NIST Interagency Report 7697

National Institute of
Standards and Technology
U.S. Department of Commerce

Common Platform Enumeration: Dictionary Specification Version 2.3

Paul Cichonski
David Waltermire
Karen Scarfone

NIST Interagency Report 7697

Common Platform Enumeration: Dictionary Specification Version 2.3

Paul Cichonski
David Waltermire
Karen Scarfone

C O M P U T E R S E C U R I T Y

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

August 2011

U.S. Department of Commerce

Rebecca M. Blank, Acting Secretary

National Institute of Standards and Technology

Patrick D. Gallagher, Under Secretary for Standards and Technology and Director

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analysis to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Interagency Report discusses ITL's research, guidance, and outreach efforts in computer security and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Interagency Report 7697
38 pages (Aug. 2011)

Acknowledgments

The authors, Paul Cichonski and David Waltermire of the National Institute of Standards and Technology (NIST), and Karen Scarfone of Scarfone Cybersecurity wish to thank their colleagues who reviewed drafts of this document and contributed to its technical content. The authors would like to acknowledge Harold Booth of NIST, Brant A. Cheikes and Mary Parmelee of the MITRE Corporation, Adam Halbardier and Christopher McCormick of Booz Allen Hamilton, Seth Hanford of Cisco Systems, Inc., Tim Keanini of nCircle, Kent Landfield of McAfee, Inc., Jim Ronayne of Varen Technologies, Shane Shaffer of G2, Inc., and Joseph L. Wolfkiel of the US Department of Defense for their insights and support throughout the development of the document.

Abstract

This report defines the Common Platform Enumeration (CPE) Dictionary version 2.3 specification. The CPE Dictionary Specification is a part of a stack of CPE specifications that support a variety of use cases relating to information technology (IT) product description and naming. An individual CPE dictionary is a repository of IT product names, with each name in the repository identifying a unique class of IT product in the world. This specification defines the semantics of the CPE Dictionary data model and the rules associated with CPE dictionary creation and management. This report also defines and explains the requirements that IT products and services, including CPE dictionaries, must meet for conformance with the CPE Dictionary version 2.3 specification.

Trademark Information

Table of Contents

List of Figures

List of Tables

1. Introduction

Common Platform Enumeration (CPE) is a standardized method of describing and identifying classes of applications, operating systems, and hardware devices present in an enterprise's computing assets. CPE can be used as a source of information for enforcing and verifying information technology (IT) management policies relating to these assets, such as vulnerability, configuration, and remediation policies. IT management tools can collect information about installed products, identify products using their CPE names, and use this standardized information to help make fully or partially automated decisions regarding the assets.

CPE consists of several modular specifications. Combinations of the specifications work together in layers to perform various functions. This specification, CPE Dictionary, defines a standardized method for creating and managing CPE dictionaries. A *dictionary* is a repository of CPE names and metadata associated with the names. Each CPE name in the dictionary identifies a single class of IT product in the world. The word "class" here signifies that the object identified is not a physical instantiation of a product on a system, but rather the abstract model of that product. Although organizations may use a CPE name to represent either a single product class or a set of multiple product classes, a CPE dictionary stores only bound forms of well-formed CPE names (WFNs) that identify a single product class, not a set of product classes. These single product-class WFNs in bound form are referred to as *identifier names*. An example of a WFN and its bound forms is shown below.

WFN:

```
wfn:[part="o",vendor="microsoft",product="windows_vista",version="6\.0",
update="sp1",edition=NA,language=NA,sw_edition="home_premium",
target_sw=NA,target_hw="x64",other=NA]
```

WFN bound to a URI:

```
cpe:/o:microsoft:windows_vista:6.0:sp1:~-~home_premium~-~x64~-
```

WFN bound to a formatted string:

```
cpe:2.3:o:microsoft:windows_vista:6.0:sp1:-:-:home_premium:-:x64:-
```

NIST hosts the Official CPE Dictionary[1], which is the authoritative repository of identifier names. The goal of the CPE stack of specifications is to provide entities within the IT industry a standardized way to describe and identify IT products, and the Official CPE Dictionary provides the mechanism to support this interoperability for product identifiers. The authoritative nature of the Official CPE Dictionary allows organizations to search for and find identifier names in one centralized place without worrying about dealing with conflicts between federated dictionaries.

Organizations may create their own extended CPE dictionaries, which are used to store identifier names not present in the Official CPE Dictionary. There are several reasons why extended dictionaries are needed. For example, an organization may have to create identifier names for proprietary products that are only useful within that organization, such as internally developed software not found outside the organization. Another possible reason is that an IT company may want to use identifiers for their unreleased products that do not yet have official identifier names; once the identifier names have stabilized, the company would submit them to the Official CPE Dictionary. Finally, an organization may discover new products in use that do not yet have official identifiers; the organization could create identifiers, add them to its own extended dictionary, submit them for inclusion in the Official CPE Dictionary, and use them internally while waiting for their addition to the Official CPE Dictionary.

[1] The Official CPE Dictionary is available at http://nvd.nist.gov/cpe.cfm.

1.1 Purpose and Scope

This document defines the specification for CPE Dictionary version 2.3. This includes formally defining the concept of a CPE dictionary, the rules and policies relating to dictionary instantiation and management, and the data model that represents all dictionary concepts and relationships. This document also establishes the concept of an Official CPE Dictionary, as well as the process for how organizations can extend the Official CPE Dictionary using extended CPE dictionaries.

This report only applies to version 2.3 of CPE Dictionary. All other versions are out of the scope of this report, as are all CPE specifications other than CPE Dictionary. Operational guidelines, such as suggested processes for submitting new entries to a dictionary, are also out of the scope of this report.

1.2 Audience

This report is intended for two main audiences: the creators and maintainers of CPE dictionaries, and IT management tool developers. Readers of this report should already be familiar with CPE naming and name matching concepts and conventions, as specified in [CPE23-N] and [CPE23-M].

1.3 Document Structure

The remainder of this report is organized into the following major sections and appendices:

- Section 2 defines selected terms and abbreviations used within this specification.
- Section 3 provides an overview of related specifications or standards.
- Section 4 defines the conformance rules for this specification.
- Section 5 defines the CPE dictionary data model.
- Section 6 provides information and requirements related to CPE dictionary creation and maintenance.
- Section 7 defines operations for using a CPE dictionary, such as looking up a particular identifier name or searching for dictionary entries that belong to a particular set of products.
- Section 8 provides pseudocode that implements concepts defined in other sections of the specification.
- Appendix A lists normative and informative references.
- Appendix B provides a change log that documents significant changes to major drafts of the specification.

1.4 Document Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in Request for Comment (RFC) 2119.[2]

Text intended to represent computing system input, output, or algorithmic processing is presented in `fixed-width Courier font`.

Normative references are listed in Appendix A. These references use the citation convention of a square bracket notation containing an abbreviation of the overall reference citation, followed by a colon and

[2] RFC 2119, “Key words for use in RFCs to Indicate Requirement Levels”, is available at http://www.ietf.org/rfc/rfc2119.txt.

subsection citation where applicable (e.g., [CPE23-N:7.2] is a citation for the CPE 2.3 Naming specification, Section 7.2).

This specification adheres to all rules and conventions defined lower in the CPE stack of specifications. The CPE Naming Specification defines the concept of a Well-Formed CPE Name (WFN) that is a logical representation of a CPE name [CPE23-N:5.1]. Wherever possible, this specification uses WFN representation of CPE names to limit the dependency on any particular CPE name binding. However, readers should keep in mind that CPE dictionaries store bound CPE names, not WFNs.

This document uses an abstract pseudocode programming language to specify expected computational behavior. Pseudocode is intended to be straightforwardly readable and translatable into actual programming language statements. Note, however, that pseudocode specifications are not necessarily intended to illustrate efficient or optimized programming code; rather, their purpose is to clearly define the desired behavior, leaving it to implementers to choose the best language-specific design which respects that behavior. In some cases, particularly where standardized implementations exist for a given pseudocode function, we describe the function's behavior in prose.

