RETIRED DRAFT

March 29, 2016

The attached DRAFT document (provided here for historical purposes):

> Draft NIST Interagency Report (NISTIR) 7831, *Common Remediation Enumeration (CRE) Version 1.0* (posted for public comment on December 6, 2011)

has been RETIRED, and additional development has been discontinued.

Information on other NIST cybersecurity publications and programs can be found at: http://csrc.nist.gov/.

The following information was originally posted with the attached DRAFT document:

Dec. 6, 2011

NIST IR-7831

DRAFT Common Remediation Enumeration (CRE) Version 1.0

NIST announces the public comment release of Draft NIST Interagency Report (NISTIR) 7831, Common Remediation Enumeration Version 1.0. NISTIR 7831 defines the Common Remediation Enumeration (CRE) specification. CRE is part of an emerging suite of enterprise remediation specifications that enable automation and enhanced correlation of enterprise remediation activities. Each CRE entry represents a unique remediation activity and is assigned a globally unique CRE identifier (CRE-ID). This specification describes the core concepts of CRE and the technical components of a CRE entry, outlines how CRE entries are created, and defines the technical requirements for constructing CRE entries.

NIST requests public comments on draft NISTIR 7831 by ***January 20, 2012***. Comments should be sent to remediation-comments @nist.gov

National Institute of
Standards and Technology

U.S. Department of Commerce

NIST Interagency Report 7831
(Draft)

Common Remediation Enumeration (CRE) Version 1.0 (Draft)

Gerard T. McGuire
David Waltermire
Jonathan O. Baker

NIST Interagency Report 7831 (Draft)

Common Remediation Enumeration (CRE) Version 1.0 (Draft)

Gerard T. McGuire
David Waltermire
Jonathan O. Baker

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

December 2011

U.S. Department of Commerce

John Bryson, 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 7831 (Draft)
40 pages (Dec. 2011)

Acknowledgments

The authors, Gerard T. McGuire of The MITRE Corporation, David Waltermire of the National Institute of Standards and Technology (NIST), and Jonathan O. Baker of The MITRE Corporation wish to thank colleagues who reviewed drafts of this document and contributed to its technical content. The authors would like to acknowledge Mike Kinney of The National Security Agency (NSA), James K. Ronayne of Varen Technologies, Matt Kerr of G2 Inc., Matthew Wojcik, Charles M. Schmidt, John Wunder, and Franklin Haskell of The MITRE Corporation, and Karen Scarfone of Scarfone Cybersecurity for their insights and support throughout the development of this document.

Abstract

This document defines the Common Remediation Enumeration (CRE) 1.0 specification. CRE is part of a suite of enterprise remediation specifications that enable automation and enhanced correlation of enterprise remediation activities. Each CRE entry represents a unique remediation activity and is assigned a globally unique CRE identifier (CRE-ID). This specification describes the core concepts of CRE, the technical components of a CRE entry, outlines how CRE entries are created, the technical requirements for constructing a CRE-ID, and how CRE-IDs may be assigned. CRE-IDs are intended to be boundary objects that are broadly useable in enterprise security management products and information domains that participate in remediation activities or make assertions about remediation actions.

Trademark Information

Contents

List of Appendices

List of Tables

1. Introduction

Common Remediation Enumeration (CRE) is a scheme for identifying and describing remediation actions. For the purposes of CRE the definition of *remediation*, as presented in NIST Interagency Report (IR) 7670, *Proposed Open Specifications for an Enterprise Remediation Automation Framework* [IR7670], is: "…a set of actions that result in a change to an IT asset's configuration that bring it into compliance with policies, correct discovered vulnerabilities or misconfigurations, change settings or controls in response to events, or to install, remove or disable software which include patches." CRE enables automation and enhanced correlation of enterprise remediation activities.

CRE will facilitate communication about remediations between different groups and tools, much like CVE[1] and CCE[2] currently do for vulnerabilities and configuration items. Each CRE entry represents a unique remediation activity and is assigned a globally unique CRE identifier (CRE-ID). This specification describes the core concepts of CRE, the technical components of a CRE entry, and outlines how CRE entries are created. This includes the technical requirements for constructing a CRE-ID and assigning CRE-IDs. CRE-IDs are intended to be boundary objects[3] that are broadly usable in enterprise security management products and information domains that participate in remediation activities or make assertions about remediation actions.

1.1 Purpose and Scope

This document provides the definitive technical specification for CRE version 1.0.[4] Additionally, this document presents guidance and requirements for CRE publishers to ensure consistency in the creation of CRE entries, the assignment of CRE-IDs, and the exchange of CRE data.

The scope of this document is limited to CRE version 1.0. Other versions of CRE and related specifications are not addressed here. Future versions of CRE will be defined in distinct revisions of this document, each clearly labeled with a document revision number and the appropriate CRE version number.

1.2 Audience

This document is intended for three primary audiences:

- Content authors, editors, and CRE repository publishers seeking guidance on the proper assignment of CRE-IDs and the creation of CRE entries.
- Software developers and system integrators seeking to use or exchange CRE data in their products or service offerings.
- Policy makers seeking to include precise remediation actions in their system configuration policies.

The intended audience for this document is presumed to be familiar with the concepts presented in [IR7670].

1.3 Document Structure

This document is organized into the following major sections and appendices:

- Section 2 provides a list of normative references for the document.
- Section 3 defines selected terms and abbreviations used in the document.
- Section 4 describes the conformance requirements for CRE creators and users.
- Section 5 defines the composition of a CRE-ID.

---

1 http://cve.mitre.org/

2 http://cce.mitre.org/

3 Bowker, G. C., and Star, S. L. *Sorting Things Out: Classification and Its Consequences (Inside Technology)*, MIT Press, Boston, 1999.

4 CRE is one of a number of specifications described in the Enterprise Remediation Automation Framework overview and published in [IR7670].

- Section 6 defines the composition and technical details of a CRE entry.
- Section 7 discusses the role of content decisions and their place in establishing trust for federated repositories.
- Section 8 describes the management and governance of CRE repositories.
- Section 9 provides requirements for products and services using CRE-IDs and CRE Entries.
- Section 10 presents information on selecting CRE repositories.
- Appendix A details the motivating use cases for CRE.
- Appendix B describes the CRE Data Exchange Format which can be used to exchange CRE entries.
- Appendix C contains XML examples of CRE entry use and deprecation.
- Appendix D provides a change log that documents significant changes to major drafts of this 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 [RFC2119].

1.5 Feedback

Feedback to improve this specification is welcomed and encouraged. Comments can be sent to the remediation standardization development email list at remediation-dev@nist.gov. Further information regarding this list can be found at the NIST community web site: http://scap.nist.gov/community.html.

2. Normative References

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[ILSR], IANA, IANA Language Subtag Registry (ILSR), available at <http://www.iana.org/assignments/language-subtag-registry>

[IR7670], NIST, NIST IR 7670, *Proposed Open Specifications for an Enterprise Remediation Automation Framework (Draft)*, February 2011, available at <http://csrc.nist.gov/publications/PubsDrafts.html#NIST-IR-7670>

[IR7698], NIST, NIST IR 7698, *Common Platform Enumeration: Applicability Language Specification Version 2.3*, August 2011, available at <http://csrc.nist.gov/publications/PubsNISTIRs.html>

[RFC1035], IETF, RFC 1035, *Domain Names – Implementation and Specification*, November 1987, available at <http://www.ietf.org/rfc/rfc1035.txt>

[RFC2119], IETF, RFC 2119, *Key words for use in RFCs to Indicate Requirement Levels*, March 1997, available at <http://www.ietf.org/rfc/rfc2119.txt>

[RFC5646], IETF, RFC 5646, Tags for Identifying Languages, September 2009, available at <http://www.ietf.org/rfc/rfc5646.txt>

3. Terms, Definitions, and Abbreviations

For the purposes of this document, the following terms, definitions, and abbreviations apply.

3.1 CRE Terminology

Remediation concepts

- **Remediation method** – The specific steps or actions taken to perform a remediation.
- **Remediation effect** – The result of successfully applying a remediation method. This includes not only the changes to an IT asset's configuration state (e.g., settings, software), but also other aspects such as the delay or required action (e.g., reboot) before the change becomes effective, the duration for which the change is effective, and other side effects above and beyond making the change.
- **Remediation instance** – The conceptual combination of a specific remediation method and its remediation effect, including platform and (optionally) parameters.
- **Remediation location** – The system on which the remediation is ultimately applied. Note that this may be different from the system on which the remediation method is performed.

