MITRE

Common Event Expression

CEE Profile Specification

Version 0.6

The CEE Board
July 2011

This page intentionally left blank.

LICENSE

Disclaimers

This page intentionally left blank.

Acknowledgments

This CEE Profile specification combines two portions of CEE into a single specification: the CEE Dictionary and Event Taxonomy (CDET) vocabulary and the CEE Event Log Recommendations profiles. This document and related concepts are the result of the combined efforts of the CEE Board and various individuals from the CEE Community who provided ideas, recommendations, and inputs. The CEE Board wishes to thank their colleagues who reviewed drafts of this document, providing comments that contributed to its development. Specific contributions the CEE Board would like to acknowledge include Raffael Marty for the concept of the Data Dictionary and Taxonomy, and Eric Fitzgerald for much of the implementation concepts. The CEE Board also gratefully acknowledges and appreciates the many contributions from individuals representing government agencies and the private sector, including Rainer Gerharts, Steve Grubb, Sanford Whitehouse, Dr. Anton Chuvakin and David Corlette. Additionally, the CEE Board would like to thank Francis Duff (MITRE) and Linda K. Jones (MITRE) for helping organize and draft this document.

This page intentionally left blank.

Abstract

Two important components of the Common Event Expression (CEE) Architecture [1] are the CEE Dictionary and Event Taxonomy (CDET), and the CEE Event Log Recommendations (CELR). As these components share common requirements and have interdependencies, CDET and CELR are combined into a single, machine-interpretable CEE Profile document specification.

The CDET component consists of a dictionary and taxonomy portions. The CDET Dictionary defines a common terminology, which can be used to describe the various properties of an event instance. The CDET Taxonomy provides a common event classification system to help identify similar events. By combining the Dictionary and Taxonomy, end users and products can use the same terms to describe the same event characteristics, producing a more unified record of an event.

CELR provides the ability to identify recommended event types and event properties for IT devices. Logging recommendations, to include specific events and event properties, are identified with the use of "event profiles." The CELR profiles are defined based on a collection of best practices from various sources, including information assurance recommendations, requirements, forensics, and inputs from the CEE Community.

CEE Profiles are intended for use with the CEE Log Syntax (CLS) and the CLS Encodings, but may be used within other contexts. CLS and the CLS Encodings are defined in related documents.

The CEE Profile, CEE Dictionary and Event Taxonomy, and CEE Event Log Recommendations are developed by MITRE, in collaboration with industry and government. This specification is built upon the foundation outlined in the *Common Event Expression Whitepaper* [2].

KEYWORDS: CEE, CEE Profile, CDET, CDET Dictionary, CDET Taxonomy, CELR, Event Profile, Log, Event Log, Audit Log, SIEM, Log Management

This page intentionally left blank.

Table of Contents

Appendices

List of Figures

List of Tables

1 Introduction

Common Event Expression (CEE) enables an open, practical, and industry-accepted standard for the description, representation, and exchange of event records between electronic systems. To do this, CEE partially relies on standardized vocabularies, in the form of the CEE Dictionary and Event Taxonomy (CDET), for recording descriptions of events. The objective of this event vocabulary is to correct the limited interoperability due to the inconsistent use of terms used to describe and categorize events. This specification is a coordinated industry initiative, developed by a community of vendors, researchers, and end users. Further details regarding the CEE Architecture and its various components are in the *Common Event Expression: Architecture Overview* document [1].

The overall purpose of the CDET Dictionary and CDET Taxonomy is to improve the audit process and users' ability to effectively interpret and analyze event log and audit data. Individually, the CDET Dictionary enables the standardized recording of event characteristics using common event field definitions. The CDET Taxonomy defines lists of event categories, which allow events to be categorized based on the type of event. The CEE Log Syntax (CLS) [3] defines how these definitions can be used within a CEE Event.

MITRE coordinates this CDET specification and associated Dictionary and Taxonomy repositories as part of the overall CEE Architecture and in coordination with MITRE's larger *Making Security Measurable* standardization initiative (http://measurablesecurity.mitre.org).

1.1 Scope

This document is an introduction and technical specification of the CEE CDET component, including the CDET Dictionary and CDET Taxonomy. Other CEE components (e.g., CLS, CLT) will be defined in their own specification documents. The vocabulary defined within the Dictionary and Taxonomy is based on inputs from the CEE Community, the CEE Board, and The MITRE Corporation.

1.2 Purpose

This document introduces the CEE Dictionary and Event Taxonomy (CDET) components to the CEE Community for validation and approval. The CEE Community is vital to the success and adoption of CEE; their feedback and discussion is needed to produce an open, practical, and industry-accepted standard. Comments and recommendations should be submitted to the CEE Discussion List (`cee-discussion-list@lists.mitre.org`) or to the MITRE CEE Team (`cee@mitre.org`).

1.3 Document Organization

This document is organized into the following sections:

- ***Introduction:*** identifies the scope, purpose, and approach for this document
- ***CEE Architecture:*** provides an overview of the CEE Architecture and design
- ***CEE Profile:*** defines the objective, profile types, and specifications for a CEE Profile document
- ***CDET Taxonomy:*** provides technical specifications for the CDET Taxonomy Tags
- ***CDET Dictionary:*** provides technical specifications for the CDET Dictionary Fields
- ***CELR Event Profile:*** defines the structure for an event, including required and optional components, event text, and how to extend other event profiles

- ***Maintenance:*** provides an overview of the change management and compliance processes for CEE and CDET
- ***Summary***: identifies the components that were discussed and summarizes the next steps to improve CEE as well as the CDET Dictionary and Taxonomy
- ***Appendix A***: identifies the external documents referred to within this specification
- ***Appendix B***: provides a full XML 1.0 Schema that specifies the format of a valid CEE Dictionary and Event Taxonomy XML document
- ***Appendix C***: identifies all of the changes introduced with each version of the CEE CDET specification

2 CEE Architecture

Organizations routinely undertake the expensive task of auditing their electronic systems. Some audits are performed to identify problems, reduce unnecessary overhead, or to maintain compliance with regulatory laws. Every log may contain critical information about prior and ongoing electronic events. Examples of some of these events include electronic events such as logon, connect, and write, or physical events such as building access or equipment pressure readings. These electronic events reflect status, threats, and other observable environment changes that allow an enterprise to maintain constant situational and informational awareness.

Today, there is no standard for representing and describing these events in logs. This has become a significant data management problem, since enterprise-wide situational awareness depends on the ability to process and analyze event data. The CEE Architecture addresses this audit problem by standardizing the event-log relationship by normalizing the way events are recorded, shared, and interpreted (Figure 1).

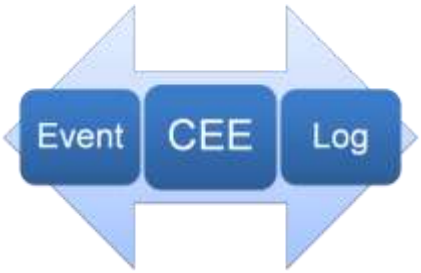


Figure 1: Standardizing Event-Log Relationships

2.1 Overview

The CEE Architecture standardizes the representation of events by providing tools similar to the dictionaries, grammar books, communication mediums (e.g., letters, e-mails, newspapers), and guidance used to support natural languages (e.g., English, Spanish, Latin). The architecture groups these tools across four areas: terminology dictionaries, representation (e.g., grammar rules), transport (e.g., e-mails, web services), and recommendations. These areas map directly to the four CEE Architecture components: CEE Dictionary and Event Taxonomy (CDET), CEE Log Syntax (CLS), CEE Log Transport (CLT), and the CEE Event Log Recommendations (CELR). The CEE Dictionary and Event Taxonomy provide a controlled vocabulary for the consistent description of event details. CLS defines the event language and event encodings for representing CEE events. CLT defines the requirements for secure, reliable recording and transmission of events. The CELR provides Event Profiles for commonly used and product-generated CEE events.

A CEE Profile combines both CDET definitions and CELR Event Profiles into a single, machine-readable document.

2.2 Design Goals

Due to the many uses of event records, CEE is designed to address many diverse audit needs. To encourage widespread adoption, the criteria and considerations for the Architecture must address current community requirements as well as deficiencies identified with previous standardization attempts. Described below are general design goals for the CEE Architecture. Included with each goal is an explanation of why it is important to CEE's success.

1. **Encoding Neutral**: CEE Architecture shall support multiple ways of encoding event records in order to support the variety of event log environments. To maximize usability and promote adoption, the CEE standard shall focus on the event records and the event attributes, while supplying multiple encoding choices.

2. **Flexible**: CEE Architecture shall provide options in the choice of event fields and syntax encodings to provide flexibility to the end user within constraints defined by the CEE standard to preserve interoperability.
3. **Extensible:** CEE Architecture shall focus on addressing representative event records for a typical organization's environment.
4. **Compatible:** CEE shall strive to utilize or provide compatibility with widely used standards (e.g., XML, JSON, Syslog).
5. **Comprehensive:** All event fields and field values shall be represented within CEE or shall be capable of being specified within a compatible extension.
6. **Maintainable:** CEE shall be defined in a manner to ensure that maintenance and updates have minimal impact on CEE and component specifications.
7. **Easily Implementable**: CEE and its components shall be defined such that it is easy to implement by both event record producers and consumers.

3 CEE Profile

CEE Profiles are a crucial piece of CEE as they allow the CDET and CELR components of CEE to be combined into a single, machine-readable document. The intention behind these profiles is to encourage sharing and reuse of events, event type tags, and event fields. To this end, CEE Profile documents must be made publicly available under a royalty-free, non-discriminatory, perpetual license.

Each CEE Profile is machine-readable file that is used primarily to identify the event data to record in logs. A good analogy for a profile is a "cookbook" in which the "recipes" identify the events and event properties and fields to record. The structure for a CEE Profile document is specified below.

3.1 CEE Profile Types

While there is only one CEE Profile document format, it can be used to provide any one of three (3) possible CEE Profile types: base, function, and product. The difference between each of these profile types is explained below.

Examples of each of these CEE Profile documents are provided in Appendix C.

3.1.1 Base Profile

CEE provides a special type of profile called the **CEE Base Profile**. Each base profile must define one event profile, known as the **base event profile**—the minimal, fundamental structure for all event records. The base profile is defined to improve event record compatibility and ensure that products are providing at least the minimal amount of record data. All users of CEE should be familiar with the CEE Base Profile.

Within CEE, the core Base Profile document is named `cee-base-profile.xml` and is published to the profiles section of the CEE website. The CEE base event profile (`cee_base`) is defined in this base profile. For compatibility, the fields, field types, tags, and tag types are defined within a separate profile document and included within the CEE Base Profile.

3.1.2 Function Profile

A CEE Profile created for a function, or **CEE Function Profile**, identifies the recommended events, to include the event details, which should be logged when a device completes that function or activity. Within CEE, a function is considered a particular activity or capability, which is completed on or by an IT device. Examples of functions include user authentication, account management, firewall, application, etc. CEE Function Profiles are used to define recommended events to log related to a particular function or capability. In the user authentication example, the events that comprise user authentication may be the login, session start, session end, and logout events therefore these events would be identified in the profile.

Every function profile must extend a CEE Base Profile and may extend other CEE Function Profiles. The process for specifying extensions via the profile Include element is later discussed in Section 3.3.5. All new field definitions in the function profile should build on those defined in the base profile, while all event profiles must extend a base event profile[1].

[1] This extension may be indirect. For example an event profile may extend another event profile that extends a base event profile.

Function profiles can be specific to an environment or a risk posture. Some of the sources of information used to create a function profile include requirement and policy documents, NIST publications, best practices and recommendations by subject matter experts (SMEs), and security checklists.

3.1.3 Product Profile

A **CEE Product Profile** can be created for a product in order to identify the specific events and fields that it can produce.

Similar to CEE Function Profiles, product profiles should extend the fields and event profiles of a CEE Base Profile, and may also extended CEE Function Profiles as necessary. For example, if a product supports the concept of user accounts and permits users to login and logout, the product should model its user management events based on the recommendations in the "user authentication" CEE Function Profile.

3.2 CEE Profile Specification

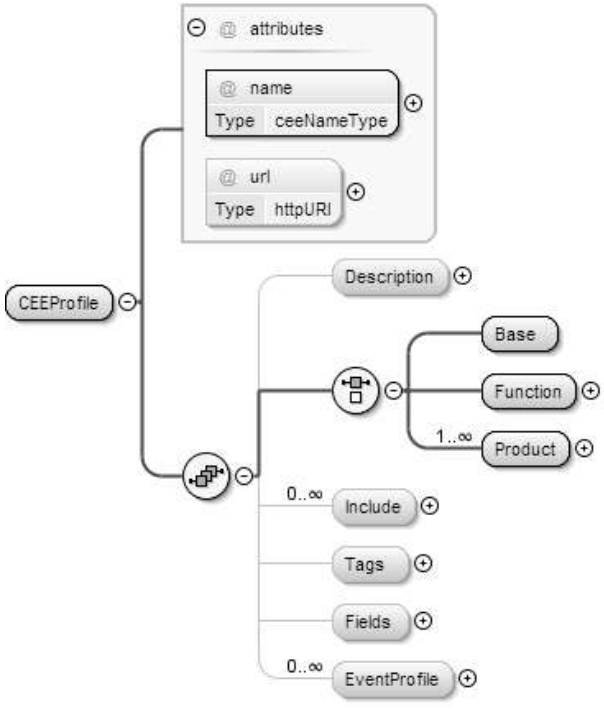
CEE Profiles are comprised of components, some of which are unique to the intended profile use. For example, a CEE Base Profile defines tag and field definitions as part of the CEE Dictionary and Event Taxonomy (CDET), which a CEE Function Profile may not require.

Any CEE Profile Document consists of six (6) main components: a description, profile type (base, product, or function), include statements, CDET Taxonomy Tags, CDET Dictionary Fields, and zero or more Event Profile definitions. An overview of these components can be found in Table 1.

3.3 CEE Profile Definition

The `CEEProfile` element is used to define new CEE Profile. The only one CEE Profile should be defined per document, with the `CEEProfile` element being the document root.