When reading pseudocode the following should be kept in mind:

- All pseudocode functions are *pass by value*, meaning that any changes applied to the supplied arguments within the scope of the function do not affect the values of the variables in the caller's scope.
- In a few cases, the pseudocode functions reference (more or less) standard library functions, particularly to support string handling. In most cases semantically equivalent functions can be found in the GNU C library, cf. http://www.gnu.org/software/libc/manual/html_node/index.html#toc_String-and-Array-Utilities.

This document uses qualified names to refer to specific XML elements. A qualified name associates a named element with a namespace. The namespace identifies the XML model, and the XML schema is a definition and implementation of that model. A qualified name declares this schema to element association using the format '*prefix*:*element-name*'. The association of prefix to namespace is defined in the metadata of an XML document and varies from document to document. This document uses the conventional mappings listed below.

Prefix	Namespace	Schema
cpe_dict	http://cpe.mitre.org/dictionary/2.0	CPE Dictionary 2.3 schema
cpe_dict_ext	http://scap.nist.gov/schema/cpe-extension/2.3	CPE Dictionary 2.3 schema extension
cpe-name	http://cpe.mitre.org/naming/2.0	CPE Naming 2.3 schema
xml	http://www.w3.org/XML/1998/namespace	Common XML attributes
xsd	http://www.w3.org/2001/XMLSchema	XML Schema

2. Definitions and Abbreviations

This section defines selected terms and abbreviations used within the document. This section builds on the terms defined in the CPE Naming specification [CPE23-N] and the CPE Name Matching specification [CPE23-M], and does not repeat them here.

2.1 Definitions

CPE Dictionary: A repository of identifier CPE names (WFNs in bound form) and associated metadata.

Deprecated Identifier Name: An identifier name that is no longer valid because it has either been replaced by a new identifier name or set of identifier names or was created erroneously.

Dictionary Contributor: An organization or person that submits new identifier CPE names to a dictionary for inclusion.

Dictionary Creator: An organization that instantiates a CPE dictionary that conforms to the guidance within this specification. A dictionary creator is the organization that is ultimately responsible for the dictionary.

Dictionary Maintainer: An organization that manages a CPE dictionary and all processes relating to that CPE dictionary. In the majority of cases, the organization that serves as the dictionary creator for a specific CPE dictionary will also serve as the dictionary maintainer. Otherwise, generally the dictionary maintainer is supporting the dictionary on behalf of the dictionary creator.

Dictionary Management Documents: A set of documentation that captures the rules and processes specific to a CPE dictionary.

Dictionary Search: The process of determining which identifier names within a CPE dictionary are members of a source name that represents a set of products.

Dictionary User: An organization, individual, product, or service that consumes a CPE dictionary for any purpose.

Extended CPE Dictionary: A dictionary that an organization may create to house identifier names not found in the Official CPE Dictionary.

Identifier Lookup: The process of determining if a single identifier name exists in a CPE dictionary.

Identifier CPE Name: A bound representation of a CPE WFN that uniquely identifies a single product class. Also referred to as an "identifier name".

Known Data: A category of information that may be present within an attribute of a CPE name. Known data represents any meaningful value about a product (e.g., "sp1", "2.3.4", "pro", NA), but does not include the logical value ANY.

Official CPE Dictionary: The authoritative repository of identifier names, which is hosted by NIST.

Official Identifier CPE Name: Any bound representation of a CPE WFN that uniquely identifies a single product class and is contained within the Official CPE Dictionary.

2.2 Abbreviations

CPE	Common Platform Enumeration
IR	Interagency Report
IT	Information Technology
ITL	Information Technology Laboratory
NIST	National Institute of Standards and Technology
RFC	Request for Comment
URI	Uniform Resource Identifier
WFN	Well-Formed CPE Name
XML	Extensible Markup Language

3. Relationship to Existing Specifications and Standards

This section explains the relationships between this specification and related specifications or standards.

3.1 Other CPE Version 2.3 Specifications

CPE version 2.3 was constructed using a modular, stack-based approach, with each major component defined in a separate specification. Functional capabilities are built by layering these modular specifications. This architecture opens opportunities for innovation, as novel capabilities can be defined by combining only the needed specifications, and the impacts of change can be better compartmentalized and managed.

The CPE Dictionary version 2.3 specification builds upon the CPE Naming version 2.3 [CPE23-N] and CPE Name Matching version 2.3 [CPE23-M] specifications to define the concept of a CPE dictionary.

3.2 CPE Version 2.2

The CPE version 2.3 specifications, including this specification, collectively replace [CPE22]. CPE version 2.3 is intended to provide all the capabilities made available by [CPE22] while adding new features suggested by the CPE user community.

The primary differences between CPE Dictionary versions 2.2 and 2.3 include:

- Updated deprecation logic that includes one-to-many CPE deprecation
- Updated change history and provenance data requirements
- Built-in one-to-one mapping between CPE version 2.2 and version 2.3 names; the version 2.3 name is embedded in the version 2.2 name so that the instance document will validate against the version 2.2 schema.

4. Conformance

Products and organizations may want to claim conformance with this specification for a variety of reasons. For example, a product may want to assert that it uses official identifier names properly. Organizations may want to claim conformance if they are creating and/or maintaining CPE dictionaries.

This section provides the high-level requirements that a product or organization must meet if they are seeking conformance with this specification. The majority of the requirements listed in this section reference other sections in this document that fully define the high-level requirement.

4.1 Dictionary Use Conformance

This section contains the set of requirements for IT products asserting conformance with this specification for dictionary use. Products may claim conformance with this specification to show that the product uses identifier names properly. All products claiming conformance with this specification MUST adhere to the following requirements:

1. Products using CPE names as identifiers MUST only use identifier names that meet the criteria defined in Section 6.1. Products using identifier names SHOULD only use official identifier names, which are located in the Official CPE Dictionary. If an official identifier name is not available, the product MUST use an identifier name contained within an extended CPE dictionary to which it has access.

2. Products using a CPE dictionary MUST adhere to the requirements in Section 7.1 and 7.2 for identifier lookups and dictionary searches.

3. When a product consumes or outputs an identifier name, that product MUST first determine if the identifier name is deprecated and act appropriately if it is, in compliance with the process defined in Section 7.3. This requirement MAY be ignored if the product is intending to communicate information about deprecated identifier names.

4.2 Dictionary Creation and Maintenance Conformance

Organizations creating or maintaining CPE dictionaries may claim conformance with this specification if the dictionary meets the requirements in this section. This section first provides the generic requirements for all dictionaries, then provides requirements specific to the dictionary type (i.e. Official CPE Dictionary or extended CPE dictionary).

1. The data within a CPE dictionary MUST adhere to the data model defined in Section 5; this includes all entries in the dictionary containing the mandatory elements defined in Section 5.

2. Organizations creating/maintaining a CPE dictionary MAY extend the CPE dictionary data model defined in Section 5. These organizations MUST use only the extension points defined in Section 5.7 when extending the CPE dictionary data model.

3. All entries in a CPE dictionary MUST adhere to the minimum set of identifier name acceptance criteria defined in Section 6.1.

4. A CPE dictionary MUST adhere to the identifier name deprecation process defined in Section 6.2.

5. A CPE dictionary MUST capture the identifier name provenance data defined in Section 6.3.

The following subsections provide additional requirements specific to each dictionary type.

4.2.1 Official CPE Dictionary

Additional requirements for the Official CPE Dictionary are:

1. The Official CPE Dictionary SHOULD NOT further restrict the acceptance criteria defined in Section 6.1 because doing so would require all extended CPE dictionaries to implement the same restrictions.

2. The Official CPE Dictionary MUST house its own set of management documents as specified in Section 6.4.

4.2.2 Extended CPE Dictionaries

Additional requirements for extended CPE dictionaries are:

1. Organizations MAY use extended CPE dictionaries to store identifier names for proprietary products.

2. Organizations MAY use extended CPE dictionaries to store identifier names not yet found in the Official CPE Dictionary, but these organizations SHOULD submit these new identifier names to the Official CPE Dictionary as official identifier names.

3. An extended CPE dictionary SHOULD house its own set of management documents as specified in Section 6.4. If a dictionary does not have its own documents, the dictionary creator or maintainer SHOULD reference the Official CPE Dictionary Management Documents as applicable to the extended dictionary.

