as opposed to "binary"), since XML is a text formatting language. We provide typing information for the attributes of the

classes in the data model however, to convey to the reader the type of data the model expects for each attribute.

Each data type in the model has specific formatting requirements in an XML IODEF description. These requirements are set forth in this section.

3.4.1 Integers

Integer attributes are represented by the INTEGER data type. Integer data MUST be encoded in Base 10 or Base 16.

Base 10 integer encoding uses the digits '0' through '9' and an optional sign ('+' or '-'). For example, "123", "-456".

Base 16 integer encoding uses the digits '0' through '9' and 'a' through 'f' (or their upper case equivalents), and is preceded by the characters "0x". For example, "0x1a2b".

3.4.2 Real Numbers

Real (floating-point) attributes are represented by the REAL data type. Real data MUST be encoded in Base 10.

Real encoding is that of the POSIX "strtod" library function: an optional sign ('+' or '-') followed by a non-empty string of decimal digits, optionally containing a radix character, then an optional exponent part. An exponent part consists of an 'e' or 'E', followed by an optional sign, followed by one or more decimal digits. For example, "123.45e02", "-567,89e-03".

IODEF-compliant applications MUST support both the '.' and ',' radix characters.

3.4.3 Characters and Strings

Single-character attributes are represented by the CHARACTER data type. Multi-character attributes of known length are represented by the STRING data type.

Character and string data have no special formatting requirements, other than the need to occasionally use character references (see Sections [4.3.2.1](#) and [4.3.2.2](#)) to represent special characters.

3.4.4 Bytes

Binary data is represented by the BYTE (and BYTE[]) data type.

Binary data MUST be encoded in its entirety using character code references (see [Section 4.3.2.2](#)).

3.4.5 Enumerated Types

Enumerated types are represented by the ENUM data type, and consist of an ordered list of acceptable values. Each value has a rank (number) and a representing keyword.

Within an IODEF message, the enumerated type keywords are used as attribute values, and the ranks are ignored. However, those IODEF-compliant applications that choose to represent these values internally in a numeric format MUST use the rank values identified in this memo.

3.4.6 Date-Time Strings

Date-time strings are represented by the DATETIME data type. Each date-time string identifies a particular instant in time; ranges are not supported.

Date-time strings are formatted according to a subset of ISO 8601:2000 [[13](#)], as show below. Section references in parentheses refer to sections of the ISO 8601:2000 standard.

1. Dates MUST be formatted as follows:

YYYY-MM-DD

where YYYY is the four- digit year, MM is the two-digit month (01-12), and DD is the two- digit day (01-31). ([Section 5.2.1.1](#), "Complete representation -- Extended format.")

2. Times MUST be formatted as follows:

hh:mm:ss

where hh is the two-digit hour (00-24), mm is the two-digit minute (00-59), and ss is the two-digit second (00-60). ([Section 5.3.1.1](#), "Complete representation -- Extended format.")

Note that midnight has two representations, 00:00:00 and 24:00:00. Both representations MUST be supported by IODEF-compliant applications, however, the 00:00:00 representation SHOULD be used whenever possible.

Note also that this format accounts for leap seconds. Positive leap seconds are inserted between 23:59:59Z and 24:00:00Z and are represented as 23:59:60Z. Negative leap seconds are achieved by the omission of 23:59:59Z. IODEF-compliant applications MUST support leap seconds.

3. Times MAY be formatted to include a decimal fraction of seconds, as follows:

hh:mm:ss.ss or
hh:mm:ss,ss

As many digits as necessary may follow the decimal sign (at least one digit must follow the decimal sign). Decimal fractions of hours and minutes are not supported. ([Section 5.3.1.3](#), "Representation of decimal fractions.")

IODEF-compliant applications MUST support the use of both decimal signs ('.' and ',').

Note that the number of digits in the fraction part does not imply anything about accuracy -- i.e., "00.100000", "00,1000" and "00.1" are all equivalent.

4. Times MUST be formatted to include (a) an indication that

the time is in Coordinated Universal Time (UTC), or (b) an indication of the difference between the specified time and Coordinated Universal Time.

- a. Times in UTC MUST be formatted by appending the letter 'Z' to the time string as follows:

```
hh:mm:ssZ
hh:mm:ss.ssZ
hh:mm:ss,ssZ
```

([Section 5.3.3](#), "Coordinated Universal Time (UTC) -- Extended format.")

- b. If the time is ahead of or equal to UTC, a '+' sign is appended to the time string; if the time is behind UTC, a '-' sign is appended. Following the sign, the number of hours and minutes representing the different from UTC is appended, as follows:

```
hh:mm:ss+hh:mm
hh:mm:ss-hh:mm
hh:mm:ss.ss+hh:mm
hh:mm:ss.ss-hh:mm
hh:mm:ss,ss+hh:mm
hh:mm:ss,ss-hh:mm
```

The difference from UTC MUST be specified in both hours and minutes, even if the minutes component is 0. A "difference" of "+00:00" is equivalent to UTC. ([Section 5.3.4.2](#), "Local time and the difference with Coordinated Universal Time -- Extended Format.")

5. Date-time strings are created by joining the date and time strings with the letter 'T', as shown below:

```
YYYY-MM-DDThh:mm:ssZ
YYYY-MM-DDThh:mm:ss.ssZ
YYYY-MM-DDThh:mm:ss,ssZ
YYYY-MM-DDThh:mm:ss+hh:mm
```

YYYY-MM-DDThh:mm:ss-hh:mm
YYYY-MM-DDThh:mm:ss.ss+hh:mm
YYYY-MM-DDThh:mm:ss.ss-hh:mm
YYYY-MM-DDThh:mm:ss,ss+hh:mm
YYYY-MM-DDThh:mm:ss,ss-hh:mm

([Section 5.4.1](#), "Complete representation -- Extended format.")

In summary, IODEF date-time strings MUST adhere to one of the nine templates identified in Paragraph 5, above.

3.4.7 NTP Timestamps

NTP timestamps are represented by the NTPSTAMP data type, and are described in detail in [[14](#)] and [[15](#)]. An NTP timestamp is a 64-bit unsigned fixed-point number. The integer part is in the first 32 bits, and the fraction part is in the last 32 bits.

Within IODEF descriptions, NTP timestamps MUST be encoded as two 32-bit hexadecimal values, separated by a period ('.'). For example, "0x12345678.0x87654321".

3.4.8 Port Lists

Port lists are represented by the PORTLIST data type, and

consist of a comma-separated list of numbers (individual integers) and ranges (N-M means ports N through M, inclusive). Any combination of numbers and ranges may be used in a single list. For example, "5-25,37,42,43,53,69-119,123-514".

3.4.9 Unique Identifiers

There are several types of unique identifiers used in this specification. All types are represented by STRING data types.

These identifiers are implemented as attributes in the relevant

XML elements, and must have unique values as follows:

1. If specified, the attribute of the Authority class ([Section 5.2.6.8](#)) OrganizationID MUST have a value that is globally unique. It may be a combination of the Registry name and unique CSIRT ID in this Registry. FIRST or industry associations normally maintain registries.

The default value is "unknown", which indicates that the authority or CSIRT doesn't have unique identifiers.

2. The Incident, Attacker, Evidence, Victim, Source, Target, Node, User, Process, Service, Address, and UserID classes (see correspondent sections) are provided with "ident" attribute, which if specified, MUST have a value that is unique across all IODEF Descriptions created by the particular CSIRT or Authority.

The "ident" attribute value MUST be unique for each particular combination of data identifying an object, not for each object. Objects may have more than one ident value associated with them. For example, an identification of a host by name would have one value, while an identification of that host by address would have another value, and an identification of that host by both name and address would have still another value. Furthermore, different analyzers may produce different values for the same information.

The "ident" attribute by itself provides a unique identifier only among all the "ident" values created/stored by a particular CSIRT or IHS. But when combined with the unique "OrganizationID" value for the CSIRT, there is no requirement for global uniqueness. The default value is "0", which indicates that the CSIRT/IHS cannot generate unique identifiers.

The specification of methods for creating the unique values

contained in these attributes is outside the scope of this document.

3.4.10 Personal names

Format for the Personal name data is used the same as in LDAP. It is supposed that normally personal names are obtained from different Directories used by CSIRTs for their daily work.

Current suggestion for the personal name formats are as follows:

Name Surname

Or

Surname, Name

It is possible to use personal handle from the official (IP or DNS) databases: RIPE NCC, InterNIC, etc. In this case element's attribute will indicate type of personal name presentation and indirectly point on used Registry or database.

3.4.11 Organization names

Organization name is presented in form of its full name, short name or identification code retrieved from official Registries.

It is possible to use organization handle (or organization role from the official (IP or DNS) databases: RIPE NCC, InterNIC, etc. In this case element's attribute will indicate type of personal name presentation and indirectly point on used Registry or database.

3.4.12 Postal addresses

Format for the Postal addresses data is used the same as in LDAP. It is supposed that postal addresses are obtained from the Incident reports or from different Directories used by CSIRTs for their daily work.

Building, Street, Zip-code, City, Country

Or

Post Office Box, Zip-code, City, Country

3.4.13 Telephone and Fax numbers

Telephone and Fax numbers are expressed in format recommended by ITU documents.

+ (international code) (local code) (tel. Number)

[4.](#) The IODEF Data Model and XML DTD

In this section, the individual components of the IODEF data model are explained in details. UML diagrams of the model are provided to illustrate the relationship between components. Likewise, relevant sections of the XML DTD are presented to describe how the model is translated into XML.

4.1 Data Model Overview

The relationship between the principal components of the data model is shown in Figure 5.1 (cardinality and attributes are omitted).

IODEF-Description is the top-level container class for all IODEF documents. Recognizing that incidents might require different types of data, sub-classes of this root class called incident descriptions are defined. There are presently two types of descriptions defined: the Incident class to describe an incident and the IncidentAlert class to allow seamless support for IODEF alerts.

It is important to note that the data model does not define the events that constitute an incident. The notion of an incident is very site-specific. For example, a port scan may be identified by one CSIRT as a single incident with multiple victims. Another CSIRT might separate this activity as multiple incidents each from a single source to a single victim. Regardless, once the creator of the report has determined a logical grouping of events that constitute an incident, the data model dictates how that description should be formatted.

```
+-----+
| IODEF-Description |
+-----+
```

/ _ \
|
+-----+

						+-----+	
						IncidentAlert	
						+-----+	
+-----+		+-----+		+-----+		+-----+	
Incident <>-		Attack <>-		Source <>-		Node	
+-----+		+-----+		+-----+		+-----+	
						<>- User	
						+-----+	
						+-----+	
						<>- Process	
						+-----+	
						+-----+	
						<>- Service	
						+-----+	
						+-----+	
						<>- Program	
						+-----+	
						<>- OS	
				+-----+		+-----+	
				+-----+		+-----+	
				<>- Target <>-		Node	
				+-----+		+-----+	
						+-----+	
						<>- User	
						+-----+	
						+-----+	
						<>- Process	
						+-----+	
						+-----+	
						<>- Service	
						+-----+	
						+-----+	
						<>- Program	

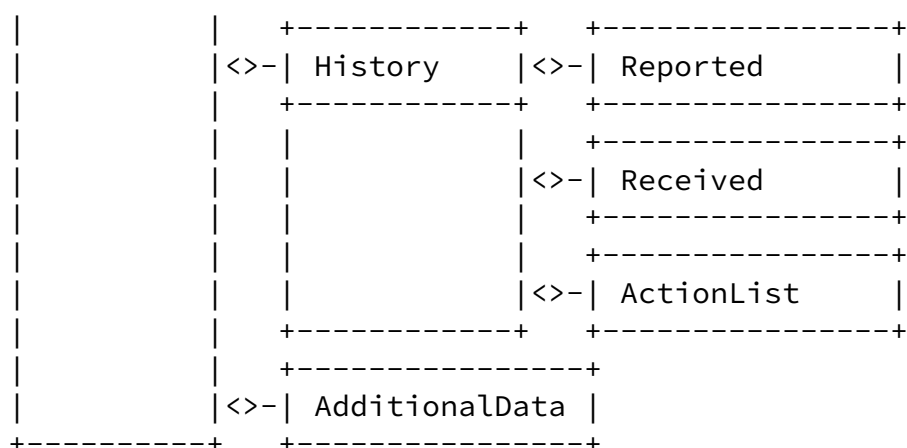


Figure 4.1 Data model overview

Note: The IODEF data model in graphical form can be found at [\[18\]](#).

The individual classes are described in the following sections.

4.2 The IODEF-Description Class

IODEF-Description is the root container class of the IODEF data model. There are currently two main types (subclasses) of IODEF-Description: Incident and IncidentAlert. A third Experimental class is also included temporarily for testing.

Since DTDs do not support subclassing (see [Section 4.3.4](#)), the inheritance relationship between the IODEF-Description and the Incident and IncidentAlert subclasses shown in Figure 5.1 has been replaced with an aggregate relationship.

NOTE: The use of aggregation to implement an inheritance relationship is done throughout the data model.

The IODEF-Description class is declared in the IODEF DTD as follows:

```
<!ENTITY % attlist.IODEF
    version          CDATA          #FIXED    '0.0'
">
<!ELEMENT IODEF-Description          (
    (Incident | IncidentAlert)*
)>
<!ATTLIST IODEF-Description
    %attlist.IODEF;
>
```

The IODEF-Description class has a single attribute:

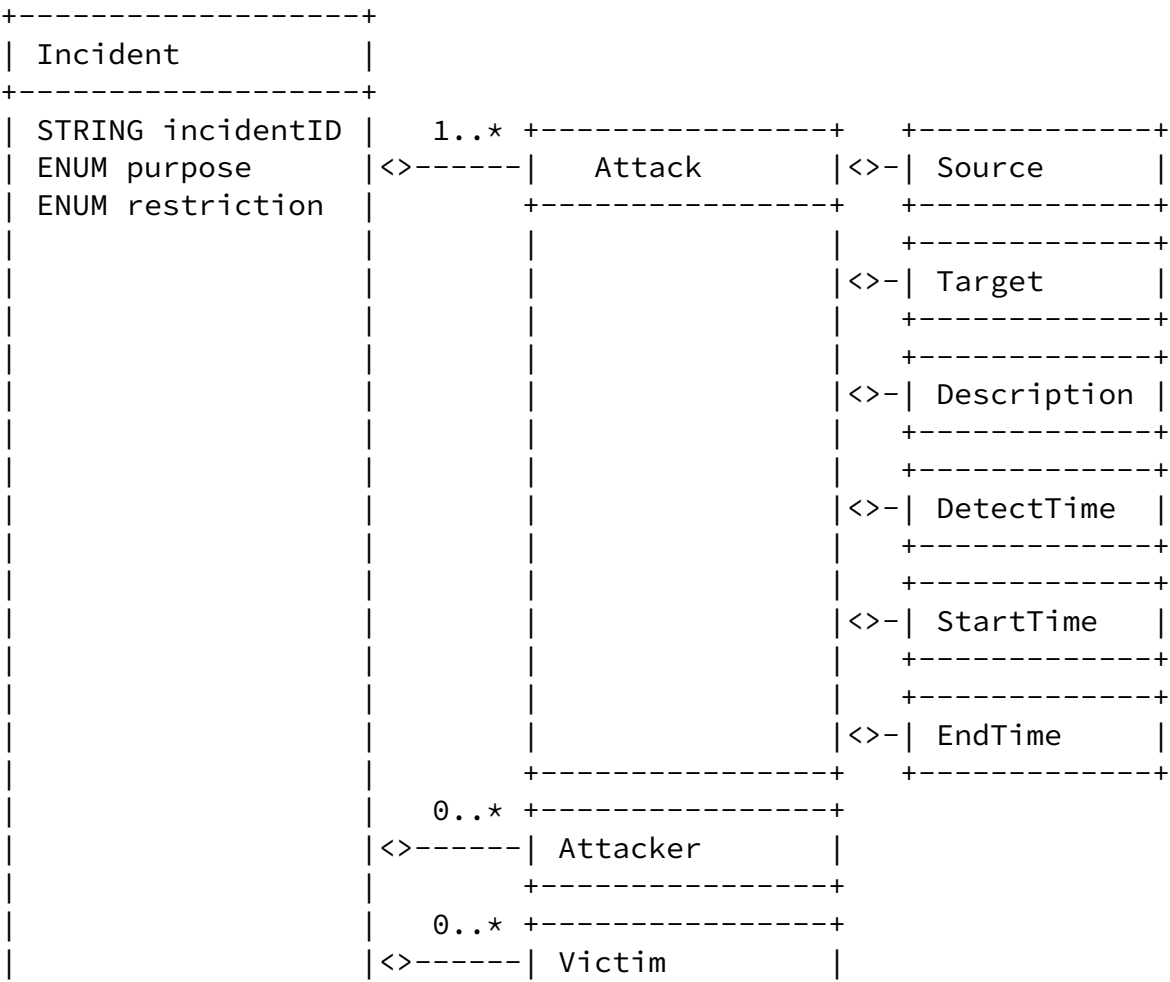
version

The version of the IODEF-Description specification (this document) to which the document conforms. Applications specifying a value for this attribute MUST use the value "0.0".

4.3 The Incident Class

For a given incident, the CSIRT will create an instance of the Incident class. The information used to populate this class will come from the reporting infrastructure that the CSIRT already has in place. Thus, direct reports from their constituency, IDS alert messages, or collaboration with other CSIRTs could serve as potential input.

An Incident description is composed of several aggregate classes, as shown in Figure 4.2. The aggregate classes themselves are described in Sections [4.2.4.1](#) - [4.2.4.10](#).



History

Zero or one. A log of the actions taken by the CSIRT(s) in the course of investigating the incident.

AdditionalData

Zero or more. Additional information about the incident included by CSIRT that cannot be readily expressed in the data model.

The Incident class is represented in the XML DTD as follows:

```
<!ENTITY % attvals.purpose "  
    ( unknown | report | handling | communication |  
      statistics | experimental )  
">  
<!ENTITY % attvals.restriction "  
    ( default | public | internal |  
      restricted )  
">  
<!ELEMENT Incident (  
    Attack+, Attacker*, Victim*, Method*, Evidence?,  
    CorrelationIncident?, Authority, History?, AdditionalData*)>  
<!--ATTLIST Incident  
    incidentID          CDATA          '0'  
    purpose              %attvals.purpose;  'experimental'  
    restriction          %attvals.restriction;  'default'  
>
```

The Incident class has three attributes:

IncidentID

Required. A unique identifier for the Incident (see [Section 3.4.9](#)).

purpose

Optional. The purpose of the incident being reported to the CSIRT.

Rank	Keyword	Description
----	-----	-----

0	unknown	Purpose of the incident is unknown
1	report	Incident report
2	handling	Incident is being handled
3	communication	Incident is being sent to another team
4	statistics	Incident was reported for statistical purposes
5	experimental	Experimental

restriction

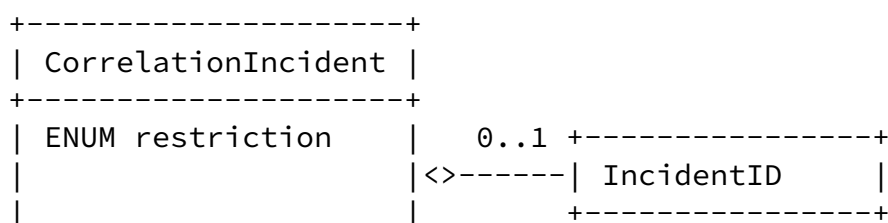
Optional. Sets a restriction on the usage of the data in element.

Rank	Keyword	Description
----	-----	-----
0	default	Restriction level is defined by external policy applied to overall CSIRT process
1	public	No restriction is applied to element
2	internal	Data is for company's (or constituency) internal use
3	restricted	Use strictly for Incident managers at CSIRT

4.4 The CorrelationIncident Class

The CorrelationIncident class represents information related to the correlation of current incident. It is intended as a way by which to logically group previously reported incidents as related.