Remediation artifacts

- **Remediation statement** – A prose description of a remediation instance, including its method and effect.
- **Remediation implementation** – The creation of a script, software module, or set of detailed, human-oriented instructions that enable the application of a specific remediation.

CRE artifacts

- **CRE-ID** – A globally unique formatted string that is associated with a single CRE entry.
- **CRE entry** – A set of fields that describe a specific remediation instance and whose structure conforms to the CRE XML schema (see Appendix B).
- **CRE producer** – An organization or individual that issues new CRE-IDs and CRE entries.
- **CRE publisher** – An organization or individual that maintains a CRE repository.
- **CRE consumer** – A product that uses a CRE entry as a key resource. Typical CRE consumers are tools that directly perform, manage, or report remediation actions.
- **CRE referrer** – A product or data that contains or cites CRE-IDs.
- **CRE repository** – A collection of CRE entries maintained by a single CRE publisher. Documentation provided as part of the CRE repository must include the processes and content decisions that guide the publication of CRE entries within that repository (see Section 8). In most cases a CRE repository is associated with a single CRE producer. In other cases it may be desirable to aggregate CRE entries from multiple CRE repositories. When aggregating CRE entries, the CRE publisher would most likely not be the same as the CRE producer.

3.2 Acronyms and Abbreviations

CCE	Common Configuration Enumeration
CPE	Common Platform Enumeration
CRE	Common Remediation Enumeration
CRE-DEF	Common Remediation Enumeration Data Exchange Format
CRE-ID	Common Remediation Enumeration Identifier
CVE	Common Vulnerabilities and Exposures
DISA	Defense Information Systems Agency
DNS	Domain Name System

ERI	Extended Remediation Language
IR	Interagency Report
ITL	Information Technology Laboratory
NIST	National Institute of Standards and Technology
NSA	National Security Agency
RFC	Request for Comment
SCAP	Security Content Automation Protocol
SDDL	Security Descriptor Definition Language
USGCB	United States Government Configuration Baseline
UTC	Coordinated Universal Time
XML	Extensible Markup Language

4. Conformance

This section provides the high-level requirements that MUST be met for conformance with this specification. There are three primary roles described below: CRE publisher, CRE consumer, and third-party CRE referrer.

4.1 CRE Publisher

A *CRE publisher* is an organization that maintains a CRE repository and issues CREs for use by some community of interest. CRE publishers claiming conformance to this specification SHALL comply with the following requirements:

1. Create well-formed CRE-IDs according to the requirements defined in Section 5.
2. Publish well-formed CRE entries as defined in Section 6 and Appendix B, and as defined in the data model in Appendix B.
3. Adhere to the repository governance requirements defined in Section 8.
4. If republishing, follow the repository selection guidance as described in Section 10, and support the ability to consume CRE entries from other repositories in the format defined in Appendix B.
5. Make an explicit claim of conformance to this specification in any documentation provided to end users.

A CRE publisher may republish CRE entries that originated with another organization. These CRE entries MUST reuse the original CRE-ID, description, and platform. Changing any of these fields requires a new CRE.

4.2 CRE Consumer

A *CRE consumer* is a product or service that uses CRE-IDs to identify, perform, report, and correlate remediations. Products and services claiming conformance to this specification SHALL comply with the following requirements:

1. Use well-formed CRE-IDs as described in Section 5.
2. Use CRE-IDs from a repository that complies with the requirements in Section 4.1.
3. Implement CRE entries as described in Section 9.
4. Follow the repository selection guidance as described in Section 10.
5. Make an explicit claim of conformance to this specification in any documentation provided to end users.

4.3 Third-Party CRE Referrer

A *third-party CRE referrer* is an organization that maintains a document, data model, or database that incorporates CRE-IDs. Such organizations claiming conformance to this specification SHALL comply with the following requirements:

1. Use well-formed CRE-IDs as described in Section 5.
2. Comply with the CRE entry concept (honoring properties such as the version and deprecation status, etc.) as described in Section 6.
3. Follow the repository selection guidance as described in Section 10.
4. Make an explicit claim of conformance to this specification in any documentation provided to end users.

5. CRE-ID Definition

A *CRE-ID* is a case-insensitive textual identifier that can be used to uniquely name a single remediation instance. A CRE-ID is simply a non-semantic unique identifier; it does not carry any information about the remediation. Each CRE-ID has four components that MUST be represented in the following format:

```
<PREFIX>:<NAMESPACE>:<ID>-<CHECK_DIGIT>
```

The components of a CRE-ID are:

- PREFIX – A text value that MUST be "cre".
- NAMESPACE – The namespace to which the identifier belongs. It MUST be represented as a reverse Domain Name System (DNS) [RFC1035] name that represents the organization of the CRE producer. Using a reverse DNS name provides a hint about the origin of the id, allows organizations to manage their own collections of IDs, and tends to avoid CRE collisions. A valid reverse-DNS style string is limited to letters, numbers, periods, and the hyphen character. These namespace strings MAY have any number of parts, and CRE consumers and referrers processing them SHALL treat them as case-insensitive. (That is, com.ABC is considered identical to com.abc). The namespace property of all CRE-IDs MUST be the same for all CRE-IDs issued by a single CRE producer.
- ID – MUST be a positive integer value that is unique relative to the "namespace" component. Leading zeros MUST NOT be included.
- CHECK_DIGIT – MUST be a Luhn[5] check digit that is calculated by processing the numeric value of the ID component.

Examples of valid CRE-IDs are:

- **`cre:org.notforprofit:5270-4`**
- **`cre:gov.exampleagency:5270-4`**

These examples can also be used to illustrate some subtleties; note that the check digit remains the same in both examples because the check digit algorithm is run only on the numeric portion of the identifier.[6] The check digit (**4**) is the result of performing the Luhn algorithm on **5270 only**. The fact that both CRE-IDs have the same ID number is also not problematic because the IDs have different namespaces.

[5] The Luhn algorithm, which is defined in Annex B of ISO/IEC 7812-1 and further described in http://www.tayloredge.com/reference/Mathematics/Luhn.pdf. The web page http://www.ee.unb.ca/cgi-bin/tervo/luhn.pl provides a utility for calculating Luhn values.

[6] The Luhn check digit only encodes the numeric portion of the identifier. It is expected that the final release of CRE version 1.0 will use another algorithm that will also protect the namespace.

6. CRE Entry Definition

A *CRE entry* is a set of properties that describe a specific remediation instance (e.g., a single configuration setting change, the application of a patch, installation/de-installation of software, a system reboot). The following sections describe the overall traits of a CRE entry, the properties that comprise a CRE entry, and the overall methodology to apply to CRE creation.

6.1 Traits of a CRE Entry

This section describes the primary traits of a CRE entry to be considered when creating and publishing new CRE entries. Each entry in a CRE repository MUST be *static* and *unique*.

6.1.1 Static

The meaning of a CRE entry, as defined by the description, platform, and parameter properties (see Sections 6.2.1 through 6.2.3), MUST NOT change. Once published, a CRE entry may be implemented by CRE consumers, aggregated by other CRE publishers, and referenced by third-party referrers. The requirement that CRE entries are static allows for references to and uses of CREs that do not change over time. For example, a security guide author using CRE-IDs can be confident that the definition of the corresponding CRE entry will not change after being published. Any change to the meaning of a CRE places an undue maintenance burden on the users of that CRE, undermining its usefulness.

Note that new translations of descriptions, and changes to metadata and references do not constitute a change in a CRE entry's meaning.

6.1.2 Unique

A CRE entry with the same meaning MUST NOT be repeated within the scope of a CRE-ID's NAMESPACE component (see Section 5). This requirement allows for consistent communication about remediation actions across organizations and tools by ensuring that remediation actions can always be correlated.

6.2 CRE Entry Properties

To minimize the labor involved in creating a CRE entry and the overall storage load on repositories, and to improve the ability to find matches, a CRE entry consists of only the minimum amount of data required to differentiate one remediation instance from another. A CRE entry has the following properties:

- A CRE-ID as described in Section 5.
- A textual, human-oriented description of the entry as described in Section 6.2.1.
- An optional list of parameters applicable to the entry as described in Section 6.2.2.
- The platform on which the entry is valid, expressed as a Common Platform Enumeration (CPE) Applicability Language expression as described in Section 6.2.3.
- Supporting references as described in Section 6.2.4.
- Supporting metadata related to the CRE entry as described in Section 6.2.5.