4. An extended CPE dictionary MUST adhere to all acceptance criteria restrictions implemented in the Official CPE Dictionary. An extended CPE dictionary MAY further restrict the Official CPE Dictionary acceptance criteria, but MUST not conflict with them.

5. An extended CPE dictionary SHOULD NOT contain identifier names that conflict with the Official CPE Dictionary. If an extended dictionary does contain conflicting names, the extended dictionary's maintainers SHOULD try to resolve the conflict with the Official Dictionary.

6. An extended CPE dictionary MUST NOT include any names that are not unique with respect to itself or to the Official CPE Dictionary. Section 6.1.3 defines CPE name uniqueness.

5. CPE Dictionary Data Model

This section defines the data model that all CPE dictionaries MUST implement. The data model does not prescribe a specific binding or implementation; it merely describes the data that is required to support the technical use cases. Any CPE dictionary binding MUST implement all data model elements and properties defined in this section. Additionally, any CPE dictionary binding MAY extend this data model, but extensions MUST only occur at the valid extension points defined in Section 5.7.

A CPE dictionary is a collection of identifier names and metadata associated with these identifiers. To support this, the CPE Dictionary data model revolves around one core element called *cpe-item* that holds all the information relating to a single identifier name. The *cpe-item* element has evolved out of the CPE Dictionary 2.2 XML Schema to support backwards compatibility. The *cpe-item* element contains a *cpe23-item* element that captures all CPE 2.3 specific data, including provenance data and an upgraded deprecation system.

This data model makes special accommodations to ensure that existing version 2.2 CPE dictionaries are forward compatible with the CPE Dictionary 2.3 XML schemas. Also, any XML schema based bindings generated from this data model can produce instance data that validates against the CPE Dictionary 2.2 XML schema, if the *cpe23-item* element is used as the "any" element in the CPE Dictionary 2.2 XML schema and the *title* element is always specified for each *cpe-item* element. To support this, the CPE dictionary's XML specification is split into two schemas: the main schema corresponds to the CPE Dictionary 2.2 XML schema (with the exception of making the *title* element optional instead of mandatory), and the schema extension contains the CPE Dictionary 2.3 specific elements and attributes.

The tables below define the elements of the data model and their properties. Several of the elements allow organizations to supply additional information that is not covered by these properties. However, such information is not part of the official CPE Dictionary data model, and there are no requirements for dictionary users to access or process this information.

In the tables below, types prefixed with "cpe_dict:" are from the CPE Dictionary 2.3 XML schema, "cpe_dict_ext:" prefixes indicate types from the CPE Dictionary 2.3 XML schema extension, and "cpe-name" prefixes indicate types from the CPE Naming 2.3 XML schema. These schemas, which are the authoritative XML binding definitions, can be found at http://scap.nist.gov/schema/cpe/2.3/cpe-dictionary_2.3.xsd, http://scap.nist.gov/schema/cpe/2.3/cpe-dictionary-extension_2.3.xsd, and http://scap.nist.gov/schema/cpe/2.3/cpe-naming_2.3.xsd, respectively.

5.1 The <cpe_dict:cpe-list> Element

The `<cpe_dict:cpe-list>` element contains all of the dictionary entries and dictionary metadata. The table below describes the `<cpe_dict:cpe-list>` element's properties.

Property	Type	Count	Description
generator (element)	cpe_dict:GeneratorType	0-1	Information related to the generation of the document, including the dictionary schema version, the time of the dictionary's generation, and the name and version of the application used to generate it.
cpe-item (element)	cpe_dict:ItemType	1-n	A container for a single dictionary entry (identifier name) and its metadata.

Property	Type	Count	Description
	xsd:any	0-n	Additional elements that are not part of the CPE namespace. These MAY contain additional information that a user MAY choose to use or not, but this information is not required to be used or understood.

The rest of Section 5 provides additional information on the <cpe_dict:generator> and <cpe_dict:cpe-item> elements.

5.2 The <cpe_dict:generator> Element

The <cpe_dict:generator> element contains information about the generation of the dictionary file. The properties of the <cpe_dict:generator> element are listed below.

Property	Type	Count	Description
product_name (element)	xsd:string	0-1	The name of the application used to generate the file.
product_version (element)	xsd:string	0-1	The version of the application used to generate the file.
schema_version (element)	xsd:decimal	1	The version of the schema that the document was written in and that should be used for validation.
timestamp (element)	xsd:dateTime	1	When the file was generated.
	xsd:any	0-n	Additional elements that are not part of the CPE namespace. These MAY contain additional information that a user MAY choose to use or not, but this information is not required to be used or understood.

5.3 The <cpe_dict:cpe-item> Element

The <cpe_dict:cpe-item> element contains all of the information for a single dictionary entry (identifier name), including metadata. The properties of the <cpe_dict:cpe-item> element are defined below.

Property	Type	Count	Description
name (attribute)	cpe-name:cpe22Type	1	The identifier name bound in CPE 2.2 format.
deprecated (attribute)	xsd:boolean	0-1	Whether or not the name has been deprecated. Default value is false.
deprecated_by (attribute)	cpe-name:cpe22Type	0-1	The name that deprecated this name, bound in CPE 2.2 format.
deprecation_date (attribute)	xsd:dateTime	0-1	When the name was deprecated.
title (element)	cpe_dict:TextType	0-n	Human-readable title of the name. To support uses intended for multiple languages, the <cpe_dict:title> element supports the @xml:lang attribute. At most one <cpe_dict:title> element MAY appear for each language.

Property	Type	Count	Description
notes (element)	cpe_dict:NotesType	0-n	Optional descriptive material. To support uses intended for multiple languages, the `<cpe_dict:notes>` element supports the `@xml:lang` attribute. At most one `<cpe_dict:notes>` element MAY appear for each language.
references (element)	cpe_dict:ReferencesType	0-1	External references to additional descriptive material. Each reference consists of a piece of text (intended to be human-readable) and a URI (intended to be a URL pointing to a real resource).
check (element)	cpe_dict:CheckType	0-n	Calls out a check, such as an OVAL definition, that can confirm or reject an IT system as an instance of the named platform. Includes a REQUIRED check system specification URI (e.g., the URI for a particular version of OVAL), a REQUIRED check identifier, and an OPTIONAL external file reference (for example, a pointer to the file where the check is defined).
cpe23-item (element)	cpe_dict_ext:itemType	1	Element that captures all CPE 2.3 specific data including the CPE 2.3 formatted string binding of the name, provenance data, and deprecation data.
	xsd:any	0-n	Additional elements that are not part of the CPE namespace. These MAY contain additional information that a user MAY choose to use or not, but this information is not required to be used or understood.

5.4 The <cpe_dict_ext:cpe23-item> Element

The `<cpe_dict_ext:cpe23-item>` element contains all CPE 2.3 specific data related to a given identifier name.

Property	Type	Count	Description
name (attribute)	cpe_dict_ext:namePattern	1	The identifier name bound to a formatted string [CPE23-N:6.2].
provenance-record (element)	cpe_dict_ext:provenanceRecordType	0-1	Container for all provenance information for the given identifier name.
deprecation (element)	cpe_dict_ext:deprecationType	0-n	Element holding one or more deprecation entries for the given identifier name. It is possible for a single identifier name to have multiple deprecations that occur at different time periods.
	xsd:any	0-n	Additional elements that are not part of the CPE namespace. These MAY contain additional information that a user MAY choose to use or not, but this information is not required to be used or understood.

5.5 The <cpe_dict_ext:provenance-record> Element

The `<cpe_dict_ext:provenance-record>` element contains the provenance information for the given identifier name.

Property	Type	Count	Description
submitter (element)	cpe_dict_ext:organizationType	1	The organization responsible for submitting the identifier name.
authority (element)	cpe_dict_ext:organizationType	1-n	The authority or authorities responsible for endorsing the identifier name. Multiple authorities may endorse the same identifier name.
change-description (element)	cpe_dict_ext:changeDescriptionType	1-n	A description of any changes made to the identifier name or associated metadata.
	xsd:any	0-n	Additional elements that are not part of the CPE namespace. These MAY contain additional information that a user MAY choose to use or not, but this information is not required to be used or understood.