The CorrelationIncident class is composed of three aggregate classes, as shown in Figure 4.3.



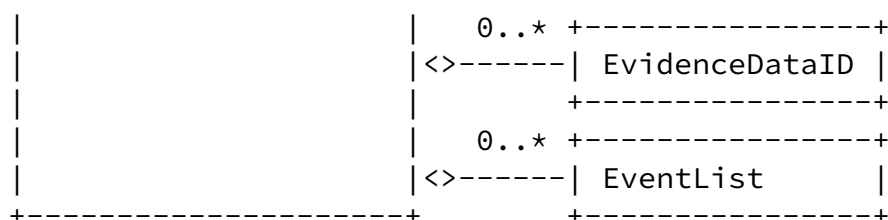


Figure 4.3 – The CorrelationIncident Class

The aggregate classes that constitute CorrelationIncident are:

IncidentID

Zero or one. STRING. Identifier of current Incident. If not included into CorrelationIncident class, this value may be derived from the top class Incident attribute.

EvidenceDataID

Zero or more. Evidence data that are linked to current Incident.

EventList

One or more. Lists all events which are investigated together, or have another common denominator.

This is represented in the XML DTD as follows:

```
<!ELEMENT CorrelationIncident (
  IncidentID?, EventList*, EvidenceDataID*
)>

<!ATTLIST CorrelationIncident
  restriction          %attvals.restriction;  'default'
>
```

The CorrelationIncident class has one attribute:

restriction

Optional. Sets a restriction on the usage of the data in element.

4.5 IncidentAlert Class

The IncidentAlert class is used as a container for IDMEF Alert messages.

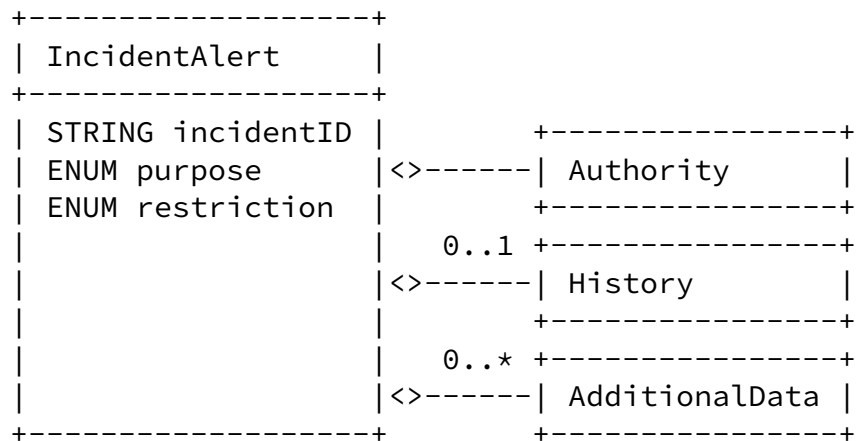


Figure 4.4 The IncidentAlert Class

The aggregate classes that constitute IncidentAlert are:

Authority

Exactly one. The CSIRT or authority that created the incident.

History

Zero or one. A log of the actions taken by the CSIRT(s) in course of investigating the incident.

AdditionalData

Zero or more. A container for the IDMEF Alert message.

This is represented in the XML DTD as follows:

```
<!ELEMENT IncidentAlert (
  Authority, History?, AdditionalData*)>
<!ATTLIST Incident
```

```

IncidentID      ID      #IMPLIED
purpose         %attvals.purpose;    'unknown'
restriction     %attvals.restriction; 'default'
>

```

The IncidentAlert class has three attributes:

```

IncidentID
  Optional. A unique identifier for the alert, see Section 3.4.9.

purpose
  Optional. The purpose of the incident being reported to the
  CSIRT.

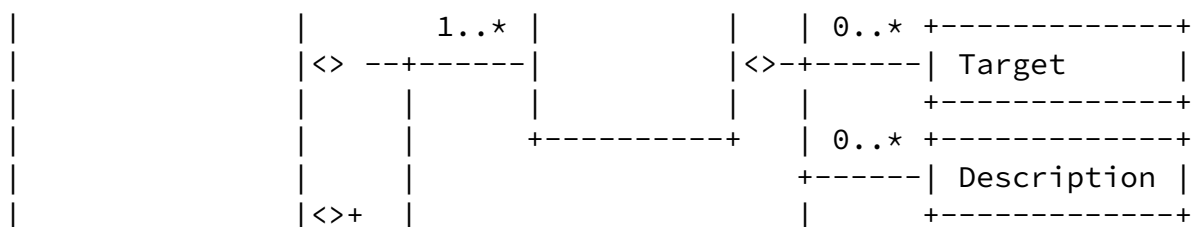
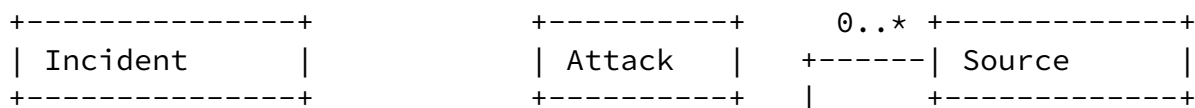
restriction
  Optional. Sets a restriction on the usage of the data in
  element.

```

4.6 The Core Classes

The core classes (Attack, Source, Target, Attacker, Victim, Method, Evidence, Authority, History, and AdditionalData) are the main parts of the Incident and IncidentAlert classes, as shown in Figure 5.5.

NOTE: The IODEF data model reuses the Source and Target classes (as well as their subclasses) from the IDMEF [3].



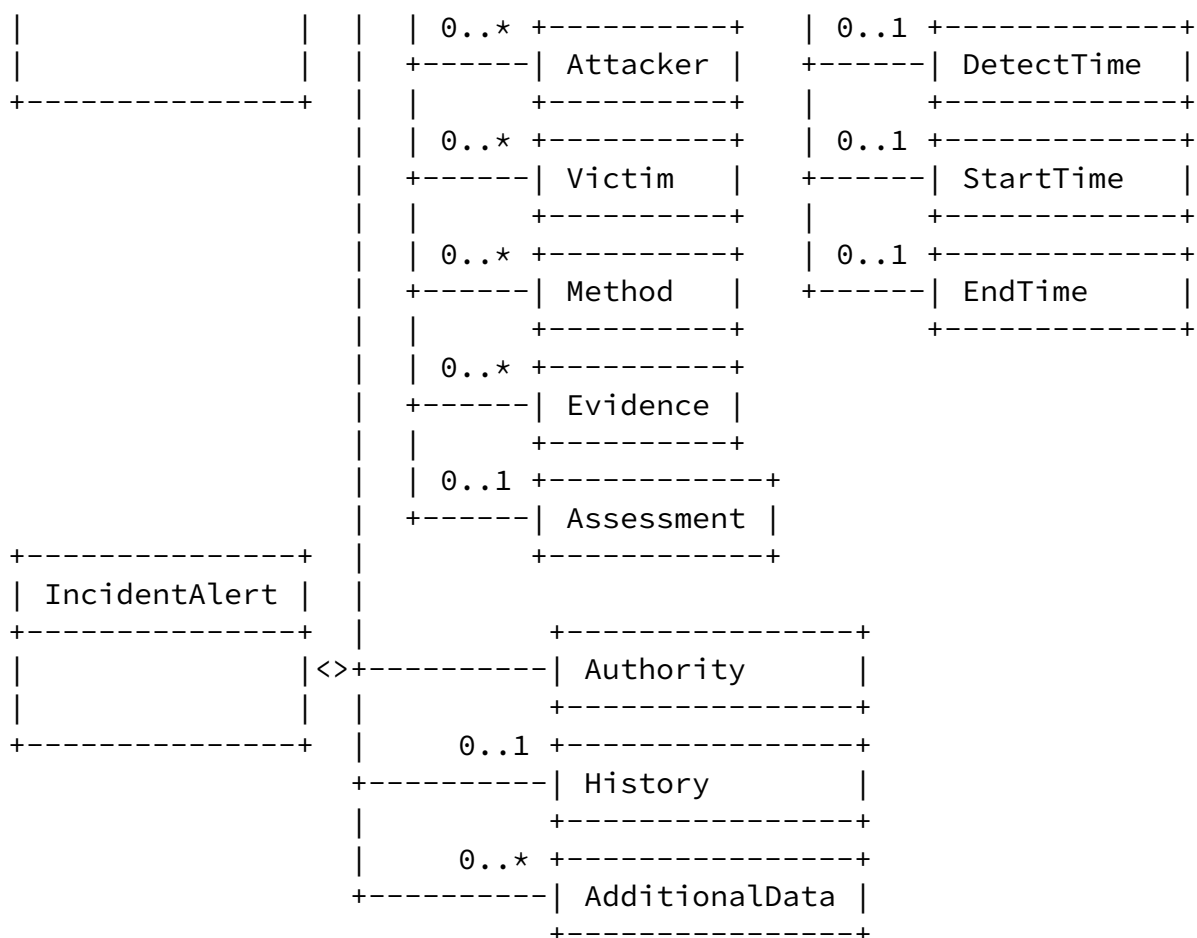
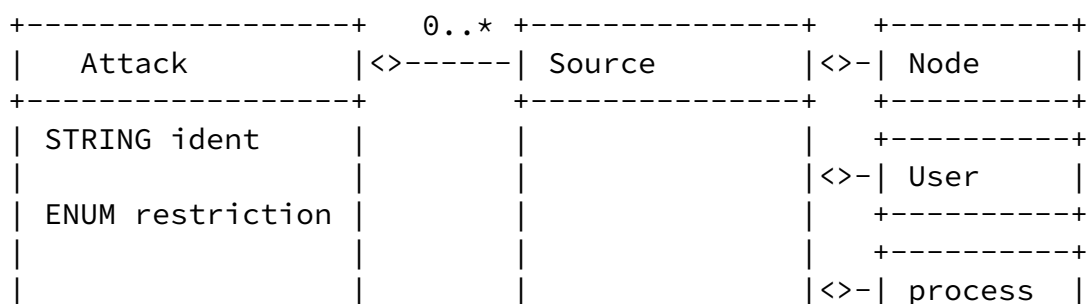


Figure 4.5 The IODEF Core Classes

4.6.1 The Attack Class

The Attack class contains information about the security events that constitute the incident.



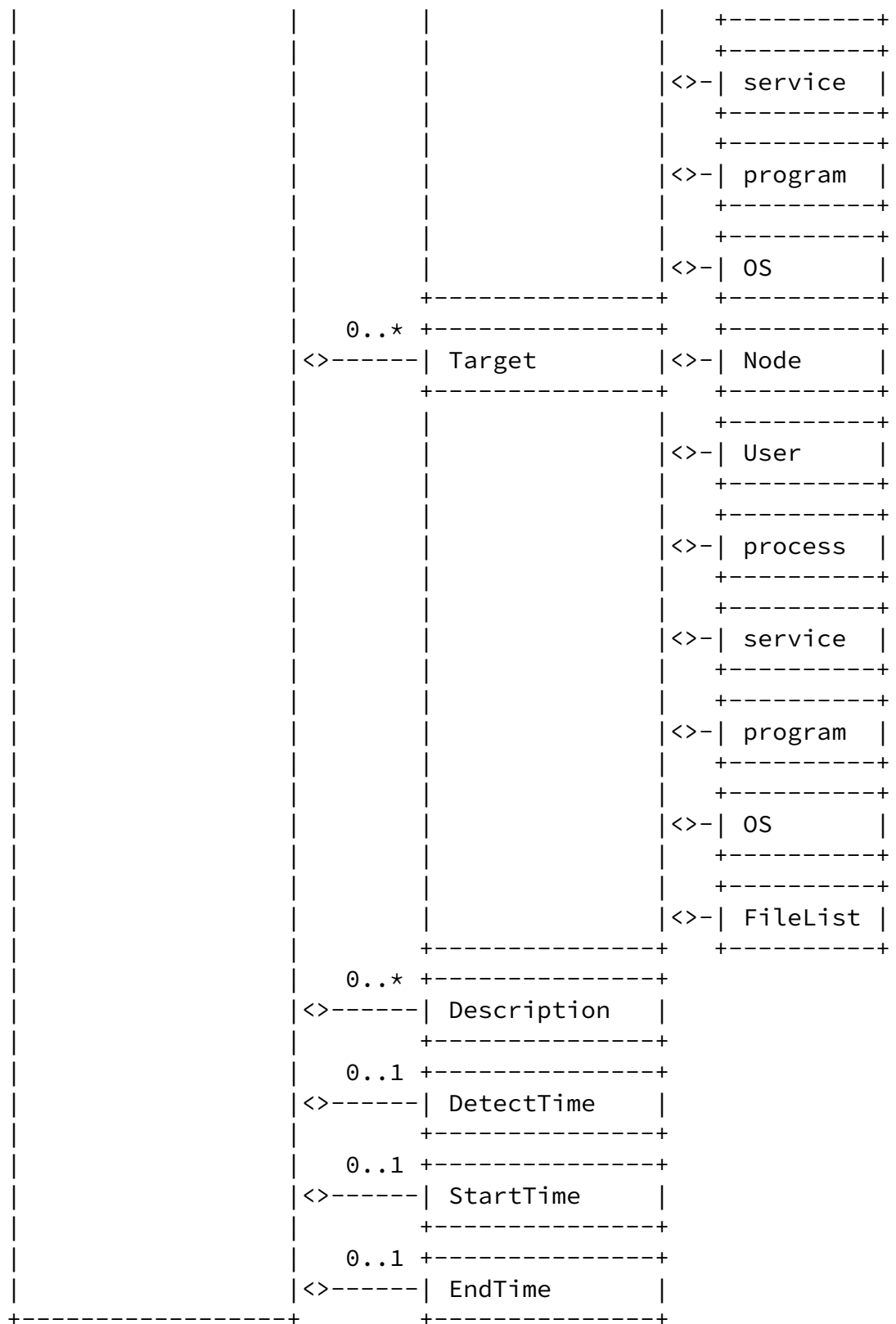


Figure 4.6 The Attack Class

The aggregate classes that constitute Attack are:

Source

Zero or more. The source(s) of the event(s) causing the incident.

Target

Zero or more. The target(s) of the event(s) in the incident.

Description

Zero or more. A free-form textual description by the CSIRT or report of the incident events.

DetectTime

Zero or one. The time when the incident activity was first detected by the reporter. In the case of more than one event, the time the first event was detected. In some circumstances, this time may not be the same as the RegistrationTime used in the History class.

StartTime

Zero or one. The start time of the incident activity.

EndTime

Zero or one. The end time of the incident activity.

This is represented in the XML DTD as follows:

```
<!ELEMENT Attack    (
    Source*, Target*, Description*,
    DetectTime?, StartTime?, EndTime?)>

<!ATTLIST Attack
    %attlist.global;
    restriction      %attvals.restriction;    'default'
    ident            CDATA                     #IMPLIED
>
```

The Attack class has two attributes:

ident

Optional. A unique identifier for this Attack class (see [Section 3.4.9](#)).

restriction

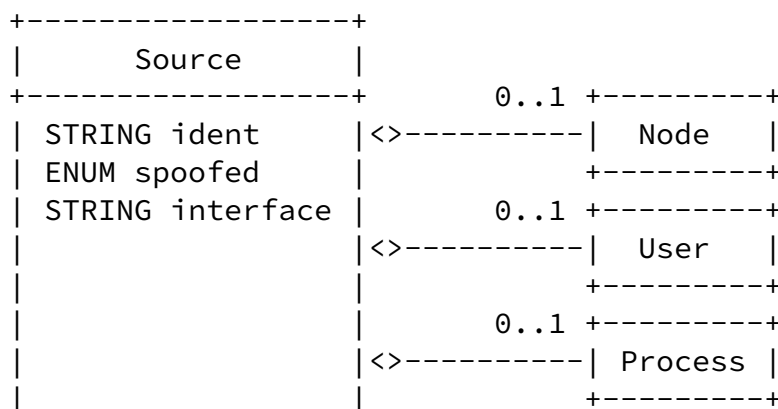
Optional. Sets a restriction on the usage of the data in element.

4.6.2 The Source Class

The Source class contains information about the possible source(s) of the incident event(s). An event may have more than one source (e.g., in a distributed denial of service attack). For the purpose of compatibility, the Source class has been reused from the IDMEF. Hence, the Source class from an IDMEF message can be included unmodified into the IODEF-Description class with the same semantics. Likewise, the data in an IDMEF-originating Source class could be decomposed between the IODEF Source and Attack classes.

The definition of the Source class in the IODEF data model is a superset of the IDMEF definition. Two new classes have been added: OS and program.

The Source class is composed of four aggregate classes, as shown in Figure 4.7.



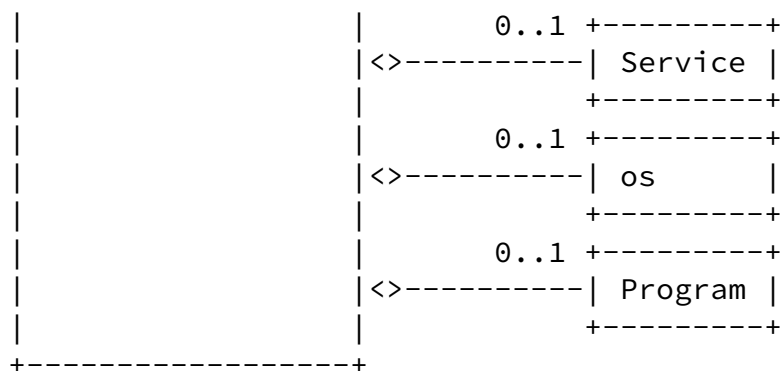


Figure 4.7 The Source Class

The aggregate classes that constitute Source are:

Node

Zero or one. Information about the host or device that appears to be causing the events (network address, network

name, etc.).

User

Zero or one. Information about the user that appears to be causing the event(s).

Process

Zero or one. Information about the process that appears to be causing the event(s).

Service

Zero or one. Information about the network service involved in the event(s).

os

Zero or one. The operation system running on the Node from which the Attack originated.

program

Zero or one. The program that caused the Attack, that is

running in the Process.

This is represented in the XML DTD as follows:

```
<!ENTITY % attvals.yesno  "
    ( unknown | yes | no )
">
<!ELEMENT Source  (
    Node?, User?, Process?, Service?, os?, program?
)>
<!ATTLIST Source
    ident      CDATA          '0'
    spoofed    %attvals.yesno; 'unknown'
    interface  CDATA          #IMPLIED
>
```

The Source class has three attributes:

ident

Optional. A unique identifier for this Source class (see [Section 3.4.9](#)).

spoofed

Optional. An indication of confidence as to whether this is the true Attack source. The permitted values for this attribute are shown below. The default value is "unknown".

Rank	Keyword	Description
----	-----	-----
0	unknown	Accuracy of source information unknown
1	yes	Source is believed to be a decoy
2	no	Source is believed to be "real"

interface

Optional. Specifies the interface on which the source of the event(s) was detected.

