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

NIST Interagency Report 7800
(Draft)

Applying the Continuous Monitoring Technical Reference Model to the Asset, Configuration, and Vulnerability Management Domains (DRAFT)

David Waltermire, Adam Halbardier, Adam Humenansky, and Peter Mell

NIST Interagency Report 7800 (Draft)

Applying the Continuous Monitoring Technical Reference Model to the Asset, Configuration, and Vulnerability Management Domains (DRAFT)

David Waltermire, Adam Halbardier, Adam Humenansky, and Peter Mell

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

January 2012

U.S. Department of Commerce

Secretary John E. Bryson

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 7800
29 pages (Jan. 2012)

Acknowledgments

The authors would like to recognize the following individuals for their participation on the continuous monitoring (CM[1]) research team, insightful ideas, and review of this work: Stephen York and Peter Sell from the National Security Agency, as well as Larry Feldman and Zach Ragland from Booz Allen Hamilton.

The authors would also like to thank the United States Chief Information Officer Council's Information Security and Identity Management Subcommittee (ISIMC) on Continuous Security Monitoring for its leadership and direction as we created this publication. In particular, we would like to thank the current co-chairs:[2] John Streufert from the Department of State, Kevin Dulany from the Office of the Secretary of Defense, and Timothy McBride from the Department of Homeland Security.

Trademark Information

Abstract

This publication binds together the CM workflows and capabilities described in NIST IR 7799 to specific data domains. It focuses on the Asset Management, Configuration, and Vulnerability data domains. It leverages the Security Content Automation Protocol (SCAP) version 1.2 for configuration and vulnerability scan content, and it dictates reporting results in an SCAP-compliant format. This specification describes an overview of the approach to each of the three domains, how they bind to specific communication protocols, and how those protocols interact. It then defines the specific requirements levied upon the various capabilities of the subsystems defined in NIST IR 7799 that enable each data domain.

[1] The acronym CM in this publication is not to be confused with other NIST 800 series publications that use the abbreviation CM to denote "Configuration Management."

[2] The co-chairs are listed on the Office of Management and Budget website https://max.omb.gov/community/display/Egov/Continuous+Monitoring+Working+Group+Members.

Table of Contents

List of Figures

List of Tables

1. Introduction

This specification is intended to define the implementation of asset, configuration, and vulnerability management data domains within the continuous monitoring (CM) architecture. The specification binds the subsystems, workflows, and interfaces defined in NIST IR 7799 [IR 7799], to the details of the three data domains. Gaps are called out where they exist in the current security automation landscape, and in some cases, near-term solutions are defined to fill those gaps. The three domains are described in the following paragraphs:

Asset management is a broad description for activities related to assets across an enterprise. NIST IR 7693 Asset Identification [IR 7693] defines an asset as "Anything that has value to an organization, including, but not limited to, another organization, person, computing device, information technology (IT) system, IT network, IT circuit, software (both an installed instance and a physical instance), virtual computing platform (common in cloud and virtualized computing), and related hardware (e.g., locks, cabinets, keyboards)." Asset management often involves understanding the complex relationships between assets within an organization's area of responsibility. CM is positioned to assist enterprises in gathering data on those relationships more regularly, as well as provide a framework to feed that information into other aspects of the security architecture.

Configuration management is primarily focused on the configuration status of computing devices across an enterprise. It involves determining compliance by collecting detailed information about specific configuration settings and comparing that data against an organization's policy. The resulting information may then be used as an input for aggregated reporting purposes or for targeted remediation. The continuous monitoring architecture provides a framework within which an organization may routinely collect configuration information from across its networks to compare against applicable policies.

Vulnerability management is concerned with understanding the security posture of an organization with respect to known vulnerabilities. It involves collecting information regarding vulnerabilities and patch levels of assets across the enterprise. The information is often fed into aggregate reporting, and may also be used to do targeted or large-scale remediation of discovered vulnerabilities. CM provides a framework which integrates more seamlessly with other security needs and facilitates regular, recurring scans.

1.1 Purpose and Scope

The purpose of this document is to bind the NIST IR 7799 [IR 7799] subsystems, workflows, and interfaces to low-level data domain-specific specifications for asset, configuration, and vulnerability management. The specifics of the low-level specifications, as well as the higher level information, documented in NIST IR 7799 are out of scope of this specification, as are any data domain-specific bindings outside of asset management, configuration management, or vulnerability management.

1.2 Audience

This publication is intended for those developing, testing, validating, or procuring information technology tools that are conformant with the CM reference model presented in NIST IR 7756 [IR 7756] and NIST IR 7799 [IR 7799]. This publication is written for a technical audience and assumes detailed knowledge of NIST IR 7756 and NIST IR 7799. The audience should also be familiar with SCAP version 1.2 as documented in NIST SP 800-126 Revision 2 [SP 800-126].