Each `CEEProfile` element must have an identifier that uniquely references that CEE Profile. CEE Product and Functional Profiles must include an element that clearly states the product or function to which the profile pertains.



3.3.1 name Attribute

The `name` attribute is required and contains the name of this profile document. The name attribute must match the CEE Profile's filename, minus the extension. For example, if the `CEEProfile name` value is *cee_base_profile*, it must be contained in the file: *cee_base_profile.xml*.

The format of the CEE Profile's name is restricted to no more than 32 ASCII letter (`[a-zA-Z]`), digit (`[0-9]`), or underscore (_) characters.

3.3.2 url Attribute

The `url` attribute is recommended and must contain a URL to a publicly available copy of this CEE Profile. This URL shall act as a unique reference for the particular CEE Profile document. I.e., the `href` URL should properly resolve to a CEE Profile XML document that is identical in content.

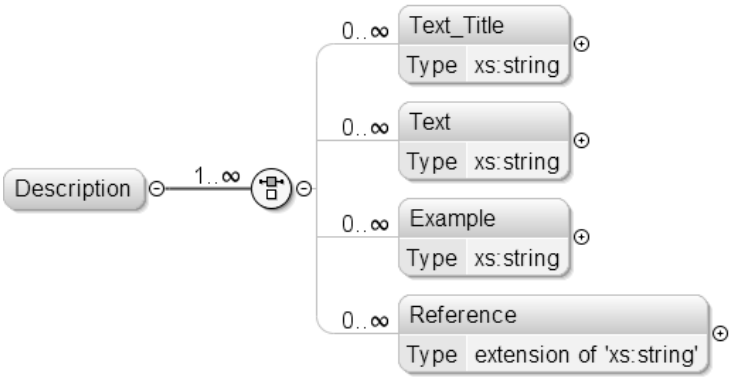
New or updated profiles should be placed along a different URL path and must contain a different `CEEProfile url` attribute value.

Table 1: CEE Profile Components

Component	Description	Occurs
Description	Meta information regarding the profile such as the names of contributors, profile descriptions, and other user-provided data	0..1
Base *or* **Function** *or* **Product**	Identifies the profile type as well as any product(s) or function for which this profile defines the event records	1
Include	Allows for externally defined CEE Profiles to be included	0..*
Tags	A listing of Taxonomy tag and tag type definitions, compliant with the *CEE Dictionary and Event Taxonomy* (CDET) specification [4]	0..1
Fields	A list of Dictionary field and field type definitions, compliant with the *CEE Dictionary and Event Taxonomy* (CDET) specification [4]	0..1
EventProfile	Templates or "recipes" for the events that may be recorded by a product or supported as part of a function profile	0..*

3.3.3 Profile Description Element

A CEE Profile's `Description` element is a container for information such as human-readable labels, descriptions, external references, and examples. This `Description` element is common amongst all profile's components.



The `Text_Title` element provides a user-friendly label or title for the profile. `Text_Title`s can be provided in many languages, as identified by the optional `xml:lang` attribute.

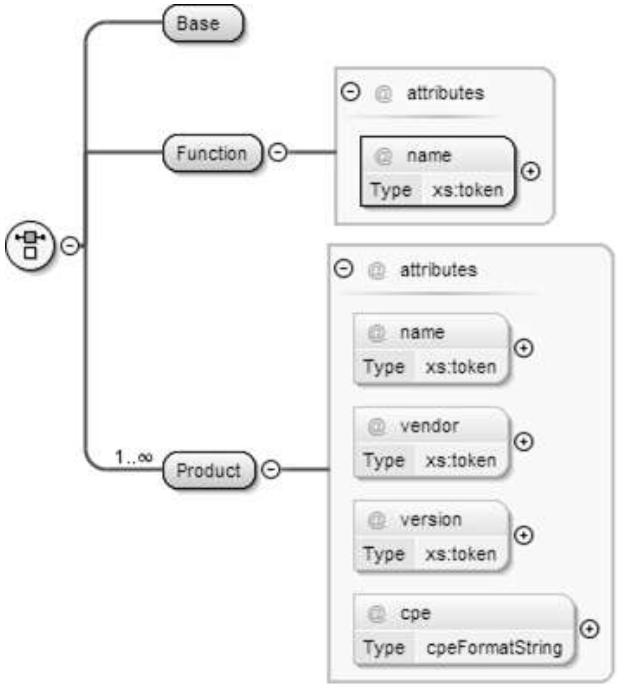
The `Text` element provides detailed information about the profile, examples, clarifications, and guidance for proper use. Like the title, the text can be multi-lingual and identified with the `xml:lang` attribute.

A `Reference` element provides an external pointer to a relevant description or similar concept. The reference can be given a unique identifier using the `ref_id` attribute or a URL to visit in the `url` attribute. A profile may reference a web page that provides more information on the product, function, or requirements.

Finally, the `Example` element contains examples of proper usage and values.

3.3.4 Profile Type (Base, Function, or Product Element)

Each CEE Profile must identify what type of profile it is by specifying a `Base`, `Function`, or `Product` element. Additionally, there needs to be a way to identify the product (for Product Profiles) or function (for Function Profiles).



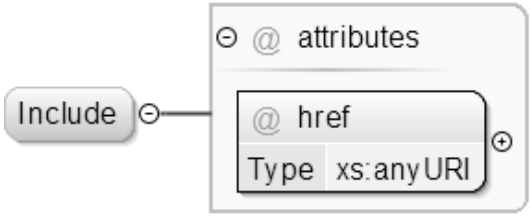
The Product Profile requires the specification of one or more products to which the CEE Profile is associated. To do this, list one or more products using the `Product` element. Each Product element supports the indication of the product using either the `product`, `vendor`, and `version` information attributes, or providing a Common Platform Enumeration (CPE) format string (`cpe:`) [5] value for the `cpe` attribute.

Likewise, a Function Profile requires that the profile document identify the events necessary to properly record that function or activity. The function type are specified using the `name` attribute in a `Function` element.

3.3.5 Include Element

The `Include` element is used to reference previously defined profiles.

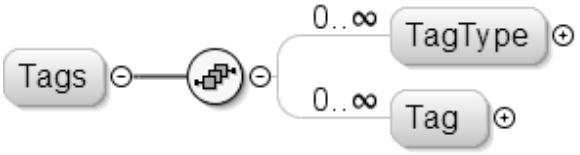
An Include statement uses the `href` attribute to provide the URL of one other CEE Profile document. Each included profile should be processed and merged into the current profile.



As the CEE Base Profile includes the latest version of the CEE Dictionary and Event Taxonomy (CDET) repository, most profiles may only need to include a reference to the CEE Base Profile.

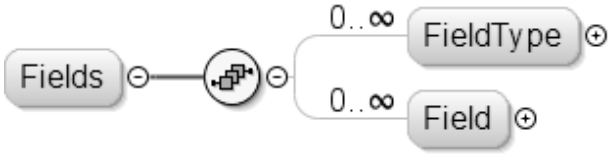
3.3.6 Tags Element

The `Tags` element is used to provide a listing of CDET Taxonomy Tags and TagTypes. The structure of the Tags element contains a listing of zero or more `TagType` definitions followed by zero or more `Tag` definitions. More information on Tags and TagTypes is provided in the CDET Taxonomy section of this document, primarily in sections 0 and 4.3.



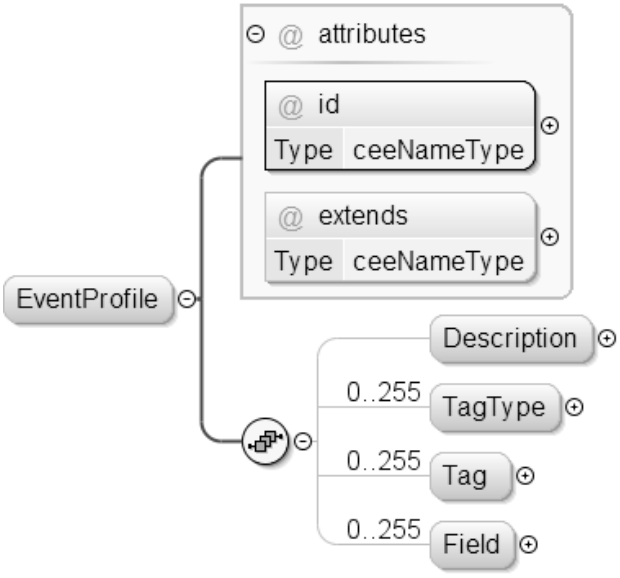
3.3.7 Fields Element

Similar to the `Tags` element, the `Fields` element contains CDET Dictionary definitions. First, zero or more `FieldType`s are defined, followed by zero or more `Field` elements. The CDET Dictionary section provides more information, including the format and requirements for FieldType and Field definitions (§5.4 and 5.2, respectively).



3.3.8 EventProfile Element

Each Profile can have zero or more CEE Event Log Recommendations (CELR) Event Profile definitions. Each Event Profile is specified within an `EventProfile` element, whose format is defined in Section 6.1.



This page intentionally left blank.

4 CDET Taxonomy

Events are categorized in many different ways depending on the event producer, event consumer, the environment, and the requirements of the end user. As all of these entities have different needs and perspectives, it is not possible to develop a single, hierarchical event taxonomy. Instead, the taxonomy portion of the CEE Dictionary and Event Taxonomy (CDET) standardizes these event categorization concepts using a multi-dimensional tag space to describe concepts that are shared across the event space.

Within this event concept tag space, a **tag type** represents each dimension, or concept. Each individual unit along a dimension is a **CEE Tag**. For example, the event action is a common concept for events – this concept is described with an *action* tag type. Within the *action* tag type, common event actions can be listed as CEE Tags: *login*, *move*, *delete*, *write*, *execute*, etc.

The CDET Taxonomy provides a collection of CEE Tags that can be used to express event categorization concepts. The goal is to provide a common vocabulary, through a community-defined collection of tags and tag type, to help identify event records that are conceptual related. Using Taxonomy tags, event producers can provide obvious and consistent event categorization identifiers. For example, users and event consumers can leverage these categories to improve event correlation or easily locate certain classes of events.

Common tag types include event action, status, and object, and might include other categorizations such as attack type, device type, or other categorizations that are required by the event consumer. An example list of event tag types and names are in Table 2.

Table 2: Example CEE Tag Types and Names

Tag Type	Tag Name
action	start, stop, execute, read, delete, login
object	file, acct, app, db, system, malware
status	success, failure, error
attack	dos, exploit, xss, buffer-overflow
service	firewall, http, audit, db

These tags can be specified within an event record to indicate that event's categorization. For example, an event could be tagged with a `login action`, `success status`, and `db service` tags, indicating that the event probably pertains to a successful login to a database.

4.1 CDET Taxonomy Specification

The CDET Taxonomy defines CEE Tags and associated tag types (Figure 2). Each tag type represents one possible way of grouping related events to better support filtering, searching, and correlation. A CEE Tag is an individual name that represents a term or concept used to describe an event. For example, a "hit" action tag can be defined and added to all events that describe someone hitting or being hit.

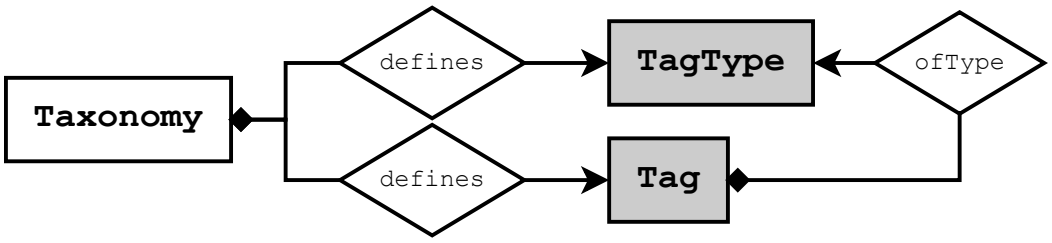
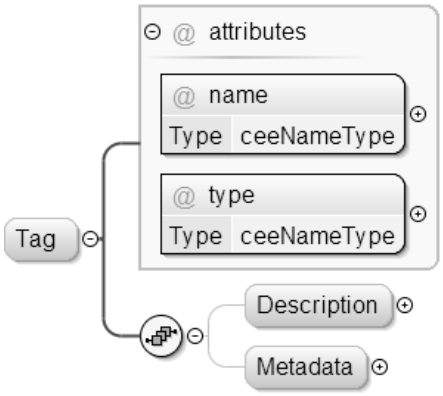


Figure 2: Taxonomy Model

4.2 CEE Tag Definition

The `Tag` element is used to define new CEE Tags. Each CEE Tag represents a single classifier for an event. For example, a "login" tag denotes all events describing a logon or login event type. Similarly, an "execute" tag might represent all types of events pertaining to program, file, or command executions.



Each `Tag` element must identify a unique tag name and associate that tag with a specific tag type. Optional definition details include providing a description and identifying any metadata, such as relations with other `Tag` elements.

4.2.1 Tag name Attribute

The CEE Tag's `name` is a required attribute whose value is a unique sequence of characters that is used to reference this specific tag definition. Each tag name must start with a letter (`[A-Za-z]`) or underscore ('_' `U+005F`) and be followed by no more than 31 letter, digit (`[0-9]`), and underscore characters.

That is, each CEE Tag name must be between 1 and 32 characters in length and conform to the regular expression: `[A-Za-z_][A-Za-z0-9_]{0,31}`

4.2.2 Tag type Attribute

A CEE Tag's `type` attribute indicates the tag type to which this tag belongs. It is required that the value of the `type` attribute reference the name (`TagType/@name`) of a defined tag type.

Tag types and the `TagType` definition are discussed in §4.3.

4.2.3 Tag Description Element

A CEE Tag's `Description` element is a container for information such as human-readable text, tag titles, external references, and examples. It is defined in the same way as Section 3.3.3: Profile Description Element.

4.2.4 Tag Metadata Element

The CEE Tag's `Metadata` element contains information that may be useful, but is not critical to the processing of a CEE Tag definition. CDET Taxonomy producers may wish to provide metadata to provide product-specific mappings or other specific information.