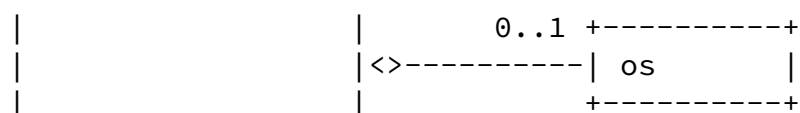
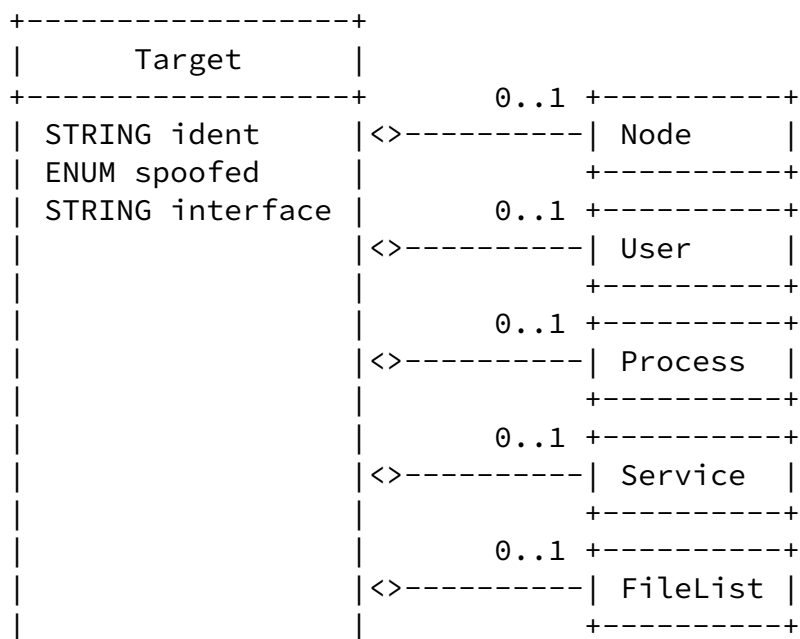
4.6.3 The Target Class

The Target class contains information about the possible target(s) of the incident event(s). An event may have more than one target (e.g., in the case of a port sweep).

For the purpose of compatibility, the Target class has been reused from the IDMEF. Hence, the Target class from an IDMEF message can be included unmodified into the IODEF-Description class with the same semantics. Likewise, the data in an IDMEF-originating Source class could be decomposed between the IODEF Target and Attack classes.

The definition of the Target class in the IODEF data model is a superset of the IDMEF definition. Two new classes have been added: OS and program.

The Target class is composed of four aggregate classes, as shown in Figure 4.8.



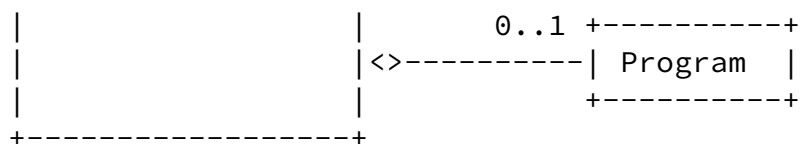


Figure 4.8 The Target Class

The aggregate classes that constitute Target are:

Node

Zero or one. Information about the host or device at which the event(s) (network address, network name, etc.) is being directed.

User

Zero or one. Information about the user at which the event(s) is being directed.

Process

Zero or one. Information about the process at which the event(s) is being directed.

Service

Zero or one. Information about the network service involved in the event(s).

FileList

Zero or one. Information about file(s) involved in the event(s).

os

Zero or one. The operation system running on the targeted Node.

program

Zero or one. The program running as the Process, which was targeted in the Attack.

This is represented in the XML DTD as follows:

```

<!ENTITY % attvals.yesno    "
    ( unknown | yes | no )
">
<!ELEMENT Target    (

```

```

    Node?, User?, Process?, Service?, FileList?, os?, program?
  )>
  <!ATTLIST Target
    ident      CDATA          '0'
    decoy      %attvals.yesno; 'unknown'
    interface  CDATA          #IMPLIED
  >

```

The Target class has three attributes:

ident

Optional. A unique identifier for this Target class (see [Section 3.4.9](#)).

spoofed

Optional. An indication of confidence as to whether this is the true Attack target. The permitted values for this attribute are shown below. The default value is "unknown".

Rank	Keyword	Description
----	-----	-----
0	unknown	Accuracy of target information unknown
1	yes	Target is believed to be a decoy
2	no	Target is believed to be "real"

interface

Optional. Specifies the interface on which the event(s) against the Target were detected.

4.6.4 The Method Class

The Method class provides information about the method used by the Attacker in the incident. This class can reference well-known vulnerability or exploit databases, as well as allow for a free-form description of the activity.

The Method class is composed of two aggregate classes, as shown in Figure 4.9.

```

+-----+
| Method |
+-----+
| STRING ident | 0..* +-----+
| ENUM restriction |<>-----| Classification |
|               |               +-----+

```

		0..* +-----+
	<>-----	Description
+-----+		+-----+

Figure 4.9 The Method Class

The aggregate classes that constitute Method are:

Classification

Zero or more. A reference to a well-known vulnerability or exploit databases.

Description

Zero or more. A free-form text description of the attack.

This is represented in the XML DTD as follows:

```
<!ELEMENT Method (
  Classification*, Description*)>
<!ATTLIST Method
  ident          ID              #IMPLIED
  restriction     %attvals.restriction;  'default'
>
```

The Method class has two attributes:

ident

Optional. A unique identifier for the element (see [Section 4.4.9](#)).

restriction

Optional. Sets a restriction on the usage of the data in element.

4.6.5 The Attacker Class

The Attacker class augments information found in the Source

class with further details related to the entity(ies)/person(s) identified as the source(s) of the incident activity.

NOTE: Information found in the Attacker class might be derived based on address and network information found in the Source class. However, particular algorithm or procedure is defined by Incident Handling System

```
+-----+
| Attacker |
+-----+
```

```
| STRING ident | 0..1 +-----+
| ENUM restriction | <>-----| Contact |
| | | +-----+
| | | 0..1 +-----+
| | | <>-----| Location |
| | | +-----+
| | | 0..1 +-----+
| | | <>-----| IRTcontact |
+-----+ +-----+
```

Figure 4.10 The Attacker Class

The aggregate classes that constitute Attacker are:

Contact

Zero or one. Contact information for the entity/person identified as an Attacker.

Location

Zero or one. Location of Attacker's node or system. This is a general definition of location that may depend on network structure or company's geographical distribution.

IRTcontact

Zero or one. Contact information for the CSIRT or Network Security manager serving the Node's network.

Attacker is represented in the XML DTD as follows:

```
<!ELEMENT Attacker (
  Contact?, Location?, IRTcontact?)>
<!ATTLIST Attacker
  ident          ID          #IMPLIED
  restriction     %attvals.restriction;  'default'
>
```

The Attacker class has two attributes:

ident
Optional. A unique identifier for the Attacker, see [Section 4.4.9](#).

restriction
Optional. Sets a restriction on the usage of the data in element.

4.6.6 The Victim Class

The Victim class augments information found in the Target class with further details related to the entity(ies)/person(s) identified as the target(s) of the incident activity.

NOTE: Information found in the Victim class might be derived based on address and network information found in the Target class. However, particular algorithm or procedure is defined by Incident Handling System.

```
+-----+
| Victim |
+-----+
| STRING ident      | 0..1 +-----+
| ENUM restriction  | <>-----| Contact      |
|                  |          +-----+
|                  | 0..1 +-----+
|                  | <>-----| Location      |
|                  |          +-----+
|                  | 0..1 +-----+
```

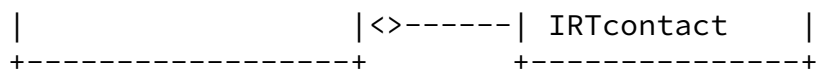


Figure 4.11 The Victim Class

The aggregate classes that constitute Victim are:

Contact

Zero or one. Contact information for the entity/person identified as a Victim.

Location

Zero or one. Location of Victim's node or system. This is a general definition of location that may depend on network structure or company's geographical distribution.

IRTcontact

Zero or one. Contact information for the CSIRT or Network Security manager serving the Node's network.

Victim is represented in the XML DTD as follows:

```

<!ELEMENT Victim (
  Contact?, Location?, IRTconact?)>
<!ATTLIST Victim
  ident ID #IMPLIED
  restriction %attvals.restriction; 'default'
>
  
```

The Victim class has two attributes:

ident

Optional. A unique identifier for the Victim element, see [Section 4.4.9](#).

restriction

Optional. Sets a restriction on the usage of the data in element.

4.6.7 The Evidence Class

The Evidence class contains evidence related to the current incident. This evidence may consist of multiple pieces of data, each in a different format, including textual information (e.g., logfiles, malicious scripts, list of changes in file system) and binary (e.g., disk images) objects.

```
+-----+
| Evidence      |
+-----+
| STRING ident  | 0..* +-----+
| ENUM restriction |<>-----| EvidenceData |
+-----+                +-----+
```

Figure 4.12 The Evidence Class

The aggregate class that constitutes Evidence is:

EvidenceData

Zero or more. Container for evidence data related to the current incident.

Evidence is represented in the XML DTD as follows:

```
<!ELEMENT Evidence    (
    EvidenceData*)>

<!ATTLIST Evidence
    ident          ID          #IMPLIED
    restriction    %attvals.restriction;  'default'
>
```

The Evidence class has two attributes:

restriction

Optional. Sets a restriction on the usage of the data in element.

ident

Optional. A unique identifier for the Evidence element, see [Section 4.4.9](#).

4.6.8 The Assessment Class

The Assessment class is used to provide the CSIRT's assessment of an event - its impact, actions taken in response, and confidence. For the purpose of compatibility the Assessment Class is reused from the IDMEF.

The Assessment class is composed of three aggregate classes, as shown in Figure 4.13.

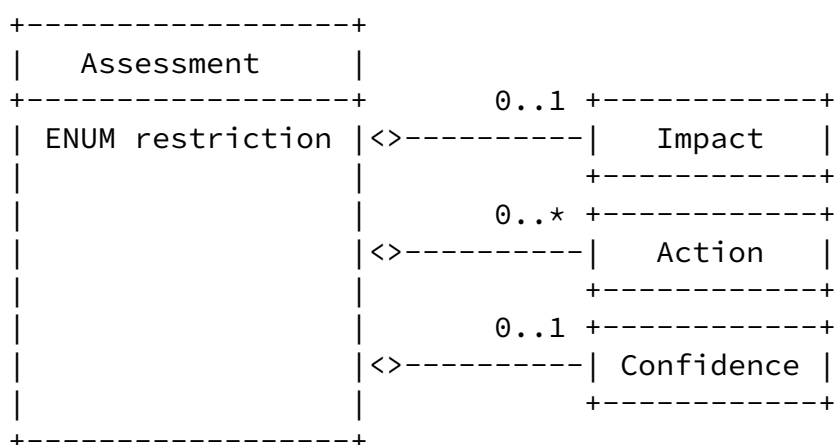


Figure 4.13 - The Assessment Class

The aggregate classes that make up Assessment are:

Impact

Zero or one. The CSIRT's assessment of the impact of the event on the target(s).

Action

Zero or more. The action(s) taken by the CSIRT in response to the event.

Confidence

A measurement of the confidence the CSIRT has in its evaluation of the event.

This is represented in the XML DTD as follows:

```
<!ELEMENT Assessment (
    Impact?, Action*, Confidence?
```

```
)>
<!ATTLIST Assessment
    restriction          %attvals.restriction;    'default'
    %attlist.global;
>
```

4.6.8.1 The Impact Class

The Impact class is used to provide the CSIRT's assessment of the impact of the event on the target(s). It is represented in the XML DTD as follows:

```
<!ENTITY % attvals.severity          "
    ( low | medium | high )
">
<!ENTITY % attvals.completion        "
    ( failed | succeeded )
">
<!ENTITY % attvals.impacttype        "
    ( admin | dos | file | recon | user | other )
">
<!ELEMENT Impact      (#PCDATA | EMPTY)* >
<!ATTLIST Impact
    severity          %attvals.severity;      #IMPLIED
    completion        %attvals.completion;    #IMPLIED
    type              %attvals.impacttype;    'other'
    %attlist.global;
>
```

The Impact class has three attributes:

severity

An estimate of the relative severity of the event. The permitted values are shown below. There is no default value.

Rank	Keyword	Description
----	-----	-----
0	low	Low severity
1	medium	Medium severity
2	high	High severity

completion

An indication of whether the CSIRT believes the attempt that the event describes was successful or not. The permitted values are shown below. There is no default value.

Rank	Keyword	Description
----	-----	-----
0	failed	The attempt was not successful
1	succeeded	The attempt succeeded

type

The type of attempt represented by this event, in relatively broad categories. The permitted values are shown below. The default value is "other."

Rank	Keyword	Description
----	-----	-----
0	admin	Administrative privileges were attempted or obtained
1	dos	A denial of service was attempted or completed
2	file	An action on a file was attempted or completed
3	recon	A reconnaissance probe was attempted or completed
4	user	User privileges were attempted or obtained
5	other	Anything not in one of the above categories

All three attributes are optional. The element itself may be empty, or may contain a textual description of the impact, if the CSIRT is able to provide additional details.

4.6.8.2 The Action Class

The Action class is used to describe any actions taken by

the CSIRT owning current Incident Object in course of its handling or investigating. It is represented in the XML DTD as follows:

```
<!ENTITY % attvals.actioncat          "  
    ( block-installed | notification-sent | taken-offline |  
      other )  
">  
<!ELEMENT Action      (#PCDATA | EMPTY)* >  
<!ATTLIST Action  
    category          %attvals.actioncat;      'other'  
    %attlist.global;  
>
```

Action has one attribute:

category

Optional. The type of action taken by CSIRT or automatic Intrusion detection tools. The permitted values are shown below. The default value is "other."

Rank	Keyword	Description
----	-----	-----
0	block-installed	A block of some sort was installed to prevent an attack from reaching its destination. The block could be a port block, address block, etc., or disabling a user account.
1	notification-sent	A notification message of some sort was sent out-of-band (via pager, e-mail, etc.). Does not include the transmission of this alert.
2	taken-offline	A system, computer, or user was taken offline, as when the computer is shut down or a user is logged off.
3	other	Anything not in one of the above categories.

The element itself may be empty, or may contain a textual description of the action, if the description of the taken actions needs to be expressed in free language.

4.6.8.3 The Confidence Class

The Confidence class is used to represent the CSIRT's best estimate of the validity of its Incident Assessment. It is represented in the XML DTD as follows:

```
<!ENTITY % attvals.rating          "  
    ( low | medium | high | numeric )  
">  
<!ELEMENT Confidence (#PCDATA | EMPTY)* >  
<!--ATTLIST Confidence  
    rating          %attvals.rating;          'numeric'  
    %attlist.global;  
>
```

The Confidence class has one attribute:

rating
The CSIRT's rating of its assessment validity. The

permitted values are shown below. The default value is "numeric."

Rank	Keyword	Description
----	-----	-----
0	low	The CSIRT/triage has little confidence in its validity
1	medium	The CSIRT/triage has average confidence in its validity
2	high	The CSIRT/triage has high confidence in its validity
3	numeric	The CSIRT/triage has provided a posterior probability value indicating its confidence in its validity

This element should be used only when the CSIRT/triage can produce meaningful information. Systems that can output only a rough heuristic should use "low", "medium", or "high" as the rating value. In this case, the element content should be omitted.

Systems capable of producing reasonable probability estimates should use "numeric" as the rating value and include a numeric confidence

value in the element content. This numeric value should reflect a posterior probability (the probability that an attack has occurred given the data seen by the detection system and the model used by the system). It is a floating point number between 0.0 and 1.0, inclusive. The number of digits should be limited to those representable by a single precision floating point value, and may be represented as described in [Section 3.4.2](#).

NOTE:

It should be noted that different types of Incident handling Systems may compute confidence values in different ways and that in many cases, confidence values from different CSIRTs should not be compared (for example, if the CSIRTs use different methods of computing or representing confidence, or are of different types or configurations). Care should be taken when implementing systems that process confidence values (such as event correlators) not to make comparisons or assumptions that cannot be supported by the system's knowledge of the environment in which it is working.

4.6.9 The Authority Class

The Authority class names and provides contact information for the CSIRT who created and is handling the incident.

+-----+

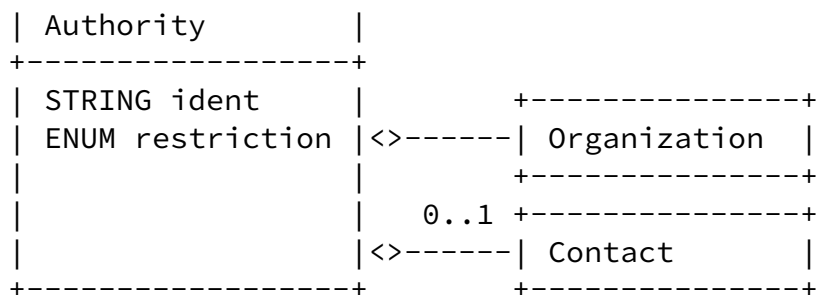


Figure 4.14 The Authority Class

The aggregate classes that constitute Authority are:

Organization

Exactly one. Name or organization handling the current incident.

Contact

Zero or one. Contact information for the Organization handling the incident.

Authority is represented in the XML DTD as follows:

```
<!ELEMENT Authority (
  Organization, Contact? )>
<!ATTLIST Authority
  ident ID #IMPLIED
  restriction %attvals.restriction; 'default'
>
```

The Authority class has two attributes:

ident

Optional. A unique identifier for the Authority element (see [Section 3.4.9](#)).

4.6.10 The History Class

History class maintains a log of significant events that occurred in the course of handling the incident. This class tracks who initially reported the incident, documents the

interaction between the reported and the investigating CSIRT, and lists any other actions taken by the CSIRT. When handling evidence, the History class can provide a chain of custody.

The History class is composed of three aggregate classes, as shown in Figure 4.15.

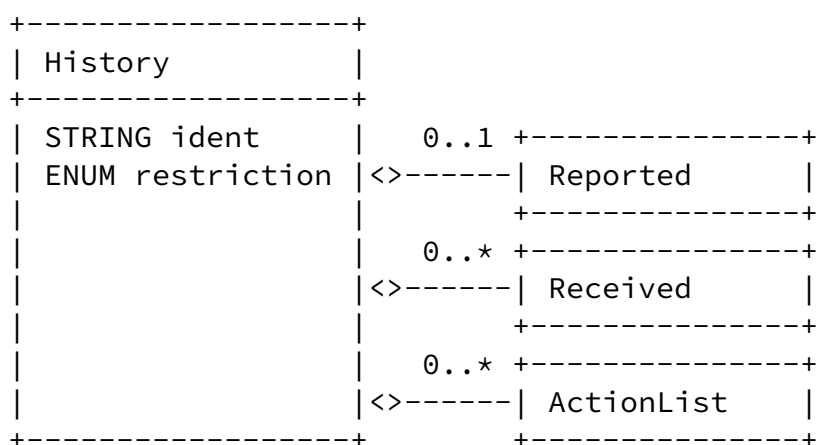


Figure 4.15 The History Class

The aggregate classes that constitute History are:

Reported

Zero or one. Identifies who initially reported the incident.

Received

Zero or more. The communications of the CSIRT when handling the incident.

ActionList

Zero or more. The actions taken by the CSIRT in the course of handling the incident.

This is represented in the XML DTD as follows:

```

<!ELEMENT History (
  Reported?, Received*, ActionList* )>
<!ATTLIST History
  ident ID #IMPLIED
  restriction %attvals.restriction; 'default'
>

```

The History class has two attributes:

ident

Optional. A unique identifier for the History element ([Section 3.4.9](#)).

restriction

Optional. Sets a restriction on the usage of the data in element.

4.6.11 The AdditionalData Class

The AdditionalData class is used to provide information that cannot be represented by the data model. AdditionalData can be used to provide atomic data (integers, strings, etc.) in cases where only small amounts of additional information needed to be represented. However, the class can also be used to extend the data model and the DTD to support proprietary IODEF extensions or for encapsulating external XML document such as IDMEF messages. Detailed instructions for extending the data model and the DTD are provided in [Section 5](#).

The AdditionalData element is declared in the XML DTD as follows:

```
<!ENTITY % attvals.adtype    "  
    ( boolean | byte | character | date-time | integer |  
      ntpstamp | portlist | real | string | xml )  
">  
<!ELEMENT AdditionalData    ANY >  
<!--ATTLIST AdditionalData  
    type      %attvals.adtype;    'string'  
    local    CDATA                #IMPLIED  
>
```

The AdditionalData class has two attributes:

type

Required. The type of data included in the element content. The permitted values for this attribute are shown below. The default value is "string".