The `<cpe_dict_ext:submitter>` and `<cpe_dict_ext:authority>` elements have the same structure:

Property	Type	Count	Description
system-id (attribute)	xsd:anyURI	1	Unique URI representing the organization.
name (attribute)	xsd:token	1	Human readable name of the organization.
date (attribute)	xsd:dateTime	1	The date the organization performed an action relative to an identifier name. For example, the date the organization submitted or endorsed a particular identifier name.
description (element)	xsd:token	0-1	A high-level description of the organization acting as the submitter or authority.

The `<cpe_dict_ext:change-description>` element has the following structure:

Property	Type	Count	Description
change-type (attribute)	cpe_dict_ext:changeTypeType	1	The type of change that occurred. The possible values are listed below the table.
date (attribute)	xsd:dateTime	0-1	When the change occurred.
evidence-reference (element)	cpe_dict_ext:evidenceReferenceType	0-1	Supporting evidence for any change to a name or associated metadata, including a link to external information relating to the change.
comments (element)	xsd:token	0-1	Comments explaining the rationale for the change.

The possible values for the `@change-type` attribute are:

- ORIGINAL_RECORD: Used when the name is first added to the dictionary.
- AUTHORITY_CHANGE: Used when the authority behind the name is modified.
- DEPRECATION: Used when the name is first deprecated.
- DEPRECATION_MODIFICATION: Used when additional deprecation entries are recorded for a deprecated name.

The <cpe_dict_ext:evidence-reference> element contains a URI, which references a specific piece of evidence, and an @evidence attribute, which has one of the following possible values:

- CURATOR_UPDATE: The curator of the dictionary discovered information that led to a change.
- VENDOR_FIX: The vendor of the product identified in the name released or submitted information that led to a change.
- THIRD_PARTY_FIX: A third party released or submitted information that led to a change.

5.6 The <cpe_dict_ext:deprecation> Element

A <cpe_dict_ext:deprecation> element (from the <cpe_dict_ext:cpe23-item> element) contains the deprecation information for a specific deprecation of a given identifier name. If a name is deprecated multiple times, there would be a separate instance of the deprecation element for each deprecation.

Property	Type	Count	Description
date (attribute)	xsd:dateTime	0-1	When the deprecation occurred.
deprecated-by (element)	cpe_dict_ext:deprecatedInfoType	1-n	The list of names that deprecated the identifier name. The names in the list do not have to be identifier names; they MAY contain wildcards and represent sets of products.

The deprecated-by element has the following structure:

Property	Type	Count	Description
name (attribute)	cpe-name:cpe23Type	0-1	The name that is deprecating the identifier name.
type (attribute)	cpe_dict_ext:deprecationTypeType	1	The type of deprecation.

The possible values for the type attribute are:

- NAME_CORRECTION: Deprecation is of type Identifier Name Correction
- NAME_REMOVAL: Deprecation is of type Identifier Name Removal
- ADDITIONAL_INFORMATION: Deprecation is of type Additional Information Discovery

5.7 Extension Points

Organizations may need to capture data not defined in the CPE Dictionary data model. Any organization serving as a CPE dictionary creator or maintainer MAY extend the <cpe_dict_ext:cpe23-item>, <cpe_dict_ext:cpe-list>, and <cpe_dict_ext:provenance-record> elements to capture additional, organization specific data. If organizations extend these elements, this extension MUST only occur by adding additional properties to these elements to capture different types of data, or by restricting the values for specific properties. Organizations MUST NOT define rules conflicting with the properties already defined for these elements.

6. Dictionary Creation and Maintenance

This section provides information and requirements related to creating and maintaining CPE dictionaries. It defines requirements for valid identifier names within dictionaries and explains the process to be followed when an identifier name needs to be replaced or removed from a dictionary. This section also briefly discusses requirements for capturing identifier name provenance data. Finally, this section explains CPE dictionary management documents and provides related requirements.

6.1 Acceptance Criteria

A CPE dictionary holds a collection of identifier names that serve to identify classes of products. A dictionary maintainer MUST only permit valid identifier names in the dictionary.[3] An identifier name is valid if it meets the acceptance criteria defined in the following sub-sections, which ensure a certain degree of interoperability across all CPE dictionaries. The criteria focus on ensuring that each identifier name within a CPE dictionary identifies a single product class. Dictionary maintainers MAY further restrict the acceptance criteria defined in this section, but MUST NOT relax the criteria. The maintainers of the Official CPE Dictionary SHOULD NOT further restrict the criteria since that would require all extended dictionaries to implement the same restrictions.

6.1.1 Logical Values and Special Characters

The CPE Naming specification reserves a set of logical values and special characters for use within a CPE attribute value [CPE23-N:5.3]. The CPE Dictionary specification permits a subset of these to be contained within identifier names.

6.1.1.1 NA Logical Value

The NA value within a CPE attribute signifies that the attribute is Not Applicable. For example, a vendor may distribute the first release of a product that contains no update, but later releases an update to this first release in the form of a service pack. This example contains two distinct products, the first product that contains no update, and the second product that does contain an update. The identifier name created to represent the first release of the product should contain an update attribute with a value of NA (i.e., update=NA) since the update attribute is known to be not applicable to the product. The identifier name created to represent the second release of the product should contain an update attribute with a value of sp1 (i.e., update="sp1").

Identifier names MUST NOT use the NA logical value as the part, vendor, or product attribute value, but MAY use the NA logical value for any other CPE attributes' values to indicate that the attributes are not applicable.

6.1.1.2 ANY Logical Value

The use of the ANY logical value for a CPE attribute value signifies that there was not enough information to populate the specific CPE attribute at the time of name creation. Identifier names MUST NOT use the ANY logical value as the part, vendor, product, or version attribute value, but MAY use the ANY logical value for any other CPE attributes' values to indicate a lack of information.

[3] Pseudocode that implements this rule is found in Section 8.2, in the function `accept-name`.

It is common for the amount of information known about a product to grow over time. To support this, it is necessary to evolve the identifier name itself by creating a new, more specific identifier name and deprecating the old identifier name. Section 6.2 defines the formal deprecation process for tracking changes to identifier names within a dictionary.

6.1.1.3 Special Characters

The CPE Naming specification reserves a set of special characters for use within a CPE attribute value [CPE23-N:5.3.2]. The CPE Name Matching specification assigns specific meaning to a subset of these special characters: the asterisk, which is a multi-character wildcard, and the question mark, which is a single-character wildcard [CPE23-M:5]. This wildcard concept works well when creating expressions to represent sets of product classes, but not for attempting to identify a single product class. Therefore, identifier names MUST NOT contain any asterisk or question mark characters within attribute values unless those characters are quoted. However, CPE dictionaries MAY contain these values within names not serving as identifiers (e.g., within deprecation logic). CPE names containing these values are not valid identifier names.[4]

6.1.2 CPE Name Completeness

Each identifier name MUST contain known data (e.g., not the ANY logical value) for the part, vendor, product, and version attributes' values. This ensures that each identifier name contains the minimum amount of data required to identify a unique product class. The ANY logical value does not increase the completeness of an identifier name; only known data can increase the identifier name completeness. Known data refers to any value that represents a distinct aspect of a product (e.g. "sp1", "2.0", "Microsoft", "ios"). For attributes other than the part, vendor, and product attributes, known data includes the NA logical value since it represents the known fact that no data exists for a specific CPE attribute.

6.1.3 CPE Name Uniqueness

Every identifier name MUST be unique in the sense that it MUST NOT encompass a more complete identifier name within the dictionary. In terms of the set relations and algorithms in the CPE Name Matching specification [CPE23-M:6], this means that the identifier name SHALL NOT be a SUPERSET of any identifier name within the dictionary unless the two names are the same (EQUAL).[5] Before adding a new identifier name to a dictionary, the dictionary maintainer MUST use CPE name matching to determine CPE uniqueness. If the dictionary is an extended dictionary, both the extended dictionary and the Official CPE Dictionary MUST be checked.

The two scenarios that MUST be checked for are:

1. New name too general (the new name is a SUPERSET of the old name, but the new name is not EQUAL to the old name). The creator of a new name may not have had the full set of information required to populate all of its attributes with known data. CPE dictionaries support this situation, but they MUST NOT include such a name if a more complete version of that name exists in the dictionary.