Currently, this specification provides relational metadata that defines the conceptual relationships between two CEE Tags. Examples of these relationships include denoting that *login* is the inverse of *logout*, and that a *read* action is subclass of an *access* action.

Individual CEE Tag relations are defined using one of the relation elements and specifying the name of the tag to which it is related using the `value` attribute. There are three (3) types of relation elements:

- `subclassOf` relation: this CEE Tag is a subclass (i.e., more specific instance) of the referenced `Tag`. This can be thought of as a parent-child relationship. For example, an "administrator" may be considered a subclass of "account."

- `equivalentTo` relation: this CEE Tag describes functionally equivalent concepts to that of the referenced tag. For example, a "logon" action is equivalent to a "login" action.
- `inverseOf` relation: this CEE Tag describes a concept that is the inverse of that of the referenced tag. That is a tag's inverse undoes (or does the opposite) of what the tag defined. For example, "create" is the inverse of "remove" and "logout" is the inverse of "login." An inverse relation currently only applies to actions.

These relations allow for more expressivity. When using a product that supports the Taxonomy tag relation information, a user searching for a tag such as "account" would also receive all subclasses (e.g., administrator). Similarly, the equivalent relationships allow users to search for something "equivalent to a file access," which would include all events marked as access as well as equivalent concepts.

It should be treated as an error when a tag relation is used to create an association between tags of different tag types. Similarly, it is forbidden for a CEE Tag to specify a relation with itself (i.e., self-referencing relation).

4.2.5 Valid Tag Examples

```
      <Tag name="access" type="actionTag">
         <Description>
            <Text_Title>Access Event</Text_Title>
            <Text>A file, user account, network share, or other object has
been accessed. If more is known regarding the access, use a more precise
action such as 'read', 'write', or 'execute'.</Text>
         </Description>
      </Tag>
      <Tag name="failure" type="statusTag">
         <Description><Text_Title>Event Failed</Text_Title></Description>
      </Tag>
      <Tag name="success" type="statusTag">
         <Description>
            <Text_Title>Event Success</Text_Title>
            <Text>The event completed successfully. For example, a successful
user authentication event would be an instance where the authentication
activity was successfully completed and the user was fully
authenticated.</Text>
         </Description>
      </Tag>
```

4.2.6 Invalid Tag Examples

```
      <Tag name="bad_entry" type="undefinedType"/>
      <Tag name="bad_entry2" type="actionTag">
          <Metadata>
              <subclassOf value="bad_entry"/>
          </Metadata>
      </Tag>
```

There are multiple problems with the above example. First, the *bad_entry* `Tag` definition references an undefined tag type. In addition, the relationship from *bad_entry2* to *bad_entry* tries to establish a relationship between tags belonging to different categories.

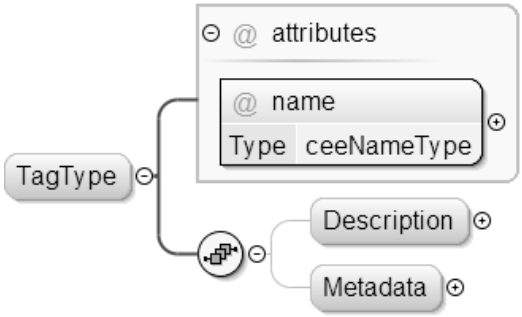
```
<Tag name="bad_entry3" type="object">
    <Metadata>
        <subclassOf value="bad_entry3"/>
    </Metadata>
</Tag>
```

The *bad_entry3* `Tag` definition has the single problem that it defines a relationship with itself and CDET does not support self-referencing tag definitions.

4.3 TagType Definition

A `TagType` element is used to define an individual CEE Tag type—a single class, category, or method of grouping similar types of events. This can also be thought of as one dimension of an event taxonomy.

Each `TagType` definition must minimally specify a unique name for that definition. Optional details may be added such as a description.



4.3.1 TagType name Attribute

The tag type's `name`[2] is a required attribute whose value is a unique sequence of characters that is used to reference this specific tag type definition. Each tag type's name must start with a letter (`[A-Za-z]`) or underscore ('_' `U+005F`) followed by no more than 31 letter, digit (`[0-9]`), and underscore characters. Each tag type name must be no longer than 32 characters and conform to the regular expression: `[A-Za-z_][A-Za-z0-9_]{0,31}`

4.3.2 TagType Description Element

A tag type's `Description` element is a container for information such as human-readable text, tag type titles, external references, and examples. It is defined in the same way as Section 3.3.3: Profile Description Element.

4.3.3 TagType Metadata Element

The tag type's `Metadata` element contains information that may be useful, but is not critical to the processing of a tag type. Taxonomy producers may wish to provide metadata to provide product-specific mappings or other specific information.

Currently, there is no defined metadata specific to a tag type.

[2] The format of the tag type name is identical to that of the Tag name (§3.2.1).

5 CDET Dictionary

The CDET Dictionary defines a collection of CEE Event field definitions and field value types. A CDET field definition describes one characteristic or property of an event (e.g., start time, action, status, user account name). Each field definition is associated with a name and a field type, which defines the format for valid values for that field. For example, a *file_name* field has values of a *string* type and a *src_ipv4* field has values corresponding to an *ipv4Address* type.

Each field and field type definition has a unique name that is used to identify the proper Dictionary definition. Therefore, the use of the CDET Dictionary names is similar to using a standard natural language dictionary. Users can look up the meaning of, or locate the proper term to describe a certain event property. For example, an event producer may wish to provide the account name of the user account affected by an event. To do so, they would search through the CEE Dictionary fields would inform that the correct event field to use would be *acct_name*. Similarly, if a product records an event field entitled *prod_proc_id*, the Dictionary would explain that this is the name for a field describes the process identifier of the process that produced this event.

5.1 CDET Dictionary Specification

The CDET Dictionary defines fields (`Field`) and field types (`FieldType`) (Figure 3). Each field type defines the type and syntax for valid values. Field definitions are enclosed within `Field` elements are used to describe various event properties. Each field definition may be associated with one field type.

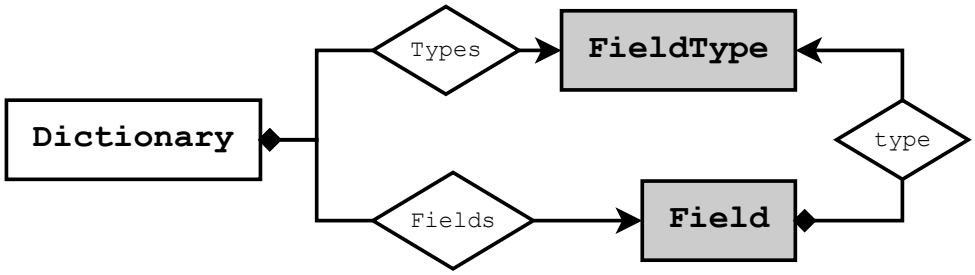
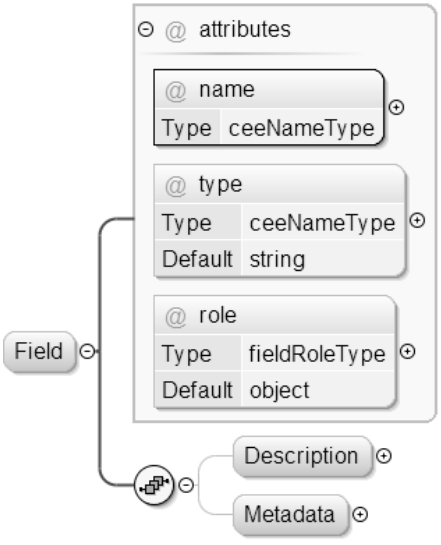


Figure 3: Dictionary Model

5.2 Field Definition

The `Field` element defines an event field, which represents a single event property. An event property may be a source IPv4 address, file name, user account name, destination port number, etc. Each field definition must have a unique, identifying name and be associated with one field type.

Within CEE, each field instance appearing in an event record has a name and a value list. The name should correspond to one field definition within a CDET Dictionary. The value list is a series of zero or more individual values, though typical field instances contain exactly one value. Each individual value should meet the criteria specified by the field's field type. If this occurs, then the field instance is validated against that field definition.



For example, an event record might have a field instance with the name *file_name* and a value list consisting of "File1.txt" and "File2.jpg". The corresponding *file_name* field definition in the CDET Dictionary specifies the cardinality of the field as being between 0 and 100 and having a type of a "string" field type. Since both values of the *file_name* field instance are within the specified cardinality and meet the criteria of being a "string" field type, the *file_name* field instance validates against the *file_name* field definition.

5.2.1 Field name Attribute

The Field definition's `name` attribute defines a unique sequence of characters that references the specific field definition and uses an identical format to that of the Taxonomy Tag name (§4.2.1). Each Field name must start with a letter (`[A-Za-z]`) or underscore ('_' `U+005F`) character followed by no more than 31 letter, digit (`[0-9]`), and underscore characters.

That is, each field type name must be no longer than 32 characters and conform to the following regular expression: `[A-Za-z_][A-Za-z0-9_]{0,31}`

5.2.2 Field role Attribute

The role attribute in a field definition indicates the role in the event to which the field corresponds. Currently, CEE recognized three (3) event roles: producer, subject, and object. The producer is the event record producer, or the source of the event record. The event subject reflects the initiator of the event action, while the event object is the target of the event.

Without this distinction, it would be difficult to tell whether a system name was the name of the system who produced the event record, initiated the event (e.g., uploaded files), or was the target of the event.

Each field can correspond to exactly one role. Not all events will have fields from each role.

5.2.3 Field type Attribute

The field definition's `type` attribute indicates the field type (as a reference to the proper `FieldType` definition) to which all valid values corresponding to this field definition must adhere (§5.4). If no type is specified, it should be assumed that the type is the default, "string" field type.

For example, the type can be used to define that fields corresponding to a *file_name* Field definition have values of type *string*, or that fields corresponding to an *event_time* Field definition have values of type *timestamp*.

5.2.4 Field Description Element

A field's `Description` element is a container for information such as human-readable text, field titles, external references, and examples. It is defined in the same way as Section 3.3.3: Profile Description Element.

5.2.5 Field Metadata Element

The field's `Metadata` element contains information that may be useful, but is not critical to the processing of a field. Dictionary producers may wish to provide metadata to provide product-specific mappings or other specific information.

5.2.6 Valid Field Examples

```
      <Field name="acct_name" role="object" xml:id="acct_name"
type="string">
         <Description>
            <Text_Title>Account Name</Text_Title>
            <Text>The name of the user account</Text>
         </Description>
      </Field>
      <Field name="file_name" role="object" xml:id="file_name"
type="string">
         <Description>
            <Text_Title>File Name</Text_Title>
         </Description>
      </Field>
      <Field name="file_path" role="object" xml:id="file_path"
type="string">
         <Description>
            <Text_Title>File Path</Text_Title>
            <Text>The path to the file that is the object of the event,
excluding the file name</Text>
         </Description>
      </Field>
      <Field name="time" role="object" xml:id="time" type="timestamp">
         <Description>
            <Text_Title>Event Start Time</Text_Title>
            <Text>An ISO8601 compliant timestamp designating the date,
time, and timezone offset when the event began</Text>
         </Description>
      </Field>
```

5.2.7 Invalid Field Examples

```
      <Field name="UndefTypeField" type="UndefinedType">
        <Description>
          <Text>A field to hold all of your base.</Text>
        </Description>
      </Field>
      <Field name="ABadField">
          <StringRestriction>
              <Length>1</Length>
          </StringRestriction>
      </Field>
```

All of these field definitions are invalid and are so for obvious reasons. The *UndefTypeField* field definition is trying to use a type that has yet to be defined and its `relatesToTag` metadata does not reference a CEE Tag definition. Similarly, the *ABadField* definition tries to declare the field type restrictions within the scope of the field.

5.3 Reserved Fields

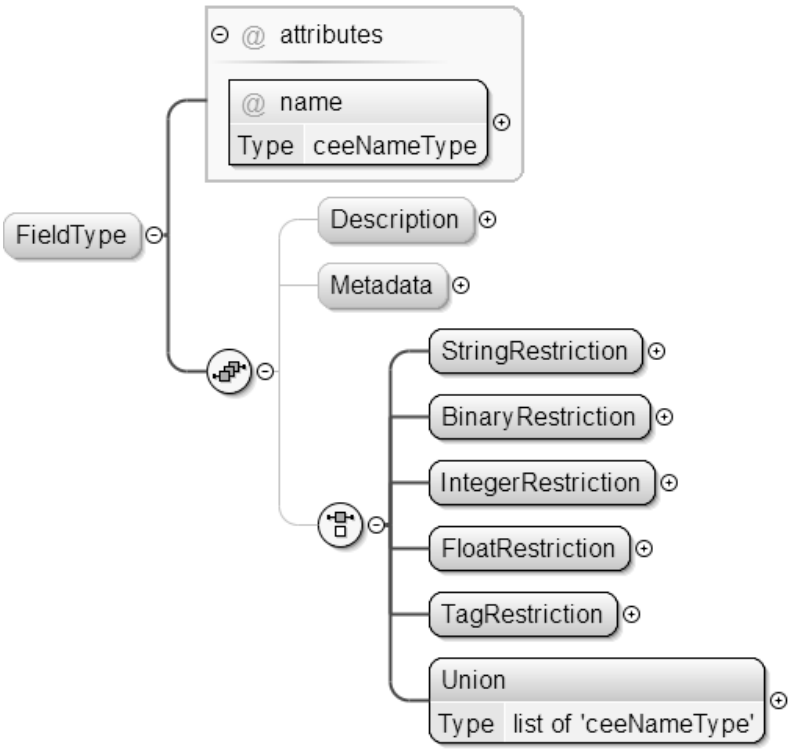
This specification reserves several field names and corresponding definitions for the sole use of CEE. Currently, no one outside of CEE can define or use field names beginning with "`cee`" (ABNF `%x63 %x65 %x65`). Other reserved field names belong to core fields required in CEE Events and defined in the CLS specification. CLS defined fields are listed below:

Table 3: Reserved Field Names

Field Name	Text
id	An id field is intended to identify semantically similar CEE Event records.
action	The action field holds an action TagType Tag that indicates the type of action occurring within the event
status	The status field holds a status TagType Tag that indicates the status or outcome of the event action
time	The time field indicates the date, time, and timezone offset indicating the time when the event began.
p_sys_id	The p_sys_id field indicates the host system that produced the event.
p_prod_id	The p_prod_id field indicates the product that produced the event, such as the process, application, or platform.