Rank	Keyword	Description
----	-----	-----
0	boolean	The element contains a boolean value, i.e., the strings "true" or "false"
1	byte	The element content is a single 8-bit byte (see Section 3.4.4)
2	character	The element content is a single character (see Section 3.4.3)
3	date-time	The element content is a date-time string

		(see Section 3.4.6)
4	integer	The element content is an integer (see Section 3.4.1)
5	ntpstamp	The element content is an NTP timestamp (see Section 3.4.7)
6	portlist	The element content is a list of ports (see Section 3.4.8)
7	real	The element content is a real number (see Section 3.4.2)
8	string	The element content is a string (see Section 3.4.3)
9	xml	The element content is XML-tagged data (see Section 5.2)

local

Optional. A string describing the meaning of the element content if used by CSIRT for a purpose not described in this document. These values will be vendor/implementation dependent. The method for ensuring that managers understand the string is outside the scope of this specification.

4.7 The Time Classes

The data model provides four classes for representing time. The three classes DetectTime, StartTime, EndTime are aggregates of the Attack classes. The support DateTime class is used for marking up date and time information in the aggregate classes EventList, ActionList, Reported and Received.

The definition of the Time classes in this document are the same as in the IDMEF to ensure compatibility.

4.7.1 The DetectTime Class

The time when the incident activity was first detected by the reporter.

It is represented in the XML DTD as follows:

```
<!ELEMENT DetectTime          (#PCDATA) >
<!ATTLIST DetectTime
    ntpstamp          CDATA          #REQUIRED
>
```

The DATETIME format of the <DetectTime> element content is described in [Section 3.4.6](#).

The DetectTime class has one attribute:

ntpstamp

Required. The NTP timestamp representing the same date and time as the element content. The NTPSTAMP format of this attribute's value is described in [Section 3.4.7](#).

If the date and time represented by the element content and the NTP timestamp differ (should "never" happen), the value in the NTP timestamp MUST be used.

4.7.2 The StartTime Class

The start time of the incident activity.

It is represented in the XML DTD as follows:

```
<!ELEMENT StartTime          (#PCDATA) >
```

The DATETIME format of the <StartTime> element content is described in [Section 3.4.6](#).

4.7.3 The EndTime Class

The end time of the incident activity.

It is represented in the XML DTD as follows:

```
<!ELEMENT EndTime          (#PCDATA) >
```

The DATETIME format of the <StartTime> element content is described in [Section 3.4.6](#).

4.7.4 The DateTime Class

The supportive class to mark up date and time information.

It is represented in the XML DTD as follows:

```
<!ELEMENT DateTime          (#PCDATA) >
```

The DATETIME format of the <DateTime> element content is described in [Section 3.4.6](#).

4.8 The Support Classes

The support classes constitute the major part of the core classes and are shared between them.

The IODEF reuses a number of support classes from the IDMEF:

- + Node, Address, User, UserId, Process, Service - as compound classes for the Source and Target classes or for the Attacker and Victim classes;

- + WebService, SMTPService - as used in Service class;
- + Classification - as a component of the Method class.

4.8.1 The Node Class

The Node class is used to identify hosts and other network devices (routers, switches, etc.).

The Node class is composed of five aggregate classes, as shown in Figure 4.16.

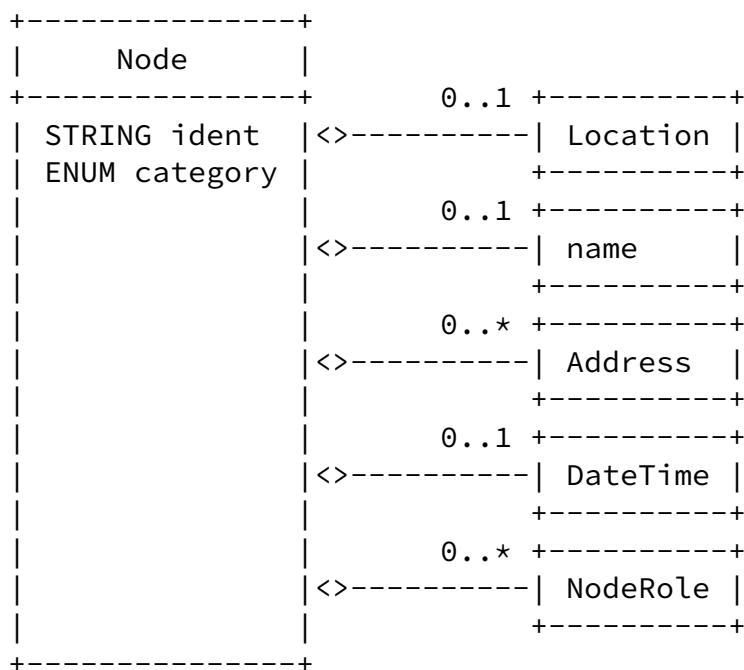


Figure 4.16 The Node Class

The aggregate classes that constitute Node are:

location

Zero or one. STRING. The physical location of the equipment.

name

Zero or one. STRING. The name of the equipment. This information MUST be provided if no Address information is given.

Address

Zero or more. The network or hardware address of the equipment. Unless a name (above) is provided, at least one address must be specified.

DateTime

Zero or one. Date and time when the resolution between the name and address was performed. This information SHOULD be provided if both an Address and name are given.

NodeRole

Zero or more. The intended role of the node.

This is represented in the XML DTD as follows:

```
<!ENTITY % attvals.nodecat
    ( unknown | ads | afs | coda | dfs | dns | hosts |
      kerberos | nds | nis | nisplus | nt | wfw )
">
<!ELEMENT Node
    (
      location?, (name | Address), Address*, DateTime?, NodeRole*
    )>
<!ATTLIST Node
    ident                CDATA                '0'
    category             %attvals.nodecat;     'unknown'
>
```

The Node class has two attributes:

ident

Optional. A unique identifier for the node, see [Section 3.4.9](#).

category

Optional. The "domain" from which the name information obtained, if relevant. The permitted values for this attribute are shown below. The default value is "unknown".

Rank	Keyword	Description
----	-----	-----
0	unknown	Domain unknown or not relevant

1	ads	Windows 2000 Advanced Directory Services
2	afs	Andrew File System (Transarc)
3	coda	Coda Distributed File System
4	dfs	Distributed File System (IBM)
5	dns	Domain Name System
6	hosts	Local hosts file
7	kerberos	Kerberos realm
8	nds	Novell Directory Services
9	nis	Network Information Services (Sun)
10	nisplus	Network Information Services Plus (Sun)
11	nt	Windows NT domain
12	wfw	Windows for Workgroups

4.8.1.1 The Address Class

The Address class represents a network, hardware, or application address.

The Address class is composed of two aggregate classes, as shown in Figure 4.17.

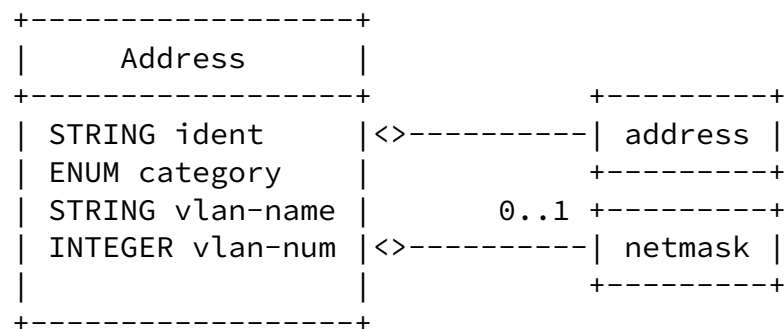


Figure 4.17 The Address Class

The aggregate classes that constitute Address are:

address

Exactly one. STRING. The address whose format is governed by the category attribute.

netmask

Zero or one. STRING. The network mask for the address, if appropriate.

This is represented in the XML DTD as follows:

```

<!ENTITY % attvals.addrcat
    ( unknown | atm | e-mail | lotus-notes | mac | sna | vm |

```

```

    ipv4-addr | ipv4-addr-hex | ipv4-net | ipv4-net-mask |
    ipv6-addr | ipv6-addr-hex | ipv6-net | ipv6-net-mask )
">
<!ELEMENT Address
    address, netmask?
)>
<!ATTLIST Address
    ident          CDATA          '0'
    category       %attvals.addrcat; 'unknown'
    vlan-name      CDATA          #IMPLIED
    vlan-num       CDATA          #IMPLIED
>

```

The Address class has four attributes:

ident

Optional. A unique identifier for the address (see [Section 3.4.9](#)).

category

Optional. The type of address represented. The permitted values for this attribute are shown below. The default value is "unknown".

Rank	Keyword	Description
----	-----	-----
0	unknown	Address type unknown
1	atm	Asynchronous Transfer Mode network address
2	e-mail	Electronic mail address (RFC 822)
3	lotus-notes	Lotus Notes e-mail address
4	mac	Media Access Control (MAC) address
5	sna	IBM Shared Network Architecture (SNA) address
6	vm	IBM VM ("PROFS") e-mail address
7	ipv4-addr	IPv4 host address in dotted-decimal notation (a.b.c.d)

8	ipv4-addr-hex	IPv4 host address in hexadecimal notation
9	ipv4-net	IPv4 network address in dotted-decimal notation, slash, significant bits (a.b.c.d/nn)
10	ipv4-net-mask	IPv4 network address in dotted-decimal notation, slash, network mask in dotted-decimal notation (a.b.c.d/w.x.y.z)
11	ipv6-addr	IPv6 host address
12	ipv6-addr-hex	IPv6 host address in hexadecimal notation

13	ipv6-net	IPv6 network address, slash, significant bits
14	ipv6-net-mask	IPv6 network address, slash, network mask

vlan-name

Optional. The name of the Virtual LAN to which the address belongs.

vlan-num

Optional. The number of the Virtual LAN to which the address belongs.

4.8.1.2 The NodeRole Class

The NodeRole class is used to represent the intended role of a particular node.

The NodeRole class is composed of a single attribute represented in the XML DTD as follows:

```
<!ENTITY % attvals.noderolecat
    ( unknown | client | server-internal | server-public |
      www | mail | messaging | streaming | voice | file |
      ftp | p2p | name | directory | credential | print |
      application | database | infra | log )
">
```

```

<!ELEMENT NodeRole ()>
<!--ATTLIST NodeRole
      category    %attvals.noderolecat;    'unknown'
-->

```

The NodeRole class has one attribute:

category

Optional. The intended role this Node is to fulfill. The permitted values for this attribute are shown below. The default value is "unknown".

Rank	Keyword	Description
----	-----	-----
0	unknown	Unknown role
1	client	Client computer
2	server-internal	Server with internal services
3	server-public	Server with public services
4	www	WWW server
5	mail	Mail server

6	messaging	Messaging server (e.g. NNTP, IRC, instant)
7	streaming	Streaming-media server
8	voice	Voice server (e.g. SIP, H.323)
9	file	File server (e.g. SMB, CVS, AFS)
10	ftp	FTP server
11	p2p	Peer-to-peer server (e.g. Napster)
12	name	Name server (e.g. DNS, WINS)
13	directory	Directory server (e.g. LDAP, finger, whois)
14	credential	Credential server (e.g. domain controller, Kerberos)
16	print	Print server
17	application	Application server
18	database	Database server
19	infra	Infrastructure server (e.g. router, firewall, DHCP)
20	log	Log server (e.g. syslog)

4.8.2 The User Class

The User class describes a user account on a system. It is primarily used as a "container" class for the UserId aggregate class, as shown in Figure 4.18.

More than one UserId can be used within the User class to indicate attempts to transition from one user to another, or to provide complete information about a user's (or process') privileges.

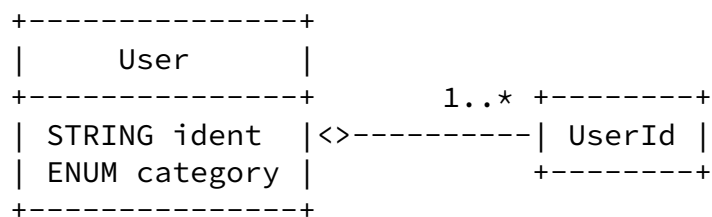


Figure 4.18 The User Class

The aggregate class contained in User is:

UserId
One or more. The user.

This is represented in the XML DTD as follows:

```
<!ENTITY % attvals.usercat "
```

```
( unknown | application | os-device )
">
<!ELEMENT User (
  UserId+
)>
<!ATTLIST User
  ident          CDATA          '0'
  category       %attvals.usercat; 'unknown'
>
```

The User class has two attributes:

ident

Optional. A unique identifier for the user (see [Section 3.4.9](#)).

category

Optional. The type of user represented. The permitted values for this attribute are shown below. The default value is "unknown".

Rank	Keyword	Description
----	-----	-----
0	unknown	User type unknown
1	application	An application user
2	os-device	An operating system or device user

4.8.2.1 The UserId Class

The UserId class describes a specific user account on a system.

The UserId class is composed of two aggregate classes, as shown in Figure 4.19.

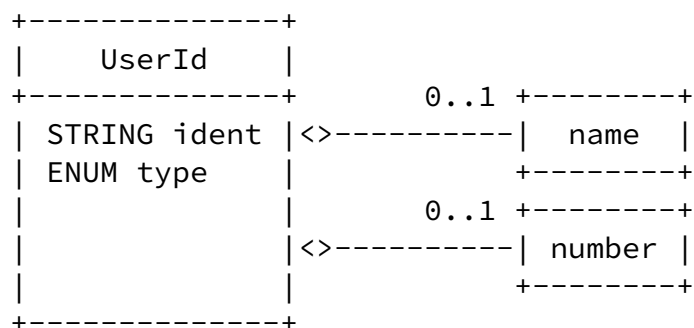


Figure 4.19 The UserId Class

The aggregate classes that constitute UserId are:

name
Zero or one. STRING. A user or group name.

number
Zero or one. INTEGER. A user or group number.

This is represented in the XML DTD as follows:

```
<!ENTITY % attvals.idtype
    ( current-user | original-user | target-user |
      user-privs | current-group | group-privs | other-privs)
">
<!ELEMENT UserId
    (
      name | number | (name, number)
    )>
<!ATTLIST UserID
    ident    CDATA          '0'
    type     %attvals.idtype; 'original-user'
>
```

The UserId class has two attributes:

ident
Optional. A unique identifier for the user id (see [Section 3.4.9](#)).

type
Optional. The type of user information represented. The permitted values for this attribute are shown below. The default value is "original-user".

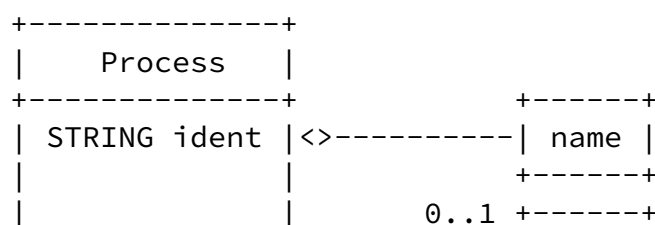
Rank	Keyword	Description
----	-----	-----
0	current-user	The current user id being used by the user or process. On Unix systems, this would be the "real" user id.
1	original-user	The actual identity of the user or process being reported on. On those systems that (a) do some type of auditing and (b) support extracting a user id from the "audit id" token, that value should be used. On those systems that do not support this, and where the user has logged into the system, the "login id" should be used.
2	target-user	The user id the user or process is attempting to become. For example,

- on Unix systems when the user attempts to use "su," "rlogin," "telnet," etc.
- 3 user-privs Another user id the user or process has the ability to use. On Unix systems, this would be the "effective" user id. Multiple UserId elements of this type may be used to specify a list of privileges.
 - 4 current-group The current group id (if applicable) being used by the user or process. On Unix systems, this would be the "real" group id.
 - 5 group-privs Another group id the group or process has the ability to use. On Unix systems, this would be the "effective" group id. On BSD-derived Unix systems, multiple UserId elements of this type would be used to include all the group ids on the "group list."
 - 6 other-privs Not used in a user, group, or process context, only used in the file context. The file permissions assigned to users who do not match either the user or group permissions on the file. On Unix systems, this would be the "world" permissions.

4.8.3 The Process Class

The Process class describes a process being executed on a system.

The Process class is composed of five aggregate classes, as shown in Figure 4.20.



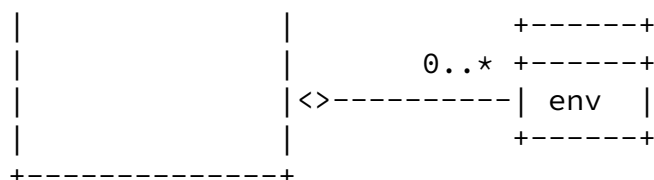
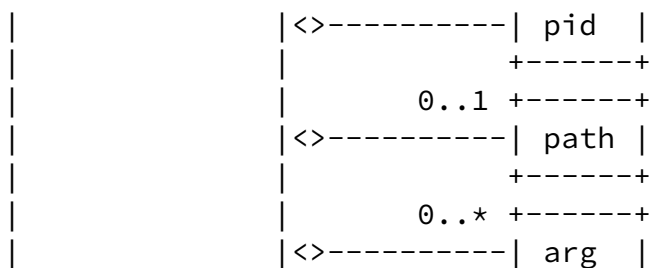


Figure 4.20 The Process Class

The aggregate classes that constitute Process are:

name

Exactly one. STRING. The filename of the program being executed. This is a short name; path and argument information are provided elsewhere.

pid

Zero or one. INTEGER. The process identifier of the process.

path

Zero or one. STRING. The full path of the program being executed.

arg

Zero or more. STRING. A command-line argument to the program. Multiple arguments may be specified (they are assumed to have occurred in the same order they are provided) with multiple uses of arg.

env

Zero or more. STRING. An environment string associated with the process; generally of the format "VARIABLE=value". Multiple environment strings may be specified with multiple

uses of env.

This is represented in the XML DTD as follows:

```
<!ELEMENT Process (
  name, pid?, path?, arg*, env*
)>
<!ATTLIST Process
  ident          CDATA          '0'
>
```

The Process class has one attribute:

ident
Optional. A unique identifier for the process (see [Section 3.4.9](#)).

4.8.4 The Service Class

The Service class describes a network service. It can identify services by name, port, and protocol.

When Service occurs as an aggregate class of Source, it is understood that the service is one from which activity of interest is originating; and that the service is "attached" to the Node, Process, and User information also contained in Source. Likewise, when Service occurs as an aggregate class of Target, it is understood that the service is one to which activity of interest is being directed; and that the service is "attached" to the Node, Process, and User information also contained in Target.

The Service class is composed of four aggregate classes, as shown in Figure 4.21.

```
+-----+
|  Service  |
+-----+          0..1 +-----+
| STRING ident |<>-----|  name  |
```

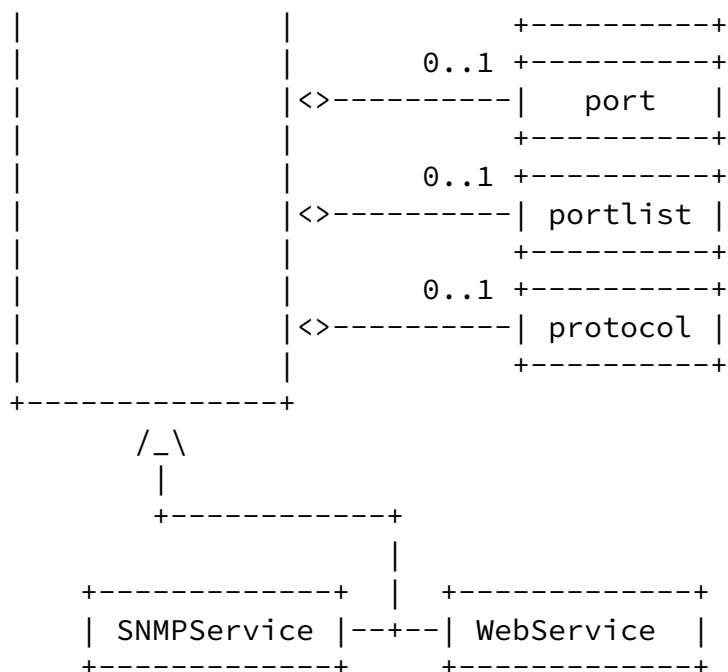


Figure 4.21 - The Service Class

The aggregate classes that constitute Service are:

name

Zero or one. STRING. The name of the service. Whenever

possible, the name from the IANA list of well-known ports SHOULD be used.

port

Zero or one. INTEGER. The port number being used.

portlist

Zero or one. PORTLIST. A list of port numbers being used; see [Section 3.4.8](#) for formatting rules.

protocol

Zero or one. STRING. The protocol being used.

