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

NIST Interagency Report 7756 (Draft)

**Jointly developed with the
Department of Homeland Security**

CAESARS Framework Extension: An Enterprise Continuous Monitoring Technical Reference Architecture (Draft)

Peter Mell, David Waltermire, Harold Booth, Timothy McBride, Alfred Ouyang

NIST Interagency Report 7756 (Draft)

CAESARS Framework Extension: An Enterprise Continuous Monitoring Technical Reference Architecture (Draft)

Peter Mell, David Waltermire, Harold Booth, Timothy McBride, Alfred Ouyang

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

February 2011

U.S. Department of Commerce

Gary Locke, Secretary

National Institute of Standards and Technology

Dr. Patrick D. Gallagher, 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 7756
46 pages (February 2011)

Acknowledgments

The authors would like to thank the original research team that developed the Department of Homeland Security (DHS) Federal Network Security's seminal work on continuous monitoring architectures. The Continuous Asset Evaluation, Situational Awareness, and Risk Scoring (CAESARS) architecture,[1] created with MITRE support, formed the foundation of this work.

Also, we would like to recognize the following individuals for their participation on the continuous monitoring research team, insightful ideas, and review of this work: David Minge and Peter Sell from the National Security Agency; Valery Feldman, Amit Mannan, Joe Debra, and Zach Ragland from Booz Allen Hamilton; and Mark Crouter from MITRE.

Finally, we would 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 co-chairs:[2] Colonel Michael Jones from the US Army, John Streufert from Department of State (DoS), and Timothy McBride from DHS (also one of the authors).

This publication was authored through the cooperative work of the following organizations:

- National Institute of Standards and Technology (Peter Mell, David Waltermire, and Harold Booth)
- Department of Homeland Security (Timothy McBride)
- MITRE (Alfred Ouyang)

Abstract

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. It also extends CAESARS to allow for large implementations that need a multi-tier architecture. The goal of this document is to facilitate enterprise continuous monitoring by presenting a reference architecture that enables organizations to aggregate collected data from across a diverse set of security tools, analyze that data, perform scoring, enable user queries, and provide overall situational awareness. The architecture design is focused on enabling organizations to realize this capability by leveraging their existing security tools and thus avoiding complicated and resource intensive custom tool integration efforts.

Audience

This publication is intended for those planning to implement, develop products for, or support enterprise continuous monitoring capabilities. The architecture is broadly applicable to diverse networks including industry, civilian government, state government, tribal, and military networks. Expected users of this document include Chief Information Security Officers, Chief Technology Officers, security tool vendors, security tool testing laboratories, security program managers, and enterprise architects.

[1] http://www.dhs.gov/xlibrary/assets/fns-caesars.pdf.
[2] Co-chairs as listed on Office of Management and Budget website https://max.omb.gov/community/display/Egov/Continuous+Monitoring+Working+Group+Members, 1/24/2011.

While not a pre-requisite, greater understanding of the CAESARS framework and our extensions will be gained by also reading the DHS CAESARS architecture.[3]

[3] http://www.dhs.gov/xlibrary/assets/fns-caesars.pdf.

Table of Contents

List of Figures

1. Introduction and Document Overview

1.1 Introduction

The United States Office of Management and Budget (OMB) issued a memo in April 2010 requesting that the Department of State, Department of Justice, and Department of the Treasury coordinate with the Department of Homeland Security (DHS) to evaluate their continuous monitoring (CM) best practices and scale them across the government.[4] As a result of this evaluation, DHS released the *Continuous Asset Evaluation, Situational Awareness and Risk Scoring (CAESARS) Reference Architecture Report* version 1.8.[5] The CAESARS report provides a reference architecture, based on security automation standards, that guides organizations in deploying enterprise CM implementations.

In October 2010, the Federal Chief Information Officer Council's Information Security and Identity Management Committee's (ISIMC) subcommittee on CM saw the need to create a technical initiative to expand upon the CAESARS architecture. Responding to this need, a team of researchers from the National Security Agency's (NSA) Information Assurance Directorate, the DHS Federal Network Security CAESARS team, and the National Institute of Standards and Technology's (NIST) Information Technology Laboratory worked together. This publication is one of the outcomes of this joint research effort in support of the ISIMC's CM subcommittee.

The NSA involvement ensured the architecture's applicability to U.S. military and national security systems with consideration for integrating with existing Department of Defense programs such as Computer Network Defense. DHS' participation ensured applicability to its Cyberscope program[6] as well as consistency with the CAESARS vision. NIST's participation led to an architectural design that could support industry as well as government and a design well integrated with existing and emerging security automation standards.

This report describes the resulting CAESARS Framework Extension (FE), building upon the CAESARS architecture to make it more broadly applicable to the entire U.S. government including the Department of Defense, Intelligence Community, and civilian agencies. This work was also designed to be applicable to industry, state governments, and tribal networks through using a flexible architecture able to handle diverse customers and uses. In part, this was accomplished by extending CAESARS to allow for large implementations that need a multitier CM architecture. In addition, much work was done in enabling additional functionality, providing more granularity within subsystem specifications, and further leveraging security automation standards (e.g., for communication payloads and application interfaces).

The end goal of CAESARS FE is to enable enterprise CM by presenting a technical reference architecture that allows organizations to aggregate collected data from across a diverse set of security tools, analyze that data, perform scoring, enable user queries, and provide overall situational awareness. The focus is on primarily supporting cyber operations with compliance reporting as a by-product of actual security monitoring and improvement. This design is focused on enabling organizations to realize this capability by leveraging their existing security tools and minimizing custom tool integration efforts.

This report provides the high-level design for meeting this goal. However, eventual success will require additional subsystem specifications, interface descriptions, and communication protocols. Achieving this

[4] The Department of Homeland Security's Continuous Asset Evaluation, Situational Awareness and Risk Scoring Reference Architecture Report, version 1.8, page xi (http://www.dhs.gov/xlibrary/assets/fns-caesars.pdf).
[5] http://www.dhs.gov/xlibrary/assets/fns-caesars.pdf.
[6] http://www.whitehouse.gov/sites/default/files/omb/assets/memoranda_2010/m10-15.pdf .

goal will also require the input and participation of security tool vendors and their customers as we develop and finalize these lower level specifications.

Security tool vendor participation is critical because their security sensors and controllers will need to be modestly instrumented to support the architecture and related security automation standards. Fortunately, much of this has already been accomplished through the NIST Security Content Automation Protocol (SCAP) Validation Program.[7] In addition, new functionality in data aggregation, analysis, and event management products will be needed. Fortunately again, many existing products contain much of the needed functionality and should require only modest instrumentation to support the architecture.

If successful, CAESARS FE and the security products that support it will enable organizations to compose diverse security products together into a hierarchical data aggregation architecture that supports a large variety of CM consumers from both the security disciplines and general information technology (IT) management domains. The challenge will be to minimally define the required functionality so that security tool vendors can cost-effectively participate while ensuring a necessary level of interoperability between vendor products. This will require ongoing discussions, collaboration, and development within government and industry.

1.2 Document Overview

This report begins in Section 2 with a discussion of the definition of CM. From this definition, essential technical characteristics are derived that support the specific CM enterprise architecture (EA) view presented in Section 3. This EA view reveals the need for a set of goals that help in the review of the capabilities and limitations of the original CAESARS architecture in Section 4. Section 5 presents the framework extension to CAESARS that expands upon the original functionality and addresses the limitations. Section 6 discusses use cases and related architectural workflow. Section 7 describes future work that needs to be done. Appendix A summarizes the acronyms used in this report. Appendix B provides details on the use cases and associated workflow that exercise the CAESARS FE architecture to achieve specific goals. These use cases assist in conveying an understanding of the purpose of the various subsystems and lower level components.

[7] NIST Security Content Automation Protocol Validation Program, http://scap.nist.gov/validation/index.html.

2. Defining and Scoping Continuous Security Monitoring

This section discusses several definitions of continuous monitoring (CM) and extracts from these definitions essential technical characteristics for CM implementations. It also discusses the scope of CM to focus the reader on what will be provided by this architecture and what will need to be provided by existing IT operations.

2.1 Definitions

CM can be described in its broadest sense by the following text:

> *Continuous monitoring is ongoing observance with intent to provide warning. A continuous monitoring capability is the ongoing observance and analysis of the operational states of systems to provide decision support regarding situational awareness and deviations from expectations.*

This definition applies to both Cybersecurity and general IT domains (e.g., network management). In this publication, we focus on the Cybersecurity domains but the architecture presented is applicable to the general IT domains as well. Because of the effort and expense involved in creating an effective CM solution, such solutions should be leveraged for as many uses as possible. We strive in this publication to support use across both Cybersecurity and IT management domains.

To focus on Cybersecurity, we now redefine CM in the context of security risk management using the draft NIST Special Publication (SP) 800-137 definition:

> *Information security continuous monitoring is defined as maintaining ongoing awareness of information security, vulnerabilities, and threats to support organizational risk management decisions.*

For purposes of designing a technical reference architecture in this publication, we provide a more granular and process-focused description. From this we extract essential characteristics for CM implementations.

> *Continuous security monitoring is a risk management approach to Cybersecurity that maintains an accurate picture of an organization's security risk posture, provides visibility into assets, and leverages use of automated data feeds to quantify risk, ensure effectiveness of security controls, and enable prioritization of remedies.*

The essential characteristics for CM that can be derived from this definition are the following:

- Maintains an accurate picture of an organization's security risk posture
- Provides visibility into assets
- Leverages automated data feeds
- Quantifies risk
- Ensures continued effectiveness of security controls
- Informs automated or human-assisted implementation of remedies
- Enables prioritization of remedies
- Identifies deviations from expected results.

These characteristics support the EA view of CM provided in Section 3.

2.2 Scoping and External System Interfaces

It is the intent of the architecture presented in this publication to clearly scope and bound our technical CM solution. Thus we make a delineation between what capabilities a technical CM implementation provides (e.g., providing analysis of events) and the *external* systems to which it interfaces. Therefore, there are multiple external systems that will interface with any CM capability. For example, CM implementations must interface with asset management systems for a CM capability to determine what assets exist.

These external systems and technologies can be categorized to include at least 11 domains (see Figure 1) that could interface with a CM capability.[8]

1) Vulnerability Management
2) Patch Management
3) Event Management
4) Incident Management
5) Malware Detection
6) Asset Management
7) Configuration Management
8) Network Management
9) License Management
10) Information Management
11) Software Assurance

Figure 1. Continuous Monitoring Data Domains

Although the tools supporting these domains are not a core part of the technical CM capability, they need to be instrumented to interface with CM solutions. For this reason, they are included in our architecture but are clearly shown as external entities so that we can describe the needed interface requirements.

[8] NIST Special Publication 800-137, Information Security Continuous Monitoring for Federal Information Systems and Organizations, Appendix D.1. http://csrc.nist.gov/publications.

3. Enterprise Architecture View for Continuous Monitoring

This section presents an enterprise architecture (EA) view for CM with the purpose of illuminating high-level goals for the CAESARS FE technical reference architecture and associated implementations.