1.3 Document Structure

The remainder of this document is organized into the following major sections:

- Section 2 defines the terms used within this specification and provides a list of common acronyms that are used throughout the document.
- Section 3 describes how this specification relates to current standards and specifications.
- Section 4 describes the three data domains included in this specification and how they are addressed.
- Section 5 describes dependencies between the three Layer 2 data domains (the asset, configuration, and vulnerability management)[3].
- Section 6 details requirements for Layer 1 specifications to fulfill this specification.
- Section 7 details the Layer 1 specifications leveraged to fulfill the requirements described in Section 6.
- Section 8 documents a near-term data model for transporting asset information.
- Section 9 details reporting results and how information will be packaged.
- Section 10 details specific requirements for each data domain to implement the [IR 7799] subsystems.
- Appendix A contains the normative references used within the document.

1.4 Document Conventions

When referencing a specification listed in Appendix A, the short-form identifier is written between brackets, such as [XMLS].

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

Both inline and indented XML forms use qualified names to refer to specific XML elements. A qualified name associates a named element with a namespace. The namespace identifies the

[3] IR 7799 Section 1.2 defines the CM model layers (i.e., Layers 1-5) and their interactions.

specific XML schema that defines (and consequently may be used to validate) the syntax of the element instance. 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 generally will vary from document to document. In this specification, the conventional mappings listed in Table 1 are used.

Table 1. Conventional XML Mappings

Mappings Prefix	Namespace URI	Schema
ai	http://scap.nist.gov/schema/asset-identification/1.1	Asset Identification 1.1
am	gov:cm:asset:model	Asset Data Model (defined in this specification)

2. Terms and Abbreviations

This section defines a set of common terms and abbreviations used within this specification. Some of these terms may have other meanings outside the context of Continuous Monitoring. These definitions are provided to alleviate confusion with ambiguous terminology and are to be used within the context of this document. Some of these terms are taken directly from [IR 7693] and [SP 800-18].

2.1 Terms

Asset: Anything that has value to an organization, including, but not limited to, another organization, person, computing device, information technology (IT) system, IT network, IT circuit, software (both an installed instance and a physical instance), virtual computing platform (common in cloud and virtualized computing), and related hardware (e.g., locks, cabinets, keyboards) [IR 7693].

Asset Identification: The attributes and methods necessary for uniquely identifying a given asset. A full explanation of asset identification is provided in [IR 7693].

Computing Device: A machine (real or virtual) for performing calculations automatically (including, but not limited to, computer, servers, routers, switches) [IR 7693].

Data Domain: A specific class of cyber security data, methodologies, and procedures, such as configuration management, vulnerability management, license management, software assurance, etcetera; used in this document as a potential applicable function of the Continuous Monitoring capability.

Subsystem: A discrete block of functionality in the CM architecture. The architecture is designed around subsystems interacting with each other over interfaces.

System: A discrete set of information resources organized for the collection, processing, maintenance, use, sharing, dissemination, or disposition of information. [SP 800-18].

Task: A command (typically represented in XML) to instruct a subsystem to perform an action.

Well-formed: For markup languages such as html and XML, it describes an element that is either: opened and subsequently closed; empty, and thus must be terminated; or properly nested without overlap.

2.2 Acronyms

ARF	Asset Reporting Format
CAESARS	Continuous Asset Evaluation, Situational Awareness, and Risk Scoring
CCE	Common Configuration Enumeration
CM	Continuous Monitoring
CPE	Common Platform Enumeration
CVE	Common Vulnerability Enumeration
DA	Data Aggregation subsystem
IETF	Internet Engineering Task Force
IR	Interagency Report
IT	Information Technology
ITL	Information Technology Laboratory
NIST	National Institute of Standards and Technology
NIST IR	National Institute of Standards and Technology Interagency Report
OVAL	Open Vulnerability and Assessment Language
RFC	Request for Comment
SCAP	Security Content Automation Protocol
SDS	Source Data Stream
W3C	World Wide Web Consortium
XCCDF	Extensible Configuration Checklist Description Format
XML	Extensible Markup Language

3. Relationship to Existing Standards and Specifications

NIST IR 7800 is not a stand-alone document, but one that relies on and heavily references other NIST publications. This section of the document provides a brief introduction to these other standards and specifications and details how they are important to the understanding of the NIST IR 7800.

