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

Continuous Monitoring Reference Model Workflow, Subsystem, and Interface Specifications (Draft)

Peter Mell, David Waltermire, Adam Halbardier, Larry Feldman

NIST Interagency Report 7799 (Draft)

Continuous Monitoring Reference Model, Workflow, and Specifications (Draft)

Peter Mell, David Waltermire, Adam Halbardier, Larry Feldman

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

Dr. 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 7799
75 pages (Jan. 2012)

Acknowledgments

The authors would like to recognize the following individuals for their participation on the continuous monitoring research team, insightful ideas, and review of this work: Stephen York and Peter Sell from the National Security Agency as well as Adam Humenansky 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 Committee (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:[1] 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.

Abstract

This publication provides the technical specifications for the continuous monitoring (CM^2) reference model presented in NIST IR 7756. These specifications enable multi-instance CM implementations, hierarchical tiers, multi-instance dynamic querying, sensor tasking, propagation of policy, policy monitoring, and policy compliance reporting. A major focus of the specifications is on workflows that describe the coordinated operation of all subsystems and components within the model. Another focus is on subsystem specifications that enable each subsystem to play its role within the workflows. The final focus is on interface specifications that supply communication paths between subsystems. These three sets of specifications (workflows, subsystems, and interfaces) are written to be data domain agnostic, which means that they can be used for CM regardless of the data domain that is being monitored. A companion publication, NIST IR 7800, binds these specifications to specific data domains (e.g., asset, configuration, and vulnerability management). The specifications provided in this document are detailed enough to enable product instrumentation and development. They are also detailed enough to enable product testing, validation, procurement, and interoperability. Taken together, the specifications in this document define an ecosystem where a variety of interoperable products can be composed together to form effective CM solutions. If properly adopted, these specifications will enable teamwork, orchestration, and coordination among CM products that currently operate distinctly. For the computer security domain, this will greatly enhance organizational effectiveness and efficiency in addressing known vulnerabilities and technical policy requirements, and decision making.

[1] 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.

[2] 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."

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. This publication is written for a technical audience and assumes detailed knowledge of NIST IR 7756.

Table of Contents

List of Figures

1. Introduction and Document Overview

This publication provides technical specifications for the continuous monitoring (CM) reference model presented in NIST IR 7756. A brief overview of the model (descriptions of subsystems, components, and interfaces) is provided, but the rationale and benefits of the model are described in NIST IR 7756 and are not repeated here.

1.1 Technical Areas

A major focus of this publication is on workflows that describe the coordinated operation of all the subsystems and components within the model. These workflows provide specific CM functionality including the ability to implement multi-instance CM systems arranged in coordinating, hierarchical tiers. The workflows also enable CM users to issue dynamic or ad-hoc queries (even reaching down through multiple CM instances) and task sensors to collect the needed data. Lastly, the workflows enable propagation of organizational policy, monitoring compliance with that policy, and reporting of the resulting compliance posture to outside entities or other CM instances.

Another focus of this publication is on the subsystem specifications. These subsystem specifications serve the essential purpose of enabling each subsystem to perform its role within the workflows. Each subsystem is described independently so that it can be developed as an independent product that interoperates with the CM model. As much as is possible, the authors recommend that subsystems be implemented as independent modules that can be composed to form a complete CM solution. This not only provides for a clean design but enables plug-and-play compatibility among subsystem implementations, thereby enabling organizations to compose CM solutions from a diverse set of best of breed tools. Independent implementation also may assist vendors who choose to partner with, or acquire, other companies that support different portions of the CM model as it will simplify the integration process.

Lastly, this publication focuses on the interface specifications that provide communication paths between subsystems and their components. These interfaces are therefore required to facilitate the aforementioned workflows. The interface specifications provide the details necessary to enable product implementation, by describing actual communication capabilities between products as opposed to expected inputs and outputs. To do this, they leverage web services, the Extensible Markup Language (XML), and the Asset Reporting Format (ARF)[3].

These specifications (workflows, subsystems, and interfaces) are written to be data domain agnostic, which means that they can be used for CM regardless of the data domain that is being monitored. This provides a useful level of abstraction, but it does require that the data domain agnostic CM model and specification be "bound" to specific *data domains* in order to be implemented. The NIST IR 7800, entitled "Applying the Continuous Monitoring Technical Reference Model to the Asset, Configuration, and Vulnerability Management Domains" provides guidance on this binding to leverage existing data domain specific specifications. The context of NIST IR 7800 includes asset management, configuration management, and vulnerability

[3] http://scap.nist.gov/specifications/arf

management through bindings to the Security Content Automation Protocol[4] (SCAP), Asset Identification[5], and ARF. CM product developers and CM architects can focus on a particular derivation of the model that best covers their desired subset of the available CM data domains.

1.2 Specification Layers

There are five layers of specifications that support each other in enabling implementation of the CM model described in NIST IR 7756. Figure 1 shows information about the five layers.