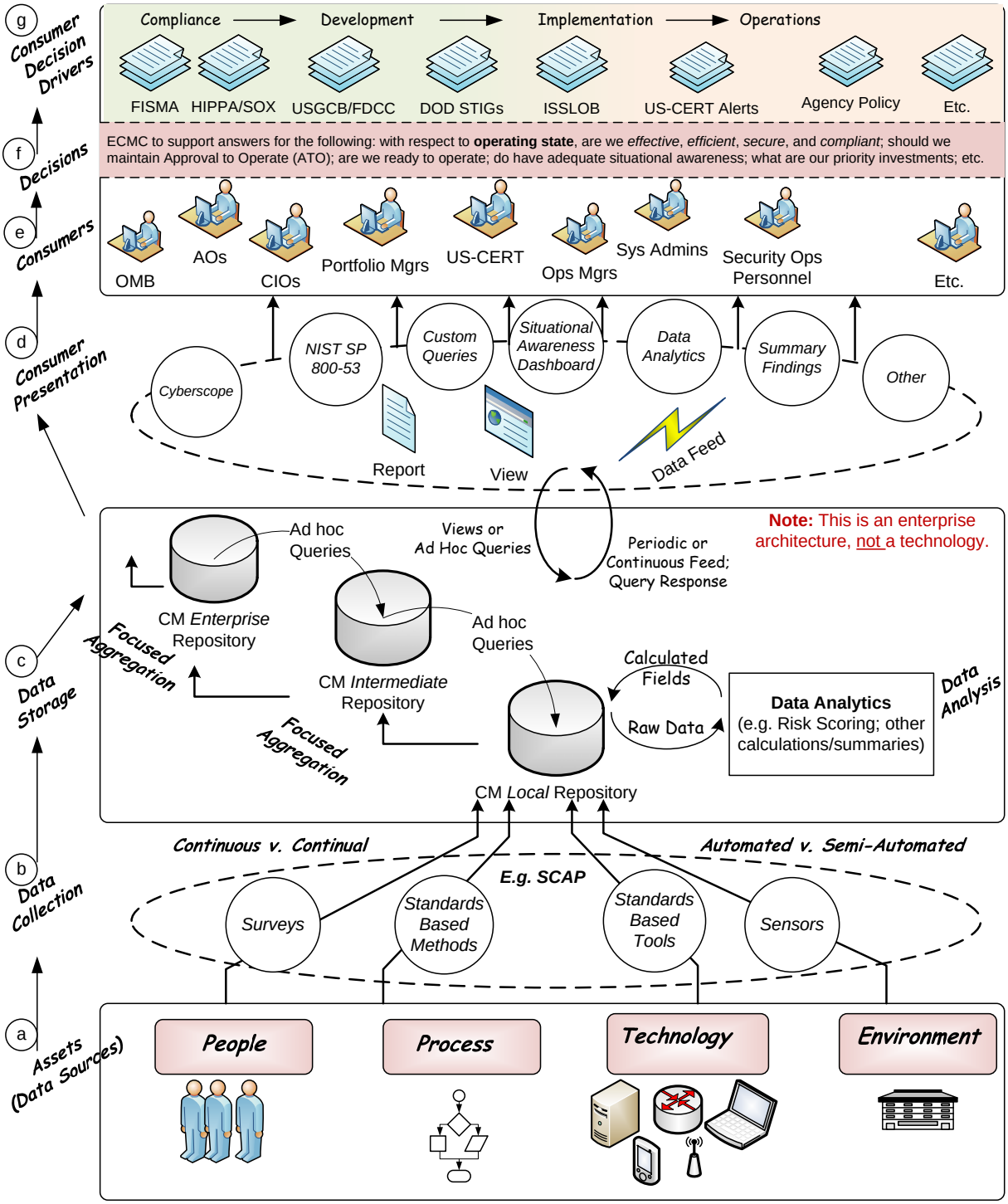

Figure 2. Enterprise Architecture View of Continuous Monitoring

Figure 2 displays the EA view of the enterprise continuous monitoring capability (ECMC). This is not a technical architecture but is illustrative of higher level goals that need to be implemented by the CAESARS FE technical architecture. The following subsections describe each level of the EA view and make general observations regarding the required technical architecture. They start with an explanation of how data sources feed data collection activities and then show how that data is stored, analyzed, and presented to consumers who make decisions based on a variety of drivers.

a. Data Sources

The data sources for CM include the categories of people, process, technology, and environment. While many CM implementations may focus initially on technology, a CM technical architecture should be general enough to allow inclusion of the other categories. The people, process, and environment data types do not always lend themselves to fully automated data collection efforts and in most cases will require some human data collection effort.

b. Data Collection

A variety of methods, both automated and manual, can be used to collect data. Focus should be on utilizing standards-based methods within tools for performing data collection to reduce integration costs, enable plug-and-play compatibility of subsystems, and enable CM implementations to incorporate diverse sets of security implementations. However, not all data feeds are currently standardized and the technical architecture needs to allow for acceptance and processing of proprietary data payloads. Human generated data (e.g., from user surveys or from security compliance documentation) should be collected using mechanisms that harness automation and that leverage standardized methods. Also, the appropriate frequency for data collection needs to be determined for each data feed. Some data feeds will be truly continual (always on) while others will be continuous (collected periodically at some set interval). While the frequency interval is usually time based, in some cases it may be more appropriate for it to be event driven.

c. Data Storage and Analysis

The collected data will initially reside at a local repository near the point of collection and then may be aggregated at higher tiers in the organization. Having CM data available at each tier enables users at that tier to have an appropriately abstracted view into the organization's security posture. Normally, users at each tier need only an abstracted view of the lower level CM data, and thus it is not necessary to duplicate all data up through each tier. Replicating all low-level security data at multiple levels within an organization could pose an increased security risk, along with authorization and scalability challenges. For this reason, the CM EA view shows a "focused aggregation" occurring when transferring CM data from one repository to a higher level repository. Only the data needed by the next higher tier is transferred up and duplicated. Determining these necessary "predefined views" is an important step in any CM implementation. Ideally, the majority of the CM data and the most sensitive of that data will stay at the local repository level. At this level, the information is closest to the authoritative sources (i.e., data collectors), allows for fine-grained access control, is the timeliest copy of the data, and poses the least aggregation risk.

This model for performing data aggregation using predefined views has been used successfully for CM implementations. However, it is likely that users at higher tiers will have an operational need to occasionally query data that is outside of the available predefined view. In such cases, organizations may be tempted to aggregate more and more of the data at all tiers. At an extreme, they may attempt to aggregate and duplicate all CM data at all tiers. This may result in network bandwidth and data storage challenges while presenting additional security risks. To alleviate this need, the EA CM view enables users at one tier to issue operational, or "ad hoc", queries that are propagated down to lower tiers for data retrieval. If the requested data is not available in local repositories, the operational query from a higher level tier may trigger additional data collection at the lower tiers. The technical CM architecture will need

to allow for this operational querying while putting into place the necessary task management systems so that such operational queries are reviewed, approved, and do not result in a degradation of the local network or systems.

A final consideration on data storage is that the same technical architecture (e.g., interfaces, protocols, and payloads) should be used for data aggregation regardless of which tier is communicating. This avoids having to create different CM aggregation solutions for differing tiers.

d. Consumer Presentation
Each tier within the data storage layer will provide a view of the data to consumers. The presentation layer needs to be flexible enough to satisfy diverse data display needs because the CM implementation needs to support many types of consumers. Primarily the CM implementation should support the operational mission in helping to secure an organization (likely through situational awareness dashboards). It will also need to support compliance reporting, executive-level reporting, and reporting for non-security use cases.

e, f, and g: Consumers, Decisions, and Consumer Decision Drivers
There are many types of consumers that need CM data ranging from system administrators to the organization Chief Information Officer to possibly external compliance or auditing entities. These consumers need to make decisions (especially those regarding effectiveness, efficiency, security, and compliance) based on a set of drivers. The CM architecture must provide them the necessary information to make these decisions.

4. Foundational Work

DHS has conducted seminal CM work published as the *Continuous Asset Evaluation, Situational Awareness and Risk Scoring (CAESARS) Reference Architecture Report.*[9] CAESARS was the result of a DHS evaluation of successful CM implementations within the civilian government: Department of State, Department of Justice, and Department of the Treasury. DHS found commonality and strengths in the approaches of these civilian agency custom solutions and used this as the basis for creating the CAESARS reference architecture. The CAESARS architecture fulfills many, but not all, of the goals of the CM EA view.

4.1 Overview of the CAESARS Reference Architecture

CAESARS enables organizations to implement a single CM instance that consists of four subsystems: Sensor, Database, Presentation/Reporting, and Analysis/Risk Scoring. All subsystems, except the Database Subsystem, may contain multiple tools providing independent observation or analysis. A single database is used to aggregate monitoring data from the Sensor Subsystem and all its distinct sensor products and their instantiations throughout the enterprise. This central database is also used by the Presentation/Reporting and Analysis/Risk Subsystems as their source of monitoring data. An enterprise service bus (ESB) is used for all inter-subsystem communication.

A review of the subsystem descriptions follows as well as Figure 3[10] showing a "contextual description of the CAESARS system." All quotations in this subsection are taken from the CAESARS publication version 1.8[11] published in September 2010.

[9] http://www.dhs.gov/xlibrary/assets/fns-caesars.pdf.
[10] Diagram is from CAESARS version 1.8, page 11.
[11] http://www.dhs.gov/xlibrary/assets/fns-caesars.pdf.

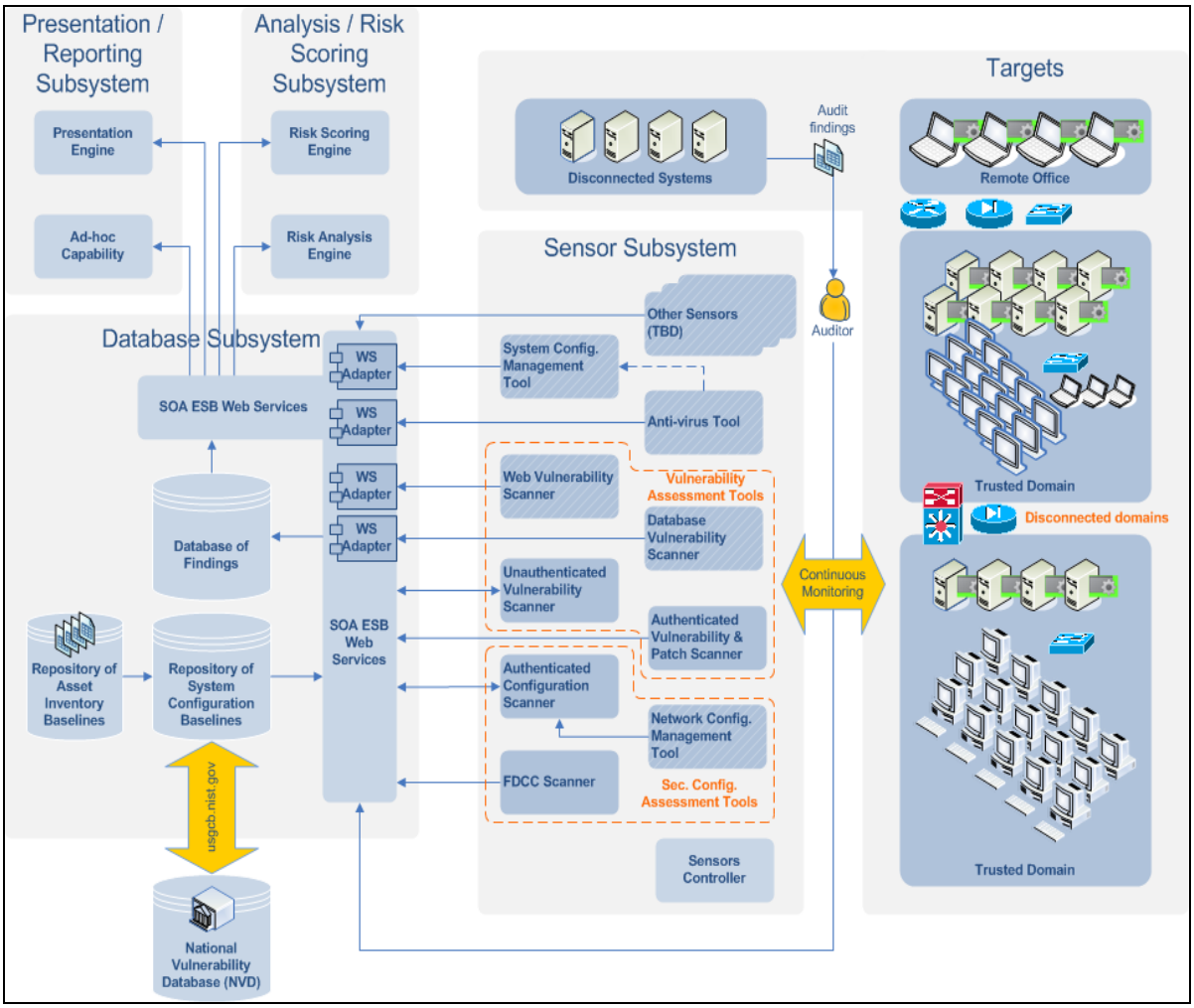

Figure 3. Contextual Description of the CAESARS System

4.1.1 Sensor Subsystem

"The Sensor Subsystem includes the totality of the IT assets that are the object of CAESARS' monitoring activities. It includes all platforms upon which CAESARS is expected to report, including end-user devices, database servers, network servers, and security appliances." Figure 3 shows nine sensor types, and it is important to note that DHS has ongoing government-wide procurements for these product types within its Information Systems Security Line of Business.[12]

4.1.2 Database Subsystem

"The purpose of the CAESARS Database is to serve as the central repository of all CAESARS data, including both raw data that is reported from the sensors on the individual platforms of the Sensor Subsystem and data that is developed within the CAESARS Analysis/Risk Scoring Subsystem as a result of 'cleansing', pre-processing, analysis, and scoring processes. . . . [The CAESARS database] also includes any tools that are required by the CAESARS Database/Repository to perform data pull operations from the Sensor Subsystem platforms." Of note is that the Database Subsystem

[12] http://www.us-cert.gov/GFIRST/presentations/Information_Systems_Security_Line_of_Business_ISSLOB_Overview.pdf.

encompasses the “repository of configuration baselines” and thus it contains machine-readable descriptions of the required baseline security posture of the organization’s systems as well as data on known vulnerabilities and their severity (e.g., from the National Vulnerability Database [NVD]). It also encompasses a “repository of asset inventory baselines,” although this asset inventory database is not described in much detail.