5.4 FieldType Definition

A field type definition is contained within a `FieldType` element and is required to have a name and format, along with an optional description. The format of the `FieldType` defines the range of acceptable values for the field and must be a string restriction, binary restriction, integer restriction, or union.



5.4.1 FieldType name Attribute

The field type definition's name is specified by a `name` attribute that uses a format identical to that used by the Taxonomy `Tag` and `TagType` `name` (§4.2.1, 4.3.1).

The field type's `name` is a required attribute whose value is a unique sequence of characters that is used to reference this specific field type definition. Each field type name must begin with a letter (`[A-Za-z]`) or underscore ('_' `U+005F`) character followed by no more than 31 letter, digit (`[0–9]`), and underscore characters.

Each field type name must conform to the regular expression: `[A-Za-z_][A-Za-z0-9_]{0,31}`

5.4.2 FieldType Description Element

A field type's `Description` element is a container for information such as human-readable text, field type titles, external references, and examples. It is defined in the same way as Section 3.3.3: Profile Description Element.

5.4.3 FieldType Metadata Element

The field type's `Metadata` element contains information that may be useful, but is not critical to the processing of a field type. CDET Dictionary publishers may wish to provide metadata to provide product-specific mappings or other specific information.

Currently, there is no defined metadata specific to a field type.

5.4.4 FieldType format

The field type's format specifies how values corresponding to that field type definition are represented. For example, this could be used to define a *port* field type as a type of integer with a minimum value of "1" (one) and a maximum value of "255."

A CDET field type format can be defined using a **restriction** or **union** of already defined `FieldTypes` (**Error! Reference source not found.**).

5.4.4.1 FieldType Restrictions

A restriction is used to place a limitation on the acceptable values supported by a field type. These field type restrictions are intended to be similar to the facets used by the XML Schema `xs:restriction` element in both names and functions.

Within a restriction, one or more restriction constraints can be specified. Current restriction constraints are `MinValue`, `MaxValue`, `Length`, `MinLength`, `MaxLength`, `Pattern`, `Enumeration`, `ABNF`, `TagType`, and `AnyTagType`.

Unlike the XML Schema `xs:Restriction`, CEE divides the eight (8) constraints across five (5) restriction groups: a `StringRestriction`, `BinaryRestriction`, `IntegerRestriction`, `FloatRestriction`, and a `TagRestriction`. The first four restrictions specify that the value of an associated field as be derived from CEE core *string*, *binary*, or *integer* field type. The `TagRestriction` is a special restriction that allows for a field to be one or more CEE Tags (Section 0).

5.4.4.1.1 MinValue Constraint (IntegerRestriction, FloatRestriction)

The `MinValue` constraint is identical to the XML Schema `xs:minInclusive` restriction and is used within an `IntegerRestriction` or `FloatRestriction` to express the minimal, inclusive value. This is the equivalent of saying that all expected values should be "greater than or equal to (≥)" the value specified within the element's text node.

It is important to keep in mind that the base field type specified within a restriction should have a natural order, whereby two items have a natural ordering.

The `MinValue` constraint can be used to mimic the XML Schema `xs:positiveInteger` type by declaring a new *PositiveIntType* field type with a `MinValue` integer restriction of "1" (one).

```
<FieldType name="PositiveIntType">
    <IntegerRestriction>
        <MinValue>1</MinValue>
    </IntegerRestriction>
</FieldType>
```

Table 4: FieldType Restrictions

Restriction	Applies to	Text
MinValue	`IntegerRestriction` `FloatRestriction`	The minimum value of an integer or floating-point number
MaxValue	`IntegerRestriction` `FloatRestriction`	The maximum value of an integer or floating-point number
MinLength	`StringRestriction` `BinaryRestriction`	The minimum length (in octets) of a string
MaxLength	`StringRestriction` `BinaryRestriction`	The maximum length (in octets) of a string
Length	`StringRestriction` `BinaryRestriction`	The exact length (in octets) of a string
Pattern	`StringRestriction`	An XML Regular Expression pattern the string must match [6]
ABNF	`StringRestriction` `BinaryRestriction`	An Augmented Backus-Naur Form (ABNF) grammar to which the string must conform [7]
Enumeration	`StringRestriction`	A specific value which the string must match
TagType	`TagRestriction`	The value must be a Taxonomy Tag of the specified TagType
AnyTagType	`TagRestriction`	The value must be a Taxonomy Tag having any Tag Type

5.4.4.1.2 MaxValue Constraint (IntegerRestriction, FloatRestriction)

The `MaxValue` constraint is identical to the XML Schema `xs:maxInclusive` restriction and is used within an `IntegerRestriction` or `FloatRestriction` to express the maximal, inclusive value. This is the equivalent of saying that all expected values should be "less than or equal to (≤)" the value specified as the constraint.

Like the `MinValue` constraint, it is important that the base field type specified within a restriction should have a natural order, whereby two items have a natural ordering.

The `MaxValue` constraint can be used to define a range field that holds a decimal number between 0.0 and 10.0 by declaring a field type with a `MinValue` and `MaxValue` restriction.

```
<FieldType name="DecimalRangeType">
    <FloatRestriction>
        <MinValue>0.0</MinValue>
        <MaxValue>10.0</MaxValue>
    </FloatRestriction>
</FieldType>
```

5.4.4.1.3 Length Constraint (StringRestriction, BinaryRestriction)

The `Length` constraint is similar to the XML Schema `xs:length` restriction. It is used within a `StringRestriction` or `BinaryRestriction` to specify that all expected values should be of a specific length, measured in the number of octets (if binary) or Unicode characters (if a string).

For example, by setting the `Length` constraint to "1" (one) on a *binary* field type, a "byte" field type can be defined that accepts only values of a single byte in length.

```
<FieldType name="byte">
    <BinaryRestriction>
        <Length>1</Length>
    </BinaryRestriction>
</FieldType>
```

If the `Length` constraint is specified, it is forbidden to also specify a `MaxLength` or `MinLength` constraint.

5.4.4.1.4 MinLength Constraint (StringRestriction, BinaryRestriction)

The `MinLength` constraint is similar to the XML Schema `xs:minLength` restriction and can be used in a `StringRestriction` or `BinaryRestriction` to specify the minimum number of octets or string length, for all expected values. In the case of `MinLength`, all values are treated as sequences of octets (if binary) or Unicode characters (if a string).

The `MinLength` constraint can be used to specify a new *NoEmptyStringType* field type that does not allow empty string values.

```
<FieldType name="NoEmptyStringType">
    <StringRestriction>
        <MinLength>1</MinLength>
    </StringRestriction>
</FieldType>
```

If both `MaxLength` and `MinLength` constraints are specified, it is required that the value `MaxLength` be greater than or equal to that of `MinLength`.

5.4.4.1.5 MaxLength Constraint (StringRestriction, BinaryRestriction)

The `MaxLength` constraint is similar to the XML Schema `xs:maxLength` restriction and can be used in a `StringRestriction` or `BinaryRestriction` to specify the maximum number of octets for all expected values. For `MaxLength`, all values are treated as a sequence of octets (if binary) or Unicode characters (if a string).

The `MaxLength` constraint can be used to specify a *HostnameType* field type that sets that the maximum length of a valid host name to "255" characters (and the minimum length to "1").

```
<FieldType name="HostnameType">
    <StringRestriction>
        <MaxLength>255</MaxLength>
        <MinLength>1</MinLength>
    </StringRestriction>
</FieldType>
```

If both `MaxLength` and `MinLength` constraints are specified, it is required that the value `MaxLength` be greater than or equal to that of `MinLength`.

5.4.4.1.6 Pattern Constraint (StringRestriction)

The `Pattern` constraint is similar to the XML Schema `xs:pattern` restriction and is used in a `StringRestriction` to express a regular expression pattern that all acceptable *string* values must match.

In the CDET `Pattern` element, the expected value format is expressed by an XML Regular Expression[3] provided within the element's text [6]. If multiple Patterns are expressed within the same Type restriction, this is equivalent to a logical "or" of the pattern regular expressions. I.e., a value is expected to match only one `Pattern`.

One use for a pattern is to define a type for a dotted-decimal IPv4 address string. A `Pattern` restriction can be defined within an *IPv4AddressType* FieldType to declare a value type accepts only a valid IPv4 address string.

```
<FieldType name="IPv4AddressType">
    <StringRestriction>
        <Pattern>((25[0-5]|2[0-4]\d|1?\d?\d)\.){3}
                 (25[0-5]|2[0-4]\d|1?\d?\d)</Pattern>[4]
    </StringRestriction>
</FieldType>
```

5.4.4.1.7 ABNF Constraint (StringRestriction, BinaryRestriction)

An alternative to using the `Pattern` constraint is the ABNF constraint. Instead of specifying a regular expression, an Augmented Backus-Naur Format (ABNF) grammar can be defined within an `ABNF` element according to the IETF RFC 5234 [7]. ABNF constraints can only be used within a `StringRestriction` or `BinaryRestriction`.

[3] The specification for XML Regular Expressions can be found at *http://www.w3.org/TR/xmlschema-2/#regexs*
[4] The `Pattern` value was split over multiple lines for readability purposes.

In order to match the *string* or *binary* values against the proper ABNF rules, the name of the primary ABNF rule must have an identical name (case-sensitive) to that of the field type definition. Notice how in the following example, the top-level ABNF rule is named identically to the name of the enclosing field type definition: *IPv4AddressABNFType*. Using an ABNF restriction, an IPv4 address field type can be defined that yields equivalent accepting states to the *IPv4AddressType* pattern defined above.

```
<FieldType name="IPv4AddressABNFType">
    <StringRestriction>
        <ABNF>
        IPv4AddressABNFType = IPV4DEC 3( "." IPV4DEC )
        IPV4DEC = (%x32 (%x35 %x30-35) / (%x30-%x34 %x30-%x39)) /
                  (%x31 2DIGIT) / (1*2DIGIT)
        </ABNF>
    </StringRestriction>
</FieldType>
```

One additional requirement is that ABNF rule names must not appear within angle brackets ("<", ">"). While this is required in the Backus-Naur Format (BNF) and permitted by IETF RFC 5234, it adds unnecessary complexity and conflicts with the XML restrictions on the less-than (< U+003C) and greater-than sign (> U+003E) characters.

5.4.4.1.8 Enumeration Constraint (StringRestriction)

The `Enumeration` constraint is similar to the XML Schema `xs:enumeration` restriction. Each `Enumeration` element specifies one acceptable value for that field type. For a *string* value to meet this field type's `StringRestriction`, it must match only one of the Enumeration constraints.

One common use of enumerations is to specify a menu of choices. For instance, all of the two-character state postal abbreviations for the United States can be specified using an `Enumeration` constraint such as in the *USStateAbbrType* field type.

```
<FieldType name="USStateAbbrType">
    <StringRestriction>
        <Enumeration>AL</Enumeration>
        <Enumeration>AK</Enumeration>
        <Enumeration>AZ</Enumeration>
        <Enumeration>AR</Enumeration>
        …
    </StringRestriction>
</FieldType>
```

5.4.4.1.9 TagType Constraint (TagRestriction)

The `TagType` element is used in a `TagRestriction` to constrain the type (TagType) of CEE Tag values that can be used for a field value. All field values must match a CEE Tag whose TagType is one of the TagTypes specified in a TagType constraint.

While a `TagRestriction` can have multiple `TagType` constraints, each constraint must specify precisely one TagType value. Each `TagType` value must correspond to a Taxonomy TagType definition (§4.3).

One example of a TagType constraint is the CEE *actionTagType* field type that defines a field of having only values of an *actionTag* TagType.

```
<FieldType name="actionTagType">
    <TagRestriction>
        <TagType>actionTag</TagType>
    </TagRestriction>
</FieldType>
```

5.4.4.1.10 AnyTagType Constraint (TagRestriction)

The `AnyTagType` constraint allows for values to be any CEE Tag, regardless of that tag's type. If a field value can be of any CEE Tag, then the `AnyTagType` element should be used instead of individual `TagType` constraints. The `AnyTagType` element must not have any associated values or attributes.

The `AnyTagType` constraint is used with the CEE *tag* field type. The CEE *tags* field allows for values consisting of any CEE Tag.

```
<FieldType name="tag">
    <TagRestriction><AnyTagType/></TagRestriction>
</FieldType>
```

5.4.4.2 FieldType Union Element

A union type is a value type that is a combination of other types.

Union value types are indicated by a `Union` element within a `FieldType` definition and require a value that is a space (`U+0020`) delimited list of the names of one or more field type names. It is forbidden to this value an empty list and it is forbidden for any one of the field type names to be a reference to the current field type definition.