6.2.1 Description

The description property of the CRE entry is a statement intended for a human audience describing the method and the effect of the remediation instance. It is used by higher-level remediation layers to define policy and aid humans in constructing relevant policy by providing enough information to differentiate one remediation from another. The

descriptions MUST be written at a technical level that addresses both implementers of products and services as well as content authors.

A description MUST completely describe the remediation method and remediation effect. The description MUST document any SPLIT, MERGE, or REPLACEMENT decision (see Section 6.3) as part of creating a new CRE entry. For example, CCE-2175-8 requires that permissions for the file %SystemRoot%\regedit.exe be assigned.

- There are multiple possible methods to change the user access rights on a file. An incorrect description for the corresponding CRE is: "Set the administrator permissions on the regedit.exe file appropriately" because the *method* (identifying the mechanism to affect this change) is omitted.
- The description "Prevent inappropriate access to files using Windows Explorer" is also incorrect because the *effect* (e.g., the specific file, permissions, and user) are not defined in the description.
- A complete description, "For specific users or groups defined by the account parameter, set the administrator permissions defined by the permissions parameter on the %SystemRoot%\regedit.exe file appropriately using Windows Explorer" unambiguously states both the method and effect.

If the immediacy of the *effect* of the remediation action is not instantaneous, then the immediacy MUST be defined within the description (e.g., on reboot, on service restart, after a certain time).

A CRE entry MAY have multiple descriptions to support language translations. There MUST be only one description per CRE entry per language. Translated descriptions MUST be equivalent in meaning.

6.2.2 Parameters

While a CRE entry represents an atomic remediation action, the actual execution of that action can lead to different outcomes. A single CRE entry might deal with the enablement of remote access, but this would cover both enabling and disabling this access. When such options exist within a CRE entry, the options are identified through the use of parameters. Observe the example used in the above section (Description - 6.2.1), which states "set the permissions… appropriately", allowing the same CRE entry to be used to both permit and deny access to the file.

When one atomic remediation action can be executed with different parameters, CRE entries SHOULD contain a list of parameters that fully identify all of the outcomes that may result. For example, the CRE entry associated with "The start-up type on telnet should be set appropriately" has a single parameter of "start-up type". A more complex expression (such as a range or a series of ranges) can be used as well. If a parameter value is needed to properly characterize a remediation action, then the list of parameters MUST be provided.

The level of detail present in the descriptions relies on the type of the parameter. The types generally include text, integer, floating point, date/time, access control list (ACL), and enumerated. The description of an enumerated parameter type MUST declare the use of each distinct value, and of all parameter types it usually requires the highest level of detail.

The description of a parameter MUST declare:

1. The purpose of the parameter
2. The general type of the parameter (e.g., text, integer, decimal, date/time), including the unit of measure if applicable.
3. The allowed range of values. Must be one of:
    a. A numeric range (simple or non-contiguous).
    b. An enumerated list of values.
    c. For custom values, such as access control lists (ACLs), the syntax and/or valid values MUST be described or documentation MUST be referenced. This capability permits complex objects to be passed without a need to explicitly list all objects and possible states.

For example, one method of setting file access permissions is via the Security Descriptor Definition Language (SDDL). It is appropriate to include a link to SDDL reference material in the parameter description: "The FILE_ACCESS_INFORMATION SDDL structure is described here: http://msdn.microsoft.com/en-us/library/windows/desktop/aa379567(v=vs.85).aspx"

A parameter MUST NOT change the method of the CRE entry. For example, a hypothetical CRE entry to change a registry setting where the registry ValueName is specified as a parameter is not permitted because the resulting remediation can affect any value; the behavior is not predictable.

The following example demonstrates a problem where the description for the parameter is insufficiently defined.

Table 1 – Invalid CRE Parameter: parameters are too broad, *effect* is uncertain

Name	Type	Description
Value	Integer	Inactivity time

The description of the parameter in Table 1 is not valid because the unit of measurement for the parameter is not defined. It is impossible to determine if the parameter value is in minutes, seconds, milliseconds, etc. A remedy for this problem is shown in Table 2. The purpose of the parameter and the applicable units are clearly stated.

Table 2 – Valid CRE Parameter: Parameter is of appropriate scope and readability

Name	Type	Description
Value	Integer	Inactivity time in seconds before screensaver is activated

The example below in Table 3 illustrates an incomplete parameter description that does not define the valid values.

Table 3 – Invalid CRE Parameter: Insufficient Parameter Description

Name	Type	Description
TCP Options	Integer	Enable/Disable RFC 1323 features

This parameter in Table 3 does not describe either the range of the parameter or an enumeration of the possible values. Each distinct value (as opposed to a range used in Table 2 above) needs to have a description. A reader of this description would need to research the range of values available for this parameter and the meaning of each specific value. The example shown in Table 4 adds this information to the description, demonstrating a complete parameter description. The parameter description describes each of the permitted values needed to perform the remediation action.

Table 4 – Valid CRE Parameter: Range and Parameter is fully explained

Name	Type	Description
TCP Options	Integer	Enable/Disable RFC 1323 features Valid Range: 0, 1, 2, 3 0 – disable RFC 1323 options 1 – window scaling enabled only 2 – timestamps enabled only 3 – both options enabled

In general, each parameter present in a CRE entry identifies one aspect of variability for the given CRE entry. Each parameter consists of a human-oriented description and a statement defining the permitted values. These values would be the values needed to perform the remediation action.

While most controls are defined by a single parameter (which may have many possible values), some controls are defined by multiple parameters. For example, a CRE to change a Windows registry setting may take two arguments, which consist of the registry value and the appropriate type.

6.2.3 Platform

The platform to which a given CRE entry applies is expressed using a CPE Applicability Language Version 2.3 or later statement [IR7698]. This platform is known as the target CPE. Each CRE entry MUST include a single designated platform; a remediation encompassing more than one platform would have a separate CRE created for each platform.

The CPE Applicability Language [IR7698] does allow for the specification of a complex platform. For example, the following XML binding of a CPE applicability language construction specifies an environment running Internet Explorer (IE) 7 or IE 8 on a Windows XP operating system. Note that the platform for the purposes of a CRE entry may include both operating system and application layers. In a statement of the format X running on Y, X is known as the target CPE, because it is the target of the applicability statement, and Y is known as the qualifying CPE because it qualifies which platforms the target must be running on.

CPE Platform Applicability Language Expression

```
Title: Microsoft Windows XP with Internet Explorer 7.x or 8.x
  All of the following conditions must be met:
    CPE match: "cpe:2.3:o:microsoft:windows_xp:*:*:*:*:*:*:*:*"
    One of the following conditions must be met:
      CPE match: "cpe:2.3:a:microsoft:internet_explorer:7.*:*:*:*:*:*:*:*"
      CPE match: "cpe:2.3:a:microsoft:internet_explorer:8.*:*:*:*:*:*:*:*"
```

A broad platform specification such as

"cpe:2.3:o:microsoft:*:*:*:*:*:*:*:*"

should not be used because all Microsoft operating systems would be accepted by this platform statement. As explained in Section 6.3 (Entry Creation Methodology), each operating platform MUST require a separate CRE entry. If the platform is too broad, then it may encompass operating systems with different methods or effects for a single CRE entry, causing that CRE entry to be invalid. And if the platform is too narrow, then there is a risk of creating additional CRE entries with the same method, effect, parameters, and other properties—essentially identical CRE entries duplicated for each too-narrow platform.

In all CRE platform specifications, all target CPEs MUST provide a vendor product. Qualifying CPEs MAY be less restrictive, such as using a wild card for the vendor product field.

Other specifications in the Security Content Automation Protocol (SCAP) family allow several platforms to apply to a particular enumeration; however, CRE does not permit such an association. The CRE shown in Table 5 is not permitted because three distinct platforms are specified in the CRE entry. Each platform will require its own distinct CRE.

Table 5 – CRE Disallowed: More than one platform

ID	cre:org.mitre:3010-6
Platform	cpe:2.3:o:microsoft:windows_vista:*:*:*:*:*:*:*:*:*, cpe:2.3:o:microsoft:windows_2008:*:*:*:*:*:*:*:*:*, cpe:2.3:o:microsoft:windows_7:*:*:*:*:*:*:*:*:*

The set of CRE entries shown in Tables 6a, 6b, and 6c is valid because each CRE entry is defined for exactly one platform. This example also demonstrates a trade-off when dealing with multiple, similar platforms, as these CREs are very similar.