4.1.3 Analysis/Risk Scoring Subsystem

“The CAESARS Analysis/Risk Scoring Subsystem [consists] of multiple analytic tools, possibly querying the same or different portions of the database [subsystem], with either local (region- or site-specific) or enterprise-wide perspectives, yet still provide confidence that no single type of analysis will influence or skew any other.” It is the CAESARS’ architecture of independent plug-and-play components that enables this capability to allow a variety of risk scoring schemes and tools to be simultaneously deployed without any of them interfering with the others.

4.1.4 Presentation/Reporting Subsystem

“The CAESARS Presentation and Reporting Subsystem can include multiple presentation tools, with either local or enterprise-wide perspectives, yet still provide confidence that no one type of presentation will influence or skew any other.” This enables a variety of display schemes serving diverse user types to be simultaneously deployed without any one of them interfering with the others.

4.2 Limitations of the CAESARS Reference Architecture

The CAESARS reference architecture provides a strong foundation on which to build a CM technical architecture that meets our CM EA goals. However, it has limitations in several areas that are required to evolve CAESARS into providing stronger capabilities.

4.2.1 Lack of Interface Specifications

CAESARS does not specify the machine-level interfaces that subsystems can use to communicate with other subsystems. This limits the plug-and-play capabilities within CAESARS and requires each implementation of CAESARS to undergo custom integration efforts.

4.2.2 Reliance on an Enterprise Service Bus

CAESARS specifies using an Enterprise Service Bus (ESB) for communication between subsystems. ESBs may not be an optimal communication mechanism for some types of communication and may not be appropriate for inclusion within certain vendors’ products.

4.2.3 Incomplete Communication Payload Specifications

CAESARS does explore payload specification for the Sensor Subsystem to communicate to the Database Subsystem. However, it does not provide a full specification for all sensor types that can be used by vendors to build CAESARS-compatible security tools. Furthermore, CAESARS does not specify the communication payloads between the other subsystems.

4.2.4 Lack of Specifications Describing Subsystem Capabilities

CAESARS describes each subsystem in general but does not provide detailed specifications. Such specifications are needed to support procurement activities, enable vendors to instrument their tools to support subsystem capabilities, and enable development of product testing methodologies in support of product validation programs (e.g., Security Content Automation Protocol validation[13]).

4.2.5 Lack of a Multi-CM Instance Capability

CAESARS is designed as a monolithic capability whereby each organization is to create a single CM instance. Because CAESARS also requires one of each subsystem, this means that all organizational security data will need to be duplicated within a single central CM database. This runs counter to the structure of many organizations (especially federal government ones) that tend toward a decentralized approach to IT management. This also runs counter to the EA CM goals whereby one wants to perform focused aggregation and to leave the most fresh and timely CM data at the leaf nodes of a hierarchical tiered structure. Thus, most large organizations will need the ability to have multiple CM instances.

4.2.6 Lack of Multi-Subsystem Instance Capability

CAESARS specifies that a CM instance have only one of each type of subsystem. However, it also allows for and encourages a variety of independent security tools within each subsystem. For example, multiple distinct presentation tools are allowed within the Presentation/Reporting Subsystem. This adds complexity because there is no discussion as to how the different tools within a subsystem communicate with and within the subsystem itself. This complexity could be eliminated by allowing multiple instances of a subsystem type and then having a separate instance for each relevant security tool (e.g., presentation system). Although users of CAESARS could avoid this problem by having only one vendor tool within each subsystem, such an approach may limit one's ability to have full coverage of platforms and security features. Having multiple tools (especially security sensors) will increase trust in the collection and the scope of what is collected and allow for a deeper analysis of the data.

4.2.7 CM Database Integration with Security Baseline Content

CAESARS integrates the "repository of system configuration baselines" and the "database of findings" within the Database Subsystem. A variety of vendor tools exist that implement one but not the other of those capabilities. To facilitate vendor adoption of CAESARS it may be advantageous to separate the configuration baselines and database findings into multiple subsystems.

In addition, different parts of an organization may need to customize policies on security posture to fit their environment and unique mission. This is usually managed vertically with policies pushed down from above and modified at the lower levels. Having a single repository of baselines within a single Database Subsystem makes it more difficult to allow customization of the expected security posture.

[13] http://scap.nist.gov/validation/index.html.

4.2.8 Lack of Detail on the Required Asset Inventory

CAESARS provides very little detail on how an asset inventory is to be maintained and how it relates to the repository of configuration baselines.

4.2.9 Requirement for Risk Measurement

CAESARS requires that the Analysis/Risk Scoring Subsystem measure security risk. According to the NIST risk management publication SP 800-37 revision 1[14], risk is defined as follows:

> "A measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of: (i) the adverse impacts that would arise if the circumstance or event occurs; and (ii) the likelihood of occurrence."

Given the requirement to calculate the likelihood of an event and the requirement to calculate the extent to which an adverse impact will harm an organization, it is difficult to measure risk and CM solutions often do not have the necessary inputs. Thus, CM architectures should allow for a more simple focus on measuring the effectiveness of the security posture or controls[15] while not eliminating the possibility of a CM system also calculating risk (if the necessary inputs are available).

[14] http://csrc.nist.gov/publications/nistpubs/800-37-rev1/sp800-37-rev1-final.pdf.

[15] These lower level security effectiveness measurements can then be used as (necessary but not sufficient) inputs to a risk measurement capability.

5. CAESARS Framework Extension

The CAESARS Framework Extension (FE) is a technical reference architecture for enterprise continuous monitoring (CM) that builds upon the Department of Homeland Security's Continuous Asset Evaluation Situational Awareness and Risk Scoring (CAESARS) reference architecture. Most of the CAESARS Subsystems remain in CAESARS FE, but modest revisions have been made to the higher level architecture to provide enhanced functionality and allow multi-tier CM implementations.

This section presents the CAESARS FE architecture, subsystem overview, subsystem interface overview, and multi-tier capability. As with CAESARS, CAESARS FE can be used to support operational security needs as well as compliance evaluation and reporting. Unlike CAESARS, CAESARS FE is designed as a data agnostic architecture allowing collection, aggregation, analysis, presentation, and reporting on a variety of IT areas (both security and general IT management). While the architecture itself is data type agnostic, CAESARS FE implementations will be focused on specific data domains. Requirements for those data domains will be addressed with follow on research (see section 7 for more details).

5.1 Variations on the CAESARS Architecture

CAESARS FE is designed to overcome the previously described limitations of the CAESARS architecture. To accomplish this, CAESARS FE provides the following capabilities:

- Enables a variety of subsystem interfaces to be defined (removing the required ESB from within the CAESARS Database Subsystem) (see 4.2.1 and 4.2.2)
- Provides detailed subsystem communications payload specifications (see 4.2.3)
- Provides detailed specifications for each subsystem that enable tool development and support product validation program or agency procurement (e.g., DHS Information Systems Security Line of Business) (see 4.2.4)
- Allows for the instantiation of multiple CM instances (see 4.2.5)
- Allows for the construction of a CM tiered hierarchical architecture (see 4.2.5)
- Allows for the existence of multiple instances of individual subsystems (see 4.2.6)
- Creates a separate subsystem for a Task Manager (created by extracting the "sensors controller" from the CAESARS Sensor Subsystem, 4.2.6)
- Creates a separate subsystem for security content server (extracting the CAESARS "repository of configuration baselines" from its Database Subsystem) (see 4.2.7)
- Further defines specifications for the asset inventory database (see 4.2.8)
- Provides flexibility to perform general enterprise measurement as opposed to being restricted to true risk measurement (see 4.2.9).

5.2 Subsystem Overview

CAESARS FE contains six distinct subsystems that together orchestrate a CM solution. These six encompass the Collection Subsystems (e.g., sensors) as well as the Situational Awareness Capability (SAC) and Content Subsystem (i.e., security baseline and benchmark repositories) (see Figure 4). We chose to include the Collection Subsystems as part of CAESARS FE (as opposed to just identifying them as integration points) because it will be necessary to instrument those systems to enable them to provide the necessary standards-based data. It is expected that the majority of the CAESARS FE subsystem requirements will be focused on the Situational Awareness Capability where the majority of the CM functionality is contained.

A goal of CAESARS FE is to enable organizations to create a CM solution through harnessing best-of-breed tools from diverse vendors. Thus, different vendor- and organization-developed tools can fulfill

different subsystems that can then be combined to form a CAESARS FE based solution. Custom integration is minimized except in the case of any necessary proprietary data feeds, and in such cases, the integration costs are reduced through the provision of standard interfaces and Extensible Markup Language (XML) protocols that can "wrap" proprietary data.

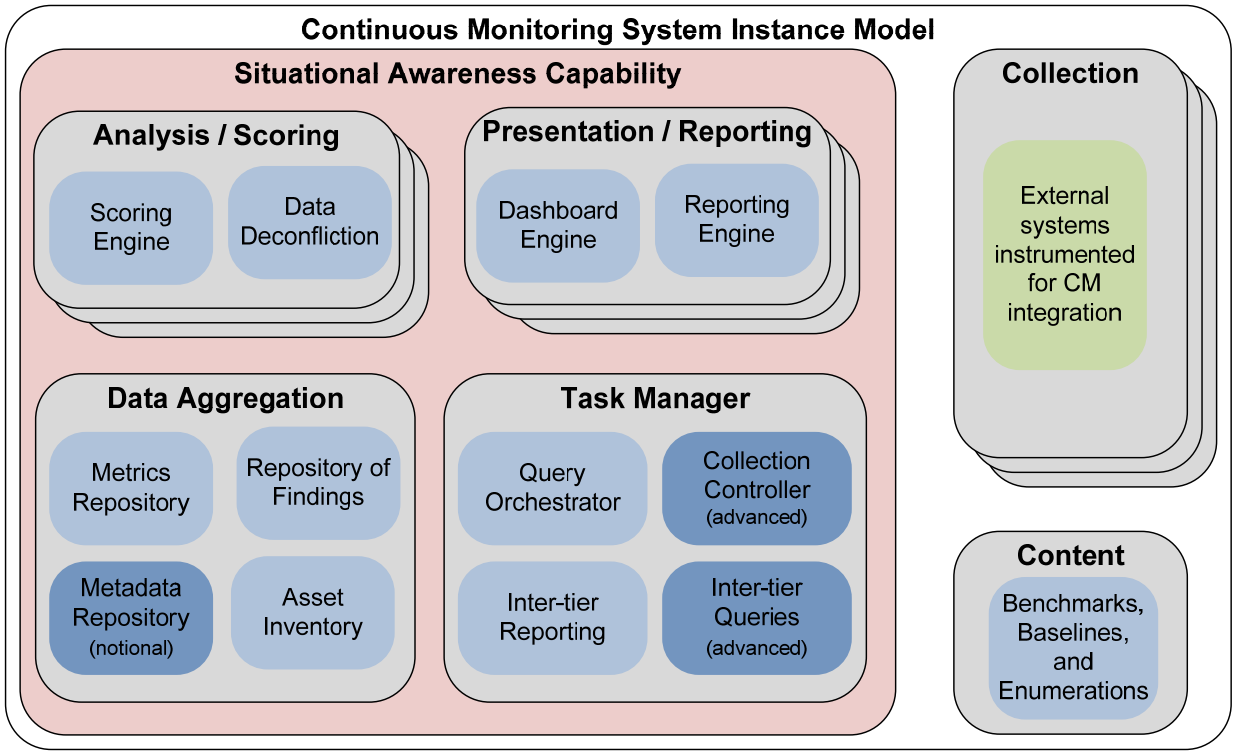

Figure 4. CAESARS Framework Extension Subsystems

The CAESARS FE Subsystems are summarized below along with information on how many of each subsystem may exist within a single CM instance. Future publications will provide detailed specifications for each subsystem.