A Service MUST be specified as either (a) a name, (b) a port,

(c) a name and a port, or (d) a portlist. The protocol is optional in all cases, but no other combinations are permitted.

Because DTDs do not support subclassing (see [Section 4.3.4](#)), the inheritance relationship between Service and the SNMPService and WebService subclasses shown in Figure 5.17 has been replaced with an aggregate relationship. Service is represented in the XML DTD as follows:

```
<!ELEMENT Service (
    ((name | port | (name, port)) | portlist), protocol?,
    SNMPService?, WebService?)>
<!ATTLIST Service
    ident          CDATA          '0'
>
```

The Service class has one attribute:

ident
Optional. A unique identifier for the service, see [Section 3.4.9](#).

4.8.4.1 The WebService Class

The WebService class augments the Service class with additional information related to web traffic.

The WebService class is composed of four aggregate classes, as shown in Figure 4.22.

```
+-----+
|  Service  |
```

```
+-----+
|  /_ \  |
|    |   |
+-----+
| WebService |
+-----+
+-----+
```

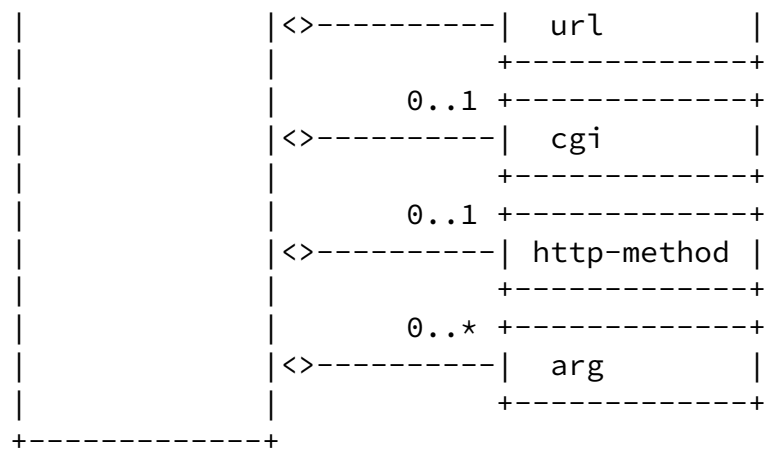


Figure 4.22 The WebService Class

The aggregate classes that constitute WebService are:

url

Exactly one. STRING. The URL in the request.

cgi

Zero or one. STRING. The CGI script in the request, without arguments.

http-method

Zero or one. STRING. The HTTP method (PUT, GET) used in the request.

arg

Zero or more. STRING. The arguments to the CGI script.

This is represented in the XML DTD as follows:

```
<!ELEMENT WebService (
  url, cgi?, http-method?, arg*
)>
```

4.8.4.2 The SNMPService Class

The SNMPService class augments the Service class with additional information related to SNMP traffic.

The SNMPService class is composed of three aggregate classes, as shown in Figure 4.23.

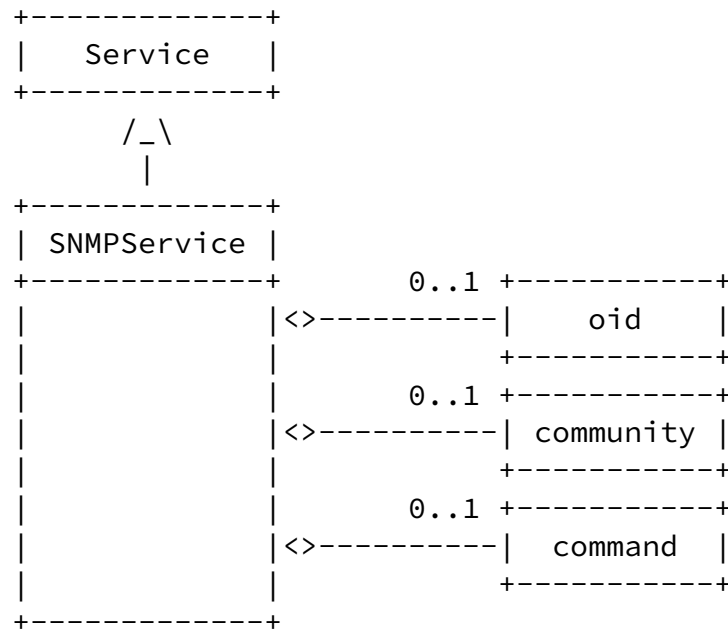


Figure 4.23 The SNMPService Class

The aggregate classes that constitute SNMPService are:

oid

Zero or one. STRING. The object identifier in the request.

community

Zero or one. STRING. The object's community string.

command

Zero or one. STRING. The command sent to the SNMP server (GET, SET. etc.).

This is represented in the XML DTD as follows:

```

<!ELEMENT SNMPService (
    oid?, community?, command?
)>

```

4.8.5 The Classification Class

The Classification class provides a way to reference an external vulnerability, exposure, or virus database.

The Classification class is composed of two aggregate classes,

as shown in Figure 4.24.

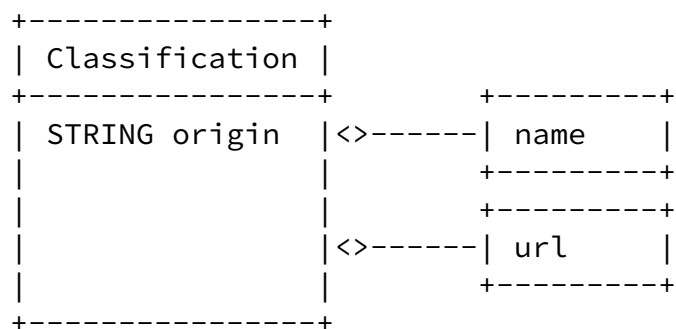


Figure 4.24 The Classification Class

The aggregate classes that constitute Classification are:

name

Exactly one. STRING. The name of the Vulnerability, Exposure or Virus (from one of the origins listed below) used by Attacker to cause Incident.

url

Exactly one. STRING. A URL at which the manager can find additional information about classified method. The document pointed to by the URL may include an in-depth description of the attack, appropriate countermeasures, or other information deemed relevant by the vendor.

This is represented in the XML DTD as follows:

```
<!ENTITY % attvals.origin          "
    ( unknown | bugtraqid | cve | vendor-specific )
">
<!ELEMENT Classification            (
    name, url
)>
<!ATTLIST Classification
    origin          %attvals.origin;          'unknown'
>
```


The Classification class has one attribute:

origin

Required. The source from which the name of the alert originates. The permitted values for this attribute are shown below. The default value is "unknown".

Rank	Keyword	Description
------	---------	-------------

Meijer, et al.	Expires October 2002	[page 75]
----------------	----------------------	-----------

Internet Draft	draft-ietf-inch-iodef-00.txt	Apr 2002
----------------	--	----------

Rank	Keyword	Description
0	unknown	Origin of the name is not known
1	bugtraqid	The SecurityFocus.com ("Bugtraq") vulnerability database identifier (http://www.securityfocus.com/vdb)
2	cve	The Common Vulnerabilities and Exposures (CVE) name (http://www.cve.mitre.org/)
3	vendor-specific	A vendor-specific name (and hence, URL); this can be used to provide product-specific information

4.8.6 The EvidenceData Class

The EvidenceData class contains textual (e.g., logfiles, malicious scripts, list of changes if file system, etc.) and binary (e.g., disc images) evidence data related to current Incident.

```

+-----+
| Evidence |
+-----+
      /_ \
      |
+-----+
| EvidenceData |
+-----+
| STRING ident |<-----0..* +-----+
| ENUM restriction |          | CorrEvidence |
+-----+

```

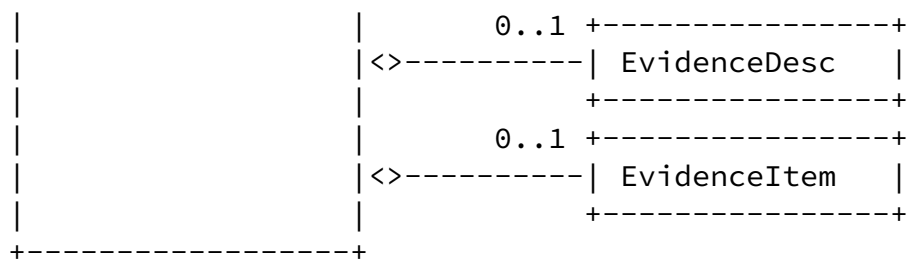


Figure 4.25 The EvidenceData Class

The aggregate classes that constitute EvidenceData are:

CorrEvidence

Zero or more. EvidenceData of the Evidence class that contains Evidence data correlated with current Evidence data.

EvidenceDesc

Zero or one. Description of the evidence found in the

EvidenceItem class.

EvidenceItem

Zero or one. Container for a particular piece of evidence data.

This is represented in the XML DTD as follows:

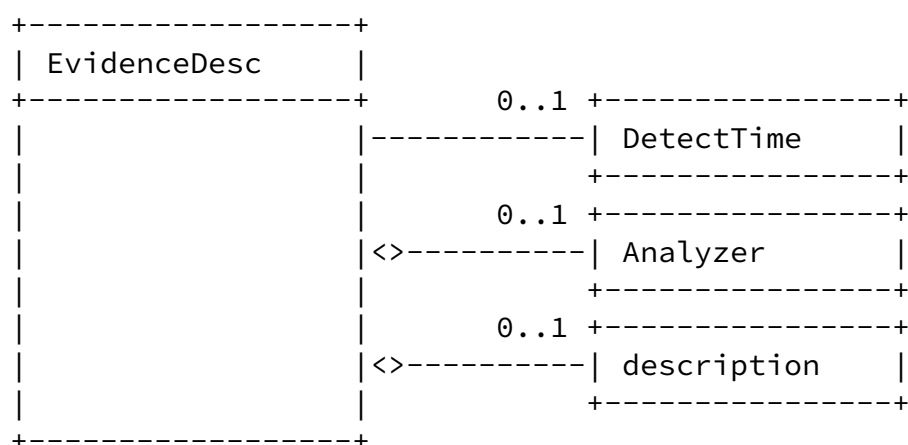
```
<!ENTITY % attvals.restriction          "
( default | public | internal |
  restricted )
">
<!ELEMENT EvidenceData                  (
  CorrEvidence*, EvidenceDesc?, EvidenceItem?
)>
<!ATTLIST EvidenceData
  ident CDATA                           #IMPLIED
  restriction %attvals.restriction; #IMPLIED
>
```

The EvidenceData class has two attributes:

Optional. A unique identifier for this EvidenceData (see [Section 3.4.9](#)).

Optional. Sets a restriction on the usage of the data in element.

The EvidenceDesc class contains meta-information about evidence in the EvidenceItem class.



Zero or one. The facility used to gather the evidence.
The analyzer SHOULD define the name of the format,

facility, tool, or device used to generate the evidence if it is not self-describing (e.g. xml). Likewise, the analyzer SHOULD define the Node which detected the evidence or from which it was extracted if this information is not represented elsewhere.

description

Zero or one. Free-form text to make comments on or annotate the evidence.

This is represented in the XML DTD as follows:

```
<!ELEMENT EvidenceDesc (
    DetectTime?, Analyzer?, description?
)>
```

4.8.6.2 The EventList Class

The EventList class contains information about events which are treated as correlated with respect to current incident.

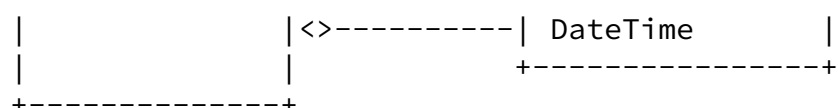
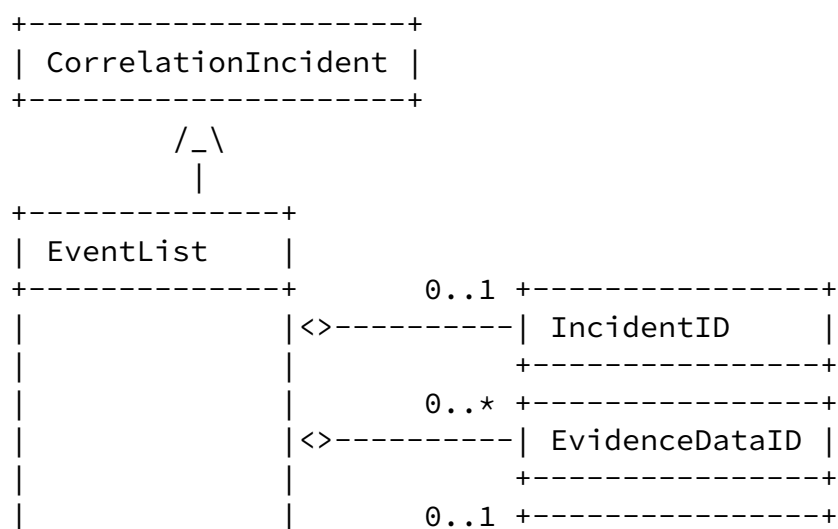


Figure 4.27 The EventList Class

The aggregate classes that constitute EventList are:

IncidentID

Zero or one. Identification number of the Incident.

EvidenceDataID

Zero or more. Identification number of the EvidenceData element related to referenced event or IncidentID.

DateTime

Zero or one. Date and time when the event occurred.

EventList is represented in the XML DTD as follows:

```
<!ELEMENT EventList (
    IncidentID?, EvidenceDataID?, DateTime?)>
<!ATTLIST EventList
    ident ID #IMPLIED
>
```

The EventList class has one attributes:

ident

Optional. A unique identifier for the EventList element (see [Section 3.4.9](#)).

4.8.7 The Organization Class

The Organization class describes a CSIRT involved in incident handling. This class is a mandatory subordinate element of the mandatory Authority class.

```
+-----+
| Authority |
+-----+
    /_ \
    |
+-----+
| Organization |
+-----+
| STRING ident |<>-----| OrganizationID |
```

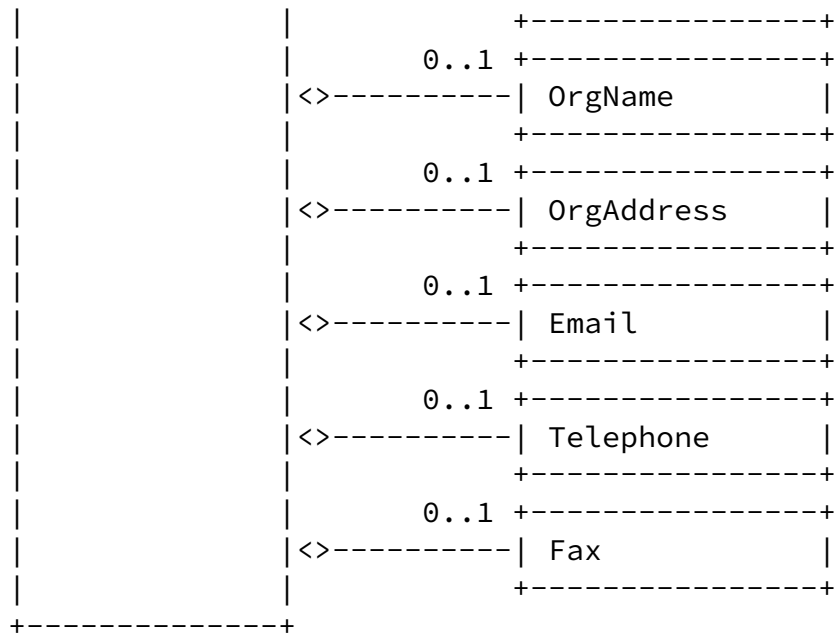


Figure 4.28 Organization Class

The aggregate classes that constitute the Organization class are:

OrganizationID

Zero or one. The identification number of the Organization. The ID can be derived from known registries such as RIPE NCC, TI, etc.

OrgName

Zero or one. Name of the organization as it used in official post address.

OrgAddress

Zero or one. Address of the organization.

Email

Zero or one. Email address of the organization.

Telephone

Zero or one. Telephone number of the organization.

Fax

Zero or one. Fax number of the organization.

At a minimum, the Organization class MUST have either a an

OrgName or OrganizationID.

Organization is represented in the XML DTD as follows:

```
<!ELEMENT Organization                               (
  (OrganizationID? | OrgName?), OrgAddress?,
  Email?, Telephone?, Fax? )>
<!ATTLIST Organization
  ident                ID                #IMPLIED
>
```

The Organization class has one attributes:

ident
Optional. A unique identifier for the Organization element
(see [Section 3.4.9](#)).

4.8.8 The Contact Class

The Contact Class contains contact information for a person or role in a CSIRT handling an incident.

```
+-----+
|  Authority  |
+-----+
      /\
      |
+-----+
| Contact    |
+-----+
| STRING ident | <>----- 0..1 +-----+
|               |               | PersonName |
|               |               +-----+
|               |               0..1 +-----+
|               | <>----- | PersonAddress |
|               |               +-----+
|               |               0..1 +-----+
|               | <>----- | ContactHandle |
|               |               +-----+
```

+-----+

Figure 4.29 Contact Class

The aggregate classes that constitute Contact class are:

PersonName

Zero or one. Name of the person responsible for handling the current incident.

ContactHandle

Meijer, et al.

Expires October 2002

[page 81]

Internet Draft

[draft-ietf-inch-iodef-00.txt](#)

Apr 2002

Zero or one. Identification number (or handle) used to refer to personal (or role) information in different Registries.

PersonAddress

Zero or one. Contact or physical Address of the person identified by PersonName.

Contact is represented in the XML DTD as follows:

```
<!ELEMENT Contact (
  PersonName?, ContactHandle?, PersonAddress?)>
<!ATTLIST Contact
  ident ID #IMPLIED
>
```

The Contact class has one attribute:

ident

Optional. A unique identifier for the Contact element (see [Section 3.4.9](#)).

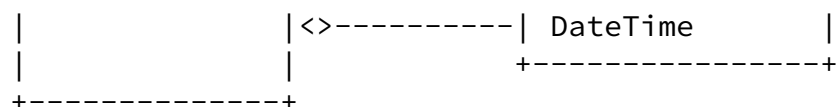
4.8.9 The Reported Class

The Reported class is subordinate class of the History class. It provided information about who and when reported current Incident, particularly identification number of the CSIRT that


```

+-----+
| History |
+-----+
      /_ \
      |
+-----+
| Reported |
+-----+
0..1 +-----+
| STRING ident | <>-----| AuthorityID |
|               | +-----+
|               | 0..1 +-----+
|               | <>-----| IncidentID |
|               | +-----+
|               | 0..1 +-----+

```



Zero or one. Date and time when message was received.

Reported is represented in the XML DTD as follows:

```
<!ELEMENT Reported (
  AuthorityID?, IncidentID?, DateTime?)>
<!ATTLIST Reported
  ident ID #IMPLIED
>
```

The Reported class has one attributes:

ident
Optional. A unique identifier for the Reported element, see [Section 3.4.9](#).