Tables 6a, 6b, 6c – This set of CRE platforms is valid

ID	cre:org.mitre:3011-4
Platform	cpe:2.3:o:microsoft:windows_vista:*:*:*:*:*:*:*:*:*:*

ID	cre:org.mitre:3012-2
Platform	cpe:2.3:o:microsoft:windows_2008:*:*:*:*:*:*:*:*:*:*

ID	cre:org.mitre:3013-0
Platform	cpe:2.3:o:microsoft:windows_7:*:*:*:*:*:*:*:*:*:*

One of the issues of narrowly defining platforms (and illustrated by the above example) is that there will be very similar CRE entries that differ primarily by different platforms. This increases the storage requirements, and if a remediation method is deprecated, several CRE entries may be affected. On the other hand, if several platforms are rolled into the same remediation, non-apparent differences may be hidden, resulting in incomplete or inaccurate application on some platforms.

6.2.4 Supporting References

Supporting references provide links to published documentation where the remediation instance has been described, such as configuration guides, vendor security bulletins, or patch descriptions. These references provide additional supporting information to explain why the CRE was created and how it is distinct from other similar CREs. For example, the following reference supports Microsoft's Windows Server May 2011 cumulative patch:

Supporting Reference URL:
http://www.microsoft.com/technet/security/bulletin/MS11-may.mspx

A reasonable supporting reference MUST have sufficient information to locate the relevant information. A general reference to a library or enclosing volume set is incomplete.

The following example of a supporting reference URL is incorrect because it appears to reference an entire web site:
http://www.msexchange.org/

If a supporting reference is provided and corresponding information is available from the original vendor, a reference to that vendor information MUST be used, and one or more references to third party documentation MAY also be included.. The following supporting reference URL is not correct:

http://easyxdm.net/wp/2011/04/13/microsoft-security-bulletin-ms11-018

The problem with this reference is that it duplicates information published by and available from the original vendor (Microsoft). The original vendor is preferred because:

- they are most knowledgeable about the product, with insights on configuration issues and vulnerabilities
- they know best practices to correct issues in order to enforce policy

The following supporting reference URL has been corrected to refer to Microsoft's bulletin directly:

http://technet.microsoft.com/en-us/security/bulletin/ms11-018

Each CRE entry SHOULD have a populated supporting reference. A CRE entry MAY have multiple supporting references.

A number of additional optional fields are defined for supporting references. See Appendix B.1.6, Table 12 for a list of these fields.

6.2.5 Metadata

Metadata about CRE entries themselves MUST contain the following:

- Creation Date – A UTC[7] date/time reference indicating when this CRE was created.
- Modification Date – A UTC date/time reference indicating when this CRE was last modified. If this is the initial entry, then this should match the creation date. The modification date is updated whenever the version number is changed or this entry is deprecated.
- Version – A numeric value, initially a value of one which is incremented for each modification of this CRE entry. Modifications made to add descriptions in other languages, update or add a reference, or modify metadata SHALL cause the version to increment. Significant changes will cause a new CRE to be created (see Section 6.1.1).
- Submitter – The name of the authoring individual and/or organizational entity.

If the CRE entry is deprecated, then the metadata MUST contain a Deprecation Entry property. A Deprecation Entry consists of a deprecation reason, notes, and one or more successor CRE-IDs (see Section 6.4 for a complete discussion). If this metadata is not present, then the CRE entry is current and not deprecated.

6.3 Entry Creation Methodology

When determining what constitutes a unique CRE entry, the *remediation method* and *remediation effect* MUST be considered. These terms are defined in Section 3.1.

6.3.1 SPLIT Decisions

If there is more than one possible *method* or the *effect* of a single CRE entry varies, that CRE entry MUST be split into multiple CRE entries. When considering remediation statements, details of the *method* and *effect* of possible approaches will be considered to determine how CRE entries are created. If either the *method* or *effect* of two statements differ, they SHALL be SPLIT and assigned multiple CRE-IDs.

A CRE entry that applies to multiple platforms MUST be split into multiple CRE entries such that each platform has its own CRE entry.

While certain remediation actions may be valid on multiple platforms, many details regarding those actions have been demonstrated to vary from one platform to the next. Examples include valid parameters, indicators, operational impacts, prerequisites, whether a reboot is required, etc.

The location of otherwise similar *methods* will be considered significant (and MUST be included within the description) in creating CRE entries. For example, a change made on an individual system will be SPLIT from a change made via a directory service, even where the eventual measurable effect on the end system is the same.

The immediacy of the *effect* of various remediation options shall be considered significant (and MUST be included within the description) when creating CRE entries. A CRE entry MUST be SPLIT based upon differences in when an option is effective (e.g., immediately, on reboot, on service restart, after a certain time). In situations where high availability is a critical business need, immediacy of effect may trump completeness when selecting between remediations. For example, a partial fix, workaround, or mitigation which takes effect immediately may be preferable to a complete fix which requires a reboot for servers which must remain running until a future scheduled maintenance window. For this reason, it is important that CREs distinguish between these options so that the correct course of action can be clearly indicated.

[7] Coordinated Universal Time - http://www.nist.gov/pml/div688/utcnist.cfm#cut

In taking *effect* into account, the permanence of the effect MUST result in separate CRE entries. For example, the effects of a CRE that disables a service until reboot are different than one that keeps the service disabled after a reboot; they would be separate CREs.

CRE entries MUST be SPLIT based on whether a remediation option inherently can be applied to only a single system or to multiple systems. For example, the same Windows Group Policy Object would be assigned different CRE-IDs when applied via a local policy versus a domain policy.

CRE entries are a "per platform" object. That is, even if the same method has the same effect across multiple platforms, a separate CRE entry MUST be created for each platform.

6.3.2 MERGE Decisions

If multiple remediations with equivalent *methods* are producing the same measurable *effect*, they SHALL be MERGED and assigned the same CRE-ID. For example, suppose that two different remediation management tools support the same remediation instance, affecting a change to the same method and causing the same effect. These remediation instances are the same CRE entry and should have the same CRE-ID. If the remediation instances are new, then a single CRE entry is to be created; if the remediation instances have multiple CRE-IDs, then they are to be merged into a single CRE entry and the old CRE entries deprecated.

As discussed in the SPLIT Decisions section (6.3.1), details of the *method* and *effect* of possible approaches MUST be considered to determine how CRE entries are assigned. If *method* and *effect* are functionally equivalent, the statements SHALL be MERGED and assigned the same CRE entry.

When considering a configuration setting, the value of the setting MUST NOT result in the creation of multiple CRE entries. A parameter MUST be used to distinguish between possible setting values. For example, if a configuration setting exists that allows a service to be enabled or disabled, a single CRE entry must be created with a parameter that defines the enablement value.

In another example, one CRE entry will be created for setting the required minimum password length (on a given platform with a given method), regardless of the specific length required. The minimum required password length would be passed as a parameter. This avoids creating separate CRE entries for 12, 16, and 20 character passwords.

6.3.3 NEW Decisions

There will be instances in which an entirely new CRE will be created. A new CRE entry MUST meet all the requirements from Section 5 (CRE-ID) and Section 6 (CRE Entry Definition). Two such situations are:

- An entirely new remediation is identified. A new configuration or vulnerability may require a remediation action that has not been used previously. Note that some searching may be necessary to determine whether a NEW entry or a SPLIT of an existing CRE is warranted.
- A clarification of a previous CRE is published. This may occur when existing policy requires a new remediation action. An example is when a better remediation method is identified for the same effect as an existing CRE entry.

6.3.4 REPLACEMENT Decisions

If a CRE is found to have a deficit in its meaning related to its description, parameters, and/or platform properties, then one or more new CRE entries MUST be generated, and the original CRE entry MUST be deprecated.

There will be instances in which a new CRE will be created as a replacement for a deprecated CRE. The intent is to provide historical documentation of the evolution of a remediation. A replacement CRE entry MUST meet:

- All the requirements from Section 5 (CRE-ID) and Section 6 (CRE Entry Definition).

- The description must contain the CRE ID of the previous CRE entry.

The replaced CRE entry MUST be deprecated and MUST itself contain a reference to the replacement.

Users of CRE entries SHOULD avoid using deprecated CRE entries.