5.2.1 Presentation/Reporting Subsystem

Presentation/Reporting Subsystem (one or more per CM instance): This is similar to the CAESARS Subsystem of the same name. However, the CAESARS presentation engine has been augmented with dashboard and reporting engine components. These components may provide a variety of views (e.g., XML feeds) to support a wide range of customers.

Note, the CAESARS ad hoc query capability is now implemented within the CAESARS FE Task Manager (see Section 5.2.6).

Dashboard Engine

As described in CAESARS, the CAESARS FE dashboard engine is the primary user interface for the CM solution. It collects the data to be displayed by issuing queries to the Task Manager. The Task Manager's query orchestrator (QO) then determines the best way to satisfy the request (e.g., by retrieving data from the Data Aggregation Subsystem or engaging the collection controller [CC] to gather the needed data).

Reporting Engine

The purpose of the CAESARS FE reporting engine is to provide report-based situational awareness information. The reporting engine interfaces with the Task Manager to retrieve data from the Data Aggregation Subsystem, or the collection, and generate the security posture reports.

5.2.2 Analysis/Scoring Subsystem

Analysis/Scoring Subsystem (one or more per CM instance): This is similar to the CAESARS Subsystem of the same name. It differs in that it provides generalized scoring services as opposed to focusing only on "risk" measurement. This does not preclude using this subsystem for risk measurement because this type of measurement is a subset of the more general concept of scoring. For security-focused CM implementations, risks scoring should be an objective.[16] Another minor difference is that the CAESARS analysis engine has been renamed the "data deconfliction component" to highlight its data cleansing function and to make it clear that this component does not perform measurement itself.

Scoring Engine and Data Deconfliction
This data deconfliction component retrieves raw data from the repository of findings to create a "cleansed" copy, which is then used by the scoring engine to create scores that will be stored within the Data Aggregation Subsystem's metrics repository. This metrics data will be tagged with the name of the Scoring Subsystem that created it. This approach enables the CM implementation to use multiple Scoring Subsystems that use different data deconfliction algorithms and different scoring equations without any conflicting manipulation of the data.

5.2.3 Data Aggregation Subsystem

Data Aggregation Subsystem (one per CM instance): This contains the "database of findings" portion of the CAESARS Database Subsystem. In addition, it contains asset inventory results, a metadata repository, and a metrics repository that stores security measurements from one or more scoring subsystems. Unlike CAESARS, this subsystem does not store organizational baselines or benchmarks. In CAESARS FE these are stored in the Content Subsystem.

Repository of Findings
The repository of findings contains the raw data that is provided by the Collection Subsystems. While not necessary, some preprocessing of the data may occur at this point to improve the quality of the data. If done, this preprocessing will alter the analysis of all Scoring Subsystems. For this reason, the primary data deconfliction may normally occur within the Scoring Subsystem's data deconfliction component so that each Scoring Subsystem can use different approaches and algorithms.

Asset Inventory
The asset inventory stores standards-based representations of asset data retrieved from the Collection Subsystems. Thus, the asset inventory component itself is not an asset inventory system but simply an aggregation database. Collection Subsystems specializing in asset inventories may feed the asset inventory, but it can also be fed by security tools that perform asset identification.

Metrics Repository
This database stores the output from the various Scoring Subsystems tagged with the name of the subsystem that created it. This enables multiple Scoring Subsystems to calculate different or overlapping metrics without one corrupting the other's data.

[16] True risk scoring can be difficult to achieve using the NIST SP 800-37 revision 1 definition, and many "risk scoring" methodologies do not demonstrate a correlation to true risk measurement. Instead, they usually measure the state of a collection of security capabilities or controls.

Metadata Repository (notional)
The metadata repository is a notional component that may eventually store higher level information about the raw data in the repository of findings or about the aggregated asset inventory data. This could include a variety of information including relationships between assets information (e.g., what assets are in what system) or the relative accuracy of different collection systems for certain metrics. If this higher level data view was standardized, it could someday enable data agnostic implementations of the Analysis/Scoring and Presentation/Reporting Subsystems. These concepts will be explored in future versions of CAESARS FE.

5.2.4 Collection Subsystem

Collection Subsystem (zero or more per CM instance): This is similar to the CAESARS "Sensor Subsystem" except that each instance of a Collection Subsystem contains a single vendor's solution (which may comprise multiple tools) that may map to one or more of the CAESARS sensor types. Although the Collection Subsystem is included in the CAESARS FE architecture, only modest instrumentation of existing tools will be required for the tools to participate. This instrumentation will primarily revolve around the ability to output particular standards-based data types (e.g., product names) and use CAESARS FE defined interfaces. Thus, the tools can be viewed as external systems that are essential to, and interface with, the CM architecture. CAESARS FE does not specify or replace the diverse security and management tools already provided by vendors and used by organizations (e.g., asset management, configuration management, and vulnerability management). It leverages those tools to feed essential data to the CM architecture.

Another change is that the CAESARS sensors controller has been moved to the CAESARS FE Task Manager Subsystem and renamed the "collection controller."

In the case of multi-tier CM implementations, if a CM instance does not have a Collection Subsystem, it will rely solely on data provided from lower level CM instances.

5.2.5 Content Subsystem

Content Subsystem (zero or one per CM instance): This is similar to the CAESARS "repository of system configuration baselines" within the Database Subsystem except that its scope has been expanded to include non-security information.

The Content Subsystem is an essential part of the CM architecture that contains *organizational policy* on the expected or required state of systems (possibly tailored to local system policies). It may also contain *supporting data elements* used within state calculations (usually retrieved from external providers). Organizational policy types of data include benchmarks (e.g., organization-wide security configuration policy) and baselines (e.g., customized security configuration policy for specific systems). Supporting data elements include enumerations of data types and associated scores (e.g., product names, vulnerability names, and vulnerability impact scores). The purpose of having this data within a CM architecture is to enable measurement of the system against a required state. Another purpose is to enable measurement using normalized data elements so that multiple Collection Subsystems report on the state of systems using comparable languages.

The following are examples of typical (but not required) types of organizational policy included within a Content Subsystem:

- Software asset baselines (allowed software and required versions)
- Security configuration benchmarks (e.g., Federal Desktop Core Configuration)

- Required patches
- Lists of authorized software
- Required network port configurations.

The following are examples of typical (but not required) types of supporting data elements included within a Content Subsystem:

- Lists of known vulnerabilities and their impact scores
- Lists of known configuration issues and their impact scores
- Lists of asset names that may be applicable (not the list of actual assets in a system).

To the greatest extent possible, this data should be represented using standards to promote reuse, modification, and normalization. The following are examples of typical (but not required) types of standards that may be leveraged:

- Extensible Configuration Checklist Description Format (XCCDF)
- Open Vulnerability and Assessment Language (OVAL)
- Open Checklist Interactive Language (OCIL)
- Common Vulnerabilities and Exposures (CVE)
- Common Configuration Enumeration (CCE)
- Common Vulnerability Scoring System (CVSS)
- Common Configuration Scoring System (CCSS)
- Common Platform Enumeration (CPE).

Using standards facilitates the communication of requirements to vendor tools. More important, it supports the aggregation and comparison of the resultant standards-based output. Without standards, it is difficult to impossible to aggregate and compare system state data from a diverse set of tools. In the security domain, the Security Content Automation Protocol[17] (SCAP) and the associated National Vulnerability Database[18] (NVD) provide a variety of standards that provide a strong foundation for policy expression and system state data normalization. CM implementations focused on security will likely leverage SCAP but are not limited to this suite of standards. Many other standards are in development that can support CM domains outside of the scope of SCAP.

A particular CM instance is not required to have a Content Subsystem, but at least one Content Subsystem must exist somewhere in the organization's CM implementation. Choosing the number of Content Subsystems for an organization's CM implementation is a matter of policy. If all parts of the organization are required to use the exact same security posture settings, only a single Content Subsystem is needed. If some requirements are universal across an organization while others can be tailored for specific environments, a hierarchy of Content Subsystems could be implemented where each tier in the hierarchy would have the ability to tailor a particular set of the requirements but not all of them. Alternately, a completely decentralized approach could be implemented where only the lowest tiers have Content Subsystems. This allows each part of the organization the greatest flexibility in defining its security requirements but will pose great challenges when trying to aggregate and compare the CM results. Section 5.4 on tiered implementations will discuss these design decisions in greater detail.

5.2.6 Task Manager Subsystem

Task Manager Subsystem (one per CM instance): The primary functions of the Task Manager are to orchestrate queries for data and handle inter-CM instance requests and reports. Common examples of this include responding to data requests from the dashboard engine, delivering predefined data views to a

[17] http://scap.nist.gov
[18] http://nvd.nist.gov

higher level CM instance, and responding to a data request from a higher level CM instance. The Presentation Subsystem is thus a primary customer of the Task Manager as are the Task Managers from other CM instances.

The Task Manager is composed of four components: query orchestrator (QO), inter-tier reporting (IR), collection controller (CC), and inter-tier queries (IQ).

Query Orchestrator

The query orchestrator (QO) component determines the best way to satisfy a request for CM data. The request may arrive from a Presentation Subsystem or from a higher level CM instance. If possible, data is simply retrieved from the Data Aggregation Subsystem. If the data is stale or incomplete, the QO may task the CC to begin collection of the needed data. If the data is available only from lower level CM instances, the QO may task the inter-tier queries component to task those lower level instances to return the requested data. Note that this activity can occur recursively as the lower tier's QO processes the request using the same logic.

Inter-tier Reporting

An important part of any CM multi-tier implementation is the ability to regularly pass predefined data views from a lower tier to a higher tier. Even organizations that view themselves as having only a single CM instance may need this capability if they need to regularly report data to external entities (e.g., DHS Cyberscope in the U.S. government). The inter-tier reporting (IR) enables this regular data transfer by periodically sending data up to a higher tier's IR component.

Collection Controller (advanced)

The collection controller (CC) component is a collector (or sensor system) orchestration capability. If the Data Aggregation Subsystem does not contain data needed by the QO, the QO may task the CC to orchestrate collection. The CC directs the Collection Subsystems to collect the needed data and to report it to the Data Aggregation Subsystem. When the collection is complete, the CC notifies the QO that the data has been collected.

Although many security tool consoles have this capability for their product's sensors, the CC will be able to orchestrate this activity over a variety of products from multiple vendors. Without this capability, the CM users will have access only to predefined data views provided by the Collection Subsystems.

The CC is labeled as an "advanced" component because the marketplace for this type of functionality is nascent and relevant orchestration standards have not been identified. Fortunately, CM implementations can function without the CC capability, but much greater functionality is to be gained through implementing it. Section 6 describes the use cases and related functionality available with and without the "advanced" components.

Inter-tier Queries (advanced)

The inter-queries (IQ) component provides a multi-tier CM communication capability. The IQ at a higher CM instance tier may send operational (or ad hoc) queries to lower tier IQ components to collect the relevant data. The lower tier IQ components then pass the requests to their QO for processing. Once the low tier IQ receives the query results, it passes those up to the higher tier IQ component. The higher tier IQ passes the data back to its QO, which stores the data in the higher tier Data Aggregation Subsystem.

This inter-tier communication capability allows for the majority of the CM data to stay at the lower tiers and for only specific data to be passed to the higher tiers upon request. As discussed previously, this has both security and functional advantages.

Without the IQ unit in a multi-tier implementation, users at the higher tiers would be restricted to querying the predefined data view sent up through the inter-tier reporting components from the lower tiers. If the CM users at the higher tier needed to drill down into a specific issue, they would need to have the predefined data aggregation view updated and expanded. Over time, the temptation would be to expand the predefined data views to pass more and more data up through all tiers of the CM implementation. The large volume of this data would cause storage and transmission issues. There also might be issues with "stale" data existing at the higher level tiers with the "fresh" data only being available at lower level tiers. Last and most important, aggregating all low-level security data at higher tiers is likely unnecessary and creates a security risk by centralizing a large amount of security sensitive data in a single repository.

One concern in this model is with the interplay between the IQ, QO, and CC components. For example, a user at a higher tier could issue a query that is passed down by the IQ to a lower tier and that triggers excessive consumption of system resources in the lower tier. This could happen, for example, if a higher level request coming in through the IQ triggered the QO to direct the CC to issue an organization-wide file scan. For this reason, the Task Manager IQ units will need to contain policies on what type of requests to automatically execute and when to execute them, on what types that need human approval, and on what types to simply deny. This will enable CM implementations to protect themselves from excessive querying from other CM instances, enable implementation of data sharing policies, and operate in a more federated model.