1. NIST IR 7756: CAESARS Framework Extension: An Enterprise Continuous Monitoring Technical Reference Architecture (Draft) - This publication presents an enterprise continuous monitoring technical reference architecture that extends the framework provided by the DHS Federal Network Security CAESARS architecture. This extension enables added functionality, defines each subsystem in more detail, and further leverages security automation standards. [IR 7756]

2. NIST IR 7799: Continuous Monitoring Reference Model Workflow, Subsystem, and Interface Specifications – This specification binds NIST IR 7799 to the data domains of asset management, configuration, and vulnerability. IR 7799 Section 1.2 defines the specification layers (i.e., Layers 1-5) and their interactions. [IR 7799]

3. NIST SP 800-126 Rev 2: Security Content Automation Protocol (SCAP) – SCAP standardizes a format and methodology for representing inventory, configuration, and vulnerability checks and checklists. This specification leverages SCAP as the standard content format for discovering configuration settings and determining the existence of vulnerabilities on a host. [SP 800-126]

4. Overview of Approaches to Address Data Domains

This section addresses how the data domains of asset management, configuration management, and vulnerability management map to the capabilities described in NIST IR 7799 [IR 7799]. For convenience, the CM system instance model from [IR 7799] is provided in Figure 1.

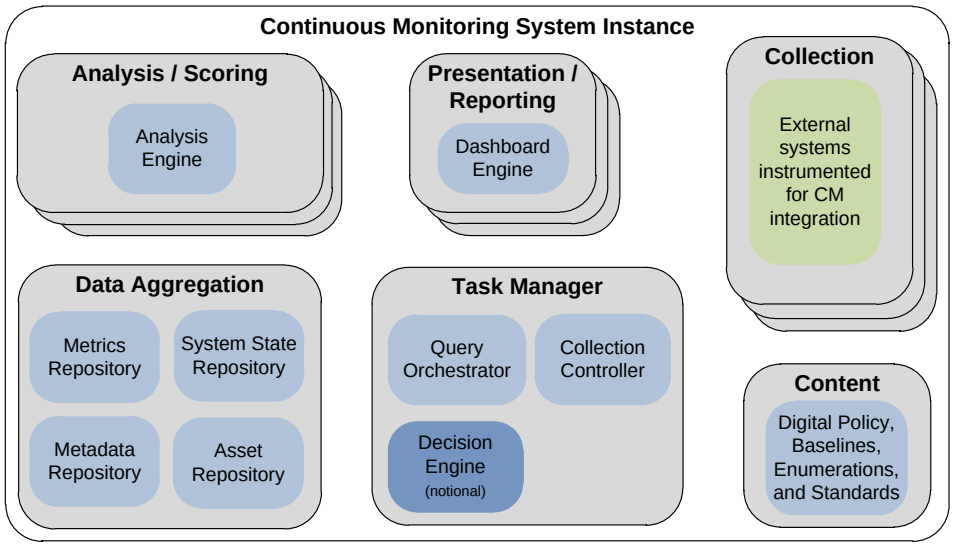


Figure 1. CM System Instance

4.1 Asset Management Data Domain

The Asset Management data domain is important and complex and acts as the foundation for the other domains within security and IT management. The configuration and vulnerability data domains depend on the Asset Management data domain to collect configuration and vulnerability data from assets across an enterprise. It would be impossible to gather and report asset information without first binding to the Asset Management data domain. This reference architecture assumes the functionality of Asset Management as a set of cohesive capabilities, leveraging existing asset management solutions, to feed data into a CM instance, which can then be leveraged by the other data domains. See Figure 2 for a modified architecture diagram showing the relevant subsystems for the Asset Management Data Domain.

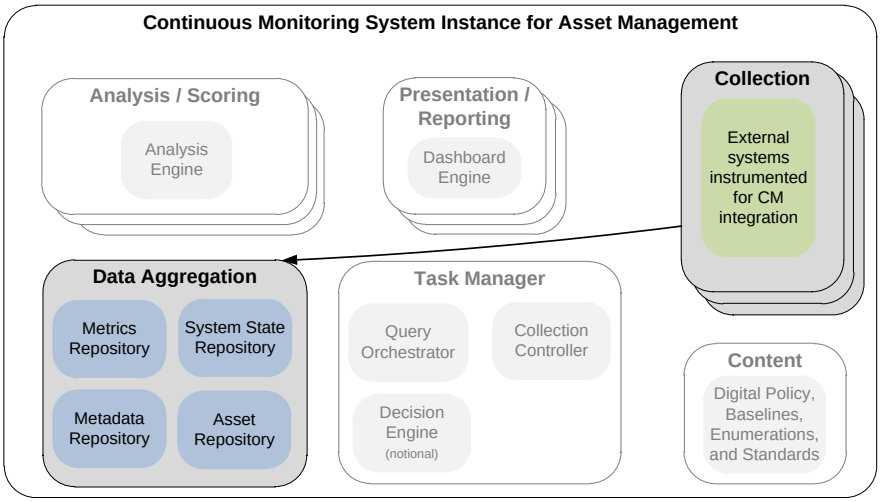


Figure 2. Continuous Monitoring Instance for Asset Management Data Domain

As Figure 2 shows, only the Collection and Data Aggregation subsystems are active in the Asset Management Data Domain. The Collection subsystem will be implemented through the use of existing tools possessing asset management capabilities and able to report on data related to assets. They will be expected to push that data to the Data Aggregation subsystem, which will be responsible for storing and managing the asset data. This data is stored via four components: the System State Repository, the Asset Repository, the Metrics Repository, and the Metadata Repository. In the current state of the reference architecture model as defined in [IR 7756], the other subsystems will not be utilized within the Asset Management data domain. The Collection subsystem is expected to operate independently, and simply push data across the I1 interface as detailed in [IR 7799] Section 5.1. In the long-term, it is expected that asset management capabilities will be developed that can leverage the full capabilities of this reference model, including tasking.