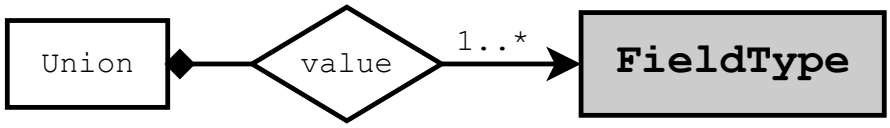


Figure 4: FieldType Union Model

For example, an *IPAddressType* field type can be defined as a union of an *IPv4AddressType* and an *IPv6AddressType*.

```
<FieldType name="IPv4AddressType">
    <StringRestriction>
        <Pattern>…</Pattern>
    </StringRestriction>
</FieldType>
<FieldType name="IPv6AddressType">
    <StringRestriction>
        <Pattern>…</Pattern>
    </StringRestriction>
</FieldType>
<FieldType name="IPAddressType">
    <Union>IPv4AddressType IPv6AddressType</Union>
</FieldType>
```

5.4.5 Valid FieldType Examples

```
<FieldType name="actionTagType">
   <TagRestriction>
      <TagType>actionTag</TagType>
   </TagRestriction>
</FieldType>
<FieldType name="emailAddress">
   <Description>
      <Text_Title>E-mail Address</Text_Title>
   </Description>
   <StringRestriction>
      <Pattern
          >[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]+</Pattern>
   </StringRestriction>
</FieldType>
<FieldType name="boolean">
   <Description>
      <Text_Title>Boolean</Text_Title>
      <Text>A Boolean value: "true" or "false"</Text>
   </Description>
   <StringRestriction>
      <Enumeration>true</Enumeration>
      <Enumeration>false</Enumeration>
      <ABNF>boolean = "true" / "false"</ABNF>
   </StringRestriction>
</FieldType>
```

5.4.6 Invalid FieldType Examples

```
<FieldType name="MyDigitType">
    <StringRestriction>
        <Pattern>[:digit:]+</Pattern>
    <StringRestriction>
</FieldType>
```

The issue within the above *MyDigitType* FieldType example is that the `Pattern` constraint is expressed using a POSIX regular expression, and is not a valid XML Regular Expression.

```
<FieldType name="InvalidIntType">
    <StringRestriction>
        <MinValue>1</MinValue>
    </StringRestriction>
</FieldType>
```

The *InvalidIntType* FieldType definition is invalid as it sets a MinValue constraint (which is an integer restriction) within a StringRestriction.

5.4.7 Reserved FieldType Definitions

In addition to the new field types defined in a CDET Dictionary, CEE defines a core set of base field types: *string*, *binary*, *integer*, *float*, and *boolean* to provide representations for strings, binary values, integers, floating-point and Boolean values. CEE also defines several commonly used string types for the representation of IPv4, IPv6, and MAC addresses, and ISO8601 timestamps and durations [8]. Finally, CEE defines a special core field type called *tag*, whose

value is a name reference one CEE Tag definition (§0) within a CDET Taxonomy. A listing of the core field types provided by this specification can be found in Table 5.

Table 5: CEE Core Field Types

Field Type	Description	Example
binary	A sequence of zero or more octets. Some CLS Encodings require binary values to be Base64 encoded	01ab
boolean	A Boolean value: "true" or "false"	true, false
duration	An ISO8601 compliant representation of a duration of time [8]	P1DT1H, PT1.123S
float	A 64-bit binary (“double”) floating point number	1.0, .5, 12.3e10, -1.1E-1
integer	An signed 64-bit integer value	12345, -392456
ipv4Address	The dotted decimal IPv4 address notation	10.0.0.1
ipv6Address	A valid IPv6 string representation (e.g., colon-hex notation) as defined in RFC4291 [9]	12AB:0:0:0:123:4567:89AB:CDEF ::13.1.68.3, ::1 1080:0:0:0:8:800:200C:417A
macAddress	A string representation of a MAC address as defined by the IEEE 802 specification [10]	01:23:45:67:89:ab 98-76-54-32-10-fe
string	A sequence of zero or more Unicode characters [11]	abcd123, ÇæøΞ¡
tag	A tag name relating to a CEE Tag definition within a CDET Taxonomy (§0)	action.login. status.success service.fw, object.file

The representation for each core field type is deferred to the CEE Log Syntax (CLS) specification and the actual syntax used to encode the event record. While a *string* value will always conceptually be a series of Unicode characters, different syntaxes may define different ways to represent them. This is even more evident for *integer* field types, where some CLS Encodings may prefer a sequence of digits and others prefer a more compact binary representation.

It is forbidden for a CDET Dictionary to redefine any CEE core field types.

6 CELR Event Profile

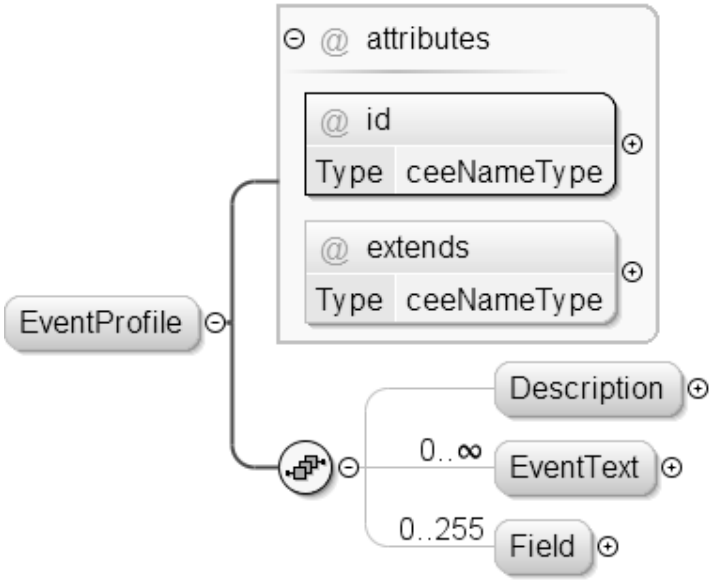
The intent of CEE Event Log Recommendations (CELR) and the Event Profiles are to provide recommendations, or guidance on events to record in logs. CELR Event Profiles were developed in support of the following use cases:

- Logging Guidance: CELR is intended to provide logging guidance to users and vendors on event, including the event details, a device should record in order to provide optimal value to log consumers.
- Platform Independent Event Data: Implementation of common event names and fields enable audit and log data to be standardized across platforms. The common event names, fields, and taxonomy descriptors improve detection, correlation and aggregation abilities.
- Machine Readable Log Policy: The capability to define a log policy that is machine-readable provides the ability to perform activities such as validating that a device is logging events according to a specified profile.
- Event Reconstruction: Event reconstruction is also known as digital forensics. Complete logs with unambiguous data assist with determining what occurred and when it occurred.

6.1 Event Profile Definitions

An Event Profile describes the event and the event structure. The event structure defines the tags, tag types, and fields that comprise the associated event record. Each event, tag, tag type, or field component is required or optional. Required components must appear in all related CEE event records[5]. Optional components may appear.

Each Event Profile definition is represented by an `EventProfile` element within a CEE Profile document.



[5] A "CEE event record" is a record of an event that is compliant with at least one of the CEE Log Syntax (CLS) Encodings defined in compliance with the CEE Log Syntax specification [5].

6.1.1 id Attribute

The `id` attribute is required and defines a unique identifier that corresponds to individual Event Profile definitions. Each `EventProfile id` must be unique within scope of the profile and should be meaningful to a human operator. Later, when recording an event, this event profile identifier can be used to associate a CEE event record to the corresponding Event Profile definition. This is done by matching the Event Profile's `id` value to the `id` field of a CEE event record.

An `EventProfile id` attribute value must is in an identical format to that of the Taxonomy Tag and Dictionary Field names; the `id` value must begin with a letter (`[A-Za-z]`) or underscore ('_' `U+005F`) character followed by no more than 31 letter, digit (`[0-9]`), and underscore characters. I.e., the `id` value must be no longer than 32 characters and conform to the following regular expression: `[A-Za-z_][A-Za-z0-9_]{0,31}`

All MITRE-provided CEE event definitions will begin with the prefix `cee_`. Event profile definitions contributed by third parties should begin with a common prefix that identifies the contributor or product, but must not begin with the prefix `cee_`.

6.1.2 extends Attribute

The `extends` attribute enhances reusability and reduces the need for duplication by allowing event profiles to extend the structure of another event profile. To extend another event profile, specify the `id` of the `EventProfile` definition to be extended. When declaring an extension, the event profile inherits all of the structure from the event profile being extended. Every event profile should be derived from the CEE base event profile: `cee_base_event`.

For example, an event definition *example1* requires fields with names `time`, `action`, `src_ip`, and `dst_ip`. If a new event definition, *example2*, were to extend the *example1* definition, it would inherit all of the structure of *example1*. In addition to the inherited structure, *example2* could also require an `acct_name` field.

Since the structure contains optional as well as required components, rules are needed to define valid and invalid extensions. In the following rules, the term **parent** refers to the event definition being extended (*example1*) and **child** refers the extended event (*example2*).

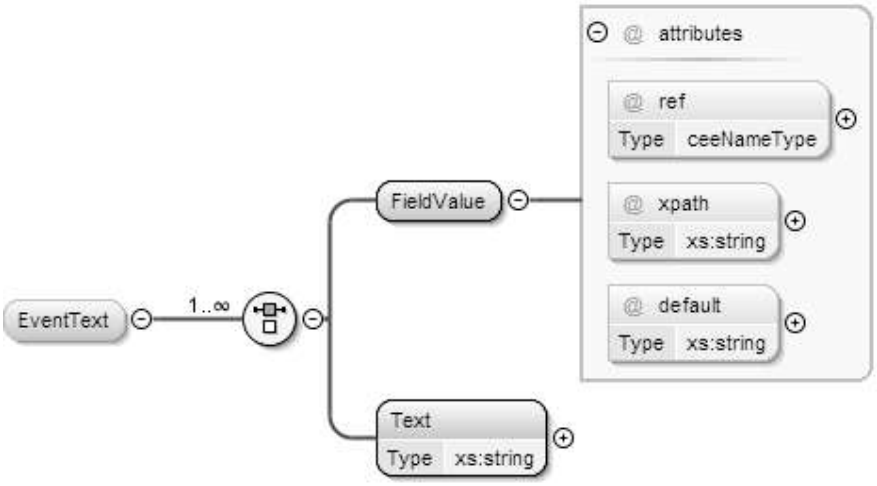
- Each component of any event definition structure must be a field.
- Each component is either "required" or "optional"; if not specified, it is required.
- Each component in *parent* must be in *child* (either implicitly or explicitly declared).
- If a component in *parent* is required, it must be required in *child*.
- If a component in *parent* is optional, it must be optional or required in *child*.

6.1.3 EventProfile Description Element

An `EventProfile`'s `Description` element is a container for information such as human-readable text describing the event type and uses, an event title, external references, and examples. It is defined in the same way as Section 3.3.3: Profile Description Element.

6.1.4 EventProfile EventText Element

The purpose of the `EventText` element is to provide human-readable translations of an event profile.



An `EventText` element is used to build a text description of the event. This is done by using a combination of `Text` and `FieldValue` elements. A `Text` element is used to define static text. While `FieldValue` elements can be used to insert values from the CEE event record fields. The value indicated by the `FieldValue` attributes should be inserted in place of the `FieldValue` element when the event record is being displayed as text.

The `FieldValue ref` attribute must identify the field name whose values should be inserted. If a field with that name does not exist, the value of the `default` attribute should be used instead.

One other option that can be used to define the substitute value is by using the `FieldValue xpath` attribute. The `xpath` value should be a valid XPath 2.0 expression [12], which should be executed against a CEE event record formatted using CEE Log Syntax (CLS) XML Encoding [13]. The output of the XPath expression will be substituted in place of the `FieldValue` element.

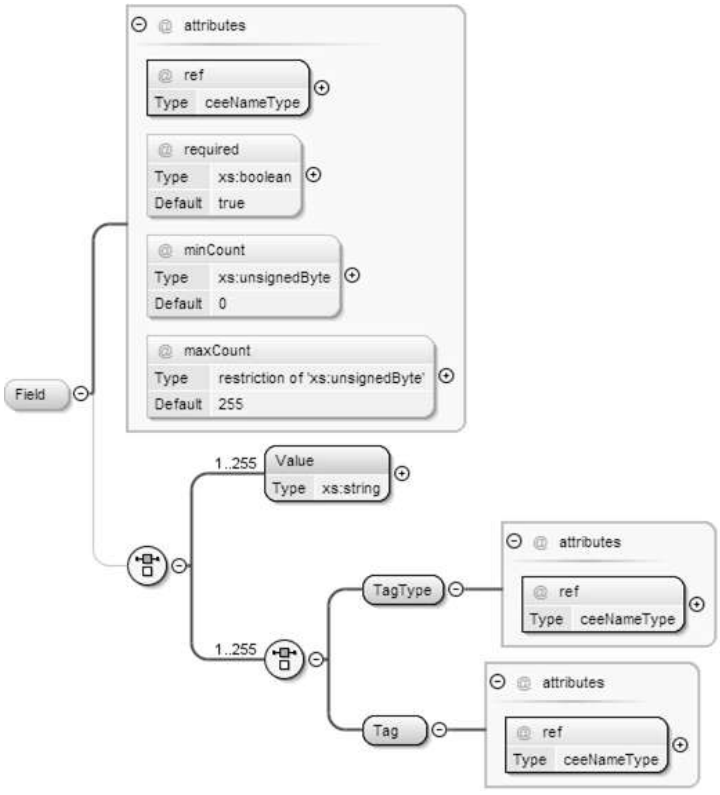
The `ref` and `xpath` attributes should not both be provided. If they are, the `xpath` is more expressive and should take precedence.

6.1.5 EventProfile Field Element

The `Field` element declares the fields and corresponding field types and values that should be present in conformant CEE event records.

The `Field ref` attribute is required and provides the name of a field that should be present in all corresponding event records. The `ref` value must correspond to the name of a valid CDET Dictionary Field definition (Section 5.2) and be defined in the local CEE Profile document, including by resolution of any Include elements (Section 3.3.5).

The `required` attribute is used to indicate whether the field must be present in a conformant CEE event



record. If `required` is true, then the field must be present. If the `required` attribute is absent, then the field must be treated as being required with the `required` attribute set to true.

The `minCount` and `maxCount` attributes can be used to set bound on the number of values the field must have in conformant event records. The `minCount` value must between 0 (zero) and 255. The `maxCount` value must between 1 and 255, and be greater than or equal to the `minCount` value.

6.1.6 EventProfile Field Values

In addition to specifying the required and optional event fields along with their value cardinality, an event profile can also define the values that the field must have.

Each `EventProfile Field` may provide 0 (zero) to 255 field values. Each field value must be identified as being a `Tag`, `TagType`, or `Value`. Value elements contain the literal text value that the corresponding field in the event record must have. The `Tag` and `TagType` elements indicate the tag or tag type value the field must have. The `ref` attribute holds an identifier that must correspond to a Tag (§4.2) or TagType (§4.3) definition in the scope of the CEE Profile document.