2. Existing name too general (the new name is a SUBSET of the old name, but the new name is not EQUAL to the old name). The new name may include information that was not available when

[4] Pseudocode that implements this rule is found in Section 8.2, in the function `contains-restricted-characters`.

[5] In CPE Name Matching, the SUPERSET is non-proper, meaning that it includes cases where the two names are the same.

previous, more general names were created. For example, there may already be a similar identifier name in the dictionary that contains the ANY logical value for fields that have known data in the new name.[6] In such a case the pre-existing name(s) SHALL be deprecated to the new, more specific name; the deprecation process defined in Section 6.2 SHALL be followed for performing this deprecation.

To provide an example (focusing on the CPE update attribute), suppose that a CPE dictionary contains the following identifier name version of a CPE:

```
wfn:[part="a",vendor="foo_company",product="bar",version="2\.3",
update="sp1",edition=ANY,language=ANY,sw_edition=ANY,target_sw=ANY,
target_hw=ANY,other=ANY]
```

The above CPE represents the hypothetical product Foo Company Bar 2.3 sp1. At some later point in time, a dictionary contributor submits the following identifier name version of a CPE for potential inclusion within the same dictionary:

```
wfn:[part="a",vendor="foo_company",product="bar",version="2\.3",
update=ANY,edition=ANY,language=ANY,sw_edition=ANY,target_sw=ANY,
target_hw=ANY,other=ANY]
```

This new name represents the hypothetical product: Foo Company Bar 2.3. The difference with this new name is that "`ANY`" is the value for the update attribute. This means that the contributor of this name does not know the correct value of the update field at the time of submission. The acceptance criteria defined in this document does not permit this submission because an identifier name already exists that contains a known value for the update field of the product. The reason for this restriction is that the less-complete name does not represent any real-world product, but instead represents a set of existing identifier name within the dictionary. If the dictionary contributor discovers a new update of the product, then the contributor should submit an identifier name containing the new update value.

This does not mean that an identifier name representing the product Foo Company Bar 2.3 with no update is not permitted (e.g., the first release of the product contained no update). The identifier name representing the product with no update would contain the NA logical value in the update attribute. The following CPE does meet the acceptance criteria since it represents a real-world product that is known to have no update (i.e., the initial release of Bar 2.3):

```
wfn:[part="a",vendor="foo_company",product="bar",version="2\.3",
update=NA,edition=ANY,language=ANY,sw_edition=ANY,target_sw=ANY,
target_hw=ANY,other=ANY]
```

6.2 CPE Dictionary Deprecation Process

All identifier names stored within a dictionary MUST be immutable because data relating to a product is captured in the identifier itself. An identifier name MUST NOT be modified because the loss of historical information will cause problems for anyone already using that identifier name. Any updates to the product data captured in the identifier name MUST occur through deprecation. For example, when a dictionary maintainer discovers more information about a product represented by an existing identifier in the

[6] Pseudocode that implements this rule is found in Section 8.2, in the function `matches-more-complete-in-dictionary`.

dictionary, they MUST deprecate the legacy identifier name in favor of a new identifier name or new set of identifier names. A deprecated identifier name is no longer considered valid.

An important distinction exists between the set of names that deprecate an identifier name and the identifier name that a CPE dictionary user selects as a replacement of the deprecated name. The CPE dictionary may link a deprecated identifier name to a set of identifier names replacing it, but this is asserting that any identifier name within the set of new identifier names is a valid replacement for the deprecated identifier name. Using this information a CPE dictionary user may, depending on the use case, decide on the appropriate name to use as a replacement, or decide to use the entire set. For example, when resolving the deprecating entries for a deprecated name found within an applicability statement, an organization may find it useful to use the entire set to avoid false negatives.

There are three types of identifier name deprecation:

- **Identifier Name Correction** – An error occurred during name creation. The error could range from a simple syntax error (e.g., typo, misspelling) to an incorrect product listing. For example, a dictionary maintainer may add a product to the dictionary, only to later discover that the product is only a component library of a larger product. In this case, the dictionary maintainer would deprecate the identifier name representing the component library to the identifier name representing the actual product containing the component. This type of deprecation is one-to-one, with the deprecated name pointing to the single identifier name that represents the correct product.
- **Identifier Name Removal** – An identifier name exists in the dictionary that does not belong and has no replacement. This condition normally results from a case where the dictionary maintainer has added a name to the dictionary that should never have been included in the dictionary. In this case, the dictionary maintainer will deprecate the legacy name without pointing to a new name.
- **Additional Information Discovery** – The dictionary maintainer adds one or more identifier names to the dictionary that are more complete than an existing identifier name. In this case, the pre-existing name really represents a set of possible products that the maintainer did not know about when originally adding the name to the dictionary. This type of deprecation may be one-to-many since the dictionary maintainer will deprecate the pre-existing name to all of the names that are more complete. This deprecation relationship is largely informational; when making this deprecation, the dictionary maintainer is asserting that any identifier name within the set of new identifier names is a valid replacement for the deprecated name. It is up to the user to decide on which individual name to use, or to use the entire set.

When deprecating an identifier name, the dictionary maintainer MUST link the deprecated identifier name to all identifier names that are replacing it through the deprecated-by relationship as defined in Section 5.6. For example, if x is a single identifier name that is being deprecated and y is a set of identifier names that is deprecating x, then "x deprecated-by y" defines a deprecation relationship between x and y.

When a dictionary maintainer decides to deprecate an identifier name, the dictionary maintainer MUST follow the deprecation process below. This process refers to the identifier name being deprecated as the *legacy name*.

1. If the dictionary maintainer is replacing the legacy name with one or more new identifier names, then the dictionary maintainer MUST add the new identifier names to the dictionary.

2. The dictionary maintainer MUST add a deprecation element to the legacy name dictionary entry to signify that it is deprecated. If the legacy name was previously deprecated, then the dictionary maintainer MUST add a new deprecation element to record the new deprecation.[7]

3. The dictionary maintainer MUST expand the new deprecation element to include one or more deprecated-by elements to record the identifier name(s) replacing the legacy name. The dictionary maintainer MUST specify the type of deprecation within the type attribute of each deprecated-by element. Additionally:

    a. If the dictionary maintainer is performing an Identifier Name Correction deprecation, then the dictionary maintainer MUST populate the name attribute of the deprecated-by element with the identifier name replacing the legacy name.

    b. If the dictionary maintainer is performing an Additional Information Discovery deprecation, then the dictionary maintainer MUST populate the name attribute of the deprecated-by element with a name representing the set of new identifier names that are replacing the legacy name. This name may contain wildcards (e.g., *, ?) to represent a set of identifier names.

4. The dictionary maintainer MUST record the change in the legacy name's provenance information.

6.3 CPE Dictionary Provenance Data

CPE dictionary maintainers MUST capture the required provenance data specified in Section 5.5. The provenance data required includes data useful in understanding the reasoning behind changes made to identifier names stored within a CPE Dictionary. This provenance data also captures the organization responsible for identifier name submissions, as well as the authority behind the submissions.

6.4 CPE Dictionary Management Documents

Every CPE dictionary has a set of supporting management documentation associated with it. This documentation improves the transparency relating to dictionary acceptance criteria, dictionary content creation decisions and processes associated with the dictionary. Each CPE dictionary creator or maintainer MUST create or reference a series of accompanying management documents capturing rules and processes specific to its dictionary. A single dictionary creator or maintainer MAY establish multiple extended CPE dictionaries. In these situations a single set of external CPE dictionary management documents MAY be referenced as the authoritative documents for each extended dictionary.

This section defines each required CPE dictionary management document.

6.4.1 Dictionary Content Management and Decisions Document

Dictionary maintainers MUST either create or reference a Dictionary Content Management and Decisions Document for each dictionary that they maintain. The document defines the procedures for identifier name creation within a particular dictionary or set of dictionaries. All procedures defined in the decisions document MUST NOT override or conflict with the requirements defined in this specification. The decisions document MUST capture the following information:

[7] Multiple deprecations may occur against a single identifier WFN at different times (e.g., it is discovered that more products are included in the deprecated-by set). To support this, the CPE Dictionary data model requires that each deprecation element within an identifier WFN record the deprecation for one instant in time.