4.2 Configuration Management and Vulnerability Management Data Domains

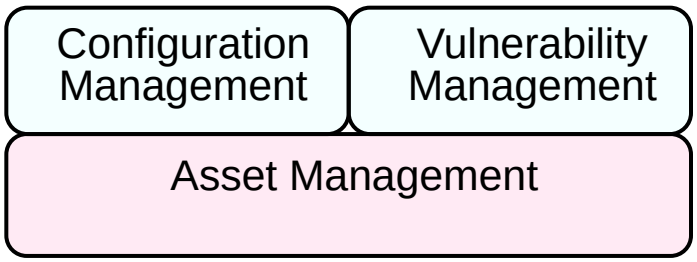
The Configuration and Vulnerability data domains exercise the full capabilities of the CM reference architecture, leveraging the [IR 7799] workflows. The SCAP specification [SP 800-126] is a mature, existing specification and can be utilized for the Configuration and Vulnerability data domains. This better enables complete workflow integration. The entirety of workflows one (WF1) and two (WF2) will be utilized in the Configuration and Vulnerability data domains. Those workflows document both simple and complex interactions that enable the automatic and dynamic collection and processing of data.

5. Layer 2 Dependencies

This section provides an overview of how the CM data domains of asset, configuration, and vulnerability management depend on each other. Subsequent sections will describe how these Layer 2 CM data domains are bound to and supported by specific Layer 1 communication specifications.

Configuration and vulnerability management can be implemented as CM capabilities independently, but both rely on the Asset Management domain. The Configuration and Vulnerability domains report against the assets that are identified by the Asset Management. The Asset Management, on the other hand, does not depend on any given data domain, but acts as a foundation for other domains in security and IT management.

Figure 3 shows the notional relationships between the CM data domains, where configuration and vulnerability management sit on top of, and depend upon, asset management.



6. Layer 1 Dependencies

This section details how the Layer 1 specifications will inform the specific data domains of configuration and vulnerability management. Each requires a naming scheme, as well as a checking and a checklist language. These Layer 1 specifications are required for the CM systems and will be mapped to existing standards in Section 7 of this document. These dependencies allow the specific data domains to be bound properly for interoperability.

Asset management will be largely defined as a black-box system in the near-term, reducing the immediate need for Layer 1 specifications to perform asset management in the CM system. Asset Management has a dependency on digital policy, as well as an asset model.

Configuration management depends on three Layer 1 abstract specification capabilities. The necessary specifications are:

1. **Configuration Naming scheme**:
   A configuration naming scheme provides a common mechanism to refer to configuration information across the CM model.

2. **Configuration Checking language**:
   One or more configuration checking languages MUST be capable of expressing instructions on how to check configuration settings across a wide variety of platforms and applications. Content expressed in these languages SHOULD be able to accept external configuration from a policy checklist language, and it SHOULD report configuration settings on the target.

3. **Configuration Checklist language**:
   One or more languages MUST be capable of expressing rules about which configuration settings to check for. They MUST be capable of reporting compliance results against a variety of platforms, and they MUST be able to use the configuration checking language(s).

Vulnerability management depends on three Layer 1 specifications. The necessary specifications are:

1. **Vulnerability Naming scheme**:
   It is necessary to depend on a vulnerability naming scheme that provides for a common mechanism to refer to vulnerability information across the CM architecture.

2. **Vulnerability Checking language**:
   One or more vulnerability checking languages MUST be capable of expressing instructions on how to check for vulnerabilities across a wide variety of platforms and applications. Content expressed in these languages SHOULD be able to accept external configuration from a policy checklist language, and it SHOULD report discovered vulnerabilities on the target.

3. **Vulnerability Checklist specification**:
   One or more languages MUST be capable of expressing rules that enable one to check for vulnerabilities. The languages MUST be capable of reporting discovered vulnerabilities on a target, and they MUST be able to use the vulnerability checking language(s).

7. Required Layer 1 Specifications

This section details how this specification depends on the usage of the Security Content Automation Protocol (SCAP) revision 2, which is an umbrella specification tying together concrete specifications and languages that implement the Layer 1 dependencies described in Section 6. This section details how existing standards map to and partially or fully meet the needs of the CM System. Specifically, SCAP ties together the following specifications.