6.2 Event Profile Excerpt

```
      <EventProfile id="cee_base_event" xml:id="cee_base_event">
        <Description>
           <Text_Title>CEE Base Event Profile</Text_Title>
           <Text>The base event structure for CEE Events. All CEE formatted events are expected
  to minimally conform to this event profile. The CEE Base Event is derived from the Syslog
  RFC5424 event structure [http://tools.ietf.org/html/rfc5424].</Text>
      </Description>
      <EventText><FieldValue ref="time"/><Text> </Text><FieldValue ref="p_sys_id"/><Text>
  </Text><FieldValue ref="p_prod_id"/><Text> </Text><FieldValue ref="id"/><Text>
  [</Text><FieldValue ref="action"/><Text> </Text><FieldValue
  ref="status"/><Text>]</Text></EventText>
      <Field ref="time" required="true"/>
      <Field ref="id" required="true"/>
      <Field ref="p_sys_id" required="true"/>
      <Field ref="p_prod_id" required="true"/>
      <Field ref="action" minCount="1" required="true"/>
      <Field ref="status" minCount="1" required="true"/>
      <Field ref="rec_id" required="false"/>
      <Field ref="crit" required="false"/>
      <Field ref="end_time" required="false"/>
      <Field ref="dur" required="false"/>
      <Field ref="tags" required="false"/>
    </EventProfile>
```

7 Evaluating Profile Conformance

A CEE event record can be tested against one or more CEE Profiles. An event record that complies with that CEE profile is said to be conformant with that profile.

To check whether a CEE Event Record, **R**, is compliant with a CEE Profile document, **C**, perform the following operations. If at any time a check fails, then **R** is not compliant with **C**.

1. Recursively resolve all `Includes` instructions for **C** to generate **C'**
2. Find the event profile, **P**, whose `id` attribute value matches that of the `id` field in **R**
3. Recursively resolve all `extends` instructions in **P** to generate **P'**
4. For each value in each field in **R**, validate that the value agrees with the field type defined in the associated field definition in **C'**
5. For each required field **P'**, validate that an identically named field exists in **R**
6. For each value of each field in **P'** where the value is a `Tag`, validate that that exact value must appears as a tag value in an identically named field in **R**, if such a field exists
7. For each value of each field in **P'** where the value is a `TagType`, validate that a tag value having that tag type appears in an identically named field in **R**, if such a field exists
8. For each value of each field in **P'** where the value is a `Value`, validate that that exact value appears as a non-tag value in an identically named field in **R**, if such a field exists

8 CEE Management

CEE is managed according to the CEE Architecture Management process [1]. This process identifies that the CEE Board is responsible for maintaining all CEE specifications (including this CEE Profile specification), CEE XML Schemas, and revisions to the CELR Event Profile, CDET Dictionary, and CDET Taxonomy definitions. The CEE Board reviews inputs and requests from the community regarding modifications, updates, and new feature requests for CEE. Figure 5 outlines the change management process for all CEE products.

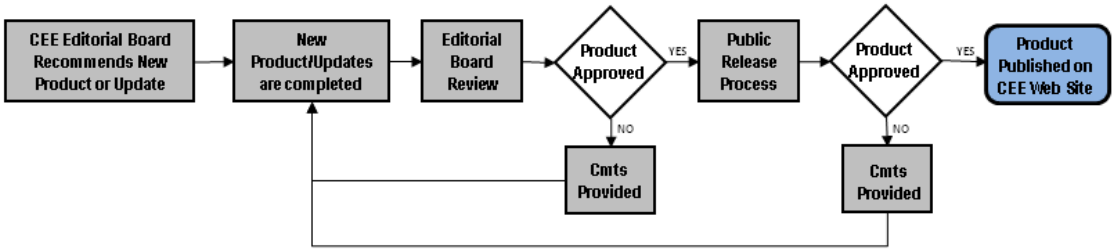


Figure 5: CEE Request Management Process

Prior to major CEE revisions or new features becoming finalized, a draft product will be made available to the CEE Community for review and comment. Once the comment period has ended, per the direction of the CEE Editorial Board, the draft document will be updated to address comments, re-reviewed, finalized, and published. In addition to the drafting and review of major revisions, the CEE Editorial Board is also responsible for approving additions or modifications to the CDET Dictionary or CDET Taxonomy instances included in the official CEE repository.

9 Summary

The combination of the CEE Dictionary and Event Taxonomy (CDET) defines a standardized vocabulary for recording events. The Dictionary and Taxonomy entries provide a collection of event field and tag definitions that can be leveraged by vendors and users to improve their products or audit processes. The wider the adoption of CEE, more people will utilize CEE terminology in their event records and the more the CEE Community will benefit.

The success of CEE depends on its ability to meet the needs of the audit and log community. Therefore, it is vital that the CEE Community contribute to and promote the successes and adoption of CEE. Your inputs, feedback, and discussion are crucial to the production of an open, practical, and industry-accepted standard. Comments concerning CEE, this CEE Specification, the CEE repository, or CEE in general, should be submitted to the CEE Discussion List (`cee-discussion-list@lists.mitre.org`) or to the MITRE CEE Team (`cee@mitre.org`).

This page intentionally left blank.

Appendix A References

[1] CEE Editorial Board. (2010, February) Common Event Expression: CEE Architecture Overview.

[2] The MITRE Corporation. (2008, June) Common Event Expression: CEE, A Standard Log Language for Event Interoperability in Electronic Systems. [Online]. http://cee.mitre.org

[3] CEE Editorial Board. (2011, June) Common Event Expression: CEE Log Syntax (CLS) Specification, Version 0.6.

[4] CEE Editorial Board. (2010, November) Common Event Expression: CEE Dictionary and Event Taxonomy Specification v0.5.1.

[5] Brant A. Cheikes, David Waltermire, and Karen Scarfone. (2012, April) Common Platform Enumeration: Naming Specification, Version 2.3 (DRAFT). [Online]. http://csrc.nist.gov/publications/drafts/nistir-7695/2nd-draft-NISTIR-7695-CPE-Naming-Apr2011.pdf

[6] W3C. (2008, November) Extensible Markup Language (XML) 1.0 (Fifth Edition). [Online]. http://www.w3.org/TR/2008/REC-xml-20081126/

[7] Dave Crocker and Paul Overell. (2008, January) Augmented BNF for Syntax Specifications: ABNF (RFC 5234). [Online]. http://tools.ietf.org/html/rfc5234

[8] International Organization for Standardization (ISO), "Data elements and interchange formats - Information interchange - Representation of dates and times," ISO, Geneva, ISO 8601:2004(E), 2004.

[9] Robert M Hinden and Stephen E Deering. (2006, February) IP Version 6 Addressing Architecture (RFC 4291). [Online]. http://tools.ietf.org/html/rfc4291

[10] IEEE Computer Society. (2002, February) IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture. [Online]. http://standards.ieee.org/getieee802/download/802-2001.pdf

[11] The Unicode Consortium. (2009) The Unicode Standard, Version 5.2.0, defined by: The Unicode Standard, Version 5.2. [Online]. http://www.unicode.org/versions/Unicode5.2.0/

[12] W3C. (2010, December) XML Path Language (XPath) 2.0 (Second Edition). [Online]. http://www.w3.org/TR/xpath20/

[13] CEE Editorial Board. (2011, June) CLS Encoding: XML.

[14] Andrew Buttner and Neal Ziring. (2009, March) Common Platform Enumeration (CPE) - Specification, Version 2.2. [Online]. http://cpe.mitre.org/files/cpe-specification_2.2.pdf

This page intentionally left blank.

Appendix B XML Schema for CEE Profile

B.1 profile_v0.6.xsd

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://cee.mitre.org" xmlns="http://cee.mitre.org"
xmlns:ns1="http://cee.mitre.org/INTERNAL" xmlns:sch="http://purl.oclc.org/dsdl/schematron"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:include schemaLocation="common.xsd"/>
  <xs:element name="CEEProfile">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" ref="Description"/>
        <xs:choice>
          <xs:element name="Base"/>
          <xs:element name="Function">
            <xs:complexType>
              <xs:attribute name="name" type="xs:token" use="required"/>
            </xs:complexType>
          </xs:element>
          <xs:element maxOccurs="unbounded" name="Product">
            <xs:complexType>
              <xs:attribute name="name" type="xs:token"/>
              <xs:attribute name="vendor" type="xs:token"/>
              <xs:attribute name="version" type="xs:token"/>
              <xs:attribute name="cpe" type="cpeFormatString"> </xs:attribute>
            </xs:complexType>
          </xs:element>
        </xs:choice>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="Include">
          <xs:complexType>
            <xs:attribute name="href" type="xs:anyURI" use="required"/>
          </xs:complexType>
        </xs:element>
        <xs:element minOccurs="0" name="Tags">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" minOccurs="0" name="TagType">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element minOccurs="0" ref="Description"/>
                    <xs:element minOccurs="0" name="Metadata">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any"
processContents="lax"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name="name" type="ceeNameType" use="required"/>
                  <xs:attribute ref="xml:id"/>
                </xs:complexType>
              </xs:element>
              <xs:element maxOccurs="unbounded" minOccurs="0" name="Tag">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element minOccurs="0" ref="Description"/>
                    <xs:element minOccurs="0" name="Metadata">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any"
processContents="lax"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name="name" type="ceeNameType" use="required"/>
                  <xs:attribute ref="xml:id"/>
```

```
                    <xs:attribute name="type" type="ceeNameType" use="required"/>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element minOccurs="0" name="Fields">
            <xs:complexType>
              <xs:sequence>
                <xs:element maxOccurs="unbounded" minOccurs="0" name="FieldType">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element minOccurs="0" ref="Description"/>
                      <xs:element minOccurs="0" name="Metadata">
                        <xs:complexType>
                          <xs:sequence>
                            <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any"
processContents="lax"/>
                          </xs:sequence>
                        </xs:complexType>
                      </xs:element>
                      <xs:choice minOccurs="0">
                        <xs:element name="StringRestriction">
                          <xs:complexType>
                            <xs:group maxOccurs="unbounded" ref="stringFacets"/>
                          </xs:complexType>
                        </xs:element>
                        <xs:element name="BinaryRestriction">
                          <xs:complexType>
                            <xs:group ref="binaryFacets"/>
                          </xs:complexType>
                        </xs:element>
                        <xs:element name="IntegerRestriction">
                          <xs:complexType>
                            <xs:group ref="integerFacets"/>
                          </xs:complexType>
                        </xs:element>
                        <xs:element name="FloatRestriction">
                          <xs:complexType>
                            <xs:group ref="floatFacets"/>
                          </xs:complexType>
                        </xs:element>
                        <xs:element name="TagRestriction">
                          <xs:complexType>
                            <xs:group ref="tagFacets"/>
                          </xs:complexType>
                        </xs:element>
                        <xs:element name="Union">
                          <xs:simpleType>
                            <xs:list itemType="ceeNameType"/>
                          </xs:simpleType>
                        </xs:element>
                      </xs:choice>
                    </xs:sequence>
                    <xs:attribute name="name" type="ceeNameType" use="required"/>
                    <xs:attribute ref="xml:id"/>
                  </xs:complexType>
                </xs:element>
                <xs:element maxOccurs="unbounded" minOccurs="0" name="Field">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element minOccurs="0" ref="Description"/>
                      <xs:element minOccurs="0" name="Metadata">
                        <xs:complexType>
                          <xs:sequence>
                            <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any"
processContents="lax"/>
                          </xs:sequence>
                        </xs:complexType>
                      </xs:element>
                    </xs:sequence>
                    <xs:attribute name="name" type="ceeNameType" use="required"/>
                    <xs:attribute default="string" name="type" type="ceeNameType"/>
```

```
                  <xs:attribute ref="xml:id"/>
                  <xs:attribute default="1" name="minCard" type="cardValueType">
                    <xs:annotation>
                      <xs:documentation>The minimum number of values that an actual instance of
the field should have. This value must be between 1 and 255, inclusive.</xs:documentation>
                    </xs:annotation>
                  </xs:attribute>
                  <xs:attribute default="1" name="maxCard" type="cardValueType">
                    <xs:annotation>
                      <xs:documentation>The maximum number of values that an actual instance of
the field should have. This value must be between 1 and 255, inclusive, and greater than or
equal to the value of the minCard parameter.</xs:documentation>
                    </xs:annotation>
                  </xs:attribute>
                  <xs:attribute default="object" name="role" type="fieldRoleType"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="EventProfile">
          <xs:complexType>
            <xs:sequence>
              <xs:element minOccurs="0" ref="Description"/>
              <xs:element maxOccurs="unbounded" minOccurs="0" name="EventText">
                <xs:complexType>
                  <xs:choice maxOccurs="unbounded">
                    <xs:element name="FieldValue">
                      <xs:complexType>
                        <xs:attribute name="ref" type="ceeNameType"/>
                        <xs:attribute name="xpath" type="xs:string"/>
                        <xs:attribute name="default" type="xs:string"/>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="Text" type="xs:string"/>
                  </xs:choice>
                  <xs:attribute ref="xml:lang"/>
                </xs:complexType>
              </xs:element>
              <xs:element maxOccurs="255" minOccurs="0" name="Field">
                <xs:complexType>
                  <xs:choice minOccurs="0">
                    <xs:element maxOccurs="255" minOccurs="1" name="Value" type="xs:string"/>
                    <xs:choice maxOccurs="255" minOccurs="1">
                      <xs:element name="TagType">
                        <xs:complexType>
                          <xs:attribute name="ref" type="ceeNameType" use="required"/>
                        </xs:complexType>
                      </xs:element>
                      <xs:element name="Tag">
                        <xs:complexType>
                          <xs:attribute name="ref" type="ceeNameType" use="required"/>
                        </xs:complexType>
                      </xs:element>
                    </xs:choice>
                  </xs:choice>
                  <xs:attribute name="ref" type="ceeNameType" use="required"/>
                  <xs:attribute default="true" name="required" type="xs:boolean"/>
                  <xs:attribute default="0" name="minCount" type="xs:unsignedByte"/>
                  <xs:attribute default="255" name="maxCount">
                    <xs:simpleType>
                      <xs:restriction base="xs:unsignedByte">
                        <xs:minInclusive value="1"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:attribute>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="id" type="ceeNameType" use="required"/>
            <xs:attribute ref="xml:id"/>
            <xs:attribute name="extends" type="ceeNameType"/>
          </xs:complexType>
```