6.4 CRE Entry Deprecation

Third-party referrers SHOULD use the most up-to-date CRE information available and SHOULD avoid using deprecated CRE entries. A deprecated CRE entry will have a CRE Deprecation Entry property within its metadata.

A Deprecation Entry property contains a deprecation reason, a notes section, and zero or more successor CRE-IDs. The number of IDs present is dependent on the deprecation reason. The deprecation reason is restricted to one of the following terms:

- SUPERSEDED_BY – This CRE entry is replaced by another CRE entry. This Deprecation Entry property MUST contain exactly one successive CRE-ID.
- SPLIT – This CRE entry is replaced by two or more CRE entries as the result of a SPLIT content decision. Each successive new CRE-ID MUST be contained within the Deprecation Entry property.
- MERGE – When combining CRE entries, a new CRE entry MUST be created and the existing CRE entries MUST be deprecated with a deprecation reason of MERGE. A deprecation of this type MUST contain the CRE-ID of the succeeding CRE entry.
- DISCONTINUED – A CRE entry that has no replacement MUST have this deprecation reason. Examples of this state will occur when corresponding APIs are discontinued or hardware is no longer supported. There is no successor CRE. Details on the deprecation SHOULD be provided in the notes section.

The notes section is OPTIONAL and MAY provide additional insight into the reasons for the deprecation of the CRE entry.

See Section 6.2.5 for more information on the Deprecation Entry property.

7. Content Decisions

CRE Content Decisions (CDs) provide guidance on creating CRE entries with some degree of consistency. These guidelines are used to ensure that CRE entries are created in a consistent fashion, independent of who is doing the creation. CRE CDs attempt to describe the ways that humans name and discuss their intentions when documenting the resolution of configuration problems. As CRE matures the CDs will necessarily evolve and the fully documented set of CDs will be a living document to accommodate this evolution. In applying CRE to a diverse set of platforms, new issues will be discovered for which a CD must be made. For CRE to be broadly used there must be consistency across federated repositories and their CDs.

There are four basic types of content decisions available to content creators as described below:

- SCOPE – This limits the scope of the problem space that CRE must consider. For example, one might SCOPE CRE assignment to include only security-relevant remediations.
- INCLUDE – This type identifies a CD that indicates that a specific type of remediation may be included in the set of assigned CREs. Examples may INCLUDE fixes that are complete, mitigations, patches, etc.
- MERGE – This type identifies a CD that will lead to combining multiple remediations into one CRE. This combined CRE may use parameters to address different instances of the CRE. For example, the same CRE can be used to turn a setting “on” and “off” varying only by a parameter value
- SPLIT – This type indicates multiple distinct CREs should be created. For example, a remediation that applies to more than one platform MUST be SPLIT and assigned a CRE for each platform.

7.1 Global Content Decisions

The CDs in this document MUST be followed by all CRE repositories. Most of these CDs are listed in Section 6.3. The following is an additional global CD:

CRE-IDs will be assigned to remediations rather than general system configuration change actions. This decision limits the scope of the problem space that CRE must consider; only issues that arise from identified security workflows need to be addressed.

7.2 Use of Local Content Decisions

The CRE repository publisher has the option to create additional CD policies and if it chooses to create them, must publish them, as described in Section 8. These CDs MAY augment but MUST NOT contradict the global CDs described in this document.

8. Repository Management and Governance

It is expected that CRE creation will be managed by many organizations using a decentralized, loosely federated approach. This section describes issues and provides guidance concerning the creation and maintenance of CRE repositories within the framework of a distributed, federated array of repositories.

8.1 CRE Repository Management Documents

Each CRE publisher MUST make available a set of managing documentation to support its use in the remediation community. The documents assist in the adoption of the CREs by third parties by describing the repository management processes and defining any local CDs in use.

8.1.1 Editorial Content Policy

A CRE publisher MUST publish applicable CDs (as described in Section 7.2) for each CRE repository.

8.1.2 Repository Management Process Document

Repository maintainers MUST publish a repository management process document for each distinct repository. The document MUST specify the lifecycle of CRE entries within the repository. This MUST include defining how CRE entry deprecation is handled, such as under what conditions entries are deprecated and how the correct replacement for the CRE is determined if a third party discovers a deprecated CRE entry.

Additional repository governance considerations will be found on NIST's Remediation Project website.[8]

8.2 Repository Entries

A CRE repository MUST have a single CRE entry for a given remediation instance; however, different CRE repositories MAY each have their own CRE entries for the same remediation instance. Coordination between CRE producers is recommended to avoid duplication of CRE entries across CRE repositories.

[8] Precise location and contents to be determined.

9. Implementing CRE Entries

Proper CRE entries are the basis for all remediation. Whether identifying, performing, reporting, or correlating remediations, the form and content of CRE entries must remain consistent. Implementers of products and services using CRE-IDs and CRE entries MUST comply with the following requirements:

Regarding CRE repository use:

1. MUST use CRE-IDs from a repository or repositories that comply with the requirements in Section 4.1. CRE repositories are composed of CRE entries which contain CRE-IDs. Organizing a collection of CRE entries is just as important as creating the CRE-IDs in making them useful. Maintaining credibility for CRE entries, regardless of who created them, is crucial.
2. MUST state which CRE repository namespaces are supported in the consuming product or service. SHOULD document the policy used to determine which CRE-IDs are supported and which repository they came from.
3. MUST define the process and the frequency used to acquire updates from any subscribed repositories.

Regarding CRE entry use:

1. MUST accurately implement the method specified in the CRE description creating the described effect.
2. MUST, for a given CRE, honor all parameters and process all valid parameter values.

10. Selecting CRE Repositories

The CRE repository sources fall into three categories:

- Primary Software Author – An OS or application manufacturer. For example, Microsoft, Adobe, and Oracle would fall into this category when issuing CREs for their respective products. A vendor's repository will likely be the most authoritative source for remediations for its own products, although those will be the only remediations there.
- Policy Authority – An agency or organization that has responsibility for the compliance for a standard. For example, DISA could publish a CRE repository for STIG compliance, or NIST may host a repository for USGCB configuration CREs. The responsible entity will obtain CRE entries from multiple sources. Repositories maintained for this reason may be the only choice for enterprises that must conform to a specific organizational policy.
- Third Party Sources – A third party publishing a collection of CREs. For organizations not requiring primary source remediations this represents an alternative. For software for which a primary software author does not produce CREs, a third party repository may be the only choice. It should be noted that repositories maintained by these parties may have duplicate entries due to their potentially having multiple sources.

Appendix A—CRE Use Cases

The creation of CRE has been motivated by several use cases within the context of an automated response to a cyber vulnerability or policy violation. The core use cases listed in this appendix address four technical tasks: responding to assessment, enforcing system configuration policy, supporting policy definition, and reporting on CRE actions and status. In addition, several use cases explore the roles specified in Section 4 (Conformance).

A common theme among all of these use cases is the use of CRE identifiers to represent system actions, disambiguate similar responses, and clarify natural language descriptions of responses.

A.1 Policy Compliance Verification and Enforcement

In this use case, the system administrators are in the role of CRE consumers. The system administrators are provided with a policy and required set of CREs to bring their systems into compliance. The designated CREs include a description of system changes required to remediate any reported vulnerabilities, misconfigurations, or prohibited software. System administrators will use the CRE to:

- Map assessment finding to desired remediation actions
- Bring newly provisioned systems into policy compliance
- Look up additional remediation data
- Reference remediation effects
- Direct a remediation tool

A.1.1 Scenario 1

A lab manager runs a monthly scan of the lab's IT systems to determine if they are in compliance with the organization's configuration baseline. One such scan determines that, due to a misconfigured registry setting, a system is no longer compliant with the organization's screen saver timeout policy. The lab manager must determine which corrective actions correspond to this finding and apply the corresponding remediation.

A.1.2 Scenario 2

A company acquires several dozen new desktops and a few database servers for the accounting department. As delivered, these systems do not meet the standard configuration required by the policy defined by the CIO's office. Each system requires several configuration changes before it can be deployed into a production environment.

A.1.3 Problem

In both cases, the responsible authority must determine the appropriate remediation for each misconfiguration. If the organization's preferred remediation is not specified with a reference to a CRE, the organization will be left to research the possible set of remediations for each finding and make the appropriate choice for the organization. Natural language remediation instructions, without a CRE reference, may contain ambiguity, there may be more than one remediation for a given misconfiguration, and remediations across several platforms may appear to be identical but rely upon different platform-specific mechanisms.