Like the CC, the IQ is labeled as an "advanced" component because the marketplace for this type of functionality is nascent, and relevant querying standards have not been identified. Fortunately, CM implementations can function without the IQ capability, but much greater functionality is to be gained through implementing it. Section 6 describes the use cases and related functionality available with and without the "advanced" components.

5.2.7 Situational Awareness Capability

Situational Awareness Capability (one per CM instance): Four of the CAESARS FE Subsystems are grouped into the Situational Awareness Capability (SAC). The SAC holds the core CM functionality including CM data storage, analysis, scoring, retrieval, and presentation. The interplay between SAC Subsystems is complex, and the available standards do not sufficiently cover the needed functionality. Thus, standardization of this capability is not currently achievable or even desirable at this time. The CM tool market is immature in this area and needs the ability to innovate and mature before any standardization efforts are pursued.

Thus, organizations will likely acquire or build their SAC Subsystems as a single proprietary unit or leverage complementary products tied together with proprietary interfaces. In the future as the CM and SAC marketplace matures, standards may be developed enabling organizations to buy best-of-breed SAC Subsystems and to cost-effectively integrate them together in a plug-and-play manner. This is a major long-term design goal of both CAESARS and CAESARS FE, but current limitations in available standards for the SAC limit our near-term ability to achieve this.

5.3 Subsystem Interface Overview

For CAESARS FE to describe a CM model that consists of plug-and-play components, it is necessary to define the minimal necessary communication paths or interfaces between the subsystems. An interface is the mechanism by which to transmit data between two subsystems (e.g., using web services). Along with each interface, there needs to be well-defined payloads (e.g. using XML) for the communications.

Most of these interfaces fall within the SAC boundary and thus do not currently lend themselves to standardization (see Section 5.2.7). In such cases, proprietary solutions are likely to be leveraged and interoperability will need to be determined through procurement evaluation or product testing.

Figure 5 shows the currently identified minimum set of interfaces for a CM implementation. Most of these interfaces are derived from the abstract use cases and associated workflow presented in Section 6 and Appendix B. The arrows show which subsystem typically initiates communications over a particular interface and do not represent directional flow of data. If a subsystem is not deployed within a CM implementation (i.e., a content or collection subsystem), obviously the related communication paths are unnecessary to that deployment. Note that in some cases, subsystems will need interfaces to communicate with entities outside of the CM implementation.

CAESARS FE plans to use web services along with the Asset Reporting Format (ARF) as a required capability for most subsystem interfaces. The subsystem and detailed interface requirements will be made available in future publications. An ESB using ARF as the primary communication language is also an option, although some custom integration may be required. Below are references to the ARF specification as well as the ARF interface description provided as a Web Service Description Language document.

Four draft documents are available:

1. Asset Reporting Format specification (http://scap.nist.gov/specifications/arf/)
2. Asset Reporting Format XML schema (http://scap.nist.gov/specifications/arf/)
3. Asset Identification specification (http://scap.nist.gov/specifications/ai/)
4. Asset Identification XML schema (http://scap.nist.gov/specifications/ai/)

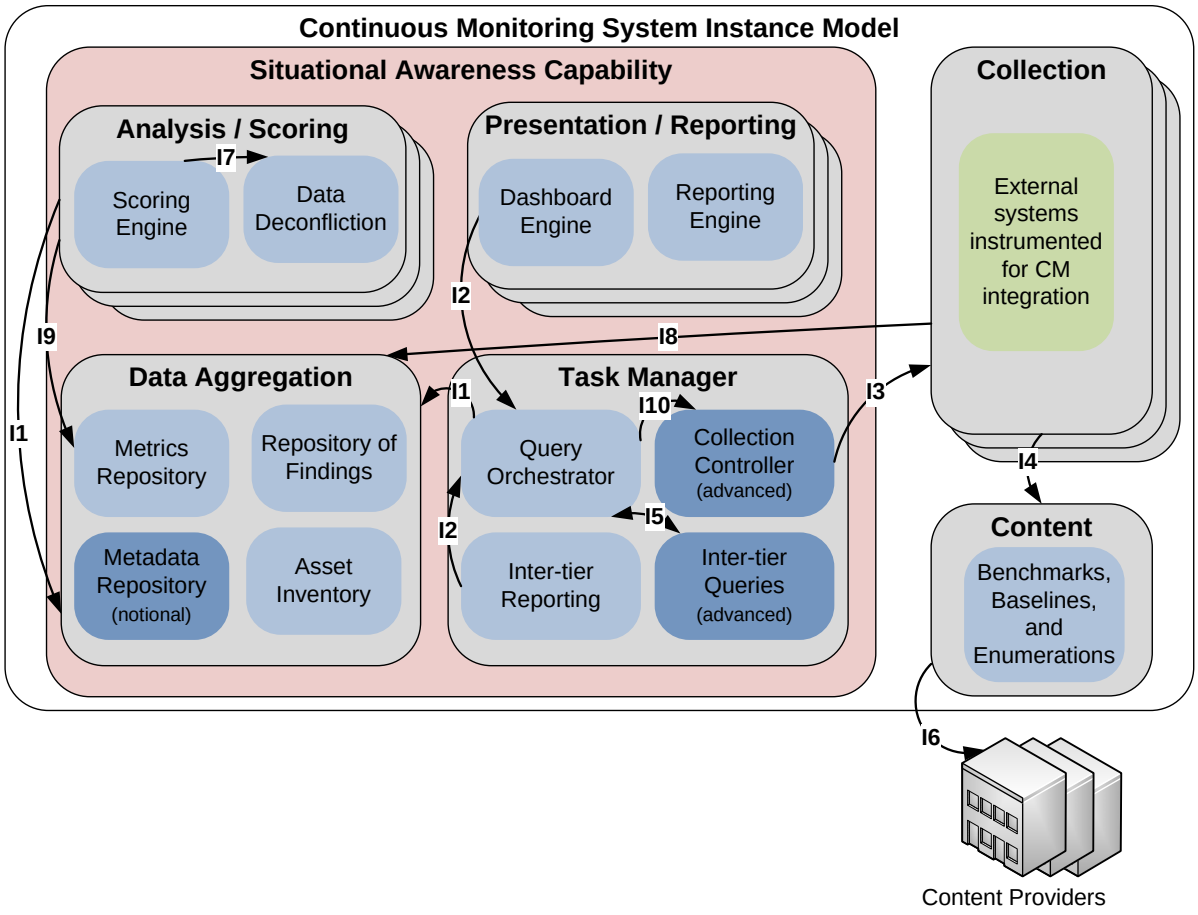

Figure 5. Continuous Monitoring System Instance Model with Interfaces

Each interface shown in Figure 5 is described below:

I1: The Data Aggregation Subsystem is accepting data retrieval requests.

I2: The QO is receiving data query requests from the dashboard engine or the IR component.

I3: The Collection Subsystem is accepting commands from the CC to modify the data being collected or the periodicity of collection.

I4: The Collection Subsystem is communicating with the Content Subsystem to retrieve the latest profile of expected security posture as required by organization policy (e.g., host configuration baseline or lists of authorized software).

I5: The Task Manager's IQ component is exchanging multi-tier queries and results with the QO.

I6: The Content Subsystem is retrieving security content (e.g., configuration benchmarks or vulnerability names) from content providers. For the security domain, CM implementations will need to interface with the National Vulnerability Database (NVD). NVD has defined and implemented web services interfaces. The Collection and Analysis Subsystems may also need this interface.

I7: The scoring engine is requesting the data deconfliction component to cleanse a set of data prior to it being scored.

I8: The Collection Subsystem is depositing data into the repository of findings and asset inventory. Note that this interface covers both asset inventory collection subsystems and security-focused collection subsystems.

I9: The scoring engine is depositing scoring results into the metrics repository.

I10: The Task Manager's QO is requesting that the CC initiate data gathering.

Note, I8 and I9 may end up using the same interface, but further research is required to understand the correct approach.

5.4 Multi-tier Capability

Large organizations often need more than one CM instance. This may be due to a need to avoid aggregating all security data into a single repository. It may also be because organizational components are structured such that they independently manage their IT and having a collection of federated CM instances is their preferred approach.

CAESARS FE assumes that CM instances are arranged in a tree structure forming a logical hierarchy as shown in Figure 6. Aggregated data reports and compliance information typically travel up the tree. Data calls and security configuration requirements typically travel down the tree. Lateral communication between nodes is possible but not explicitly part of the CAESARS FE model. Depending on the policy of the organization, CM instances may be given an element of autonomy, resulting in a more federated relationship even while preserving the hierarchy. In this case, CM instances may require human approval of requests from higher tiers prior to releasing the data or performing additional data collection. This may be important in cases where the incoming request would consume significant system resources and thus require planning and scheduling to minimize the impact of the load on the organization.

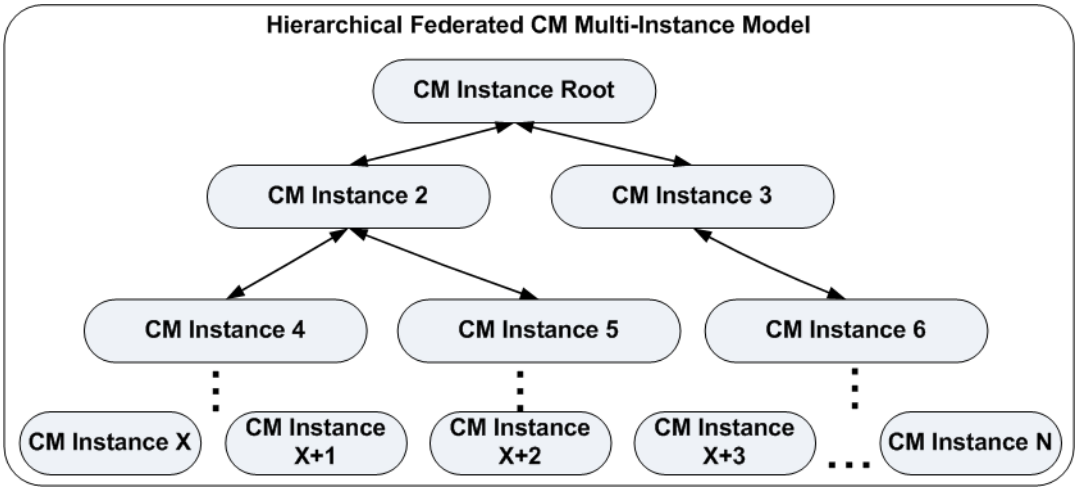

Figure 6. Federated Hierarchical Continuous Security Monitoring Instance Model

The federated hierarchical model for multiple CM instances creates additional interface considerations. As shown in Figure 7, CAESARS FE has three additional interfaces for such interactions.

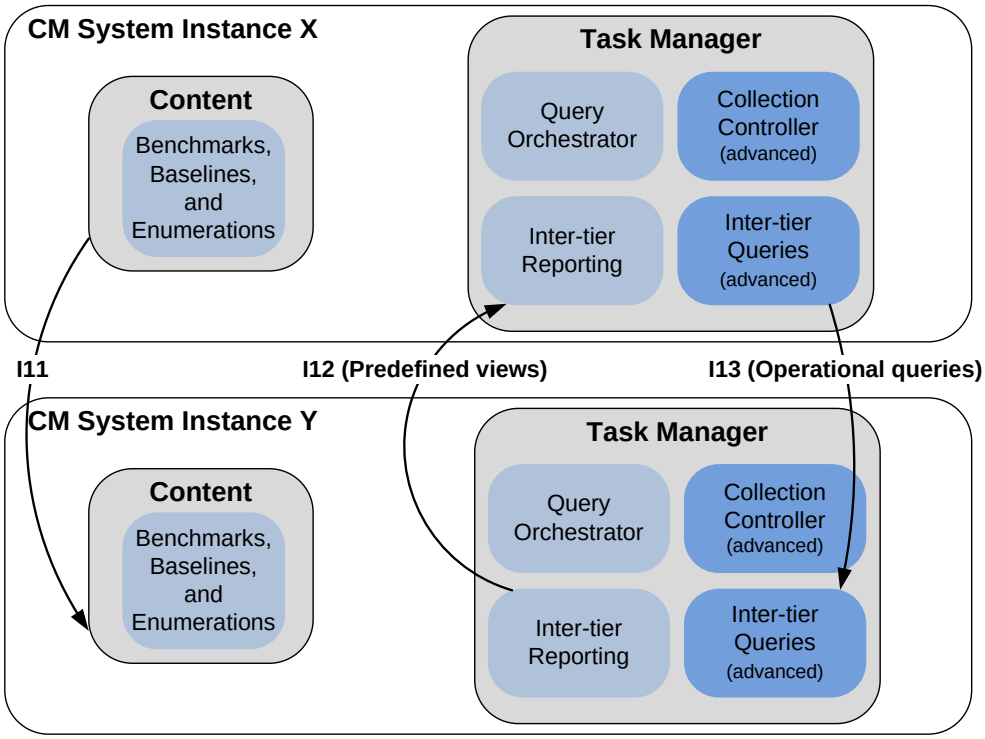

Figure 7. Continuous Monitoring Multi-Instance Model with Interfaces

Each interface shown in Figure 7 is described below:

I11: The Content Subsystem at a higher tier is communicating organizational security policy data (e.g., configuration or scoring data) to a Content Subsystem at a lower tier.