1. Rules relating to dictionary-specific acceptance criteria for identifier names.

2. Identification strategies relating to different product types. Due to the heterogeneous nature of product versioning in the IT industry, multiple disparate strategies for versioning products exist. Where possible, the document should record the different identification strategies captured within the dictionary. For example, if a specific product line uses seven digits within its version syntax, then the document should record the semantics of each digit within this syntax.

3. Automated identifier name creation strategies for specific products. These strategies may include the API calls or functions to call on certain products to populate specific attributes of an identifier name.

4. Lists of valid values for specific CPE attributes where appropriate. Valid value lists either may be global or pertain to specific CPEs. For example, it is possible to have a granular valid value list for the version attribute of a specific vendor and product, without extending the scope of this valid values list to every CPE version attribute within the dictionary.

5. Abbreviation rules for data within specific CPE attributes where appropriate. Abbreviation rules may either be global or pertain to specific CPEs.

6. Rules relating to any dictionary-specific provenance data that the dictionary records.

6.4.2 Dictionary Process Management Document

Dictionary maintainers MUST either create or reference a Dictionary Process Management Document for each dictionary that they maintain. The document defines all processes associated with a particular dictionary or set of dictionaries. All processes defined in the document MUST NOT override or conflict with the requirements defined in this specification. The process management document MUST capture the following information:

1. The scope of the dictionary.

2. The target audience of the dictionary.

3. The submission process that users must follow to submit new CPE identifiers for inclusion within the dictionary. At a minimum, this overview SHOULD include the submission format, the process for starting the submission process, and the workflow surrounding the submission process.

4. The content decisions process that the community follows to create the content decision rules that are captured in the Dictionary Content Management and Decisions Document.

5. The CPE Identifier modification process followed by dictionary maintainers.

6. The dictionary distribution process and methodology. At a minimum, this SHOULD define the binding in which the dictionary is distributed.

7. Dictionary Use

This section defines two common ways of using a CPE Dictionary. The first involves looking up a single identifier name within a dictionary. The second involves using a name representing a set of products to search against the dictionary to determine which identifier names within the dictionary are members of that set. The two methods of searching are necessary to support the type of information a user is retrieving from a dictionary. In some cases, it is necessary to retrieve an exact match from the dictionary; for example, when resolving a deprecation chain based on "Identifier Name Correction" deprecation, it is necessary to resolve only the exact identifier name listed in the deprecation metadata. In other cases, it is necessary to search for an entire set of CPE names; for example, when resolving a deprecation chain based on "Additional Information Discovery" deprecation, it is necessary to resolve all identifier names captured in the deprecation metadata.

The section also explains how identifier names should be resolved from a deprecated name. All of these operations leverage the CPE name matching algorithms[8] when performing the search operations.

7.1 Identifier Lookup

Identifier lookup involves using the WFN corresponding to a single identifier name (i.e., the source identifier name) to iterate through each entry in a dictionary to see if that source identifier exists within the dictionary—if any identifier name in the dictionary is equal to the source identifier name. Equality means that the corresponding attribute values in the two identifier names are all equal. The `CPE_EQUAL` function defined in the CPE Name Matching specification [CPE23-M:7.2] presents a formal implementation of this equality test. The two possible outcomes of an identifier lookup are:

- **Match** – The source identifier name matches an identifier name within the dictionary. The operation returns the status "EXACT-MATCH" and the WFN of the dictionary identifier name that matched. Note that since each name within the dictionary MUST be unique, at most only one identifier name could match the source identifier name.
- **No Match** – The source identifier name does not match any identifier name within the dictionary. The operation returns a null value.

Section 8.3 defines the formal implementation of this identifier lookup operation in pseudocode.

7.2 Dictionary Searching

Dictionary searching involves using a source name representing a set of products to search against a dictionary to determine which identifier names within the dictionary are members of that set. A dictionary searching routine would iterate through each entry in the dictionary to identify all that are a member of the set represented by the source name. Possible outcomes of this searching operation are:

- **Superset Match** – One or more identifier names from the dictionary belong to the set represented by the source name. In set theory language, the set represented by the source name is a superset of the matched identifier names from the dictionary. The operation returns "SUPERSET-MATCH" and the set of matching dictionary identifier names expressed as WFNs.
- **Subset Match** – The set represented by the source name is a possible subset of one or more identifier names within the dictionary. This will only occur if the source name is more specific

[8] The CPE Name Matching specification [CPE23-M] formally defines the CPE name matching algorithms used in this section, and in accompanying pseudocode implementations in Section 8.3.

than one or more names within the dictionary. Generally this result should be treated as an error; the searcher may want to notify the dictionary maintainer that a name in the wild is more specific than one or more identifier names within the dictionary. The operation returns "SUBSET-MATCH" and the set of matching dictionary identifier names expressed as WFNs.

- **No Match** – There is no relationship between the set represented by the source name and the identifier names within the dictionary. The source name is disjoint with the dictionary. When no match is found, the operation returns a null value.

Section 8.3 defines the formal implementation of this dictionary search operation in pseudocode. This pseudocode leverages the `CPE_SUPERSET` and `CPE_SUBSET` functions defined in the CPE Name Matching specification [CPE23-M:7.2] to implement the search operation.

7.3 Use of Deprecated Identifier Names

The concept of deprecated identifier names is explained in Section 6.2. CPE dictionary users MUST NOT use a deprecated identifier name, but MUST instead use an identifier name that is linked to the deprecated identifier name through the deprecated-by relationship.[9] It is important to understand that even though a set of names may deprecate one identifier name, an organization does not have to use all of these new names. The organization MAY simply pick a non-deprecated name out of the set, or the organization MAY choose to use the entire set.

When a product consumes or outputs an identifier name, that product MUST first determine if the identifier name is deprecated in the Official CPE Dictionary. If the identifier name is not present in the Official CPE Dictionary, then the product MUST determine if the identifier name is deprecated in an extended CPE dictionary to which it has access. If the identifier name is deprecated, the product MUST resolve the correct non-deprecated identifier names for use in place of the deprecated identifier name. The product MUST use the following process, which references the `dictionary-search` function defined in Section 8.3, to perform this resolution.

1. If the deprecated identifier name does not reference any identifier names within the *deprecated-by* element, then the organization MUST choose a new non-deprecated identifier name from the dictionary. This situation will occur if the deprecation is of type "Identifier Name Removal".

2. If the deprecated identifier name provides an identifier name within the *deprecated-by* element, and:

    a. If the deprecation type is "Identifier Name Correction", then the organization MUST resolve the dictionary entry containing the identifier name listed. The organization MUST resolve this identifier name using the `dictionary-search` function, passing the identifier name, the dictionary, and a value of *true* as the parameters to the function; these arguments will result in an identifier lookup operation.

    b. If the deprecation type is "Additional Information Discovery" then the organization MUST resolve the set of dictionary entries containing the identifier names. The organization MUST resolve this set of identifier names using the `dictionary-search` function, passing the set of identifier names, the dictionary, and a value of *false* as the parameters to the function; these arguments will result in a dictionary search operation.

[9] This requirement MAY be ignored when dictionary users are purposefully intending to communicate information relating to deprecated WFNs.

3. If there are multiple deprecation elements or a single deprecation element with multiple deprecated-by elements, then the organization MUST iterate through step 2 for all deprecated-by elements provided. The organization MUST take the union of all resolved sets of identifier names; this union is the correct set of identifier names that have replaced the legacy name.

4. The organization MAY encounter deprecated identifier names in the set of identifier names resolved in Step 2. In this case, the organization MUST follow the above process to replace these deprecated names with the set of names that deprecated it; this process may be recursive.

5. The final set of resolved names represents all names that replace the legacy name. Organizations MAY either use all of these names, or pick one name out of the set to use in place of the legacy name.

8. CPE Dictionary Operations and Pseudocode

This section specifies functions for basic CPE dictionary operations and provides pseudocode for the identifier name acceptance criteria and the dictionary searching operations.

8.1 Operations on a CPE Dictionary

This section defines a set of functions for performing common activities against a CPE Dictionary. These functions are relatively simple and do not require a pseudocode implementation, but the remaining sub-sections present pseudocode that may utilize these common functions.