A.1.4 How CRE Helps

By annotating the provisioning and deployment policies with CRE-IDs, the set of configuration changes can be applied as a series of repeatable, unambiguous actions. The problem of ambiguity is eliminated because there is no need to interpret natural language. Each referenced CRE contains information identifying the correct usage of that remediation. At each step of the process, all of the parties involved can use the CRE-IDs to find more information on the acceptable remediation action(s) for a given issue. In this way, they can more quickly and accurately map the appropriate remediation action into their own localized context and tool set.

A.2 IT Policy Formulation

A CRE-ID is used to designate actions to be taken when an asset is non-compliant with policy. A policy writer would reference a remediation action by the associated CRE-ID, allowing readers to:

- Direct remediation tools
- Reference remediation effects
- Look up related remediation data
- Implement one specific remediation when multiple options may be available.

A.2.1 Scenario

The cyber security department of a research facility prepares its local policy largely based on guidelines published by US Government agencies. The policy for this facility is customized and is not simply an adoption of any one standard. Instead, it uses recommendations from several sources such as the United States Government Configuration Baseline (USGCB), DISA Security Technical Implementation Guide (STIG), or a similar vendor guide (such as the Oracle Enterprise Manager Configuration Guide).

A.2.2 Problem

If CRE entries or CRE-IDs are not present, then a determination of which mechanism should be employed to make required settings is difficult. The natural language descriptions may contain ambiguity, there may be more than one way to achieve the desired state, and settings may vary across several operating systems. Different facilities may attempt to address the changes via differing mechanisms.

A.2.3 How CRE Helps

By annotating needed configuration standards with CRE-IDs, the administrators of those systems can use these identifiers to transfer required policies from the assorted sources. These CRE-IDs also allow the comparison of various actions to ensure there are no conflicts and all situations are covered. The use of CRE entries helps assure that any deviations from the stated policy are well understood. Consistency can be assured across multiple facilities.

A.3 Results of Remediation Action (Reporting)

A CRE-ID can be used as a reference when describing the results of a previously performed remediation, both at the single system and organizational level. Remediation results creators will use the CRE-ID to specify the remediations that were attempted and their outcomes.

Consumers of remediation results will use CRE-IDs to:

- Look up additional remediation data
- Reference remediation effects in event logs and reports
- Track a specific remediation across a set of systems

A.3.1 Scenario

An organization performed an automated configuration audit of several hundred systems and detected a large number of misconfigurations affecting a significant percentage of machines. The required remediations were performed. Due to the scale of the organization, a complete configuration audit will take several days to complete.

A.3.2 Problem

The resulting list of the changes made is long and the description of specific changes made can be ambiguous. Non-standard reporting can result in inaccurate assumptions and an ineffective response to vulnerabilities.

A.3.3 How CRE Helps

By listing the CRE-IDs for the remediation actions performed, the list of changes can be unambiguously determined. Furthermore, a list of remediation actions attempted but not successfully completed can also be compiled. By limiting a follow up audit to evaluating only those changes made as a result of remediation actions, a second audit of the affected systems can be performed with a faster turnaround.

A.4 Software Vendor (Creating a Repository)

In this example, the software support organization of a software developer is cast in the role of CRE publisher. They will have created CRE entries for their product. These can be patches, workaround procedures, and configuration changes. They are going to insert them into a repository for all their subscribers to access.

A.4.1 Scenario

A vulnerability has been discovered in a software product. The software support organization of the software product's creator has been made aware of it but knows it cannot have a definitive fix in the near future. They decide to produce a configuration change that will limit the product's capabilities so that the scenario allowing the vulnerability does not occur. This has the unfortunate side effect of circumscribing the product's functionality. Given this circumstance the software support organization also decides to produce a workaround that does not limit the product's functionality but limits the access to the facility using the product.

A.4.2 Problem

The vast majority of IT departments using the product (or any supported product) will be automatically notified of the vulnerability. They will have to figure out if they are using the product in a manner that will expose the vulnerability. If it does then they will have to determine if there is an existing corrective action to address the vulnerability. If the CRE structure is not present in the configuration baseline or in a written policy guide, then a determination of which fix is appropriate becomes more difficult. In this example there are two remediations. Will the IT department find them both? How will they determine which to use?

A.4.3 How CRE Helps

Use of CPE makes the connection between vulnerabilities and CRE entries very clear. Publication of new vulnerabilities and even CRE entries will trigger a search on the part of the users for appropriate CREs and vulnerabilities, respectively Regardless of the mechanism, IT departments will have in-hand CRE entries for new vulnerabilities. They can be assured that these represent the choices for remediation and can then make appropriate changes to their policy and databases.

A.5 Multi-Source Repositories

In this example, a third party is cast in the role of CRE publisher. They collect CREs from primary sources (who are typically the product developers). It is not necessarily true that an aggregator wouldn't go to other third sources, but their primary source will be the developers followed by the others in Section 8. It is possible that some enterprising individual might set up a repository for remediations for unsupported products. Legacy applications can be very expensive to update. Legacy support for end of life products can be extended by transferring support to a third-party aggregator. Support for current and legacy hardware drivers is similarly available from third parties.

A.5.1 Scenario

A vulnerability has been discovered in a software product. The software support organization of the software product's creator has been made aware of it and has a remediation. They enter it into their repository. A small business is using the product and receives notice that a vulnerability has been found in it.

A.5.2 Problem

Organizations rely upon a diverse set of applications and operating systems and keeping track of the current set of vulnerabilities and their remediations requires significant resources.

A.5.3 How CRE Helps

A third-party CRE aggregator maintains his own repository of CREs from many sources. When vulnerabilities and CREs are published by the vendor he will consume of the CRE entries from the vendor's repository and put them in his own. CREs then allow the aggregator to check his subscriber database to see who is using the affected products these latest CREs reference. He then pushes the CREs out to those subscribers. CRE aggregator subscribers will then have in hand only those CREs for the products they use. They need only to decide which CREs with which to update their policy and related databases.

A.6 Third Party Referrer (Maintaining CRE References)

Some entities may elect to maintain databases containing references to CREs, i.e. CRE IDs. This can be in addition to any CRE repositories maintained by the entity. There are reasons for doing this: it could merely be a logging operation; that is, CRE IDs could be added to a continuously updated trail of events along with timestamps, vulnerabilities found, denied accesses, and all the other myriad data items that are written to logs.
A more involved case is that of a security consulting firm who offers a combined vulnerability, weakness, bug report, patch, work-around, advisory, and remediation database. It is searchable using multiple different items and search operations. It is updated frequently. It is used by both clients and non-clients of the firm not just for quick updates and early warning but for the correlation of the different items it offers. This has definite value for IT departments that need to stay up-to-the-minute on threats and mitigations.

A.6.1 Scenario

A vulnerability has been discovered in a software product. The software support organization of the software product's creator has been made aware of it and publishes a remediation. It enters that remediation into its CRE repository.

A.6.2 Problem

The IT department for a firm maintaining a number of high volume web servers notices the vulnerability and its remediation. It needs to know not just what the vulnerability is and its remediation but what else may be related to it. If the remediation is one that is applied to multiple problems they may have already applied it. If it is completely new they may want to research the vulnerability it addresses and find any related vulnerabilities. This can be quite time-consuming.

A.6.3 How CRE Helps

A Third Party Referrer will do the work of checking all the various sources of security notices and the producers of CREs. He will then cross-reference their various attributes. Any one referencing the database will find it easier to find what the interactions may be between his existing remediations, vulnerabilities, and configuration. This enables him to make good decisions concerning what remediations to apply.

Appendix B—CRE Data Exchange Format

This section defines the XML-based representation of the CRE data model.