I12: The Task Manager IR component at a lower tier is communicating a predefined data view to an IR component at a higher tier. This data is typically aggregate statistics or compliance data.

I13: The IQ component from a higher tier is passing an operational (or ad hoc) data request down to a lower tier IQ component.

Subsequent publications will provide the lower level specifications for these interfaces. The focus of that work will be on specifying I11 so that CM instances can report predefined data views to higher level tiers.

6. Use Cases and Related Workflows

Any CAESARS FE implementation will need to implement at least two of the five abstract use cases presented in this section depending on the type of CM implementation. These use cases specify necessary workflow that must occur within the CAESARS FE architecture. They are data type agnostic in that they discuss high-level data movement and processing requirements without discussing how any particular data type is to be processed. These abstract use cases thus specify necessary functionality that will be used by any implementation of CAESARS FE regardless of the data domain to be processed. They are shown in Figure 8.

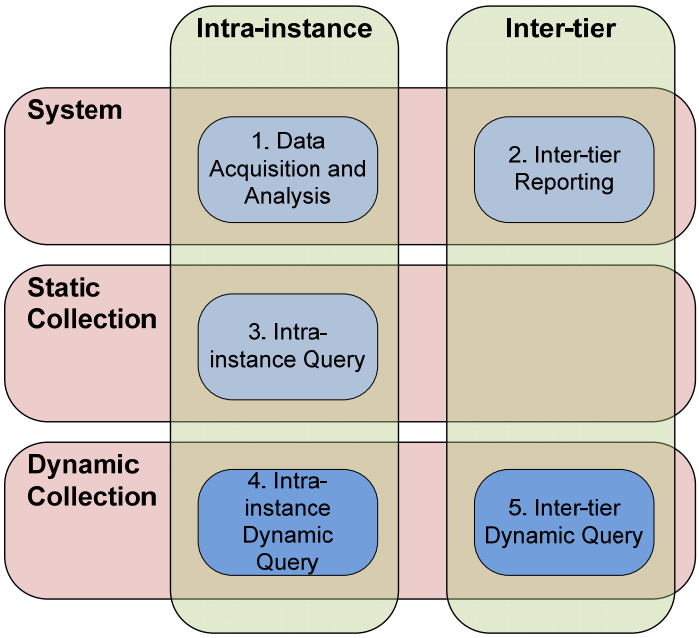

Figure 8. Abstract Use Cases

The first two use cases are labeled as “system” use cases because the related workflow is something that a CAESARS FE instantiation should automatically execute on some periodic or event-driven basis. Human actors may also trigger these activities to occur outside of the normal automated cycle.

The next use case is labeled as “static collection” because it leverages predefined data views to provide query results. Note that this method also avoids using any CAESARS FE components that are marked as “advanced” in the architecture. This simplifies the related workflow but also reduces the provided functionality. This use case handles both the workflow within a single CM instance and multi-instance communication.

The last two use cases are labeled as “dynamic collection” because they leverage the ability to collect new data as requested and are not restricted to the static predefined data collection views. Note that these use cases require the CAESARS FE components that are marked as “advanced” in the architecture. The related workflows for the “dynamic” use cases are more complex than the “static” use cases but much greater CM functionality is made available. One of these use cases handles the workflow within a single CM instance and the other handles multi-instance communication.

To conform to the CAESARS FE architecture, a CM solution must implement at least two or more abstract use cases and the associated workflow. Which use cases need to be implemented depends on

whether the CM solution contains multiple CM tiers. It also depends on whether the solution supports dynamic collection and thus incorporates the advanced components. The following are the use cases that need to be implemented for each implementation approach:

Single CM instance deployment

1. Uses static data collection: 1, 3
2. Uses dynamic data collection: 1, 4

Multiple CM instance deployment

1. Uses static data collection: 1, 2, 3
2. Uses dynamic data collection: 1, 2, 5

The details for these five use cases and their workflow specifications are provided in Appendix B.

7. Future Work

CAESARS FE currently provides a vision and an enterprise architecture for implementing enterprise-wide CM. Along with that vision is an associated technical architecture, abstract use cases, and data agnostic workflow specifications. This information is sufficient for organizations to model their CM efforts after CAESARS FE, but it will not enable them to achieve all of the design goals. Among other things, organizations need to be able to leverage their existing data collection, analysis, and event management products to compose interoperable enterprise-wide CM solutions.

To achieve this goal, CAESARS FE needs to be specified at a level low enough that vendors can instrument their products to support the architecture. It is important to provide minimal specifications that will enable the necessary functionality to avoid overburdening the participating vendors. There is also a need to focus on specifying subsystem inputs and outputs while avoiding specifying *how* a vendor is to implement subsystems and components.

The plan is to accomplish this through first specifying data agnostic requirements for each subsystem. This will fully specify the CAESARS FE architecture apart from any specific data domain. These requirements will specify how collected data is to be reported, analyzed, scored, stored, and retrieved for presentation. Once that goal is achieved, the focus will be on writing data specific requirements for CAESARS FE processing of various data domains (see Figure 1). Some of these domains already have associated mature security automation standards (e.g., CVE for vulnerability management) while others have underdeveloped or non-existent standardizations (e.g., license management). Lastly, there will be a need to synthesize data from multiple domains to provide a cross domain situational awareness picture.

Appendix A—Acronyms

This appendix contains selected acronyms and abbreviations used in the publication.

ARF	Asset Reporting Format
CAESARS	Continuous Asset Evaluation, Situational Awareness, and Risk Scoring
CC	Collection Controller (a Task Manager component)
CCE	Common Configuration Enumeration
CCSS	Common Configuration Scoring System
CM	Continuous Monitoring
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DHS	Department of Homeland Security
EA	Enterprise Architecture
ECMC	Enterprise Continuous Monitoring Capability
ESB	Enterprise Service Bus
FE	Framework Extension
FISMA	Federal Information Security Management Act
FNS	Federal Network Security
IQ	Inter-tier Query (a Task Manager component)
IR	Inter-tier Reporting (a Task Manager component)
ISIMC	Information Security and Identity Management Committee
IT	Information Technology
ITL	Information Technology Laboratory
NIST	National Institute of Standards and Technology
NIST SP	National Institute of Standards and Technology Special Publication
NSA	National Security Agency
NVD	National Vulnerability Database
OCIL	Open Checklist Interactive Language
OMB	Office of Management and Budget
OVAL	Open Vulnerability and Assessment Language
QO	Query Orchestrator (a Task Manager component)
SAC	Situational Awareness Capability
SCAP	Security Content Automation Protocol
SOA	Service-oriented Architecture
XCCDF	Extensible Configuration Checklist Description Format
XML	Extensible Markup Language

Appendix B—Use Case and Workflow Specifications

This appendix provides the details for each of the use cases presented in Section 6. These abstract use case and their associated workflow clarify the intent or purpose of each subsystem and their components and thus are essential the CAESARS FE architecture.

B1. Data Acquisition and Analysis

Use Case Name	Data Acquisition and Analysis
Use Case ID	UC1
Type	System
Scope	Intra-instance
Primary Actor	Continuous Monitoring System
Secondary Actor	Authorized CM users
Brief Description	This use case describes how the CM system acquires and analyzes data using a predefined data view. This data view may be updated periodically but not continually and is not the result of on-demand user queries (other use cases cover those scenarios). More specifically, this activity includes data collection, deconfliction, analysis, scoring, and results storage.
Parameters	None for triggers 1, 2, and 3. For trigger 4, the parameter is the data is to be retrieved (data domain specific).
Trigger	1. This use case is triggered periodically by the CM system itself to regularly update its predefined enterprise data view. In addition to using periodicity (i.e., time based), some systems may use system event-based triggers for this data acquisition use case (e.g., upon a change in threat level). 2. Authorized CM users may also manually trigger this use case to refresh the enterprise CM data. 3. UC2 may trigger UC1 to update its predefined reporting view that is to be sent to a higher tier CM instance. 4. This use case may be triggered by the Task Manager's Collection Controller. This activity occurs only as a function within the advanced use cases. For this, a parameter is provided describing the data to be collected instead of having the typical predefined data view collected.
Preconditions	Triggers 1 and 2: A predefined data view must exist that is to be populated by this use case and associated metrics for analyzing the data. The Collection, Data Aggregation, and Analysis Subsystems must be able to process and store this

	data. Trigger 3: The Collection, Data Aggregation, and Analysis Subsystems must be able to process, analyze, and store the data requested by the input parameters provided by the triggering advanced use case.
Flow of Events	1) **Collection** Subsystems collect the requested data. 2) **Collection** Subsystems populate the **Repository of Findings** component and/or **Asset Inventory** component with the data collected. 3) **Data Aggregation** Subsystem optionally performs initial data cleansing. 4) **Analysis/Scoring** Subsystem requests and receives raw data from the **Data Aggregation** Subsystem 5) **Data Deconfliction** component refines the data to prepare it for scoring 6) **Scoring Engine** component calculates metrics 7) **Analysis/Scoring** Subsystem populates **Metrics Repository** component with scoring results
Post- Conditions	Triggers 1, 2, and 3: The predefined data view has been refreshed, and the collected raw data as well as the calculated metrics are now available for the Presentation/Reporting and Task Manager Subsystems. Trigger 4: The requested data has been collected and scored. It is now available for retrieval by the Query Orchestrator.

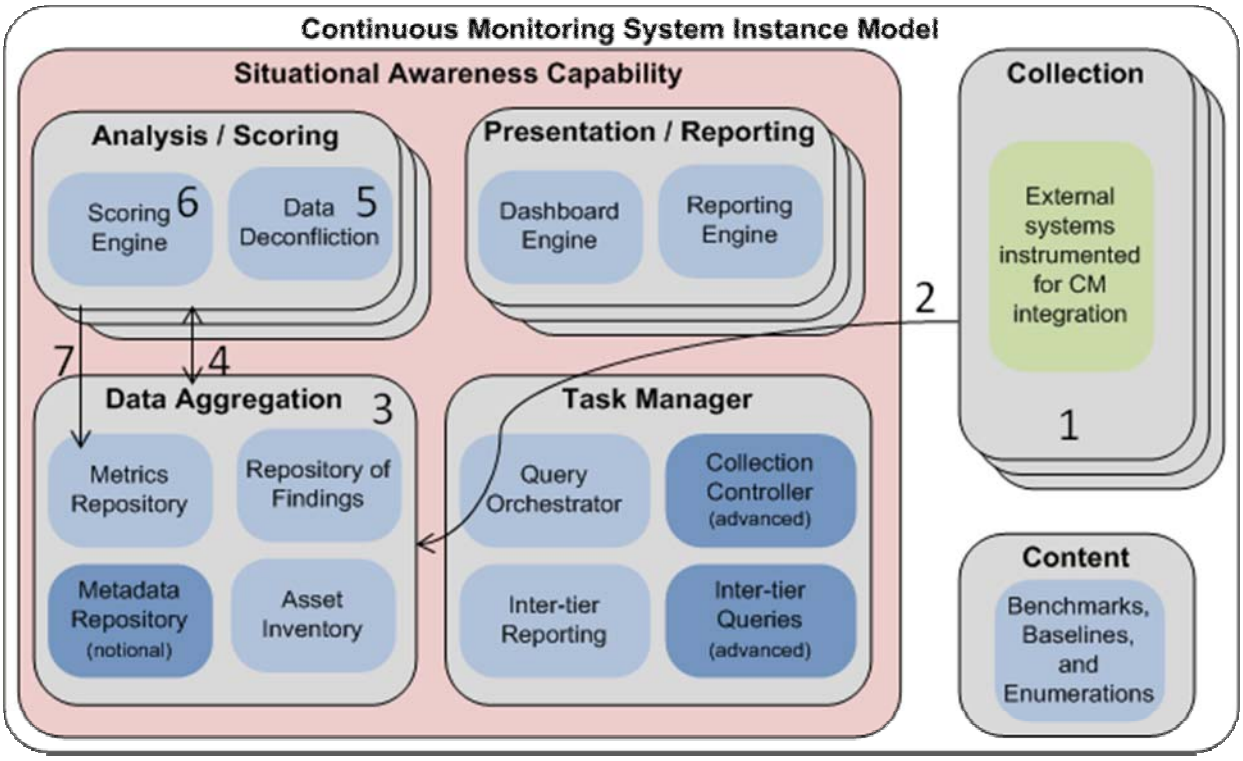

Figure 9. Use Case 1 Workflow Overlaid on Architecture

B2. Inter-tier Reporting

Use Case Name	Inter-tier Reporting
Use Case ID	UC2
Type	System
Scope	Inter-instance
Primary Actor	Continuous Monitoring System
Secondary Actor	None
Brief Description	This use case describes how a higher tier CM instance is updated with a predefined reporting view from a lower tier CM instance. This reporting view may be updated periodically but not continually and is not the result of on-demand user queries (other use cases cover those scenarios). UC2 uses UC1 to collect the data to be reported if C1 has not already been recently executed.
Parameters	1. Acceptable freshness of the data to be reported