```
        </xs:element>
      </xs:sequence>
      <xs:attribute name="name" type="ceeNameType" use="required">
        <xs:annotation>
          <xs:documentation>The name of the profile. This must match the filename without the
extension. For example, if the profile filename is 'cee_foo_profile.xml', this @name value must
be 'cee_foo_profile'.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="url" type="httpURI">
        <xs:annotation>
          <xs:documentation>The full, remote URL where this CEEProfile is
located</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>
  </xs:element>
  <xs:group name="stringFacets">
    <xs:choice>
      <xs:element name="Length">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedShort">
            <xs:maxInclusive value="2048"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="MinLength" type="xs:unsignedShort"/>
      <xs:element name="MaxLength">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedShort">
            <xs:maxInclusive value="2048"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="ABNF" type="xs:string">
        <xs:annotation>
          <xs:documentation>Restriction written according to Augmented Backus-Naur expressions
[RFC 2234]</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element maxOccurs="unbounded" name="Pattern" type="xs:string"/>
      <xs:element maxOccurs="unbounded" name="Enumeration" type="xs:string"/>
    </xs:choice>
  </xs:group>
  <xs:group name="binaryFacets">
    <xs:choice>
      <xs:element name="Length">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedShort">
            <xs:maxInclusive value="2048"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="MinLength" type="xs:unsignedShort"/>
      <xs:element name="MaxLength">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedShort">
            <xs:maxInclusive value="2048"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="ABNF" type="xs:string">
        <xs:annotation>
          <xs:documentation>Restriction written according to Augmented Backus-Naur expressions
[RFC 2234]</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:choice>
  </xs:group>
  <xs:group name="integerFacets">
    <xs:all>
      <xs:element minOccurs="0" name="MinValue" type="xs:long"/>
```

```
        <xs:element minOccurs="0" name="MaxValue" type="xs:long"/>
      </xs:all>
    </xs:group>
    <xs:group name="floatFacets">
      <xs:all>
        <xs:element minOccurs="0" name="MinValue" type="xs:double"/>
        <xs:element minOccurs="0" name="MaxValue" type="xs:double"/>
      </xs:all>
    </xs:group>
    <xs:group name="tagFacets">
      <xs:choice>
        <xs:element maxOccurs="unbounded" name="TagType" type="ceeNameType"/>
        <xs:element name="AnyTagType"/>
      </xs:choice>
    </xs:group>
    <xs:simpleType name="ceeNameType">
      <xs:restriction base="xs:string">
        <xs:pattern value="[A-Za-z_][A-Za-z0-9_]{0,31}"/>
        <xs:maxLength value="32"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="cpeFormatString">
      <xs:restriction base="xs:token">
        <xs:pattern value="cpe(:[^:]*)+"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="httpURI">
      <xs:restriction base="xs:anyURI">
        <xs:pattern value="[hH][tT][tT][pP][sS]?://.+"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="cardValueType">
      <xs:restriction base="xs:positiveInteger">
        <xs:maxInclusive value="255"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="fieldRoleType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="object"/>
        <xs:enumeration value="subject"/>
        <xs:enumeration value="producer"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:schema>
```

B.2 common.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:group name="Structured_Text_Group">
    <xs:sequence>
      <xs:choice maxOccurs="unbounded">
        <xs:element maxOccurs="unbounded" minOccurs="0" name="Text_Title">
          <xs:annotation>
            <xs:documentation> Presentation Element: This element is used to define a bold-
faced title for a subsequent block of text. </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:string">
                <xs:attribute ref="xml:lang"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="Text">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:string">
                <xs:attribute ref="xml:lang"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="Code_Example_Language"
type="Language_Type">
          <xs:annotation>
            <xs:documentation>Presentation Element: This element is used to identify the
programming language being used in the following block of Code</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="Code" type="xs:string">
          <xs:annotation>
            <xs:documentation> Presentation Element: This element is used to define a line of
code. </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="Example" type="xs:string"/>
        <xs:element minOccurs="0" name="Images">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="Image">
                <xs:annotation>
                  <xs:documentation> Presentation Element: This element is used to define an
image. </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence maxOccurs="unbounded">
                    <xs:element name="Image_Location" type="xs:string">
                      <xs:annotation>
                        <xs:documentation>This element provides the location of the image
file.</xs:documentation>
                      </xs:annotation>
                    </xs:element>
                    <xs:element name="Image_Title" type="xs:string">
                      <xs:annotation>
                        <xs:documentation>This element provides a title for the
image.</xs:documentation>
                      </xs:annotation>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
```

```
        </xs:element>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="Reference" type="Reference_Type">
</xs:element>
      </xs:choice>
    </xs:sequence>
  </xs:group>
  <xs:simpleType name="Language_Type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="C"/>
      <xs:enumeration value="C++"/>
      <xs:enumeration value="C#"/>
      <xs:enumeration value="Java"/>
      <xs:enumeration value="JSP"/>
      <xs:enumeration value="Javascript"/>
      <xs:enumeration value="ASP.NET"/>
      <xs:enumeration value="SQL"/>
      <xs:enumeration value="Python"/>
      <xs:enumeration value="Perl"/>
      <xs:enumeration value="PHP"/>
      <xs:enumeration value="SOAP"/>
      <xs:enumeration value="Ruby"/>
      <xs:enumeration value="Shell"/>
      <xs:enumeration value="PseudoCode"/>
      <xs:enumeration value=".NET"/>
      <xs:enumeration value="Assembly"/>
      <xs:enumeration value="XML"/>
      <xs:enumeration value="HTML"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="Description" type="Structured_Text_Type"> </xs:element>
  <xs:complexType name="Reference_Type">
    <xs:sequence>
      <xs:element minOccurs="0" name="Reference_Description" type="Structured_Text_Type"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="Reference_Author" type="xs:string">
        <xs:annotation>
          <xs:documentation> This element identifies an individual author of the material being
referenced. It is not required, but may be repeated sequentially in order to identify multiple
authors for a single piece of material.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element minOccurs="0" name="Reference_Title" type="xs:string">
        <xs:annotation>
          <xs:documentation> This element identifies the title of the material beingreferenced.
It is not required if the material does not have a title.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element minOccurs="0" name="Reference_Section" type="xs:string">
        <xs:annotation>
          <xs:documentation> This element is intended to provide a means of identifying the
exact location of the material inside of the publication source, such as the relevant pages of
a research paper, the appropriate chapters from a book, etc. This is useful for both book
references and internet references.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element minOccurs="0" name="Reference_Edition" type="xs:string">
        <xs:annotation>
          <xs:documentation> This element identifies the edition of the material being
referenced in the event that multiple editions of the material exist. This will usually only be
useful for book references. </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element minOccurs="0" name="Reference_Publication" type="xs:string">
        <xs:annotation>
          <xs:documentation> This element identifies the publication source of the reference
material, if one exists. </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element minOccurs="0" name="Reference_Publisher" type="xs:string">
        <xs:annotation>
          <xs:documentation> This element identifies the publisher of the reference material,
if one exists. </xs:documentation>
        </xs:annotation>
      </xs:element>
```

```
      <xs:element minOccurs="0" name="Reference_Date" type="xs:date">
        <xs:annotation>
          <xs:documentation> This element identifies the date when the reference was included
in the entry. This provides the reader with a time line for when the material in the reference,
usually the link, was valid. The date should be of the format YYYY-MM-DD. </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element minOccurs="0" name="Reference_PubDate" type="xs:string">
        <xs:annotation>
          <xs:documentation> This field describes the date when the reference was published
YYYY. </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element minOccurs="0" name="Reference_Link" type="xs:string">
        <xs:annotation>
          <xs:documentation> This element should hold the URL for the material being
referenced, if one exists. This should always be used for web references, and may optionally be
used for book and other publication references.</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="Reference_ID">
      <xs:annotation>
        <xs:documentation> The id attribute is optional and is used as a mechanism forciting
text in the entry. If an id is provided, it is placed between brackets and precedes this
reference and the matching id should be used inside of the text for the attack pattern itself
where this reference is applicable. All reference ids assigned within an entry must be unique.
</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
  <xs:element name="Block">
    <xs:annotation>
      <xs:documentation> Block is a Structured_Text element consisting of one of Text_Title,
Text, Code_Example_Language, or Code followed by another Block element. Structured_Text
elements help define whitespace and text segments. </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:choice maxOccurs="unbounded">
        <xs:group ref="Structured_Text_Group"/>
        <xs:element ref="Block">
          <xs:annotation>
            <xs:documentation> Block is a Structured_Text element consisting of one of
Text_Title,Text, Code_Example_Language, or Code followed by another Block element.
Structured_Text elements help define whitespace and text segments.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:choice>
      <xs:attribute name="Block_Nature">
        <xs:annotation>
          <xs:documentation> This attribute identifies the nature of the content
containedwithin the Block. </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Good_Code"/>
            <xs:enumeration value="Bad_Code"/>
            <xs:enumeration value="Mitigation_Code"/>
            <xs:enumeration value="Attack"/>
            <xs:enumeration value="Result"/>
            <xs:enumeration value="List"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="Structured_Text_Type">
    <xs:sequence>
      <xs:choice maxOccurs="unbounded">
        <xs:group ref="Structured_Text_Group"/>
        <xs:element ref="Block">
          <xs:annotation>
            <xs:documentation> Block is a Structured Text element consisting of one of
```

```
  Text_Title, Text, Code_Example_Language, or Code followed by another Block element.
  Structured_Text elements help define whitespace and text segments.</xs:documentation>
           </xs:annotation>
         </xs:element>
       </xs:choice>
     </xs:sequence>
   </xs:complexType>
  </xs:schema>
```