XML elements can be associated with a namespace. These namespaces are used within this appendix:

Table 7 - XML Namespaces in Use

Prefix	Namespace	Schema
cpe_lang	http://scap.nist.gov/schema/cpe/2.3/cpe-language_2.3.xsd	CPE 2.3 Applicability Language [IR7698]
xml	http://www.w3.org/XML/1998/namespace	Common XML Attributes
xsd	http://www.w3.org/2001/XMLSchema	XML Schema

The tables below define the properties of the data model. The Multiplicity column in these tables indicates how many times each property SHALL be permitted within a single instance of the parent element. Each Multiplicity entry is expressed as either a single value or a range of values. If the Multiplicity entry for a property includes the value 0, the property is OPTIONAL, otherwise the property is REQUIRED. If the Multiplicity entry for a property includes the value *, the property MAY be used more than one time, with no upper bound. Here are some examples:

- 1: the property shall be used once
- 1..*: the property shall be used at least once and may be used more than once
- 0..1: the property is optional and may be used at most once
- 0..*: the property is optional and may be used more than once

An element that has an @xml:lang attribute MAY appear multiple times within a single parent element to support multiple languages. If there are multiple instances of such a textual element within a single parent element, each instance MUST have a value for its @xml:lang attribute. Each value MUST specify the natural language locale for which the instance is written (e.g., “en” for English, “fr” for French). Values MUST be valid language tags as defined by [RFC5646]. All language and region codes used MUST be in the Internet Assigned Numbers Authority (IANA) Language Subtag Registry [ILSR].

B.1 Element Dictionary

Each data object in use will be described in this section. The description includes a table of attributes and elements within the object. Each table lists the name, data type, the cardinality (Multiplicity), and a description of each component.

B.1.1 cre_list Element

The CRE list is the data structure that defines the core structure of a CRE repository. In addition to implementing the data model described in Table 8, the CRE repository publisher must also provide supporting repository management process documentation as described in Section 8.1.2.

Table 8 – cre_list Element Properties

Property	Type	Multiplicity	Description
cre_entry (element)	CREEntryType	1..*	One or more CRE entries, each of which describes a specific remediation instance.
generator (element)	GeneratorType	1	Holds information about when a particular CRE List was compiled, what version of the schema was used, what tool compiled the document, and what version of the tool was used.

Property	Type	Multiplicity	Description
lang (attribute)	xml:lang	1	The default language for the entire document. Individual properties on other elements that support instances of the @xml:lang attribute can use it to denote the language they use, and can omit it to default to this language.

B.1.2 cre_entry Element

A CRE entry is the primary element which defines one distinct remediation action as described in Section 6. Each CRE entry defines exactly one remediation and is uniquely identified by the CRE-ID contained in the id property.

Table 9 – cre_entry Element Properties

Property	Type	Multiplicity	Description
id (element)	xsd:string	1	The CRE-ID, which is a unique identifier, handled as a string, recognized with the following regular expression: "cre:[A-Za-z0-9\.\-]+:[1-9][0-9]*-[0-9]". This element is discussed in Section 5.
description (element)	TextType	1..*	A human readable string which states the purpose of the CRE Entry. To support uses intended for multiple languages, this element supports the @xml:lang attribute. At most one description element MAY appear for each language. See Section 6.2.1.
parameters (element)	ParametersType	0..1	A list of parameters, which when combined with the CRE, describes a complete remediation action. This element is described in Section 6.2.2.
platform (element)	cpe_lang:platform	1	A CPE Applicability Language expression which designates the platform against which this CRE is valid. See Section 6.2.3.
supporting_references (element)	ReferencesType	0..1	Published documentation which can verify the correctness of the CRE action. See Section 6.2.4.
metadata (element)	MetadataType	1	Information about the entry proper: when it was created, who created it, and other information concerning the creation and disposition of the entry See Section 6.2.5.

B.1.3 generator Element

The generator element provides information about the creation and maintenance of the repository.

Table 10 – generator Element Properties

Property	Type	Multiplicity	Description
product_name (element)	xsd:string	0..1	Producer program that generated the CRE content.
product_version (element)	xsd:string	0..1	Version of the entity that generated the CRE content.

Property	Type	Multiplicity	Description
schema_version (element)	xsd:string	1	Version of the CRE Schema the CRE content is expected to validate against. This string consists of a major version, a period, and a minor version. It must match the regular expression: "[1-9][0-9]*\.[0-9]+".
timestamp (element)	xsd:dateTime	1	The date and time when this CRE repository document was created. The format for the timestamp is yyyy-mm-ddThh:mm:ss.
extension_point (element)	Any	0..*	An extension point that allows for the inclusion of any additional information associated with the generation of the CRE content.

B.1.4 metadata Element

The metadata element provides creator and creation information. It is described in Section 6.2.5.

Table 11 – metadata Element Properties

Property	Type	Multiplicity	Description
creation_date (element)	xsd:dateTime	1	The date and time when this CRE entry was created.
modification_date (element)	xsd:dateTime	1	The date and time when this CRE data store was last updated. Typically this data indicates when this entry was deprecated.
deprecated (element)	DeprecationType	0..1	This is present only when the entry has been deprecated.
submitter (element)	xsd:string	1	The name of the creating entity.
version (element)	xsd:positiveInteger	1	The version of the CRE entry.
extension_point (element)	Any	0..*	An extension point that allows for the inclusion of any additional information associated with the metadata of the CRE content.

B.1.5 supporting_references Element

The supporting_references element is a variable length list of supporting references for the CRE Entry.

Table 12 – supporting_references Element Properties

Property	Type	Multiplicity	Description
reference (element)	ReferenceType	1..*	A list of one or more supporting references.

B.1.6 reference Element

The reference element provides supporting information that provides a link to authoritative documentation which permits the validation of the CRE entry's effectiveness. This element can reference either a paper or a web

reference. If a web reference is used, then the href property MUST be populated. If a paper reference is used, then the title and section properties MUST be populated.

Table 13 – reference Element Properties

		Property	Type	Multiplicity	Description
		description (element)	TextType	0..*	A text description of the material being referenced.
		date (element)	xsd:date	1	The date which the reference was included in the entry.
Choice of one of:	Paper References	author (element)	xsd:string	0..*	A single author of the reference material. Multiple authors will require corresponding elements, one for each author.
		title (element)	xsd:string	1	The title of the material being referenced.
		section (element)	xsd:string	1	The publication source of the reference.
		publisher (element)	xsd:string	0..1	The publisher of the reference.
	Electronic References	href (element)	xsd:anyURI	1	The URL for web based references

B.1.7 deprecated Element

The presence of the deprecated element indicates that the containing CRE entry has been deprecated. Lack of this element indicates that the CRE entry is current.

Table 14 – deprecated Element Properties

Property		Type	Multiplicity	Description
Choice of one of:	superseded_by (element)	CREIDType	1	The CRE-ID of the CRE entry that replaces this CRE.
	split (element)	CREIDType	2..*	One CRE-ID for each resulting CRE entry created when this CRE is split.
	merge (element)	CREIDType	1	The CRE-ID of the CRE entry which replaces this CRE.
	discontinued (element)	xsd:null	1	The CRE entry has been deprecated with no known successor.
notes (element)		TextType	0..1	The reasoning for the deprecations is provided.

B.1.8 parameters Element

The parameters element is a variable length list of parameters for the CRE entry.

Table 15 – parameters Element Properties

Property	Type	Multiplicity	Description
parameter (element)	ParameterType	1..*	A list of one or more parameters.

B.1.9 parameter Element

The parameter element describes one parameter for a CRE entry. See Section 6.2.2.

Table 16 – parameter Element Properties

Property	Type	Multiplicity	Description
name (attribute)	xsd:NCName	1	A name for the parameter.
description (element)	TextType	1	A human understandable string that describes the purpose of the parameter.

B.1.10 TextType

The TextType type is used for several elements, including the cre_entry element's description element, the reference element's description element, the parameter element's description element, and the deprecated element's notes element.

Table 17 - TextType Properties

Property	Type	Multiplicity	Description
text (element)	xsd:string	1	A human readable string which provides text. To support uses intended for multiple languages, this element supports the @xml:lang attribute. At most one instance of this element MAY appear for each language.

Appendix C—XML Examples

This appendix contains XML examples of CRE entry use and deprecation.