- Asset Reporting Format (ARF)
- Common Configuration Enumeration (CCE)
- Common Vulnerability Enumeration (CVE)
- Common Platform Enumeration (CPE)
- Extensible Checklist Configuration Description Format (XCCDF)
- Open Vulnerability and Assessment Language (OVAL)

SCAP defines the necessary interactions between the aforementioned specifications to ensure a cohesive solution for configuration and vulnerability management. Specifically, SCAP defines a configuration capability and a vulnerability capability which are targeted at providing a well-defined format for representing content to check configuration and vulnerability information. CVSS and CCSS are Layer 3 concerns and, therefore, not covered in this specification. The following sections describe the components of SCAP in more detail.

7.1 ARF

ARF is a high-level reporting format for assets. SCAP leverages ARF as the results format for an SCAP scan. SCAP defines various restrictions on the use of ARF to enable interoperability of SCAP results.

7.2 CCE

"CCE provides unique identifiers to system configuration issues in order to facilitate fast and accurate correlation of configuration data across multiple information sources and tools."[4] SCAP leverages CCE as a common naming scheme for referring to configuration settings. This will fill the roles required listed in part one of the configuration dependencies in Section 6 as a Layer 1 configuration naming schema.

7.3 CVE

CVE provides unique identifiers to known vulnerabilities and patch information in order to "enable data exchange between security products and provide a baseline index point for evaluating coverage of tools and services."[5] SCAP leverages CVE as a common naming scheme for referring to vulnerabilities. This will fill the role listed in part one of the vulnerability dependencies in Section 6 as a Layer 1 vulnerability naming scheme.

[4] Language from website: http://cce.mitre.org/
[5] Language from website: http://cve.mitre.org/

7.4 CPE

"CPE is a standardized method of describing and identifying classes of applications, operating systems, and hardware devices present among an enterprise's computing assets. CPE does not identify unique instantiations of products on systems. Rather, CPE identifies abstract classes of products."[6] SCAP leverages CPE to identify applicable platforms for SCAP content.

7.5 OVAL

OVAL is a language for expressing definitions on how to gather state information, check configuration settings, and determine whether vulnerabilities exist on a target system. It is widely adopted within the security automation community. This language fills the need required in part two of both the configuration and vulnerability dependencies listed in Section 6 for a Layer 1 checking language and instructions.

7.6 XCCDF

XCCDF is a specification describing a language to express rules about checking for configuration and system state of a target, as well as vulnerabilities. It includes the capability to reference external checking content that expresses the technical details of how to check for configuration settings or state, as well as the ability to pass parameters to that checking content. XCCDF is flexible to allow for a range of capabilities to express configuration, vulnerability, and patch checklist content. This will fill the role required in part three of both the configuration and vulnerability dependencies listed in Section 6 as a Layer 1 checklist language for scanning configuration, vulnerability, and patch information.

[6] Language from website: http://scap.nist.gov/specifications/cpe/

8. Data Dictionary

This section captures specific data models that were developed to enable the CM reference architecture described in [IR 7799]. Specifically, this section defines the "Asset Data Model", which is necessary to transmit asset information across Interface 1 for the asset management domain. This data model is intended as a stopgap measure until a generic "Asset Data Model" specification can be written, thus filling the need.

This Asset Data Model leverages the Asset Identification [IR 7693] data model for the majority of its attributes. The Asset Identification explicitly states that "Asset Identification elements MUST NOT be used to represent information about an asset unless it is being used to identify that asset."[7] As such, it is inappropriate to directly utilize the Asset Identification data model to capture information about an asset. The model proposed in this section extends Asset Identification and incorporates the Asset Identification elements into an "Asset Data Model" where the intent of the fields can be modified.

For the Asset Model defined in this section, the root XML element is `am:computing-device`. The root element is of XML type `am:computing-device-type`. `am:computing-device-type` extends `ai:computing-device-type` and adds no elements or attributes. The inherited elements and attributes from `ai:computing-device-type` capture information about an asset. The inherited fields are documented in [IR 7693]. In addition to the identifying information elements and attributes inherited from `ai:computing-device-type`, Table 2 documents synthetic identifiers that may also be captured about an asset. In Table 2, the "Resource" column indicates the values to populate for the @resource attribute on ai:synthetic-id, and the "ID Restricted To" column indicates an enumeration of values ("n/a" means there are no restrictions).