Appendix C CEE Profile Examples

C.1 CEE Base Profile Excerpt

```
<?xml version="1.0" encoding="UTF-8"?>
<CEEProfile xmlns="http://cee.mitre.org" name="cee_base_profile"
url="http://cee.mitre.org/repository/2011/07/08/cee_base_profile.xml">
    <Base/>
    <Tags>
      <TagType name="actionTag"/>
      <TagType name="statusTag"/>
      <Tag name="access" type="actionTag">
         <Description>
            <Text_Title>Access Event</Text_Title>
            <Text>A file, user account, network share, or other object has been accessed. If
more is known regarding the access, use a more precise action such as 'read', 'write', or
'execute'.</Text>
         </Description>
      </Tag>
      <Tag name="alert" type="actionTag">
         <Description>
            <Text_Title>Alert Event</Text_Title>
         </Description>
      </Tag>
      <Tag name="allocate" type="actionTag">
         <Description>
            <Text_Title>Allocate Event</Text_Title>
         </Description>
         <Metadata>
            <inverseOf>free</inverseOf>
         </Metadata>
      </Tag>
      <Tag name="allow" type="actionTag">
         <Description>
            <Text_Title>Allowed Event</Text_Title>
         </Description>
         <Metadata>
            <inverseOf>block</inverseOf>
         </Metadata>
      </Tag>
    <Tag name="failure" type="statusTag">
      <Description>
        <Text_Title>Event Failed</Text_Title>
        <Text>The event failed due to some unmet condition, such as an incorrect
password</Text>
      </Description>
    </Tag>
    <Tag name="ongoing" type="statusTag">
      <Description>
        <Text_Title>Event Ongoing</Text_Title>
        <Text>The event has started and has yet to complete. Another event should be sent to
notify when the event completed and the final status</Text>
      </Description>
    </Tag>
    <Tag name="success" type="statusTag">
      <Description>
        <Text_Title>Event Success</Text_Title>
        <Text>The event completed successfully. For example, a successful user authentication
event would be an instance where the authentication activity was successfully completed and the
user was fully authenticated.</Text>
      </Description>
    </Tag>
    <Tag name="unknown" type="statusTag">
      <Description>
        <Text_Title>Event Status Unknown</Text_Title>
        <Text>The result state of an event occurrence was unknown. It was not known to the
observer of the event whether or not the event successfully completed.</Text>
      </Description>
    </Tag>
  </Tags>
  <Fields>
    <FieldType name="actionTagType">
```

```
        <TagRestriction>
          <TagType>actionTag</TagType>
        </TagRestriction>
      </FieldType>
      <FieldType name="boolean">
        <Description>
          <Text_Title>Boolean</Text_Title>
          <Text>A Boolean value: "true" or "false"</Text>
        </Description>
        <StringRestriction>
          <Enumeration>true</Enumeration>
          <Enumeration>false</Enumeration>
          <ABNF>boolean = "true" / "false"</ABNF>
        </StringRestriction>
      </FieldType>
      <FieldType name="cve">
        <Description>
          <Text_Title>Common Vulnerability and Exposure (CVE) Identifier</Text_Title>
          <Text>A Common Vulnerability and Exposure (CVE) identifier as listed
  [http://nvd.nist.gov] or [http://cve.mitre.org]</Text>
        </Description>
        <StringRestriction>
          <Pattern>CVE-\d{4}-\d{4}</Pattern>
        </StringRestriction>
      </FieldType>
      <FieldType name="fqdn">
        <Description>
          <Text_Title>Fully-Qualified Domain Name (FQDN)</Text_Title>
        </Description>
        <StringRestriction>
          <Pattern>([^\.]+\.)*[^\.]+</Pattern>
        </StringRestriction>
      </FieldType>
      <FieldType name="integer">
        <Description>
          <Text_Title>Integer</Text_Title>
          <Text>An unbounded integer value</Text>
        </Description>
        <StringRestriction>
          <Pattern>[+-]?[0-9]+</Pattern>
        </StringRestriction>
      </FieldType>
      <FieldType name="tag">
        <Description>
          <Text_Title>CEE Taxonomy Tag</Text_Title>
        </Description>
        <TagRestriction>
          <AnyTagType/>
        </TagRestriction>
      </FieldType>
      <FieldType name="timestamp">
        <Description>
          <Text_Title>Timestamp</Text_Title>
          <Text>A date and time according to the IETF RFC 3339
  [http://tools.ietf.org/html/rfc3339] specification, which provides an ISO 8601:2004 and
  xs:dateTime compatible definition [http://dotat.at/tmp/ISO_8601-2004_E.pdf],
  [http://www.w3.org/TR/xmlschema-2/#dateTime]</Text>
        </Description>
        <StringRestriction>
          <Pattern>\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(\.\d+)?(Z|([+-]\d{2}:\d{2}))?</Pattern>
          <ABNF>
  timestamp = DATE "T" TIME
  DATE-CENTURY   = 2DIGIT ; 00-99
  DATE-DECADE    = DIGIT  ; 0-9
  DATE-SUBDECADE = DIGIT  ; 0-9
  DATE-YEAR      = DATE-DECADE DATE-SUBDECADE
  DATE-FULLYEAR  = DATE-CENTURY DATE-YEAR
  DATE-MONTH     = 2DIGIT ; 01-12
  DATE-MDAY      = 2DIGIT ; 01-31 based on month/year
  DATE           = DATE-FULLYEAR "-" DATE-MONTH "-" DATE-MDAY
  TIME-HOUR      = 2DIGIT ; 00-23
  TIME-MINUTE    = 2DIGIT ; 00-59
  TIME-SECOND    = 2DIGIT ; 00-59, 00-60 based on leap-second rules
```

```
TIME-FRACTION  = "." 1*DIGIT
TIME-NUMOFFSET = ("+" / "-") TIME-HOUR ":" TIME-MINUTE
TIME-ZONE      = "Z" / TIME-NUMOFFSET
TIMESPEC-BASE  = TIME-HOUR ":" TIME-MINUTE ":" TIME-SECOND
TIME           = TIMESPEC-BASE [TIME-FRACTION] [TIME-ZONE]
                    </ABNF>
      </StringRestriction>
    </FieldType>
    <FieldType name="unsigned16">
      <IntegerRestriction>
        <MinValue>0</MinValue>
        <MaxValue>65535</MaxValue>
      </IntegerRestriction>
    </FieldType>
    <FieldType name="unsigned32">
      <IntegerRestriction>
        <MinValue>0</MinValue>
        <MaxValue>4294967295</MaxValue>
      </IntegerRestriction>
    </FieldType>
    <FieldType name="unsigned8">
      <IntegerRestriction>
        <MinValue>0</MinValue>
        <MaxValue>255</MaxValue>
      </IntegerRestriction>
    </FieldType>
    <Field name="acct_name" role="object" type="string" xml:id="acct_name">
      <Description>
        <Text_Title>Account Name</Text_Title>
        <Text>The name of the user account</Text>
      </Description>
      <Metadata>
        <CybOX class="Observable" field="Name" object="Account"/>
      </Metadata>
    </Field>
    <Field name="action" role="object" type="actionTagType" xml:id="action">
      <Description>
        <Text_Title>Event Action</Text_Title>
      </Description>
    </Field>
    <Field name="dst_ipv4" role="object" type="ipv4Address" xml:id="dst_ipv4">
      <Description>
        <Text_Title>Destination IPv4 Address</Text_Title>
      </Description>
    </Field>
    <Field name="dst_ipv6" role="object" type="ipv6Address" xml:id="dst_ipv6">
      <Description>
        <Text_Title>Destination IPv6 Address</Text_Title>
      </Description>
    </Field>
    <Field name="file_name" role="object" type="string" xml:id="file_name">
      <Description>
        <Text_Title>File Name</Text_Title>
      </Description>
    </Field>
    <Field name="file_path" role="object" type="string" xml:id="file_path">
      <Description>
        <Text_Title>File Path</Text_Title>
        <Text>The directory path to the file, excluding the file name</Text>
      </Description>
    </Field>
    <Field name="id" role="object" type="string" xml:id="id">
      <Description>
        <Text_Title>Event ID</Text_Title>
        <Text>A unique identifier provided by the event producer that identifies the type of
event. If the identifier is intended to be globally unique reference to a specific event
instance, use [rec_id] field instead. Examples of event identifiers are the Microsoft Windows
Event ID, the Cisco PIX ID (e.g., %PIX-2-106001), or the Sourcefire Snort snortid.</Text>
      </Description>
    </Field>
    <Field name="p_prod_cpe" role="producer" type="cpe" xml:id="p_prod_cpe">
      <Description>
        <Text_Title>Producer Product CPE Identifier</Text_Title>
```

```
        <Text>Event Producer The CPE Identifier corresponding to the product. The CPE name
should be listed in [http://nvd.nist.gov/cpe.cfm] or [http://cpe.mitre.org]</Text>
      </Description>
    </Field>
    <Field name="p_prod_id" role="producer" type="string" xml:id="p_prod_id">
      <Description>
        <Text_Title>Producer Product Identifier</Text_Title>
      </Description>
    </Field>
    <Field name="p_sys_id" role="producer" type="string" xml:id="p_sys_id">
      <Description>
        <Text_Title>Producer System Identifier</Text_Title>
      </Description>
    </Field>
    <Field name="s_proc_id" role="subject" type="string" xml:id="s_proc_id">
      <Description>
        <Text_Title>Subject Process ID</Text_Title>
      </Description>
    </Field>
    <Field name="s_proc_name" role="subject" type="string" xml:id="s_proc_name">
      <Description>
        <Text_Title>Subject Process Name</Text_Title>
      </Description>
    </Field>
    <Field name="src_ipv4" role="object" type="ipv4Address" xml:id="src_ipv4">
      <Description>
        <Text_Title>Source IPv4 Address</Text_Title>
      </Description>
    </Field>
    <Field name="src_ipv6" role="object" type="ipv6Address" xml:id="src_ipv6">
      <Description>
        <Text_Title>Source IPv6 Address</Text_Title>
      </Description>
    </Field>
    <Field name="status" role="object" type="statusTagType" xml:id="status">
      <Description>
        <Text_Title>Event Status</Text_Title>
      </Description>
    </Field>
    <Field name="tags" role="object" type="tag" xml:id="tags">
      <Description>
        <Text_Title>Event Tags</Text_Title>
        <Text>Tags describing the event type, such as the action, status, and objects involved
in the event. The tags should be chosen from the CEE Taxonomy</Text>
      </Description>
    </Field>
    <Field name="time" role="object" type="timestamp" xml:id="time">
      <Description>
        <Text_Title>Event Start Time</Text_Title>
        <Text>An ISO8601 compliant timestamp designating the date, time, and timezone offset
when the event began</Text>
      </Description>
    </Field>
  </Fields>
  <EventProfile id="cee_base_event" xml:id="cee_base_event">
    <Description>
      <Text_Title>CEE Base Event Profile</Text_Title>
      <Text>The base event structure for CEE Events. All CEE formatted events are expected to
minimally conform to this event profile. The CEE Base Event is derived from the Syslog RFC5424
event structure [http://tools.ietf.org/html/rfc5424].</Text>
      <Reference>
        <Reference_Description>
          <Block>
            <Text_Title>FAU_GEN.1</Text_Title>
            <Text>Security audit data generation</Text>
          </Block>
          <Block>
            <Text_Title>FAU_GEN.1.2</Text_Title>
            <Text>The TSF shall record within each audit record at least the following
information: a) Date and time of the event, type of event, subject identity (if applicable),
and the outcome (success or failure) of the event; and b) For each audit event type, based on
the auditable event definitions of the functional components included in the PP/ST,
[assignment: other audit relevant information].</Text>
```

```
            </Block>
          </Reference_Description>
          <Reference_Section>FAU_GEN.1.2</Reference_Section>
          <Reference_Publication>Common Criteria</Reference_Publication>
        </Reference>
      </Description>
      <EventText><FieldValue ref="time"/><Text> </Text><FieldValue ref="p_sys_id"/><Text>
  </Text><FieldValue ref="p_prod_id"/><Text> </Text><FieldValue ref="id"/><Text>
  [</Text><FieldValue ref="action"/><Text> </Text><FieldValue
  ref="status"/><Text>]</Text></EventText>
      <Field ref="time" required="true"/>
      <Field ref="id" required="true"/>
      <Field ref="p_sys_id" required="true"/>
      <Field ref="p_prod_id" required="true"/>
      <Field minCount="1" ref="action" required="true"/>
      <Field minCount="1" ref="status" required="true"/>
      <Field ref="rec_id" required="false"/>
      <Field ref="crit" required="false"/>
      <Field ref="end_time" required="false"/>
      <Field ref="dur" required="false"/>
      <Field ref="tags" required="false"/>
    </EventProfile>
  </CEEProfile>
```

C.2 CEE Function Profile

```
<?xml version="1.0" encoding="UTF-8"?>
<CEEProfile xmlns="http://cee.mitre.org" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
name="cee_firewall_profile" url="cee_firewall_profile.xml">
  <Description>
      <Text_Title xml:lang="en">CEE Function Profile: Firewall</Text_Title>
  </Description>
  <Function name="Firewall"/>
  <Include href="cee_base_profile.xml"/>
  <EventProfile id="ipv4_connection_block" extends="cee_base_event">
      <Field ref="action"><Tag ref="block"/></Field>
      <Field ref="src_ipv4" required="true"/>
      <Field ref="dst_ipv4" required="true"/>
      <Field ref="src_port" required="true"/>
      <Field ref="dst_port" required="true"/>
      <Field ref="rule_id" required="false"/>
      <Field ref="rule_val" required="false"/>
  </EventProfile>
  <EventProfile id="ipv4_connection_allow" extends="cee_base_event">
      <Field ref="action"><Tag ref="allow"/></Field>
      <Field ref="src_ipv4" required="true"/>
      <Field ref="dst_ipv4" required="true"/>
      <Field ref="src_port" required="true"/>
      <Field ref="dst_port" required="true"/>
      <Field ref="rule_id" required="false"/>
      <Field ref="rule_val" required="false"/>
  </EventProfile>
</CEEProfile>
```

C.3 CEE Product Profile

```
<?xml version="1.0" encoding="UTF-8"?>
<CEEProfile xmlns="http://cee.mitre.org" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
name="example_product_profile" url="example_product_profile.xml">
  <Description>
    <Text_Title xml:lang="en">Example CEE Product Profile</Text_Title>
  </Description>
  <Product cpe="cpe:2.3:a:Vendor:Product:Version:*:*:*:*:*:*:*"/>
  <Include href="cee_base_profile.xml"/>
  <Fields>
    <Field name="sess_dur" type="duration">
      <Description>
        <Text_Title>Session Duration</Text_Title>
      </Description>
    </Field>
  </Fields>
  <EventProfile id="acct_login" extends="cee_base_event">
    <Field ref="action"><Tag ref="login"/></Field>
    <Field ref="acct_id" required="false"/>
    <Field ref="acct_name" required="false"/>
    <Field ref="sess_id" required="false"/>
  </EventProfile>
  <EventProfile id="acct_remote_login" extends="acct_login">
    <Field ref="src_ipv6" required="true"/>
    <Field ref="src_port" required="false"/>
    <Field ref="dst_port" required="true"/>
  </EventProfile>
  <EventProfile id="acct_logout" extends="cee_base_event">
    <Field ref="action"><Tag ref="logout"/></Field>
    <Field ref="acct_id" required="true"/>
    <Field ref="sess_id" required="true"/>
    <Field ref="sess_dur" required="true"/>
  </EventProfile>
</CEEProfile>
```

Appendix D Changelog

D.1 0.6

- Expanded the specification to describe CEE Profile documents; added the event profile elements
- Changed the CDET Dictionary and CDET Taxonomy to be part of the CEE CDET component
- Updated several references and names for cross-CEE consistency
- Added the CEE/CLS core fields as reserved fields
- Renamed the definition Header and children elements to Description
- Added a FloatRestriction and TagRestriction
- Changed the naming conventions to be [0-9a-zA-Z_]{1,32} as some of the other allowed characters were problematic in the encodings

D.2 0.5.2

- Added a Description and Metadata section to all tags, tag types, fields, and field types
- Removed the Tag Relations element. Such relations should be held within the Metadata
- Moved the Text into the Description section
- Added Text_Title, Reference, and Examples into the Description
- Added 6 new core field types to bring the total core types to 11
- Added the FULL STOP ('.' `U+002E`) to the list of valid name characters

D.3 0.5.1

- Remove the concept of a Set, or field of fields. The CEE Board felt that this concept was too complex and had no viable use cases
- Renamed FieldType to FieldType
- Renamed TagSet to TagType
- Removed the concept of a FieldSet – a collection of related fields
- Simplified the XML Schema structure for both the Taxonomy and Dictionary to be less hierarchical and better usability
- Combined the concepts of Tag and Field Names and ShortNames into a single name; we want one identifier per property not multiples
- Restricted the name of Tags and Fields to be no more than 32 characters from the set of [A-Za-z0-9_-] to maximize compatibility with existing log protocols and standards. This also removed the previous Appendix B for the formal definitions and restrictions on names
- Removed the concept of alternative Tag names (AltName) – the CEE community should standardize on a single name. Vendors and third parties may use the Tag "equivalentTo"

relation metadata to define new Tag names that reflect an identical concept of an existing Tag

- Removed the "related" Tag relation as it was too generic; changed the Tag relation types to be "inverseOf", "equivalentTo", and "subclassOf"
- Added a new FieldType restriction to support grammars in Augmented Backus-Naur Form (ABNF); this should allow use to better define representations for concepts that do not condense well into a single regular expression pattern
- Added the concept of names versus qualified names, as CEE is intended to be declared across multiple Dictionary and Taxonomy documents. To enable referencing across document boundaries, the CDET must allow an identifier to be associated with a document and allow for tags and fields within that document to be referenced using a combination of the document ID and the tag/field name
- The CEE Specification now defines five core FieldTypes: *string*, *binary*, *boolean*, *integer*, and *tag*
- Added a minimum and maximum cardinality (minCard, maxCard) attributes to Field definitions. While field instances in an event record will have exactly one value, it is possible for fields and field definitions to support more. The minimum cardinality is 1, the maximum is 255
- Divided the FieldType restrictions across three Restriction groups: StringRestriction, BinaryRestriction, and IntegerRestriction. This is due to the fact that length requirements are not shared between binary and strings (octet vs. Unicode characters) and value range requirements only make sense for numbers