4.8.10 The Received Class

The Received class contains information about when and from whom the information about current incident was received. In particular case it may contain reference to message received from another CSIRT about current incident. This element contains only AuthorityID from maintained by CSIRT database or from definite public register like RIPE NCC database or Trusted Introducer CSIRT database, assuming that CSIRT normally can accept information only from trusted source.

```
+-----+
| History |
+-----+
/_\
```

```

      |
+-----+
| Received |
+-----+
| STRING ident |<>-----| AuthorityID |
|               |               +-----+
|               |               0..1 +-----+
|               |<>-----| IncidentID |
|               |               +-----+
|               |               0..1 +-----+

```

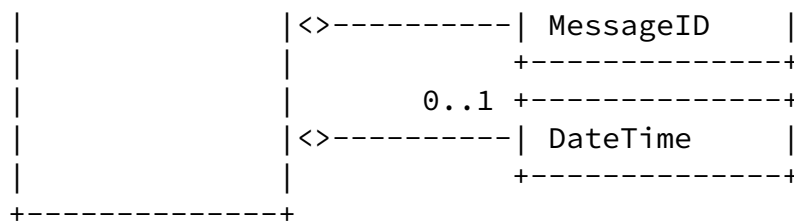


Figure 4.31 Received Class

The aggregate classes that constitute Received class are:

IncidentID

Zero or one. Identification number of the Incident as it reported by the Authority defined by AuthorityID.

MessageID

Zero or one. Identification number of the message.

AuthorityID

Zero or one. Identification number of the authority that sent referenced message.

DateTime

Zero or one. Date and time when message was received.

Received is represented in the XML DTD as follows:

```
<!ELEMENT Received (
  IncidentID?, MessageID?, AuthorityID?, DateTime?)>
<!ATTLIST Received
  ident ID #IMPLIED
>
```

The Received class has one attributes:

ident

Optional. A unique identifier for the Received element, see [Section 3.4.9](#).

4.8.11 The ActionList Class

The ActionList class represents a list of actions undertaken by the CSIRTs in the course of handling an incident.

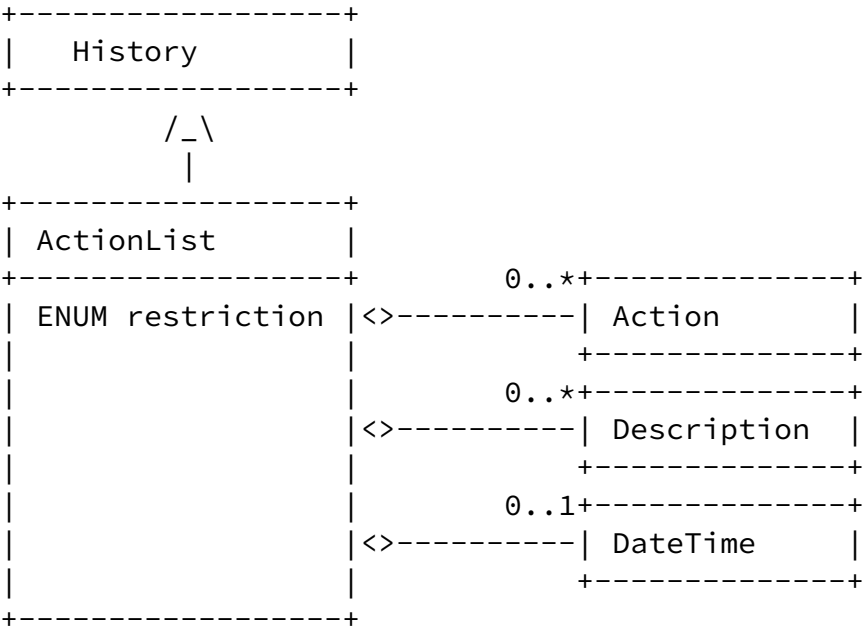


Figure 4.32 ActionList Class

The aggregate classes that constitute ActionList are:

Action

Zero or more. The action taken by the CSIRT in course of incident handling.

Description

Zero or more. A free-form text description of the action(s) taken by the CSIRT in course of handling the incident.

DateTime

Zero or one. Date and time when the action was performed.

ActionList is represented in the XML DTD as follows:

```

<!ELEMENT ActionList (
  Action*, Description*, DateTime?)>
<!ATTLIST ActionList
  ident ID #IMPLIED
  restriction %attvals.restriction; 'default'
>

```

The ActionList class has two attributes:

ident

Optional. A unique identifier for the ActionList element (see [Section 3.4.9](#)).

restriction

Optional. Sets a restriction on the usage of the data in element.

4.8.12. The FileList Class

The FileList class describes files and other file-like objects on targets. It is primarily used as a "container" class for the File aggregate class, as shown in Figure 5.33. For the purpose of compatibility the FileList Class is reused from the IDMEF.

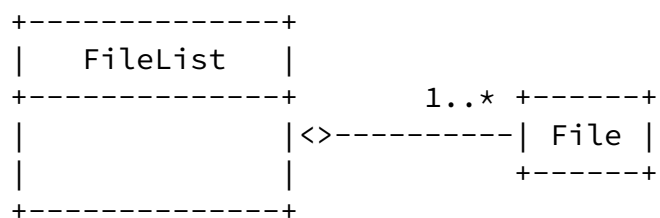


Figure 4.33 The FileList Class

The aggregate class contained in FileList is:

File

One or more. Information about an individual file, as indicated by its "category" and "fstype" attributes (see [Section 4.8.13.1](#)).

This is represented in the XML DTD as follows:

```

<!ELEMENT FileList
  File+
)>

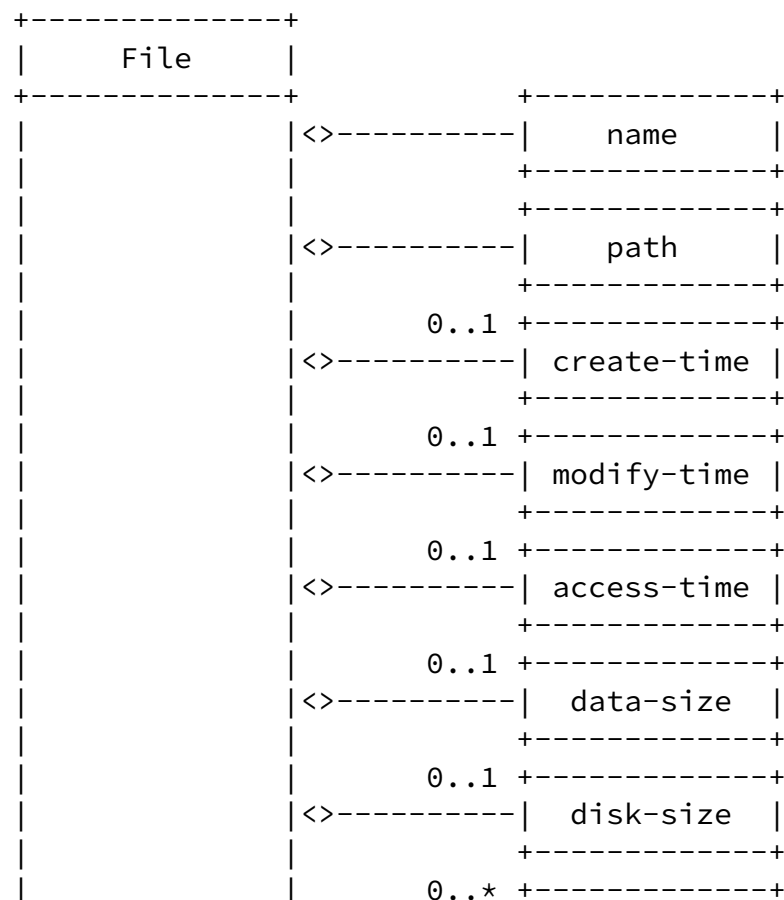
```

4.8.12.1 The File Class

The File class provides specific information about a file or other file-like object that has been created, deleted, or modified on the target. More than one File can be used

within the `FileList` class to provide information about more than one file. The description can provide either the file settings prior to the event or the file settings at the time of the event, as specified using the "category" attribute.

The File class is composed of ten aggregate classes, as shown in Figure 4.34.



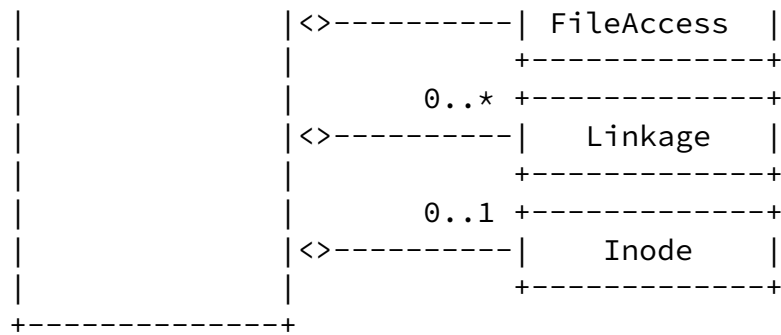


Figure 4.34 The File Class

The aggregate classes that make up File are:

name

Exactly one. STRING. The name of the file to which the

alert applies, not including the path to the file.

path

Exactly one. STRING. The full path to the file, including the name. The path name should be represented in as "universal" a manner as possible, to facilitate processing of the alert.

For Windows systems, the path should be specified using the Universal Naming Convention (UNC) for remote files, and using a drive letter for local files (e.g., "C:\boot.ini"). For Unix systems, paths on network file systems should use the name of the mounted resource instead of the local mount point (e.g., "fileserver:/usr/local/bin/foo"). The mount point can be provided using the <Linkage> element.

create-time

Zero or one. DATETIME. Time the file was created. Note that this is *not* the Unix "st_ctime" file attribute (which is not file creation time). The Unix "st_ctime" attribute is contained in the "Inode" class.

modify-time

Zero or one. DATETIME. Time the file was last modified.

access-time

Zero or one. DATETIME. Time the file was last accessed.

data-size

Zero or one. INTEGER. The size of the data, in bytes. Typically what is meant when referring to file size. On Unix UFS file systems, this value corresponds to stat.st_size. On Windows NTFS, this value corresponds to VDL.

disk-size

Zero or one. INTEGER. The physical space on disk consumed by the file, in bytes. On Unix UFS file systems, this value corresponds to 512 * stat.st_blocks. On Windows NTFS, this value corresponds to EOF.

FileAccess

Zero or more. Access permissions on the file.

Linkage

Zero or more. File system objects to which this file is linked (other references for the file).

Inode

Zero or one. Inode information for this file (relevant to Unix).

This is represented in the XML DTD as follows:

```
<!ENTITY % attvals.filecat          "  
    ( current | original )  
">  
<!ELEMENT File                      (  
    name, path, create-time?, modify-time?, access-time?,  
    data-size?, disk-size?, FileAccess*, Linkage*, Inode?  
)>  
<!ATTLIST File
```



```

    ident          CDATA          '0'
    category       %attvals.filecat; #REQUIRED
    fstype         CDATA          #REQUIRED
    %attlist.global;
>

```

The File class has three attributes:

ident
Optional. A unique identifier for this file, see [Section 3.4.9](#).

category
Required. The context for the information being provided. The permitted values are shown below. There is no default value.

Rank	Keyword	Description
0	current	The file information is from after the reported change
1	original	The file information is from before the reported change

fstype Required. The type of file system the file resides on. The name should be specified using a standard abbreviation, e.g., "ufs", "nfs", "afs", "ntfs", "fat16", "fat32", "pcfs", "joliet", "cdfs", etc. This attribute governs how path names and other attributes are interpreted.

4.8.12.2 The FileAccess Class

The FileAccess class represents the access permissions on a

file. The representation is intended to be usefule across operating systems.

The FileAccess class is composed of two aggregate classes, as shown in Figure 4.35.

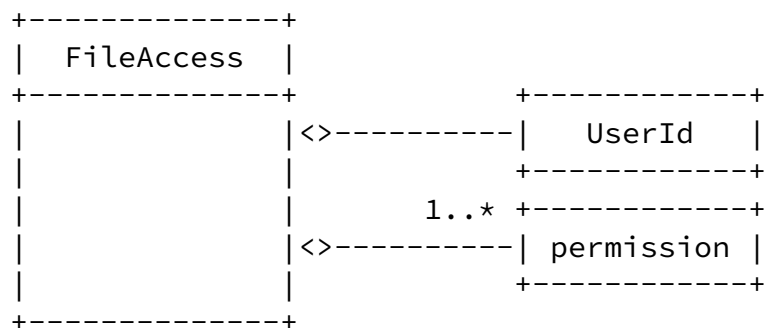


Figure 4.35 The FileAccess Class

The aggregate classes that make up FileAccess are:

UserId

Exactly one. The user (or group) to which these permissions apply. The value of the "type" attribute must be "user-privs", "group-privs", or "other-privs" as appropriate. Other values for "type" MUST NOT be used in this context.

permission

One or more. STRING. Level of access allowed. Recommended values are "noAccess", "read", "write", "execute", "delete", "executeAs", "changePermissions", and "takeOwnership". The "changePermissions" and "takeOwnership" strings represent those concepts in Windows. On Unix, the owner of the file always has "changePermissions" access, even if no other access is allowed for that user. "Full Control" in Windows is represented by enumerating the permissions it contains. The "executeAs" string represents the set-user-id and set-group-id features in Unix.

This is represented in the XML DTD as follows:

```

<!ELEMENT FileAccess (
    UserId, permission+
)>
  
```

4.8.12.3 The Linkage Class

The Linkage class represents file system connections between the file described in the <File> element and other objects in the file system. For example, if the <File> element is a symbolic link or shortcut, then the <Linkage> element should contain the name of the object the link points to. Further information can be provided about the object in the <Linkage> element with another <File> element, if appropriate.

The Linkage class is composed of three aggregate classes, as shown in Figure 4.36.

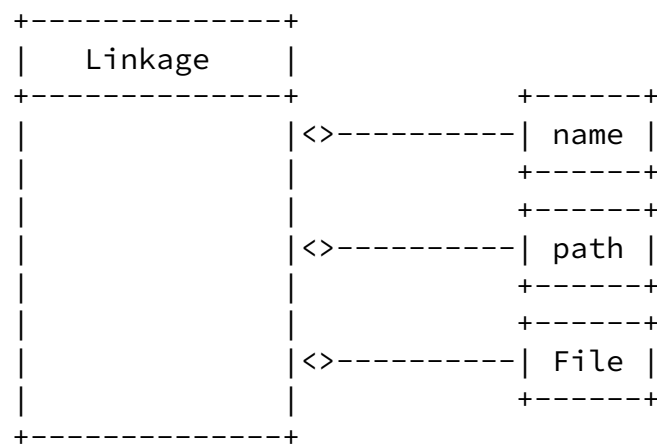


Figure 4.36 The Linkage Class

The aggregate classes that make up Linkage are:

name

Exactly one. STRING. The name of the file system object not including the path.

path

Exactly one. STRING. The full path to the file system object, including the name. The path name should be represented in as "universal" a manner as possible, to facilitate processing of the alert.

File

Exactly one. A <File> element may be used in place of the <name> and <path> elements if additional information about the file is to be included.

The is represented in the XML DTD as follows:

```

<!ENTITY % attvals.linkcat
    ( hard-link | mount-point | reparse-point | shortcut |

```

```

">
<!ELEMENT Linkage                                (
    (name, path) | File
)>
<!ATTLIST Linkage
    category                %attvals.linkcat;      #REQUIRED
>

```

The Linkage class has one attribute:

category

The type of object that the link describes. The permitted values are shown below. There is no default value.

Rank ----	Keyword -----	Description -----
0	hard-link	The <name> element represents another name for this file. This information may be more easily obtainable on NTFS file systems than others.
1	mount-point	An alias for the directory specified by the parent's <name> and <path> elements.
2	reparse-point	Applies only to Windows; excludes symbolic links and mount points, which are specific types of reparse points.
3	shortcut	The file represented by a Windows "shortcut." A shortcut is distinguished from a symbolic link because of the difference in their contents, which may be of importance to the manager.
4	stream	An Alternate Data Stream (ADS) in Windows; a fork on MacOS. Separate file system entity that is considered an extension of the main <File>.
5	symbolic-link	The <name> element represents the

file to which the link points.

4.8.12.4 The Inode Class

The Inode class is used to represent the additional information contained in a Unix file system i-node.

The Inode class is composed of six aggregate classes, as shown in Figure 4.37.

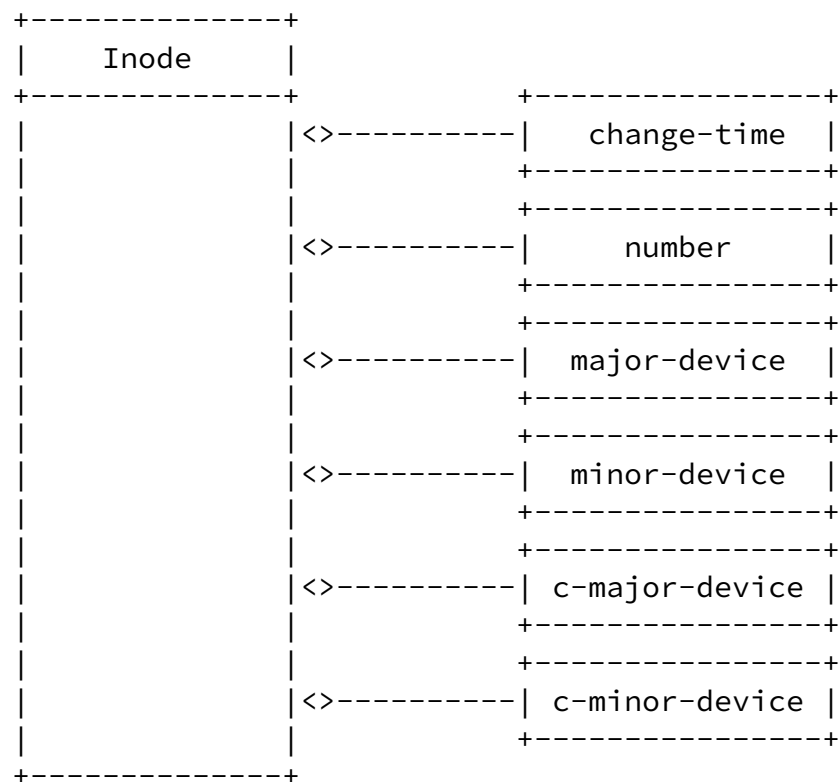


Figure 4.37 The Inode Class

The aggregate classes that make up Inode are:

change-time

Zero or one. DATETIME. The time of the last inode change, given by the st_ctime element of "struct stat".

number

Zero or one. INTEGER. The inode number.

major-device

Zero or one. INTEGER. The major device number of the device the file resides on.

minor-device

Zero or one. INTEGER. The minor device number of the device the file resides on.

c-major-device

Zero or one. INTEGER. The major device of the file itself, if it is a character special device.

c-minor-device

Zero or one. INTEGER. The minor device of the file itself, if it is a character special device.

Note that <number>, <major-device>, and <minor-device> must be given together, and the <c-major-device> and <c-minor-device> must be given together.

This is represented in the XML DTD as follows:

```
<!ELEMENT Inode (
  change-time?, (number, major-device, minor-device)?,
  (c-major-device, c-minor-device)?
)>
```

4.8.13 The Analyzer Class

The Analyzer class identifies the facility used to gather the evidence or tool generated Incident Alert. In case when Initial Incident registration is produced from the IDMEF message the Analyzer description may be taken from the IDMEF message where the Analyzer Class is mandatory and only one.

The analyzer SHOULD define the name of the format, facility, tool, or device used to generate the evidence if it is not self-describing (e.g. xml). Likewise, the analyzer SHOULD define the Node which detected the evidence or from which it was extracted if this information is not represented elsewhere.

For the purpose of compatibility the Analyzer Class is reused from the IDMEF.

The Analyzer class is composed of two aggregate classes, as shown in Figure 4.38.