Table 2. Synthetic Information for Assets

Resource	ID Restricted To	Description
mil:dod:mac	1, 2, 3	The Mission Assurance Category defined by DoD Instruction 8500.2
mil:dod:conf	PUBLIC, SENSITIVE, CLASSIFIED	The Confidentiality Level defined by DoD Instruction 8500.2
gov:nist:fips:conf	LOW, MODERATE, HIGH	The NIST FIPS 199 confidentiality level
gov:nist:fips:avail	LOW, MODERATE, HIGH	The NIST FIPS 199 availability level
gov:nist:fips:integ	LOW, MODERATE, HIGH	The NIST FIPS 199 integrity level
gov:nist:scap:own_org	n/a	The owning organization of the asset

[7] NIST Interagency Report 7693, Section 5.1, Page 7

Resource	ID Restricted To	Description
gov:nist:scap:admin_org	n/a	The administrative organization of the asset
gov:nist:scap:region	n/a	The region in which the asset is located
gov:nist:scap:location	n/a	The location of the asset
gov:nist:scap:domain	n/a	The network domain of the asset
gov:nist:scap:role	n/a	The role of the asset (e.g., "workstation", "router", etc.)
gov:nist:scap:function	n/a	The application level function of the asset (e.g., "web server")
gov:nist:scap:por	n/a	The Program-of-Record that has authority over the asset
gov:nist:scap:network	n/a	The network to which the asset is connected
gov:nist:scap:os	A CPE	The operating system installed on the asset
gov:nist:scap:admin_poc	n/a	The point of contact for the asset

The following XML snippet is an example of a populated asset model for a single computing asset.

```
<am:computing-device xmlns:ai="http://scap.nist.gov/schema/asset-identification/1.1"
xmlns:am="gov:cm:asset:model">
  <ai:synthetic-id resource="gov:nist:fips:conf" id="HIGH"/>
  <ai:synthetic-id resource="gov:nist:fips:avail" id="MODERATE"/>
  <ai:synthetic-id resource="gov:nist:fips:integ" id="LOW"/>
  <ai:connections>
    <ai:connection>
      <ai:ip-address>
        <ai:ip-v4>192.168.1.1</ai:ip-v4>
      </ai:ip-address>
    </ai:connection>
  </ai:connections>
</am:computing-device>
```

The above snippet indicates that the computing device has a FIPS confidentiality, availability, and integrity level of HIGH, MODERATE, and LOW, respectively, as well as an IP address of 192.168.1.1.

9. Collection Reporting Format Views

This section describes the types of information that will be collected in each of the data domains, as well as the format of that data.

For both Configuration and Vulnerability management domains, the collection subsystem will transport information in the format of an SCAP 1.2 result data stream (described in Section 4.4 of [SP 800-126]). The SCAP result data stream contains information about collection results for SCAP content. The configuration and vulnerability management domains use the result data stream to report information. An SCAP result data stream may contain results for SCAP collection activities for one or more targets. Information may be requested either by a query or by referencing a previously executed query by using the SCAP data stream ID. When requesting an SCAP data stream ID, all of the information related to that data stream (typically the result of a previous query) is returned.

For the Asset Management data domain, data will be reported in the asset data model format described in Section 8. In the future, when a standard data model is defined within the security automation suite of specifications, this specification may be revised to leverage that standard.

10. Continuous Monitoring Instance-Wide Specifications

The following sections detail the asset, configuration, and vulnerability management data domain-specific requirements for CM. Each subsystem MUST use communication payloads as described in the specific subsystem requirements documented in the following sections.

However, this section omits requirements for the Analysis/Scoring subsystem and Presentation/Reporting subsystem because those requirements would focus on the scoring and analysis layer (Layer 3) of the CM specification stack. Therefore, the requirements associated with those two subsystems are beyond the scope of this document.

10.1 Content Subsystem Specification

As documented in the [IR 7799] specification, the Content subsystem must provide the following capabilities:

1. **Content Provisioning**: The subsystem can accept queries for content and provide the requested digital policies and supporting content.

2. **Content Acquisition**: The subsystem can retrieve content from other content repositories.

3. **Content Persistence**: The subsystem can provide an interface to allow content maintenance tools to update the content repository.

4. **Content Propagation**: The subsystem can push content to other content subsystems (e.g., a lower tier CM instance).

5. **Content Configuration Console**: The subsystem can provide a set of controls to enable an administrator to configure the subsystem.

10.1.1 Content Provisioning

For the Configuration and Vulnerability management data domains, the Content subsystem MUST accept queries formed in the accepted SCAP-SDS format (described in Section 3.1 of [SP 800-126]). It MUST respond to the query (as defined in [IR 7799] Section 5.3.1) with a well-formed SCAP source data stream (SCAP-SDS) collection. In the case where a list of SCAP data stream IDs are requested, the response MUST include every SCAP data stream requested, along with the corresponding components. In the case where a list of CPEs is provided, all SCAP-SDSs that have a CPE applicability statement (as defined in [CPE23-LANG]) that match the list SHALL be returned; all SCAP-SDSs that have a CPE in an `<xccdf:platform>` that is EQUAL or SUBSET (as defined in [CPE23-MATCH]) to one of the items in the list MUST also be returned. For the configuration management domain, all SCAP data streams returned SHOULD be of use case “CONFIGURATION”. For the vulnerability management domain, all SCAP data streams returned SHOULD be of use case “VULNERABILITY”.