Trigger	This use case is triggered periodically by the Inter-tier Reporting component of a lower-tier CM system. In addition to using periodicity (i.e., time based), some systems may use system event-based triggers for this reporting (e.g., upon a change in threat level).
Preconditions	There must exist a predefined reporting view (i.e., raw data and/or associated metrics) that is to be regularly delivered from the lower tier CM instance to the higher tier CM instance. The Task Manager and Data Aggregation Subsystems of both CM instances must be able to process and store the reported data. The predefined data view collected in UC1 must support the UC2 predefined reporting view.
Flow of Events	1) Low-tier **Inter-Tier Reporting** component queries the low-tier **Query Orchestrator** component to retrieve the predefined reporting view. 2) Low-tier **Query Orchestrator** component checks the freshness of the needed data against the supplied freshness parameter. If data is not sufficiently fresh, execute UC1. 3) Low-tier **Query Orchestrator** component retrieves required data from low-tier **Data Aggregation** Subsystem. 4) Low-tier **Query Orchestrator** component sends data to low-tier **Inter-Tier Reporting** component. 5) Low-tier **Inter-Tier Reporting** component sends report data to high-tier **Inter-Tier Reporting** component. 6) High-tier **Inter-Tier Reporting** component sends report data to high-tier **Data Aggregation** Subsystem for storage. 7) **Data Aggregation** Subsystem now has the most up-to-date data. The process can start over based on a predefined interval.
Post-Conditions	The high-tier CM instance has been updated with a predefined reporting view from a lower tier CM instance. The higher tier CM instance can now perform analysis, presentation, and reporting on the lower tier data.

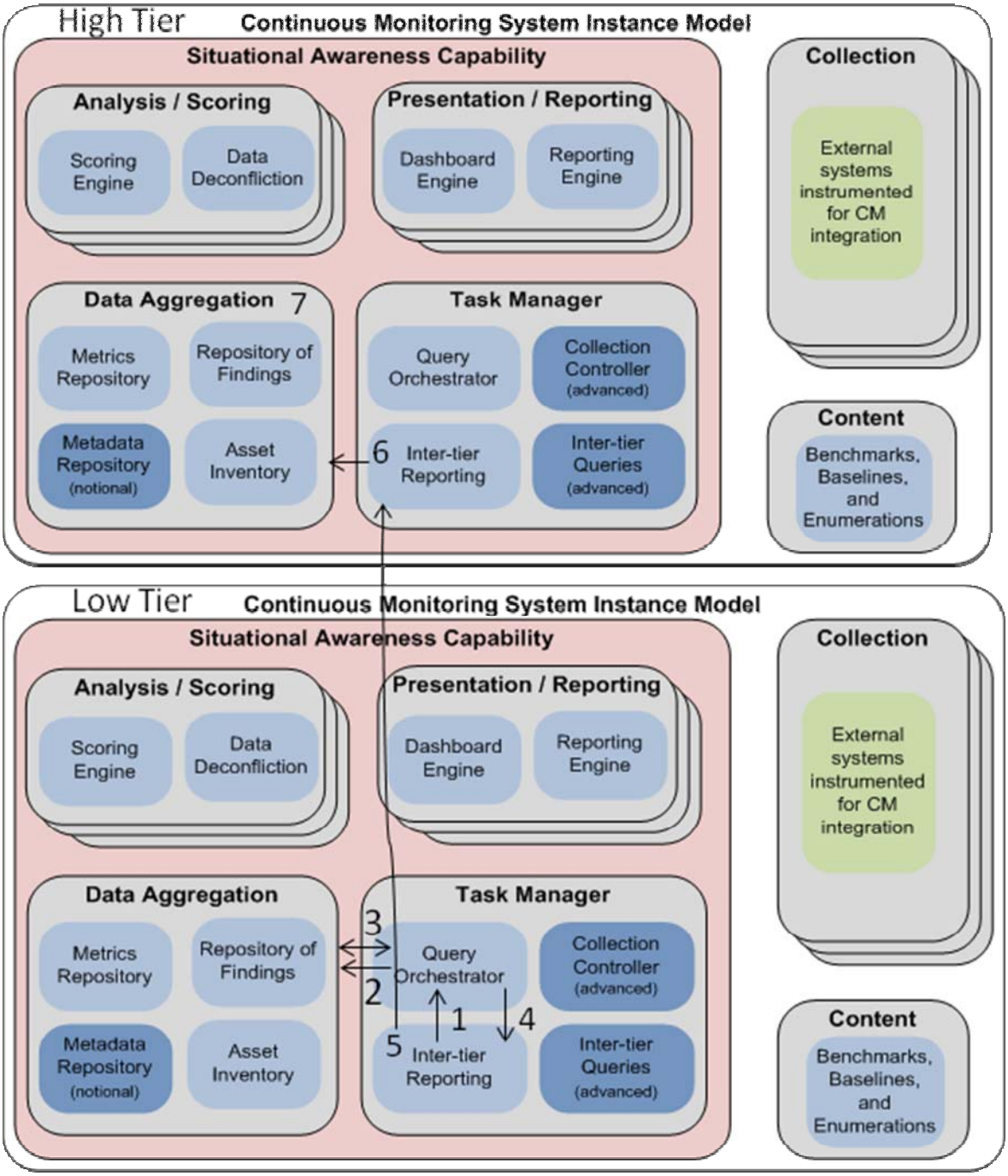

Figure 10. Use Case 2 Workflow Overlaid on Architecture

B3. Intra-instance Query

Use Case Name	Intra-Instance Query
Use Case ID	UC3

Type	Static Collection
Scope	Intra-instance
Primary Actor	Authorized CM users
Secondary Actor	None
Brief Description	This use case describes how a CM user can request data that is stored within that user's CM instance. In a multi-instance CM implementation, the available data may include results from lower tier CM instances that were retrieved using use case UC2.
Parameters	1. What data is to be retrieved (data domain specific)
Trigger	The actor initiates a request for data at the Presentation/Reporting Subsystem.
Preconditions	Within a CM instance, the process shown in UC1 must occur prior to the system being able to provide the functionality described in UC3. To avoid having to first trigger UC1, UC 4 may be used in place of UC3. In a multi-tier CM implementation, the process shown in UC2 occurs prior to the system being able to provide any data from the lower tiers. If the process in UC2 has not already taken place, only the information from the current tier will be available to present to the user.
Flow of Events	1) Actor initiates a request at the **Presentation/Reporting** Subsystem, which sends a query to the **Query Orchestrator** component. 2) **Query Orchestrator** component attempts to retrieve the requested data from the **Data Aggregation** Subsystem (in particular the **Metrics Repository, Repository of Findings,** and **Asset Inventory** components**)**. 3) **Query Orchestrator** component delivers data to **Presentation/Reporting** Subsystem or returns an error code if data retrieval was unsuccessful.
Post- Conditions	The actor is able to receive the requested information provided that it is available in the Data Aggregation Subsystem.

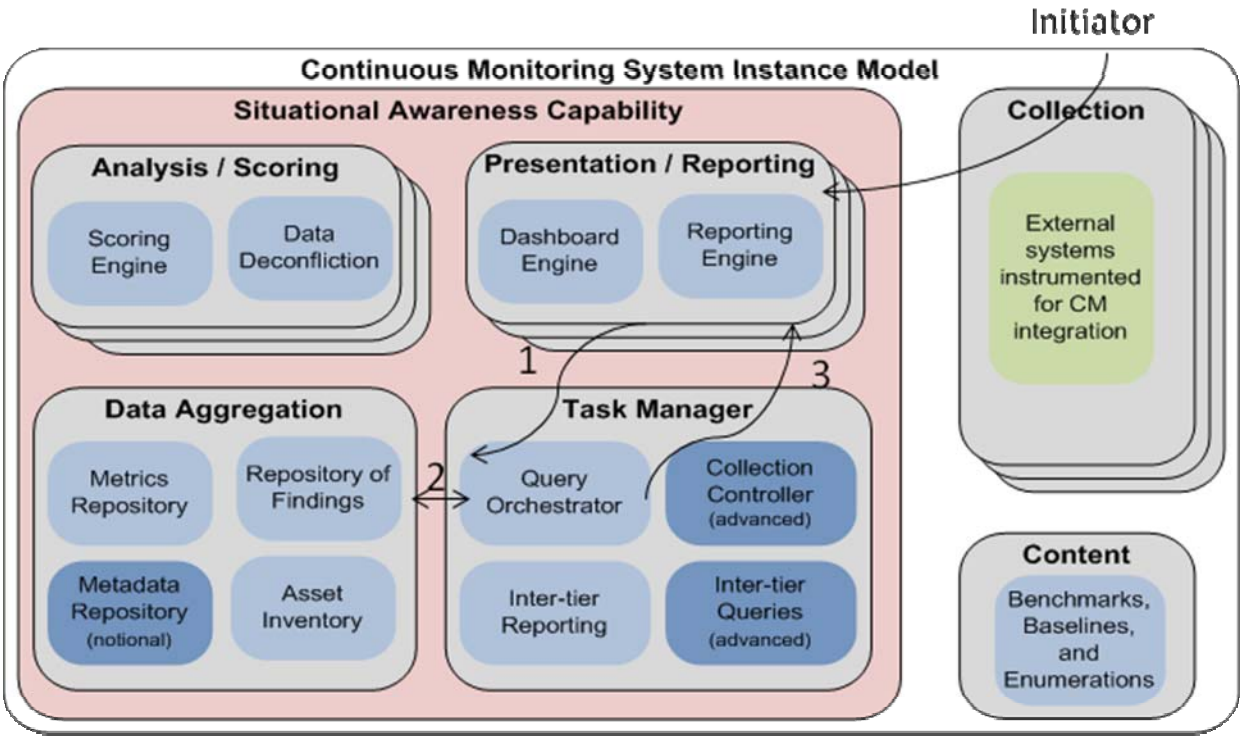

Figure 11. Use Case 3 Workflow Overlaid on Architecture

B4. Intra-instance Dynamic Query

Use Case Name	Intra-instance Dynamic Query
Use Case ID	UC4
Type	Dynamic Collection
Scope	Intra-instance
Primary Actor	Authorized CM Power Users
Secondary Actor	None
Brief Description	This use case describes how a CM user can request data that is stored within that user's CM instance or that can be collected from that CM instance's Collection Subsystems. In a multi-instance CM implementation, the available data may include results from lower tier CM instances that were retrieved using use case UC2.
Parameters	1. What data is to be retrieved (data domain specific) 2. Parameters on when to dynamically collect data (e.g., freshness

	requirements)
Trigger	The actor initiates a request for data at the Presentation/Reporting Subsystem.
Pre-conditions	Unlike UC3, UC1 does not have to execute prior to UC4 being triggered because UC4 can proactively collect the needed data. As with UC3, in a multi-tier CM implementation the process shown in UC2 must occur prior to the system being able to provide any data from the lower tiers. If the process in UC2 has not already taken place, only the information from the current tier will be available to present to the user. To avoid having to first trigger UC2, UC 5 may be used in place of UC4.
Flow of Events	1) Actor initiates request at **Presentation/Reporting** Subsystem**,** which sends a query to the **Task Manager** Subsystem. 2) **Query Orchestrator** checks to see if the requested data is available from the **Data Aggregation** Subsystem. If the data is available, skip to Step 5. *Note: In this case, UC4 becomes identical to UC3.* If the data is not available or is stale, proceed to next step. 3) **Query Orchestrator** component tasks the **Collection Controller** component to collect the needed data. a. **Collection Controller** component triggers UC1 with the requested data as the input parameter. b. When UC1 has completed, **Collection Controller** component notifies **Query Orchestrator** component that data has been collected. 4) **Query Orchestrator** component retrieves the requested data from the **Data Aggregation** Subsystem (in particular the **Metrics Repository, Repository of Findings,** and **Asset Inventory**). 5) **Query Orchestrator** component delivers the data to **Presentation/Reporting** Subsystem.
Post- Conditions	The actor is able to receive the requested information whether or not it had already been collected and stored in the Data Aggregation Subsystem.