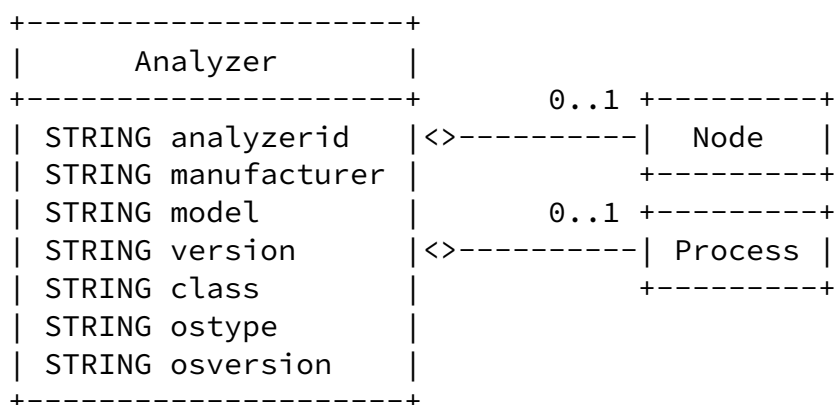


Figure 4.38 The Analyzer Class

The aggregate classes that make up Analyzer are:

Node

Zero or one. Information about the host or device on which the analyzer resides (network address, network name, etc.).

Process

Zero or one. Information about the process in which the analyzer is executing.

This is represented in the XML DTD as follows:

```
<!ELEMENT Analyzer (
```

```

    Node?, Process?
)>
<!ATTLIST Analyzer
    analyzerid          CDATA          '0'
    manufacturer        CDATA          #IMPLIED
    model               CDATA          #IMPLIED
    version             CDATA          #IMPLIED
    class               CDATA          #IMPLIED
    ostype              CDATA          #IMPLIED
    osversion           CDATA          #IMPLIED
>

```

The Analyzer class has seven attributes:

analyzerid
Optional. The attribute may be taken from the IDMEF message generated by Analyzer/IDS. For details see [IDMEF].

manufacturer
Optional. The manufacturer of the analyzer software and/or hardware.

model
Optional. The model name/number of the analyzer software and/or hardware.

version
Optional. The version number of the analyzer software and/or hardware.

class
Optional. The class of analyzer software and/or hardware.

ostype
Optional. Operating system name. On POSIX systems, this is the value returned in utsname.sysname by the uname() system call, or the output of the "uname -s" command.

osversion
Optional. Operating system version. On POSIX systems, this

is the value returned in `utsname.release` by the `uname()` system call, or the output of the `"uname -r"` command.

The "manufacturer", "model", "version", and "class" attributes' contents are vendor-specific, but may be used together to identify different types of analyzers.

4.9 Simple Classes

The simple classes do not have subclasses. The purpose of describing some of the simple classes in this section is to provide information about attributes used to describe the data of these classes.

4.9.1 The Description Class

The Description class is a general-purpose class for any natural language free-form text.

Using the XML language attribute, it is reasonable to include text in a number of different languages in different instances of the Description class. For details on declaring language attribute see [section 3.3.3](#).

```
<!ELEMENT Description xml:lang="langcode">
```

4.9.2 The IRTcontact Class

The IRTcontact class contains an IRTcontact handle to a public registry (e.g., RIPE NCC database [[18](#)], Trusted Introducer database [[19](#)]) that references contact information for the CSIRT or network security manager serving the networks referenced in the Attacker or Victim class.

This is represented in the XML DTD as follows:

```
<!ENTITY % attvals.originIRT          "  
    ( unknown | ripencc | ti | arin | apnic | afnic | local )  
">  
<!ELEMENT IRTcontact    >  
<!ATTLIST IRTcontact  
    originIRT    %attvals.originirt;    'unknown'  
>
```

IRTcontact class has one attribute:

originIRT

Required. The registry which the IRTcontact handle references. The permitted values for this attribute are shown below. The default value is "unknown".

Rank	Keyword	Description
----	-----	-----
0	unknown	Origin of the name is not known
1	ripenncc	RIPE NCC database
2	ti	Trusted Introducer database of CSIRTs
3	arin	ARIN database
4	apnic	APNIC database
5	afnic	AFNIC database
6	local	Name of IRT as it used by Incident object creator

4.9.3 The EvidenceItem Class

The EvidenceItem class is a container for the arbitrary evidence data.

This is represented in the XML DTD as follows:

```
<!ENTITY % attvals.dtype    "  
    ( boolean | byte | character | string | binary | xml |  
      file | path | url )  
">  
<!ELEMENT EvidenceItem    ANY >  
<!--ATTLIST EvidenceItem  
    dtype    %attvals.dtype;    'string'  
>
```

The EvidenceItem class has one attribute:

dtype

Required. The type of data included in the element content. The permitted values for this attribute are shown below. The default value is "string".

Rank	Keyword	Description
----	-----	-----
0	boolean	The element contains a boolean value, i.e.,

- 1 byte the strings "true" or "false"
The element content is a single 8-bit byte
(see [Section 3.4.4](#))

- 2 character The element content is a single character
(see [Section 3.4.3](#))
- 3 integer The element content is an integer (see
[Section 3.4.1](#))
- 4 string The element content is a string (see [Section
3.4.3](#))
- 5 binary The element content is base-64 encoded
binary data.
- 6 xml The element content is XML-tagged data
(see [Section 5.2](#))
- 7 file The element contains a name of file that
may be stored on any media, this
information should be necessary for CSIRT
- 8 path The element content is a path to a file
location on IHS system
- 9 url The element content is a URL to the data

4.9.4 The CorrEvidence Class

The CorrEvidence class references the ID of other related EvidenceData for correlation.

This is represented in the XML DTD as follows:

```
<!ELEMENT CorrEvidence  (
    CorrEvidenceID
)>
<!ATTLIST CorrEvidence
    IncidentID  ID      #IMPLIED
>
```

The CorrEvidence class has one attribute:

IncidentID

Optional. The type of data included in the element content.
The permitted values for this attribute are shown below.

The default value is "string".

4.9.5 The Name Class

The Name class contains the name of a contact person at a CSIRT.

This is represented in the XML DTD as follows:

```
<!ENTITY % attvals.nametype    "  
    ( dn | internic | ripencc )  
">
```

```
<!ELEMENT Name      >  
<!ATTLIST Name  
    nametype    %attvals.nametype;    'file'  
>
```

The Name class has one attribute:

nametype
Required. Type of name or source of name/role handle.

Rank	Keyword	Description
0	dn	Distinguished name (personal name). Format as described in section 3.4.10
1	internic	Name/role handle from InterNIC database
2	ripencc	Name/role handle from RIPE NCC database

[5](#). Extending the IODEF

In order to support the changing activity of CSIRTS, the IODEF data model and DTD will need to evolve along with them. To allow new features to be added, both the data model and the DTD can be extended as described in this section. As these extensions mature, they can then be incorporated into future versions of the specification.

5.1 Extending the Data Model

There are two mechanisms for extending the IODEF data model: inheritance and aggregation (see [Section 3.1.1](#)).

- + By using inheritance, new subclasses may be derived and given additional attributes or operations not found in the superclass.
- + Aggregation allows for entirely new, self-contained classes to be created and associated with a parent class.

Of the two extension mechanisms, inheritance is preferred, because it preserves the existing data model and the operations (methods) executed on the classes of the model. There are explicit guidelines for extending the XML DTD (see [Section 5.2](#)) which set limits on where extensions to the data model may be made.

5.2 Extending the XML DTD

There are two ways to extend the IODEF XML DTD:

1. The AdditionalData class (see [Section 4.2.4.5](#)) allows implementers to include arbitrary "atomic" data items (integers, strings, etc.) in an Incident or IncidentAlert class. This approach SHOULD be used whenever possible.
2. The AdditionalData class allows implementers to extend the IODEF XML DTD with additional DTD "modules" that describe arbitrarily complex data types and relationships.

To extend the IODEF DTD with a new DTD "module," these guidelines MUST be followed:

1. The IODEF description MUST include a document type declaration (see [Section 3.3.1.3](#)).
2. The document type declaration MUST define a parameter entity (see [Section 3.2.4](#)) that contains the location of the extension

DTD, and then reference that entity:

```
<!DOCTYPE IODEF-Description SYSTEM
    "/path/to/IODEF-Description.dtd"
[
    <!ENTITY % x-extension SYSTEM "/path/to/extension.dtd">
    %x-extension;
]>
```

In this example, the "x-extension" parameter entity is defined and then referenced, causing the DTD for the extension to be read by the XML parser.

The name of the parameter entity defined for this purpose MUST be a string beginning with "x-"; there are no other restrictions on the name (other than those imposed on all entity names by XML).

Multiple extensions may be included by defining multiple entities and referencing them. For example:

```
<!DOCTYPE IODEF-Description SYSTEM
    "/path/to/IODEF-Description.dtd"
[
    <!ENTITY % x-extension SYSTEM "/path/to/extension.dtd">
    <!ENTITY % x-another SYSTEM "/path/to/another.dtd">
    %x-extension;
    %x-another;
]>
```

3. Extension DTDs MUST declare all of their elements and attributes in a separate XML namespace. Extension DTDs MUST NOT declare any elements or attributes in the "IODEF" or default namespaces.

For example, the "test" extension might be declared as follows:

```
<!ELEMENT test:test (
    test:a, test:b, test:c
)>
```

```

<!ATTLIST test:test
  xmlns      CDATA    #IMPLIED
  xmlns:test CDATA    #IMPLIED
>
<!ELEMENT test:a (#PCDATA)>
<!ATTLIST test:a
  test:attr  CDATA    #IMPLIED
>

<!ELEMENT test:b (#PCDATA)>
<!ELEMENT test:c (#PCDATA)>

```

4. Extensions MUST only be included in the AdditionalData class of the Incident class whose "type" attribute is "xml". For example:

```

<IODEF-Description version="0.0">
  <Incident ident="...">
    ...
    <AdditionalData type="xml">
      <test:test
        xmlns:test="http://www.ietf.org/iodef/test.html"
        xmlns="http://www.ietf.org/iodef/test.html">
        <test:a test:attr="...">...</test:a>
        <test:b>...</test:b>
        <test:c>...</test:c>
      </test:test>
    </AdditionalData>
  </Incident>
</IODEF-Description>

```

6. Special Considerations

This section discusses some of the special considerations that must be taken into account by implementers of the IODEF.

6.1 XML Validity and Well-Formedness

It is expected that IODEF-compliant applications will normally not include the IODEF DTD in their communications. Instead, the DTD will be referenced in the document type declaration of the IODEF document (see [Section 3.3.1](#)). Such IODEF documents will be well-formed and valid as defined in [5].

Other IODEF documents will be specified that do not include the document prolog (e.g., entries in an IODEF-format database). Such IODEF documents will be well-formed but not valid.

Generally, well-formedness implies that a document has a single element that contains everything else (e.g., "<Book>"), and that all the other elements nest nicely within each other without any overlapping (e.g., a "chapter" does not start in the middle of another "chapter").

Validity further implies that not only is the document well-formed, but it also follows specific rules (contained in the Document Type Definition) about which elements are "legal" in the document, how those elements nest within other elements, and so on (e.g., a "chapter" does not begin in the middle of a "title"). A document cannot be valid unless it references a DTD.

XML processors are required to parse any well-formed document, valid or not. The purpose of validation is to make the processing of the document (what's done with the data after it's parsed) easier. Without validation, a document may contain elements in nonsense order, elements "invented" by the author that the processing application doesn't understand, and so on.

IODEF documents **MUST** be well-formed. IODEF documents **SHOULD** be valid whenever both possible and practical.

6.2 Unrecognized XML Tags

On occasion, an IODEF-compliant application may receive a well-formed, or even well-formed and valid, IODEF document containing tags that it does not understand. The tags may be either:

- + Recognized as "legitimate" (a valid document), but the application does not know the semantic meaning of the element's content; or
- + Not recognized at all.

IODEF-compliant applications **MUST** continue to process IODEF

Internet Draft

[draft-ietf-inch-iodef-00.txt](#)

Apr 2002

documents that contain unknown tags, provided that these documents are well-formed (see [Section 6.1](#)). It is up to the individual application to decide how to process (or ignore) any content from the unknown tag(s).

Special issue is related to inheritance relation between Incident/Attacker related classes IDMEF and IODEF, e.g. IODEF message may be simply wrap up into IDMEF container for the IncidentAlert class.

In particular case of relations between IODEF and IDMEF, the IODEF may be treated as IDMEF extension applying inheritance to incorporate Alert/IDMEF data structure into Attack Class of IODEF.

When Incident description is produced of IDMEF message, IODEF may use directly related data classes from IDMEF. In this context it is recommended that IHS understands both format - IODEF and IDMEF. This may be achieved by mapping part of IDMEF classes (XML tags) related to Attack description into IODEF classes. This is to be not difficult task because of initial approach to match IODEF and IDMEF XML namespaces. Otherwise IODEF parser will still be able to parse well-formed IDMEF document and recognize important XML tags, which meaning in IODEF is inherited from IDMEF.

6.3 Digital Signatures

The joint IETF/W3C XML Signature Working Group is currently working to specify XML digital signature processing rules and syntax [\[16\]](#). XML Signatures provide integrity, message authentication, and/or signer authentication services for data of any type, whether located within the XML that includes the signature or elsewhere.

The IODEF requirements [\[2\]](#) recommend that the IODEF should support content confidentiality, integrity, authentication and non-repudiation. These requirements can be achieved by the inclusion of digital signatures within an IODEF document. Additional security considerations may be applied to the communications methods and protocols used for IODEF documents exchange.

Specifications for the use of digital signatures within IODEF

documents are outside the scope of this document. If such functionality is needed, the use of the XML Signature standard is RECOMMENDED.

[7. Experimental implementation and examples](#)

Meijer, et al.

Expires October 2002

[page 103]

Internet Draft

[draft-ietf-inch-iodef-00.txt](#)

Apr 2002

There is an ongoing effort among a few European CSIRTs to implement IODEF in their daily incident handling work [[17](#)]. The results this project should be available in late 2001.

This section provides examples of IODEF encoded Incident data. The examples are provided for illustrative purposes only and do not necessarily represent the only (or even the "best") way to encode these particular incidents.

[8. The IODEF Document Type Definition](#)

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!--
```

```
*****
```

```
*****
```

```
*** Incident Object Description and Exchange Format XML DTD ***
```

```
*** Version 0.005, February 2002 ***
```

```
***
```

```
*** The use and extension of the IODEF XML DTD are described in ***
```

```
*** RFC XXXX, "Incident Object Description and Exchange Format ***
```

```
*** Data Model and Extensible Markup Language (XML) Document ***
```

```
*** Type Definition," Y. Demchenko, R. Danyliw, J. Meijer. ***
```

```
*****
```

```
*****
```

```
-->
```

```
<!--
```

```
=====
```

```
=== SECTION 1. Imported Names ===
```

```
=====
```

```

-->
<!--
| Media type, as per [RFC2045]
-->
<!ENTITY % ContentType "CDATA">
<!--
| comma-separated list of media types, as per [RFC2045]
-->
<!ENTITY % ContentTypes "CDATA">
<!--
| Character encoding, as per [RFC2045]
-->
<!ENTITY % Charset "CDATA">
<!--
| A space separated list of character encodings, as per [RFC2045]
-->

```

```

<!ENTITY % Charsets "CDATA">
<!--
| Language code, as per [RFC1766]
-->
<!ENTITY % LanguageCode "NMTOKEN">
<!--
| A single character from [ISO10646]
-->
<!ENTITY % Character "CDATA">
<!--
| Date and time information. ISO date format
-->
<!ENTITY % Datetime "CDATA">
<!--
=====
=== SECTION 2. Attribute list declarations. ===
=====
-->
<!--
| Attributes of the IODEF element. In general, the fixed value
| of this attribute will change each time a new version of
| the DTD is released.
-->
<!ENTITY % attlist.iodef "

```

```

    version                CDATA                #FIXED    '0.05'
">
<!--
| Attributes of all elements. These are the "XML" attributes
| that every element should have. Space handling, name space, and
| language.
-->
<!ENTITY % attlist.global "
    xmlns:iodef            CDATA                #FIXED
        'urn:iana:xml:ns:iodef'
    xmlns                   CDATA                #FIXED
        'urn:iana:xml:ns:iodef'
    xml:space               (default | preserve)  'default'
    xml:lang                %LanguageCode;        #IMPLIED
">
<!--
=====
=== SECTION 3. Attribute value declarations. Enumerated values ===
===                for the many element-specific attribute lists. ===
=====
-->
<!--
| Values for the Address.category attribute.
-->
<!ENTITY % attvals.addrcat "

```

```

    ( unknown | atm | e-mail | lotus-notes | mac | sna | vm |
      ipv4-addr | ipv4-addr-hex | ipv4-net | ipv4-net-mask |
      ipv6-addr | ipv6-addr-hex | ipv6-net | ipv6-net-mask )
">
<!--
| Values for the AdditionalData.type attribute.
-->
<!ENTITY % attvals.adtype "
    ( boolean | byte | character | date-time | integer | ntpstamp |
      portlist | real | string | xml )
">
<!--
| Values for the Data.type attribute used in EvidenceItem.
-->
<!ENTITY % attvals.dtype "

```

```

        ( boolean | byte | character | string | binary | xml |
          file | path | url )
    ">
<!--
| Values for the Id.type attribute.
-->
<!ENTITY % attvals.idtype "
    ( current-user | original-user | target-user | user-privs |
      current-group | group-privs )
">
<!--
| Values for the Impact.completion attribute.
-->
<!ENTITY % attvals.completion "
    ( failed | succeeded )
">
<!--
| Values for the File.category attribute.
-->
<!ENTITY % attvals.filecat "
    ( current | original )
">
<!--
| Values for the Impact.type attribute.
-->
<!ENTITY % attvals.impacttype "
    ( admin | dos | file | recon | user | other )
">
<!--
| Values for the Linkage.category attribute.
-->
<!ENTITY % attvals.linkcat "
    ( hard-link | mount-point | reparse-point | shortcut | stream |
      symbolic-link )

```

```

    ">
<!--
| Values for the Confidence.rating attribute.
-->
<!ENTITY % attvals.rating "
    ( low | medium | high | numeric )

```

```

">
<!--
| Values for the Impact.severity attribute.
-->
<!ENTITY % attvals.severity "
    ( low | medium | high )
">
<!--
| Values for the PersonName.type attribute.
-->
<!ENTITY % attvals.nametype "
    ( dn | internic | ripencc )
">
<!--
| Values for the Node.category attribute.
-->
<!ENTITY % attvals.nodecat "
    ( unknown | ads | afs | coda | dfs | dns | kerberos | nds |
      nis | nisplus | nt | wfw )
">
<!--
| Values for the NodeRole.category attribute.
-->
<!ENTITY % attvals.noderolecat "
    ( unknown | client | server-internal | server-public | www | mail |
      messaging | streaming | voice | file | ftp | p2p | name |
directory |
      credential | print | application | database | infra | log )
">
<!--
| Values for the Classification.origin attribute.
-->
<!ENTITY % attvals.origin "
    ( unknown | bugtraqid | cve | vendor-specific | irt-local )
">
<!--
| Values for the IRTcontact.originIRT attribute
-->
<!ENTITY % attvals.originIRT "
    ( unknown | ripencc | ti | arin | apnic | afnic | local )
">
<!--
| Defines purpose of the IODEF Object

```

```

-->
<!ENTITY % attvals.purpose "
    ( report | handling | communication | statistics | experimental )
">
<!--
    | Defines restriction on access to an element's content -->
<!ENTITY % attvals.restriction "
    (default | public | internal | restricted )
">
<!--
    | Values for the User.category attribute.
-->
<!ENTITY % attvals.usercat "
    ( unknown | application | os-device )
">
<!--
    | Values for yes/no attributes such as Source.spoofed and
    | Target.decoy.
-->
<!ENTITY % attvals.yesno "
    ( unknown | yes | no )
">
<!--
=====
=== SECTION 4. Top-level element declarations.  The IODEF-      ===
===      Description element and the types of Incidents it ===
===      may include.                                          ===
=====
-->
<!ELEMENT IODEF-Description ((Incident | IncidentAlert)*)>
<!ATTLIST IODEF-Description
    %attlist.iodef; CDATA #FIXED "0.01"
    %attlist.global;
>
<!ELEMENT Incident (Attack+, Attacker*, Victim*, Method*, Evidence?,
    CorrelationIncident?, Authority, History?, AdditionalData*)>
<!ATTLIST Incident
    incidentID CDATA #IMPLIED
    restriction %attvals.restriction; #IMPLIED
    purpose %attvals.purpose; #REQUIRED
    %attlist.global;
>
<!ELEMENT IncidentAlert (History?, Authority, AdditionalData+)>
<!ATTLIST IncidentAlert
    incidentID CDATA #IMPLIED
    %attlist.global;
    restriction %attvals.restriction; #IMPLIED
>

```

<!--