8.1.1 Function get_cpe_items(d)

The `get_cpe_items(d)` function takes a single CPE Dictionary, `d`, and returns all `cpe-items` associated with it. A single `cpe-item` within the dictionary represents the identifier name and all associated metadata.

8.1.2 Function get_cpe_item_WFN(item)

The `get_cpe_item_WFN(item)` function takes a single `cpe-item` element and returns the identifier WFN that it represents. Because the dictionary holds bound representations of names, it is necessary to unbind one of those to obtain a WFN. The formatted string binding should be used if available, and it can be unbound using the `unbind_fs(fs)` function defined in the CPE Naming Specification [CPE23-N:6.2.3].

8.1.3 Function get(w,a)

The `get(w,a)` function takes two arguments, a WFN `w` and an attribute `a`, and returns the value of `a`. If the attribute `a` is unspecified in `w`, the function returns the default value ANY. This function is defined in the CPE Naming Specification [CPE23-N:5.4.2].

8.1.4 Function is_deprecated(item)

The `is_deprecated(item)` function takes a single `cpe-item` and returns `true` if the `cpe-item` is deprecated, `false` if the `cpe-item` is not deprecated.

8.1.5 Function getItem(list, index)

The `getItem(list, index)` function is a helper function for retrieving an item in a list. The function will return the list item at the position specified by `index`. This function assumes a 0-based index and will return `null` if no items exist at the provided index.

8.1.6 Function strlen(s)

The `strlen(s)` function returns the length of string `s`. If the string is empty, it returns zero. It is defined as in GNU C.

8.1.7 Function substr(s,b,e)

The `substr(s,b,e)` function returns a substring of string `s`, beginning with the character at location `b` (with zero being the first character) and ending with the character at location `e`. If `b` equals `e`, it returns the character at location `b`. `b` must be less than or equal to `e`. If `b` is equal to or greater than `strlen(s)`, it returns `nil`.

8.2 Acceptance Criteria Pseudocode

This section defines the algorithm required to implement the acceptance criteria defined in Section 6.1. The core algorithm is implemented in the below pseudocode function named `accept-name`, which processes a given formatted string binding, `fs`, converts it to its WFN representation, `w`, and compares the WFN against a specific dictionary to determine if the dictionary should accept the new name. The following list provides a brief summary of the algorithm implemented in the `accept-name` function:

1. Unbind the formatted string `fs` to its WFN representation `w` because all the other functions within `accept-name` require a WFN as input. See the `unbind_fs` function in the CPE Naming Specification [CPE23-N:6.2.3].

2. Use the helper function `contains-restricted-characters` to determine if the WFN `w` contains any of the restricted characters defined in Section 6.1.1.3. If `w` contains any of the restricted characters, `accept-name` will return false.

3. Use the helper function `contains-required-attributes` to determine if the WFN `w` contains known data for all of the required attributes specified in Section 6.1.2. If `w` does not contain all of the required attributes, `accept-name` will return false.

4. Use the helper function `matches-more-complete-in-dictionary` to determine if the WFN `w` is unique within the dictionary `d`, as specified in Section 6.1.3. If `w` does not adhere to this rule, or in other words if `w` matches against a more complete name in the given dictionary `d`, then `accept-name` will return false. This function uses the `dictionary-search` function defined in Section 8.3 for matching against the dictionary.

```
1  function accept-name(fs, d)
2    ;; Top-level function to determine if the formatted string fs
3    ;; should be accepted into dictionary d based on high-level
4    ;; acceptance criteria. Assumes that the fs and its WFN
5    ;; representation meet acceptance criteria defined in the
6    ;; CPE Naming Spec.
7    w := unbind_fs(fs).
8    if contains-restricted-characters(w)
9      then return false.
10   endif.
11   if !contains-required-attributes(w)
12     then return false.
13   endif.
14   if matches-more-complete-in-dictionary(w, d)
15     then return false.
```

```
  endif.
  return true.
end.
```

Figure 8-1: accept-name function

```
function contains-restricted-characters(w)
  ;; Helper-function to determine if WFN w contains characters
  ;; not permitted in dictionary.
  ;; loop through every attribute in WFN
  foreach a in {part,vendor,product,version,update,edition,language,
                sw_edition,target_sw,target_hw,other} do
    s := get(w,a). ;; get value of attribute
    if (s = ANY or s = NA)
      ;; the attribute value is a logical
      then continue. ; to next attribute
    endif.
    ;; If we get here, s is a string value.
    n := 0.
    loop
      if n >= strlen(s)
        then break. ;; break to outer loop
      endif.
      c := substr(s,n,n).  ;; get the n'th character of s.
      if (c = "*" or c = "?")
        then
          if ((n = 0) or (substr(s,n-1,n-1) != "\"))
            then return true. ;; unquoted '*' or '?' not permitted
          endif.
        else
          ;; character is legal, move on
          n := n + 1.
          continue.
      endif.
    endloop.
  endfor.
  return false.
end.
```

Figure 8-2: contains-restricted-characters function

```
function contains-required-attributes(w)
  ;; Helper-function to determine if required attributes contain
  ;; known data. WFN syntax defined in CPE Naming Spec ensures all
  ;; attributes contain at least some data.
  foreach a in {part, vendor, product, version} do
    s := get(w,a). ;; get string value of attribute
    ;; only loop through required attributes of w
    if s = ANY
      then return false.
    endif.
```

```
11     if (a != version and s = NA)
12       then return false. ;; NA only permitted in version
13     endif.
14   endfor.
15   return true.
16 end.
```

Figure 8-3: contains-required-attributes function

```
1  function matches-more-complete-in-dictionary(w, d)
2    ;; Helper-function to determine if identifier WFN w matches a
3    ;; more complete name in the dictionary d (i.e. a superset match).
4    matches := dictionary-search(w, d, false).
5    if (size(matches) > 0)
6      then
7        if (getItem(matches, 0) = "SUPERSET-MATCH")
8          then return false.  ;; at least one superset match was found
9        endif.
10   endif.
11   return true.  ;; no match, or subset match are both okay.
12 end.
```

Figure 8-4: matches-more-complete-in-dictionary function

8.3 Dictionary Use Pseudocode

This section contains pseudocode that implements the identifier lookup and dictionary search operations defined in Section 7 with the *dictionary-search* function.[10] This function is focused on returning all non-deprecated names within a dictionary, but will not filter out deprecated names from the result set, allowing the caller to filter the names based on use case. While this function will not filter out deprecated names, it does not guarantee that it will return all matching deprecated names in the result set; it only guarantees the return of all matching non-deprecated names. When not looking for exact matches, the function will look for all superset matches within the given dictionary. The function will only look for subset matches if the given dictionary contains no superset matches. This ordering derives from the CPE Dictionary acceptance criteria that will not allow a dictionary to contain non-deprecated subset matches if a superset match is present.[11]

Table 8-1 provides a detailed overview of the `dictionary-search` function. This pseudocode leverages the accessor functions defined in Section 8.1. The code also leverages the `CPE_EQUAL`, `CPE_SUBSET`, and `CPE_SUPERSET` functions defined in the CPE Name Matching specification [CPE23-M:7.2].

[10] This function does not account for deprecation chains. For example, if an identifier WFN is found that is deprecated, a tool may want to recursively search all the identifier WFNs in the deprecation chain (i.e., the set of all identifier WFNs referenced by the deprecated-by property).

[11] The `dictionary-search` function was designed for readability; more efficient methods for implementing this logic exist.