10.1.2 Content Acquisition

For the configuration and vulnerability management data domains, the content subsystem MUST be able to acquire SCAP 1.2 configuration and vulnerability SCAP-SDS (respectively) as defined in [SP 800-126]. Specifically, when the content subsystem requests content from a different level CM instance, it MUST request an SCAP data stream(s) by one of the following:

- Data stream ID list
- List of CPEs

When retrieving content by ID, the SCAP-SDSs requested will be returned in a SCAP-SDS collection ([SP 800-126] Section 3.1). When requesting content as a list of CPEs, all SCAP-SDSs that have a CPE applicability statement (as defined in [CPE23-LANG]) that match the list will be returned; all SCAP-SDSs that have a CPE in an `<xccdf:platform>` that is EQUAL or SUBSET (as defined in [CPE23-MATCH]) to one of the items in the list will also be returned. In the configuration management data domain, only SCAP data streams marked as use case "CONFIGURATION" should be returned. In the vulnerability management data domain, only SCAP data streams marked as use-case "VULNERABILITY" should be returned.

Future enhancements to this model may support requesting content by other descriptors such as CVEs and CCEs.

SCAP-SDSs may also be pushed to the Content subsystem from content development tools over Interface 2.2.

10.1.3 Content Persistence

For the Configuration and Vulnerability management data domains, the content subsystem MUST be able to store SCAP 1.2 source data stream content. It MUST also store metadata about the content that is used for querying. Specifically, it MUST store the ID of each data stream it persists, and it MUST store all of the CPEs associated with an XCCDF benchmark (via either the `<xccdf:platform>` element or the `<cpe-lang:fact-ref>` element). Storing the metadata associated with a data stream is necessary to enable rapid re-querying of the same content.

10.1.4 Content Propagation

No requirements.

10.1.5 Content Subsystem Configuration

No requirements.

10.2 Collection Subsystem Specification

The Collection subsystem must provide the following capabilities:

1. **Task Receipt**: The Collection subsystem can receive incoming collection tasks for processing and can produce responses describing the task completion status.

2. **Content Retrieval**: The Collection subsystem can retrieve digital policy and supporting content (e.g., from the Content subsystem) needed to fulfill data collection tasks.

3. **Data Retrieval:** The Collection subsystem can retrieve CM data and collect new data as necessary to fulfill data collection tasks.

4. **Data Publication**: The Collection subsystem can publish data gathered in support of data collection tasks (e.g., to the Data Aggregation subsystem).

5. **Collection Console**: The Collection subsystem can provide a console that enables direct creation of data collection tasks and general management of the subsystem configuration.

10.2.1 Task Receipt

For the Configuration and Vulnerability management data domains, the task from the collection controller component will include one or more SCAP 1.2 source data stream IDs or a list of CPEs, a target asset list (represented as a list of asset identification elements), and if utilizing OVAL, will include OVAL directives specifying the level of OVAL results to provide.

10.2.2 Content Retrieval

The Collection subsystem MUST send a request to the Content subsystem to retrieve the content for the task. The query MUST follow the requirements in Section 10.1.2. The expected response SHOULD match what is described in Section 10.1.2.

10.2.3 Data Retrieval

For the Configuration and Vulnerability management data domains, the Collection subsystem MUST collect the information specified in the retrieved SCAP data stream(s) against the target asset list supplied in the request.

10.2.4 Data Publication

For the Configuration and Vulnerability management data domains, the Collection subsystem MUST send the configuration or vulnerability SCAP results data stream(s) to the Data Aggregation subsystem at a level of detail consistent with the task.

For the Asset management domain, the Collection subsystem MUST send asset information to the Data Aggregation subsystem as it is collected.

10.2.5 Collection Console

No requirements.

10.3 Task Management Subsystem Specification – Query Orchestrator

The Query Orchestrator (QO) MUST provide the following capabilities:

1. **Query Receipt and Response**: The QO can receive incoming queries for processing and respond with requested results.

2. **Asset Resolution:** The QO can resolve an asset population descriptor into a specific list of assets.

3. **Query Authorization**: The QO can make policy decisions regarding whether or not to let a query execute and whether human approval is required.

4. **Query Fulfillment**: The QO can coordinate query fulfillment through propagating queries, analysis tasks, and collection tasks.

5. **Analysis Task Propagation**: The QO can derive an analysis task from a query and propagate that task to an Analysis/Scoring subsystem in order to obtain the query results.

6. **Collection Task Propagation**: The QO can derive a data collection task from a query and propagate that task to the Collection controller in order to gather the data needed to support the query.