```
=====
=== SECTION 5. Subclasses of the Incident or other top-level      ===
===           elements that provide more data for specific      ===
===           types of incidents.                                ===
=====
-->
<!ELEMENT Attack (Target*, Source*, Description*, DetectTime?,
StartTime?, EndTime?)>
<!ATTLIST Attack
    ident CDATA "0"
    restriction %attvals.restriction; #IMPLIED
    %attlist.global;
>
<!ELEMENT Attacker (Contact?, Location?, IRTcontact?)>
<!ATTLIST Attacker
    %attlist.global;
    ident CDATA "0"
    restriction %attvals.restriction; #IMPLIED
    spoofed %attvals.yesno; "unknown"
>
<!ELEMENT Victim (Contact?, Location?, IRTcontact?)>
<!ATTLIST Victim
    restriction %attvals.restriction; #IMPLIED
    %attlist.global;
>
<!ELEMENT Method (Classification*, Description*)>
<!ATTLIST Method
    %attlist.global;
    ident CDATA "0"
    restriction %attvals.restriction; #IMPLIED
>
<!ELEMENT Evidence (EvidenceData)>
<!ATTLIST Evidence
    restriction %attvals.restriction;
    %attlist.global;
>
<!ELEMENT Assessment (Impact?, Action*, Confidence?)>
<!ATTLIST Assessment
    restriction %attvals.restriction; #IMPLIED
```



```

        %attlist.global;
    >
    <!ELEMENT CorrelationIncident (EventList, IncidentID,
EvidenceDataID)>
    <!ATTLIST CorrelationIncident
        restriction %attvals.restriction; #IMPLIED
        %attlist.global;
    >
    <!ELEMENT Authority (Organization, Contact*)>
    <!ATTLIST Authority

```

Meijer, et al.

Expires October 2002

[page 109]

Internet Draft

[draft-ietf-inch-iodef-00.txt](#)

Apr 2002

```

        restriction %attvals.restriction; #IMPLIED
        %attlist.global;
    >
    <!ELEMENT History (Reported?, Received*, ActionList*)>
    <!ATTLIST History
        %attlist.global;
        restriction %attvals.restriction; #IMPLIED
    >
    <!--
=====
=== SECTION 6. The AdditionalData element. This element allows  ===
===          an alert to include additional information that  ===
===          cannot be encoded elsewhere in the data model.  ===
=====
-->
    <!ELEMENT AdditionalData ANY>
    <!ATTLIST AdditionalData
        type %attvals.adtype; "string"
        %ContentType; CDATA #IMPLIED
        %attlist.global;
    >
    <!--
=====
=== SECTION 7. Elements related to identifying entities  ===
=== Attack, Attacker, Victim, Method, Evidence and Authority.  ===
=====
-->
    <!-- Elements Target and Source of IODEF are re-used from IDMEF-->
    <!ELEMENT Target (Node?, User?, Process?, Service?, program?, os?,

```

```

FileList?))>
<!ATTLIST Target
    ident CDATA "0"
    decoy %attvals.yesno; "unknown"
    interface CDATA #IMPLIED
    %attlist.global;
>
<!ELEMENT Source (Node?, User?, Process?, Service?, program?, os?)>
<!ATTLIST Source
    ident CDATA "0"
    spoofed %attvals.yesno; "unknown"
    interface CDATA #IMPLIED
    %attlist.global;
>
<!--Elements Classification of IODEF is re-used from IDMEF-->
<!ELEMENT Classification (name, url)>
<!ATTLIST Classification
    origin %attvals.origin; "unknown"
    %attlist.global;

```

```

>
<!ELEMENT EvidenceData (CorrEvidence*, EvidenceDesc?, EvidenceItem?)>
<!ATTLIST EvidenceData
    %attlist.global;
    ident CDATA "0"
    restriction %attvals.restriction; #IMPLIED
>
<!ELEMENT EvidenceDesc (DetectTime?, Analyzer?, Description?)>
<!ATTLIST EvidenceDesc
    %attlist.global;
>
<!--Elements Analyzer of IODEF is re-used from IDMEF-->
<!ELEMENT Analyzer (Node?, Process?)>
<!ATTLIST Analyzer
    analyzerid CDATA "0"
    manufacturer CDATA #IMPLIED
    model CDATA #IMPLIED
    version CDATA #IMPLIED
    class CDATA #IMPLIED
    ostype CDATA #IMPLIED
    osversion CDATA #IMPLIED

```

```

        %attlist.global;
    >
    <!ELEMENT Organization (OrganizationID?, OrgName?, OrgAddress?,
    Email?, Telephone?, Fax?)>
    <!ATTLIST Organization
        %attlist.global;
    >
    <!ELEMENT Contact (PersonName?, PersonAddress?, ContactHandle?)>
    <!ATTLIST Contact
        %attlist.global;
    >
    <!--
    =====
    == SECTION 8. Support elements used for providing detailed      ==
    == information about entities - addresses, names,                ==
    == files, events, etc.                                          ==
    ==
    =====
    -->
    <!ELEMENT Node (((name | Address), Address*), datetime?, name?,
    Address*, Location?, NodeRole*)>
    <!ATTLIST Node
        ident CDATA "0"
        category %attvals.nodecat; "unknown"
        %attlist.global;
    >
    <!ELEMENT Address (address, netmask?)>
    <!ATTLIST Address

```

```

        ident CDATA "0"
        category %attvals.addrcat; "unknown"
        vlan-name CDATA #IMPLIED
        vlan-num CDATA #IMPLIED
        %attlist.global;
    >
    <!ELEMENT NodeRole EMPTY>
    <!ATTLIST NodeRole
        category %attvals.noderolecat; "unknown"
    >
    <!ELEMENT Process (name, pid?, path?, arg*, env*)>
    <!ATTLIST Process

```

```

        ident CDATA "0"
        %attlist.global;
    >
    <!ELEMENT Service (((name | port | (name, port)) | portlist),
protocol?, SNMPService?, WebService?)>
    <!ATTLIST Service
        ident CDATA "0"
        %attlist.global;
    >
    <!ELEMENT SNMPService (oid?, community?, command?)>
    <!ATTLIST SNMPService
        %attlist.global;
    >
    <!ELEMENT User (UserId+)>
    <!ATTLIST User
        ident CDATA "0"
        category %attvals.usercat; "unknown"
        %attlist.global;
    >
    <!ELEMENT UserId (name | number | (name, number))>
    <!ATTLIST UserId
        ident CDATA "0"
        type %attvals.idtype; "original-user"
        %attlist.global;
    >
    <!ELEMENT WebService (url, cgi?, http-method?, arg*)>
    <!ATTLIST WebService
        %attlist.global;
    >
    <!--Elements  FileList with respective subelements of IODEF
are re-used from IDMEF-->
    <!ELEMENT FileList (File+)>
    <!ATTLIST FileList
        %attlist.global;
    >
    <!ELEMENT File (name, path, create-time?, modify-time?, access-time?,
data-size?, disk-size?, FileAccess*, Linkage*, Inode?)>

```

```

<!ATTLIST File
    ident CDATA "0"
    category %attvals.filecat; #REQUIRED

```

```

        fstype CDATA #REQUIRED
        %attlist.global;
    >
    <!ELEMENT FileAccess (UserId, permission+)>
    <!ATTLIST FileAccess
        %attlist.global;
    >
    <!ELEMENT Inode (change-time?, (number, major-device, minor-device)?,
        (c-major-device, c-minor-device)?)>
    <!ATTLIST Inode
        %attlist.global;
    >
    <!ELEMENT Linkage ((name, path) | File)>
    <!ATTLIST Linkage
        category %attvals.linkcat; #REQUIRED
        %attlist.global;
    >
    <!ELEMENT ActionList ((Action*, Description*, datetime?))>
    <!ATTLIST ActionList
        %attlist.global;
    >
    <!ELEMENT EventList ((IncidentID?, EvidenceDataID*, datetime?)*)>
    <!ATTLIST EventList
        %attlist.global;
    >
    <!ELEMENT Received (AuthorityID?, MessageID?, IncidentID?,
        datetime?)>
    <!ATTLIST Received
        %attlist.global;
    >
    <!ELEMENT Reported (AuthorityID?, IncidentID?, datetime?)>
    <!ATTLIST Reported
        %attlist.global;
    >
    <!ELEMENT EvidenceItem ANY>
    <!ATTLIST EvidenceItem
        dtype %attvals.dtype; "string"
    >
    <!ELEMENT CorrEvidence (CorrEvidenceID)>
    <!ATTLIST CorrEvidence
        IncidentID ID #IMPLIED
    >
    <!--
=====
=== SECTION 9. Simple elements with sub-elements or attributes. ===
=====

```

```
-->
<!--Elements Impact of IODEF is re-used from IDMEF-->
<!ELEMENT Impact EMPTY>
<!ATTLIST Impact
    severity %attvals.severity; #IMPLIED
    completion %attvals.completion; #IMPLIED
    type %attvals.impacttype; "other"
    %attlist.global;
>
<!ELEMENT Confidence EMPTY>
<!ATTLIST Confidence
    rating %attvals.rating; "numeric"
    %attlist.global;
>
<!ELEMENT DetectTime (#PCDATA)>
<!ATTLIST DetectTime
    ntpstamp CDATA #REQUIRED
    %attlist.global;
>
<!ELEMENT StartTime (#PCDATA)>
<!ATTLIST StartTime
    %attlist.global;
>
<!ELEMENT EndTime (#PCDATA)>
<!ATTLIST EndTime
    %attlist.global;
>
<!ELEMENT datetime (#PCDATA)>
<!ATTLIST datetime
    %Datetime;
    %attlist.global;
>
<!ELEMENT IRTcontact (#PCDATA)>
<!ATTLIST IRTcontact
    originIRT %attvals.originIRT; #REQUIRED
    %attlist.global;
>
<!ELEMENT ContactHandle (#PCDATA)>
<!ATTLIST ContactHandle
    originIRT %attvals.originIRT; #REQUIRED
    %attlist.global;
>
<!ELEMENT PersonName (#PCDATA)>
<!ATTLIST PersonName
    nametype CDATA #IMPLIED
    %attlist.global;
```

```
>
<!ELEMENT PersonAddress (#PCDATA)>
<!ATTLIST PersonAddress
```

```
        %attlist.global;
>
<!ELEMENT OrgName (#PCDATA)>
<!ATTLIST OrgName
        %attlist.global;
>
<!ELEMENT OrgAddress (#PCDATA)>
<!ATTLIST OrgAddress
        %attlist.global;
>
<!ELEMENT Email (#PCDATA)>
<!ATTLIST Email
        %attlist.global;
>
<!ELEMENT Telephone (#PCDATA)>
<!ATTLIST Telephone
        %attlist.global;
>
<!ELEMENT Fax (#PCDATA)>
<!ATTLIST Fax
        %attlist.global;
>
<!--Element Description may contain arbitrary description in
any/local language used by CSIRT (or IODEF object owner).
Language should be indicated in the xml:lang attribute of the
%attlist.global;
Use of different charsets/encodings for the same language should
considered.-->
<!ELEMENT Action (#PCDATA)>
<!ATTLIST Action
        %attlist.global;
>
<!ELEMENT Description ANY>
<!ATTLIST Description
        %attlist.global;
>
<!--
```

```

=====
=== SECTION 10. Simple elements with no sub-elements or      ===
===          attributes.                                         ===
=====
-->
<!ELEMENT IncidentID (#PCDATA)>
<!ATTLIST IncidentID
          %attlist.global;
>
<!ELEMENT attackID (#PCDATA)>
<!ATTLIST attackID
          %attlist.global;

```

```

>
<!ELEMENT AuthorityID (#PCDATA)>
<!ATTLIST AuthorityID
          %attlist.global;
>
<!ELEMENT MessageID (#PCDATA)>
<!ATTLIST MessageID
          %attlist.global;
>
<!ELEMENT EvidenceDataID (#PCDATA)>
<!ATTLIST EvidenceDataID
          %attlist.global;
>
<!ELEMENT CorrEvidenceID (#PCDATA)>
<!ATTLIST CorrEvidenceID
          %attlist.global;
>
<!ELEMENT OrganizationID (#PCDATA)>
<!ATTLIST OrganizationID
          %attlist.global;
>
<!ELEMENT access-time (#PCDATA)>
<!ATTLIST access-time
          %attlist.global;
>
<!ELEMENT change-time (#PCDATA)>
<!ATTLIST change-time
          %attlist.global;

```



```

>
<!ELEMENT create-time (#PCDATA)>
<!ATTLIST create-time
    %attlist.global;
>
<!ELEMENT modify-time (#PCDATA)>
<!ATTLIST modify-time
    %attlist.global;
>
<!ELEMENT c-major-device (#PCDATA)>
<!ATTLIST c-major-device
    %attlist.global;
>
<!ELEMENT c-minor-device (#PCDATA)>
<!ATTLIST c-minor-device
    %attlist.global;
>
<!ELEMENT data-size (#PCDATA)>
<!ATTLIST data-size
    %attlist.global;
>

```

```

<!ELEMENT disk-size (#PCDATA)>
<!ATTLIST disk-size
    %attlist.global;
>
<!ELEMENT major-device (#PCDATA)>
<!ATTLIST major-device
    %attlist.global;
>
<!ELEMENT minor-device (#PCDATA)>
<!ATTLIST minor-device
    %attlist.global;
>
<!ELEMENT permission (#PCDATA)>
<!ATTLIST permission
    %attlist.global;
>
<!ELEMENT address (#PCDATA)>
<!ATTLIST address
    %attlist.global;

```

```

>
<!ELEMENT buffer (#PCDATA)>
<!ATTLIST buffer
    %attlist.global;
>
<!ELEMENT cgi (#PCDATA)>
<!ATTLIST cgi
    %attlist.global;
>
<!ELEMENT command (#PCDATA)>
<!ATTLIST command
    %attlist.global;
>
<!ELEMENT env (#PCDATA)>
<!ATTLIST env
    %attlist.global;
>
<!ELEMENT netmask (#PCDATA)>
<!ATTLIST netmask
    %attlist.global;
>
<!ELEMENT community (#PCDATA)>
<!ATTLIST community
    %attlist.global;
>
<!ELEMENT Location (#PCDATA)>
<!ATTLIST Location
    %attlist.global;
>
<!ELEMENT http-method (#PCDATA)>

```

```

<!ATTLIST http-method
    %attlist.global;
>
<!ELEMENT name (#PCDATA)>
<!ATTLIST name
    %attlist.global;
>
<!ELEMENT number (#PCDATA)>
<!ATTLIST number
    %attlist.global;

```

```

>
<!ELEMENT url (#PCDATA)>
<!ATTLIST url
    %attlist.global;
>
<!ELEMENT os (#PCDATA)>
<!ATTLIST os
    %attlist.global;
>
<!ELEMENT arg (#PCDATA)>
<!ATTLIST arg
    %attlist.global;
>
<!ELEMENT oid (#PCDATA)>
<!ATTLIST oid
    %attlist.global;
>
<!ELEMENT path (#PCDATA)>
<!ATTLIST path
    %attlist.global;
>
<!ELEMENT pid (#PCDATA)>
<!ATTLIST pid
    %attlist.global;
>
<!ELEMENT port (#PCDATA)>
<!ATTLIST port
    %attlist.global;
>
<!ELEMENT portlist (#PCDATA)>
<!ATTLIST portlist
    %attlist.global;
>
<!ELEMENT program (#PCDATA)>
<!ATTLIST program
    %attlist.global;
>
<!ELEMENT protocol (#PCDATA)>
<!ATTLIST protocol

```

```
>
<!ELEMENT size (#PCDATA)>
<!ATTLIST size
      %attlist.global;
>
```

9. References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997
- [2] Arvidsson, J., Cormack, A., Demchenko, Y., Meijer J. "TERENA's Incident Object Description and Exchange Format Requirements", [RFC 3067](#), February 2001
- [3] Intrusion Detection Message Exchange Format Extensible Markup Language (XML) Document Type Definition by D. Curry - September 2001 - <http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-06.txt> - work in progress.
- [4] Taxonomy of the Computer Security Incident related terminology - http://www.terena.nl/task-forces/tf-csirt/i-taxonomy/docs/i-taxonomy_terms.html
- [5] World Wide Web Consortium (W3C), "Extensible Markup Language (XML) 1.0 (Second Edition)," W3C Recommendation, October 6, 2000. <http://www.w3.org/TR/2000/REC-xml-20001006>.
- [6] World Wide Web Consortium (W3C), "Namespaces in XML," W3C Recommendation, January 14, 1999. <http://www.w3.org/TR/1999/REC-xml-names-19990114>.
- [7] XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001. <http://www.w3.org/TR/xmlschema-0/>
- [8] Berners-Lee, T., Fielding, R.T., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," [RFC 2396](#), August 1998.
- [9] Mealling, M., "The IANA XML Registry," [draft-mealling-iana-xmlns-registry-00.txt](#), November 17, 2000, work in progress.
- [10] Rumbaugh, J., Jacobson, I., and G. Booch, "The Unified Modeling Language Reference Model," ISBN 020130998X, Addison-Wesley, 1998.
- [11] Freed, N., "IANA Charset Registration Procedures," [BCP 19](#), RFC

Internet Draft

[draft-ietf-inch-iodef-00.txt](#)

Apr 2002

2278, January 1998.

- [12] Alvestrand, H., "Tags for the Identification of Languages," [RFC 3066](#), [BCP 47](#), January 2001.
- [13] International Organization for Standardization (ISO), "International Standard: Data elements and interchange formats - Information interchange - Representation of dates and times," ISO 8601, Second Edition, December 15, 2000.
- [14] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation, and Analysis," [RFC 1305](#), March 1992.
- [15] Mills, D., "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI," [RFC 2030](#), October 1996.
- [16] Eastlake, D., Reagle, J., and D. Solo, "XML-Signature Syntax and Processing," [draft-ietf-xmlsig-core-11.txt](#), November 1, 2000, work in progress.
- [17] Incident Object Description and Exchange Format Working Group - <http://www.terena.nl/task-forces/tf-csirt/iodef/>
- [18] Incident Object Data model - <http://www.terena.nl/task-forces/tf-csirt/iodef/docs/>
- [19] RIPE NCC Database - <http://www.ripe.net/ripe/wg/db/>
- [20] Trusted Introducer Service - <http://www.ti.terena.nl/>

[10](#). Security Considerations

[11](#). IANA Considerations

[12.](#) Acknowledgements

This document was built on the work done by the Incident Object Description and Exchange Format Working-Group of the TERENA task-force TF-CSIRT.

Meijer, et al.

Expires October 2002

[page 120]

Internet Draft

[draft-ietf-inch-iodef-00.txt](#)

Apr 2002

[13.](#) Authors' Addresses:

Jan Meijer
SURFnet
Radboudburcht 273
Utrecht
The Netherlands
Phone: +31 302 305 305
Email: jan.meijer@surfnet.nl

Roman Danyliw
CERT Coordination Center
4500 Fifth Ave.
Pittsburgh PA 15213
USA
Phone: +1 412 268 7090
Email: rdd@cert.org

Yuri Demchenko
TERENA
Singel 468 D
1017 AW Amsterdam
The Netherlands
Phone: +31 205 304 488
Email: demch@terena.nl

[14.](#) Full Copyright Statement

Copyright (C) The Internet Society (2002). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

Meijer, et al.

Expires October 2002

[page 121]

Internet Draft

[draft-ietf-inch-iodef-00.txt](#)

Apr 2002

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."