Table 8-1: Description of dictionary-search function

Line Numbers	Description
1	The function accepts three arguments, the source WFN, the dictionary to search against, and a boolean flag 'exact'. When 'exact' is true the function will perform an identifier lookup based on the source WFN. When 'exact' is false the function will perform dictionary searching for all names contained the source WFN set.
11	Creates a list to store all discovered matches as well as the match type; if a match is found, then the first position of this list will contain the match type and the second position will contain the matches (either single match, or set of matches).
12 -13	If 'exact' is true, the function will enter into logic to determine if an exact match is present within d. The function will then call the findExactMatch function (defined in Figure 8-8) to determine if an exact match exists in the dictionary.
14-20	If an exact match is found, the function will populate the 'response' list with the response type of 'EXACT-MATCH' and then append the exact match to the next position in the list. The function will then return the response list, or null if no match was found.
22	The function will call the findSupersetMatches function (defined in Figure 8-6) to build the set of all superset matches found in d.
23-28	If any superset matches are found, the function will populate the 'response' list with the response type of 'SUPERSET-MATCH' and then append the set of matches to the next position in the list. The function will then return the response list.
29	The function will call the findSubsetMatches function (defined in Figure 8-7) to build the set of all subset matches found in d. NOTE: No non-deprecated subset matches should be present if a superset match was found.
30-35	If any subset matches are found, the function will populate the 'response' list with the response type of 'SUBSET-MATCH' and then append the set of matches to the next position in the list. The function will then return the response list.
36	If no matches were found the function will return null.

```
1  function dictionary-search(source, d, exact)
2    ;; For a given source WFN, source, the function will determine
3    ;; how it relates to the given dictionary, d. If the source WFN
4    ;; is a superset match to one or more identifier WFNs in d, then
5    ;; the function will return that set of names. If the source name
6    ;; is not a superset of any dictionary names, but is a subset of a
7    ;; dictionary name(s), then the function will return the set of
8    ;; dictionary names of which it is a subset. If an exact match is
9    ;; found and exact is true, the function will return the exact
10   ;; match. If no match exists the function will return null.
11   response := new List().
12   if (exact = true)
13     match := findExactMatch(source, d).
14     if (match != null)
15       then
16         response := append(response, "EXACT-MATCH").
17         response := append(response, match).
18         return response.
19       else return null.
20     endif.
```

```
21   endif.
22   supersetMatches := findSupersetMatches(source, d).
23   if (size(supersetMatches) > 0)
24     then  ;;superset matches found
25       response := append(response, "SUPERSET-MATCH").
26       response := append(response, supersetMatches).
27       return response.
28   endif.
29   subsetMatches := findSubsetMatches(source, d).
30   if (size(subsetMatches) > 0)
31     then  ;; subset matches found
32       response := append(response, "SUBSET-MATCH").
33       response := append(response, subsetMatches).
34      return response.
35   endif.
36   return null.
37 end
```

Figure 8-5: dictionary-search function

Table 8-2: Description of findSupersetMatches function

Line Numbers	Description
4	Creates a set to store all discovered superset matches.
5	Starts looping over every cpe-item in given dictionary, d.
7	Retrieves the WFN represented by the current item from the dictionary.
8-12	Passes the source and dictionary WFN to the CPE_SUPERSET and CPE_EQUAL functions (defined in the CPE Name Matching specification [CPE23-M:7.2] to determine how the two names relate. If the source is a 'superset' of the dictionary name, but not equal to the dictionary name, then the function will append the item to the set of matches.
14	The function will return the set of matches.

```
1  function findSupersetMatches(source, d)
2  ;; For a given source WFN, source, the function will find all
3  ;; superset matches contained within the given CPE dictionary, d.
4    matches := new Set().
5    foreach item in get_cpe_items(d)
6      do
7        dictionaryName := get_cpe_item_WFN(item). ;;WFN from cpe-item
8        if ((CPE_SUPERSET(source, dictionaryName) = true) and
9            (CPE_EQUAL(source, dictionaryName) = false))
10         then
11           matches := append(matches, item).
12       endif.
13   endfor.
14   return matches.
15 end
```

Figure 8-6: findSupersetMatches function

Table 8-3: Description of findSubsetMatches function

Line Numbers	Description
4	Creates a set to store all discovered subset matches.
5	Starts looping over every cpe-item in given dictionary, d.
7	Retrieves the WFN represented by the current item from the dictionary.
8-12	Passes the source and dictionary WFN to the CPE_SUBSET and CPE_EQUAL functions (defined in the CPE Name Matching specification [CPE23-M:7.2] to determine how the two names relate. If the source is a 'subset' of the dictionary name, but not equal to the dictionary name, then the function will append the item to the set of matches.
14	The function will return the set of matches

```
1  function findSubsetMatches(source, d)
2  ;; For a given source WFN, source, the function will find all
3  ;; subset matches contained within the given CPE dictionary, d.
4    matches := new Set().
5    foreach item in get_cpe_items(d)
6      do
7        dictionaryName := get_cpe_item_WFN(item). ;;WFN from cpe-item
8        if ((CPE_SUBSET(source, dictionaryName) = true) and
9            (CPE_EQUAL(source, dictionaryName) = false))
10         then
11           matches := append(matches, item).
12       endif.
13   endfor.
14   return matches.
15 end
```

Figure 8-7: findSubsetMatches function

Table 8-4: Description of findExactMatch function

Line Numbers	Description
4	Starts looping over every cpe-item in given dictionary, d.
6	Retrieves the WFN represented by the current item from the dictionary.
7-10	Passes the source and dictionary WFN to the CPE_EQUAL function (defined in the CPE Name Matching specification [CPE23-M:7.2] to determine how the two names relate. If the two names are equal then the function will return the item as the exact match.
12	If not exact matches were found the function will return null.

```
1  function findExactMatch(source, d)
2  ;; For a given source WFN, source, the function will find the
3  ;; exact match contained within the given CPE dictionary, d.
4    foreach item in get_cpe_items(d)
5      do
6        dictionaryName := get_cpe_item_WFN(item). ;;WFN from cpe-item
7        if (CPE_EQUAL(source, dictionaryName) = true)
8          then
9            return item.
10       endif.
11   endfor.
12   return null.
13 end
```

Figure 8-8: findExactMatch function

Appendix A—References

The following documents are indispensable references for understanding the application of this specification.

Normative References

[CPE22] Buttner, A. and N. Ziring, *Common Platform Enumeration (CPE)—Specification*, Version 2.2, March 11, 2009. See http://cpe.mitre.org/specification/archive/version2.2/cpe-specification_2.2.pdf.

[CPE23-M] Parmelee, M., Booth, H., Waltermire, D., and Scarfone, K., NIST Interagency Report 7696, *Common Platform Enumeration: Name Matching Specification Version 2.3*, August 2011. See http://csrc.nist.gov/publications/PubsNISTIRs.html.

[CPE23-N] Cheikes, B., Waltermire, D., and Scarfone, K., NIST Interagency Report 7695, *Common Platform Enumeration: Naming Specification Version 2.3*, August 2011. See http://csrc.nist.gov/publications/PubsNISTIRs.html.

Informative References

[SP800-117] Quinn, S., Scarfone, K., Barrett, M., and Johnson, C., NIST Special Publication 800-117, *Guide to Adopting and Using the Security Content Automation Protocol*, July 2010. See http://csrc.nist.gov/publications/PubsSPs.html#800-117.

[SP800-126] Waltermire, D., Quinn, S., Scarfone, K., and Halbardier, A., NIST Special Publication 800-126 Revision 2, *The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.2*, July 2011. See http://csrc.nist.gov/publications/PubsSPs.html#800-126.

Appendix B—Change Log

Release 0 – 9 June 2010

- Complete draft specification released to the CPE community for comment.

Release 1 – 30 June 2010

- Minor edits to audience description.
- Minor editorial changes throughout the document.
- Removed all mention of and support for the logical value UNKNOWN.
- Updated Dictionary Searching section to remove the notion of an Error result, and clarify on superset versus subset matches.
- Updated deprecation logic, and data model to include three distinct types of deprecation.

Release 2 – 2 June 2011

- Reorganized the sequence of several sections and sub-sections.
- Minor editorial changes throughout the document.
- Reorganized the data model discuss to include all elements and attributes available in version 2.3.
- Changed the overarching term "Dictionary Searching" to "Dictionary Use" to distinguish it from the lower-level term "dictionary searching".
- Clarified what results are returned by identifier lookups and dictionary searching.
- Made several minor changes to the version 2.3 schema/data model, including making several elements and attributes optional and changing a few types.

Release 3 – 30 August 2011

- Final release of CPE Dictionary 2.3 specification.
- Made minor editorial changes.
- Added prefix/namespace/schema mapping table to Section 1; updated prefixes for consistency with corresponding XML schemas.
- Adjusted some Section 5 requirements to be normative instead of informative.