7. **Query Propagation**: The QO can forward queries to the appropriate CM instances for processing and receive replies containing query results.

8. **Results Publication:** The QO can publish query results for storage.

9. **Query Console**: The QO can implement a console for managing query processing policy and query propagation to other CM instances.

10.3.1 Query Receipt and Response

No requirements.

10.3.2 Asset Resolution

No requirements.

10.3.3 Query Authorization

For the Configuration and Vulnerability management data domains, the QO MUST make a decision to authorize a query to run based on the following information: SCAP data stream IDs and/or list of CPEs, in addition to the parameters specified in [IR 7799].

10.3.4 Query Fulfillment

No requirements.

10.3.5 Analysis Task Propagation

No requirements.

10.3.6 Collection Task Propagation

No requirements.

10.3.7 Query Propagation

No requirements.

10.3.8 Results Publication

No requirements.

10.3.9 Query Console

For the Configuration and Vulnerability management data domains, the user MUST be able to see the SCAP data stream ID and/or list of CPEs when making a decision to approve and/or schedule a query to run.

10.4 Task Management Subsystem Specification - Collection Controller

There are no data domain-specific requirements for the collection controller.

10.5 Data Aggregation Subsystem Specification

The Data Aggregation (DA) subsystem MUST provide the following capabilities:

1. **Bulk Data Storage**: The DA subsystem receives and stores CM data.

2. **Interactive Data Access**: The DA subsystem provides an interactive data access and storage service.

3. **Data Maintenance Console**: The DA subsystem implements a console for administration and maintenance purposes.

10.5.1 Bulk Data Storage

For the Configuration and Vulnerability management data domains, the Data Aggregation subsystem MUST accept and store SCAP 1.2 result data stream reports. In addition, it SHOULD extract metadata from the result data stream(s) to expedite querying of the information.

Specifically, it SHOULD extract the CCE, CVE, and CPE identifiers found in the XCCDF ident elements. It SHOULD associate those identifiers with their respective XCCDF rule results and OVAL definition results. It MAY create a comprehensive metadata model extracting detailed result data from the SCAP results to expedite querying as well.

For the Asset management domain, the Data Aggregation subsystem MUST accept and store asset information received in a format consistent with the model defined in Section 8.

10.5.2 Interactive Data Access

Consumers MUST be able to access both raw data stored as blobs and data extracted from stored content. See Section 10.5.1 for details on what extracted content MUST be accessible.

10.5.3 Data Maintenance Console

No requirements.

Appendix A – References

This appendix contains normative references (used to dictate requirements) and informative references (additional information) used in the publication.

Normative References

[CPE23-LANG] NIST Interagency Report 7698, Common Platform Enumeration: Applicability Language Specification Version 2.3, April 2011. See: http://scap.nist.gov/specifications/cpe/#language

[CPE23-MATCH] NIST Interagency Report 7696, Common Platform Enumeration: Matching Specification Version 2.3, April 2011. See: http://scap.nist.gov/specifications/cpe/#matching

[IR 7693] NIST Interagency Report (IR) 7693 – Specification for Asset Identification 1.1, June 2011. See: http://csrc.nist.gov/publications/PubsNISTIRs.html#NIST-IR-7693

[IR 7756] NIST Interagency Report (IR) 7756 – CAESARS Framework Extension: An Enterprise Continuous Monitoring Technical Reference Architecture (Draft), Feb 2011. See: http://csrc.nist.gov/publications/PubsDrafts.html#NIST-IR-7756

[IR 7799] NIST Interagency Report (IR) 7799 – Continuous Monitoring Reference Model Workflow, Subsystem, and Interface Specifications (Draft), December 2011. See: http://csrc.nist.gov/publications/PubsDrafts.html#NIST-IR-7799

[RFC 2119] Internet Engineering Task Force (IETF) Request for Comment (RFC) 2119: Key words for use in RFCs to Indicate Requirement Levels, March 1997. See: http://www.ietf.org/rfc/rfc2119.txt

[SP 800-126] NIST Special Publication (SP) 800-126 Rev 2 – The Technical Specification for the Security Content Automation Protocol (SCAP) Version 1.2, September 2011. See: http://csrc.nist.gov/publications/PubsSPs.html#800-126-rev2

[SP 800-18] NIST Special Publication (SP) 800-18– Guide for Developing Security Plans for Federal Information Systems, February 2006. See: http://csrc.nist.gov/publications/nistpubs/800-18-Rev1/sp800-18-Rev1-final.pdf

[XML] W3C Recommendation Extensible Markup Language (XML) 1.0 (Fifth Edition), 26 November 2008. See: http://www.w3.org/TR/REC-xml/

[XMLS] W3C Recommendation XML Schema, 28 October 2004. See: http://www.w3.org/XML/Schema.html