C.1 Example 1

The following illustrates the use of a CRE entry to set a registry value.

```
<cre:cre_entry>
	<cre:id>cre:com.example:10521-3</cre:id>
	<cre:description>Using the RegSetKeyValue API function set the ScreenSaveActive entry in
the registry (HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Control Panel\Desktop) to
either enabled(1) or disabled(0).</cre:description>
	<cre:parameters>
		<cre:parameter name="ScreenSaveActiveValue">
			<cre:description>Enable (1) or disable (0) the screen saver password.
Valid values are 0 or 1.</cre:description>
		</cre:parameter>
	</cre:parameters>
	<cre:platform id="1701">
		<cpe:title> Microsoft Windows 7</cpe:title>
		<cpe:logical-test operator="AND" negate="0">
			<cpe:fact-ref name="cpe:2.3:o:microsoft:windows_7:*:*:*:*:*:*:*:*"/>
		</cpe:logical-test>
	</cre:platform>
	<cre:supporting_references>
		<cre:reference>
			<cre:date>2011-09-30</cre:date>
			<cre:title>Microsoft System Administration Guide</cre:title>
			<cre:section>Microsoft System Administration Guide, Section
12.5.3</cre:section>
		</cre:reference>
	</cre:supporting_references>
	<cre:metadata>
		<cre:creationDate>2011-09-15T16:00:00.377-04:00</cre:creationDate>
		<cre:modificationDate>2011-09-15T16:00:00.377-04:00</cre:modificationDate>
		<cre:version>1</cre:version>
		<cre:submitter>Example Corporation</cre:submitter>
	</cre:metadata>
</cre:cre_entry>
```

C.2 Example 2

This example illustrates the use of a CRE to set a Policy Object. This example uses multiple supporting references.

```
<cre:cre_entry>
	<cre:id>cre:com.example:31-5</cre:id>
	<cre:description>Set Maximum Password Age via the Domain Group Policy Object (Computer
Configuration\Windows Settings\Security Settings\Account Policies\Password Policy) using the
IGroupPolicyObject interface.</cre:description>
	<cre:parameters>
		<cre:parameter name="MaximumPasswordAge">
			<cre:description>The maximum number of days before a password change will
be required. An acceptable integer value will fall between 0 and 998 days
(inclusive).</cre:description>
		</cre:parameter>
	</cre:parameters>
	<cre:platform id="1701">
		<cpe:title> Microsoft Windows 7</cpe:title>
		<cpe:logical-test operator="AND" negate="0">
			<cpe:fact-ref name="cpe:2.3:o:microsoft:windows_7:*:*:*:*:*:*:*:*"/>
		</cpe:logical-test>
	</cre:platform>
	<cre:supporting_references>
```

```
                <cre:reference>
                        <cre:description>Discussion of password age
configuration</cre:description>
                        <cre:date>2011-09-30</cre:date>
                        <cre:href>http://msdn.microsoft.com/en-us/library/ms813412.aspx</cre:href>
                </cre:reference>
                <cre:reference>
                        <cre:description>Documentation for the Group Policy API:
IGroupPolicyObject interface</cre:description>
                        <cre:date>2011-10-07</cre:date>
                        <cre:href>http://msdn.microsoft.com/en-
us/library/windows/desktop/aa374177%28v=VS.85%29.aspx</cre:href>
                </cre:reference>
        </cre:supporting_references>
        <cre:metadata>
                <cre:creationDate>2011-09-15T16:00:00.377-04:00</cre:creationDate>
                <cre:modificationDate>2011-09-15T16:00:00.377-04:00</cre:modificationDate>
                <cre:version>1</cre:version>
                <cre:submitter>Example Corporation</cre:submitter>
        </cre:metadata>
</cre:cre_entry>
```

C.3 Example 3

This example illustrates the use of a CRE to set user rights. This example does not use a parameter.

```
<cre:cre_entry>
        <cre:id>cre:com.example:20941-1</cre:id>
        <cre:description>Assign the user right sedenyservicelogonright as specified for
designated security_principal(s) via INF file change.</cre:description>
        <cre:parameters>
                <cre:parameter name="SecurityPrincipals">
                        <cre:description>A list of security principals. Valid values are case-
insensitive strings that represent local and domain users, groups, and service accounts. In a
domain environment, security principals should be identified in the form: "domain\trustee name".
For local security principals use: "computer name\trustee name". For built-in accounts on the
system, use the trustee name without a domain.</cre:description>
                </cre:parameter>
                <cre:parameter name="PrivilegeValue">
                        <cre:description>A numeric value that represents grant (1) or remove (0)
for sedenyservicelogonright.  Valid values are 1 or 0. </cre:description>
                </cre:parameter>
        </cre:parameters>
        <cre:platform id="1701">
                <cpe:title> Microsoft Windows 7</cpe:title>
                <cpe:logical-test operator="AND" negate="0">
                        <cpe:fact-ref name="cpe:2.3:o:microsoft:windows_7:*:*:*:*:*:*:*:*"/>
                </cpe:logical-test>
        </cre:platform>
        <cre:supporting_references>
                <cre:reference>
                        <cre:date>2005-11-03</cre:date>
                        <cre:href>http://technet.microsoft.com/en-
us/library/bb457125.aspx</cre:href>
                </cre:reference>
        </cre:supporting_references>
        <cre:metadata>
                <cre:creationDate>2011-09-15T16:00:00.377-04:00</cre:creationDate>
                <cre:modificationDate>2011-09-15T16:00:00.377-04:00</cre:modificationDate>
                <cre:version>1</cre:version>
                <cre:submitter>Example Corporation</cre:submitter>
        </cre:metadata>
</cre:cre_entry>
```

C.4 Example 4

This example illustrates the use of a CRE to set file permissions using CACLS.exe.

```
<cre:cre_entry>
	<cre:id>cre:com.example:85-1</cre:id>
	<cre:description>Set file permissions for %SystemRoot%\System32\net.exe using
cacls.exe.</cre:description>
	<cre:parameters>
		<cre:parameter name="SDDL">
			<cre:description>A valid SDDL descriptor. The SDDL string format is
described on http://msdn.microsoft.com/en-
us/library/windows/desktop/aa379570%28v=vs.85%29.aspx</cre:description>
		</cre:parameter>
	</cre:parameters>
	<cre:platform id="1701">
		<cpe:title> Microsoft Windows 7</cpe:title>
		<cpe:logical-test operator="AND" negate="0">
			<cpe:fact-ref name="cpe:2.3:o:microsoft:windows_7:*:*:*:*:*:*:*:*"/>
		</cpe:logical-test>
	</cre:platform>
	<cre:supporting_references>
		<cre:reference>
			<cre:date>2011-07-31</cre:date>
			<cre:href>http://msdn.microsoft.com/en-
us/magazine/cc982153.aspx</cre:href>
		</cre:reference>
	</cre:supporting_references>
	<cre:metadata>
		<cre:creationDate>2011-11-30T16:00:00.377-04:00</cre:creationDate>
		<cre:modificationDate>2011-11-30T16:00:00.377-04:00</cre:modificationDate>
		<cre:version>1</cre:version>
		<cre:submitter>Example Corporation</cre:submitter>
	</cre:metadata>
</cre:cre_entry>
```

C.5 Example 5

This example illustrates how to deprecate a CRE.

```
<cre:cre_entry>
	<cre:id>cre:com.example:17-4</cre:id>
	<cre:description>Set file permissions for %SystemRoot%\System\net.exe using
cacls.exe</cre:description>
	<cre:parameters>
		<cre:parameter name="SDDL">
			<cre:description>A valid SDDL descriptor. The SDDL string format is
described at http://msdn.microsoft.com/en-
us/library/windows/desktop/aa379570%28v=vs.85%29.aspx</cre:description>
		</cre:parameter>
	</cre:parameters>
	<cre:platform id="1701">
		<cpe:title> Microsoft Windows 7</cpe:title>
		<cpe:logical-test operator="AND" negate="0">
			<cpe:fact-ref name="cpe:2.3:o:microsoft:windows_7:*:*:*:*:*:*:*:*"/>
		</cpe:logical-test>
	</cre:platform>
	<cre:supporting_references>
		<cre:reference>
			<cre:date>2011-07-31</cre:date>
			<cre:href>http://msdn.microsoft.com/en-
us/magazine/cc982153.aspx</cre:href>
		</cre:reference>
	</cre:supporting_references>
	<cre:metadata>
```

```
                <cre:creationDate>2011-11-30T16:00:00.377-04:00</cre:creationDate>
                <cre:modificationDate>2011-11-30T16:00:00.377-04:00</cre:modificationDate>
                <cre:version>1</cre:version>
                <cre:deprecated>
                        <cre:superseded_by>cre:com.example:85-1</cre:superseded_by>
                        <cre:notes>Superseded to correct file path. File path contained \System\
and should have contained \System32\.</cre:notes>
                </cre:deprecated>
                <cre:submitter>Example Corporation</cre:submitter>
        </cre:metadata>
</cre:cre_entry>
```

Appendix D—Change Log

RELEASE 0 – 6 DECEMBER 2011

CRE draft specification released for public comment.