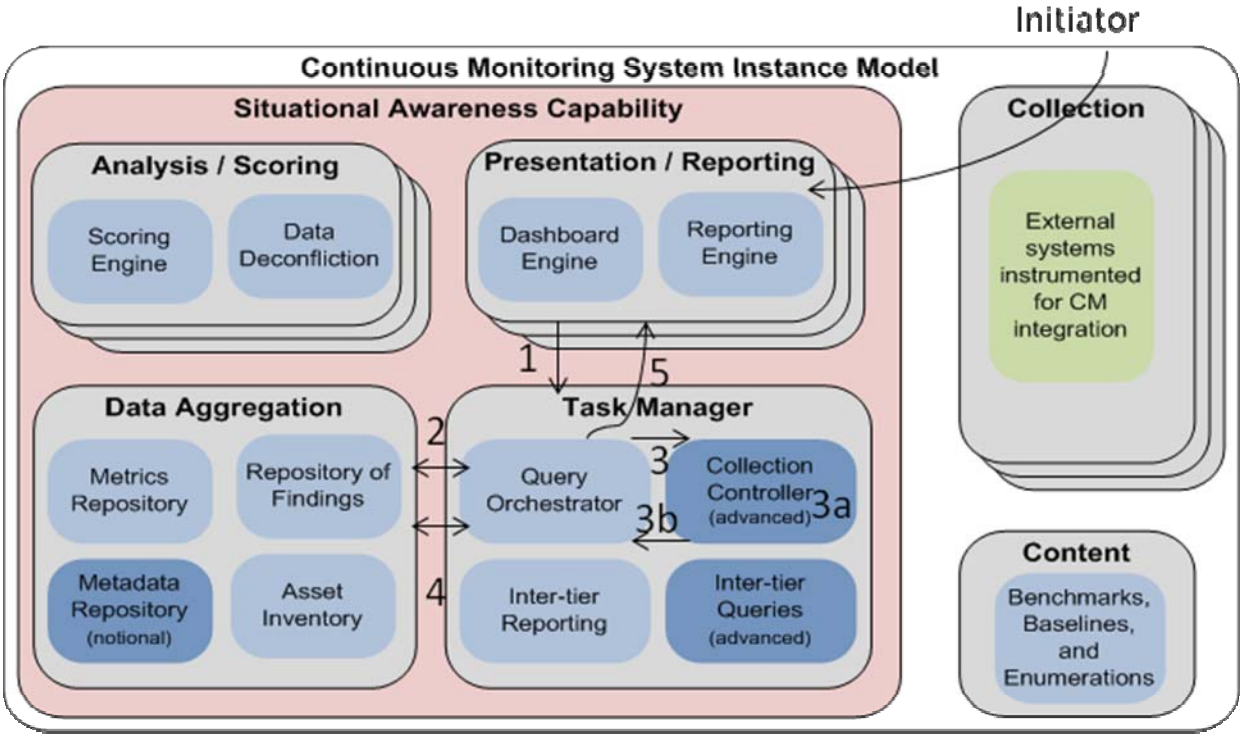

Figure 12. Use Case 4 Workflow Overlaid on Architecture

B5. Inter-tier Dynamic Query

Use Case Name	Inter-tier Dynamic Query
Use Case ID	UC5
Type	Dynamic Collection
Scope	Inter-instance
Primary Actor	Authorized CM Power Users
Secondary Actor	None
Brief Description	This use case describes how a CM user can request data that can be obtained from lower tier CM instances, can be collected from that CM instance's Collection Subsystems, or that can be retrieved from the Data Aggregation Subsystem in that user's CM instance. In a multi-instance CM implementation, the available data may include results from lower tier CM instances that were retrieved using use case UC2. If the data is not available, UC5 will recursively collect the data from lower tier CM instances. Note that this latter method provides greater functionality because arbitrary data can be collected using UC5, whereas only the predefined report

	view can be collected from UC2.
Parameters	1. What data is to be retrieved (data domain specific) 2. List of tiers to which to propagate request (possibly none) 3. Parameters on when to dynamically collect data (e.g., freshness requirements)
Trigger	Trigger 1: The actor initiates a request for data at the Presentation/Reporting Subsystem. Trigger 2: UC5 may trigger itself recursively as the request for information travels down to lower tier CM instances in the CM instance tree. If UC5 triggers itself, it starts processing at Step 2 (not Step 1) and the final results are returned to the higher tier Inter-tier Queries component (not to the current tier's Presentation/Reporting Subsystem).
Pre-conditions	Unlike UC3, UC1 does not have to execute prior to UC5 being triggered because UC5 can proactively collect the needed data from the current CM instance. However, UC5 can leverage data collected through UC1. Unlike UC4, UC2 does not have to execute prior to UC5 being triggered because UC5 can proactively collect the needed data from lower tier CM instances. However, UC5 can leverage data collected through UC2.
Flow of Events	1) Actor initiates request at a high-tier **Presentation/Reporting** Subsystem, which sends a query to its high-tier **Task Manager** Subsystem. 2) The high-tier **Query Orchestrator** component checks to see if the requested data is available from the high-tier **Data Aggregation** Subsystem. If the data is available, skip to Step 5. *Note: In this case, UC5 becomes analogous to UC3.* If the data is not available or is stale, proceed to Step 3. 3) If the high-tier **Query Orchestrator** component determines that data needs to be collected from the high-tier CM instance **Collection** Subsystems, execute Substeps 3a–3c. Otherwise, skip to Step 4. *Note that this step performs activities analogous to UC4.* a. High-tier **Query Orchestrator** component tasks the high-tier **Collection Controller** component to collect the needed data. b. High-tier **Collection Controller** component triggers UC1 with the requested data as the input parameter. c. When UC1 has completed, the high-tier **Collection Controller** component notifies high-tier **Query Orchestrator** component that data has been collected, analyzed, scored, and stored in the high-tier

	Data Aggregation subsystem. 4) If the high-tier **Query Orchestrator** component determines that data needs to be collected from lower-tier CM instances, execute sub-steps 4a–4f. Otherwise, skip to Step 5. a. High-tier **Query Orchestrator** tasks the high-tier **Inter-tier Queries** component to collect the needed data. b. High-tier **Inter-tier Queries** component sends query to the relevant low-tier **Inter-tier Queries** components (there may be multiple lower-tiers). For each lower-tier execute sub steps c-e c. The low-tier **Inter-tier Queries** component determines whether local policy will allow the request to execute (e.g., a human approval process may be triggered or pre-encoded policy logic may be used). If yes, proceed to next step. If no, return an error code. d. Low-tier **Inter-tier Queries** component tasks the low-tier **Query Orchestrator** component to collect the needed data. e. The low-tier **Query Orchestrator** component executes UC5 recursively starting at Step 2. Note that upon jumping to Step two, the current "low" tier now becomes the new "high" tier in the new instantiation of UC5. f. High-tier **Inter-tier Queries** component stores the retrieved data in the high-tier **Data Aggregation** Subsystem. High-tier **Query Orchestrator** is notified that the low-tier data is available. 5) The high-tier **Query Orchestrator** component retrieves the requested data from the high-tier **Data Aggregation** Subsystem (in particular the **Metrics Repository, Repository of Findings,** and **Asset Inventory** components). 6) If this is the initial instantiation of UC5, the high-tier **Query Orchestrator component de**livers the data to the high-tier **Presentation/Reporting** Subsystem and UC5 is complete. If this is a process recursively initiated from Step 4e, execute steps 6a and 6b. a. The high-tier **Query Orchestrator** component delivers the data to the high-tier **Inter-tier Queries** component. b. The high-tier **Inter-tier Queries** component delivers the data to a higher tier **Inter-tier Queries** component (now returning to and fulfilling step 4e of the process that instantiated this one).

Post- Conditions	The actor is able to receive the requested information whether or not it had already been collected and stored in the Data Aggregation Subsystem and whether or not it had previously been available at lower CM instance tiers.

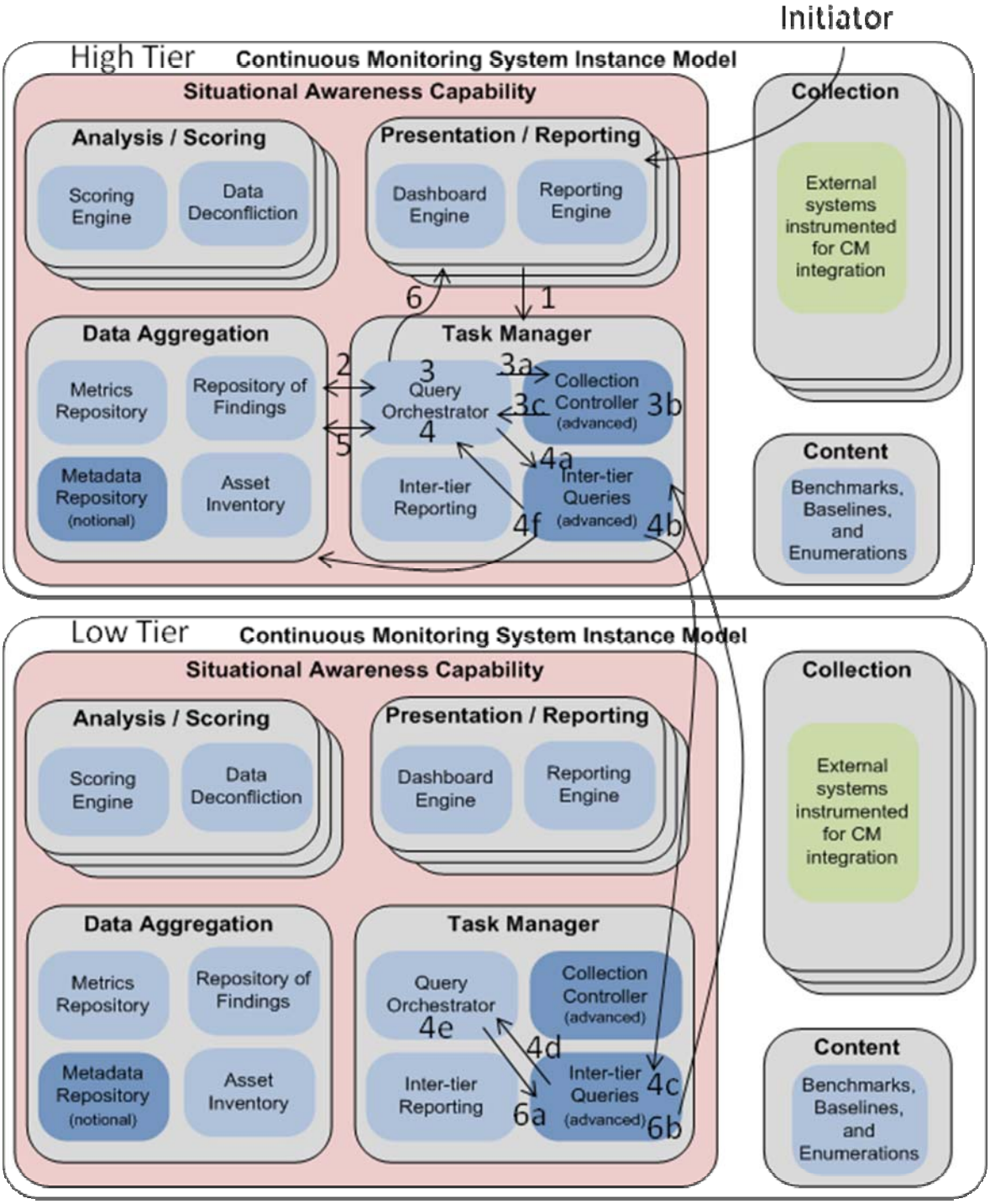

Figure 13. Use Case 5 Workflow Overlaid on Architecture