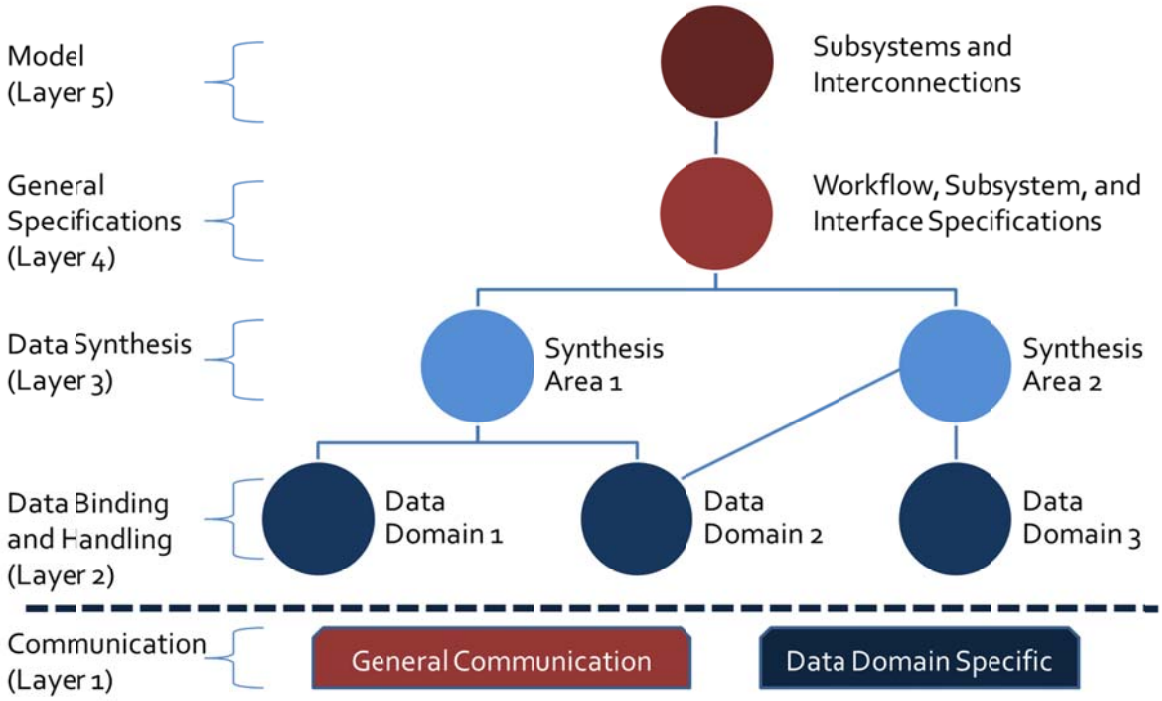


Figure 1. Continuous Monitoring Reference Model Specification Layers

Layer 5 provides the CM Reference Model itself and focuses on subsystem descriptions and their interconnections (needed communication pathways). This is covered by NIST IR 7756. Layer 4 provides the model workflows, subsystem, and interface specifications. These specifications are covered in this publication, NIST IR 7799. Layer 3 provides specifications for combining multiple CM data domains and extracting knowledge from the synthesized data sets. Layer 2 provides specifications for binding the model to specific CM data domains and also covers data domain specific requirements for the model. Layer 2 and 3 will be covered by NIST IR 7800, although the initial drafts of NIST IR 7800 will only include layer 2 specifications. These specifications will enable implementations of the model to continuously monitor the asset management, configuration management, and vulnerability management domains. Layer 3 specifications will be developed in the future for more advanced data domain integration and analysis capabilities and are not necessary for initial implementations of the model. Layer 1

[4] http://scap.nist.gov
[5] http://scap.nist.gov/specifications/ai

provides foundational communication specifications covering both data domain agnostic communications (e.g., generic reporting wrappers) and data domain specific communications (e.g., vulnerability information). Layer 1 is separated from the other layers in Figure 1 by a dashed line because these specifications have a much broader scope than CM and are necessary for, but not inherently part of, the CM Reference model.

1.3 Product Development Overview

This document enables CM product instrumentation and development. Each subsystem specification provides product development requirements applicable to specific product types. It is not expected, or desired, that any specific product adopt all of the subsystem specifications. Some of the subsystem specifications describe requirements that already exist within many Information Technology (IT) products. Thus, incorporation of these specifications should require only gentle instrumentation for those existing products. In other cases, the subsystems represent new functionality and product types (e.g., multi-product sensor orchestration and tasking and policy content repositories) that do not currently exist on the market. If vendors choose to adopt these specifications, they will likely need to develop new products. To catalyze vendor involvement we are looking into providing functioning prototypes of these capabilities[6].

This document also enables product testing, validation, procurement, and interoperability. A set of atomic requirements can be derived from each subsystem and interface specification. While not yet developed, these requirements would be used by developers to ensure conformance to the specification and by independent testing authorities to perform product validation. We are looking into the possibility of providing these derived requirements along with a product validation program[7]. This validation program, if enacted, would enable the procurement of independently validated tools to interoperate within the CM, providing workflows for a specific set of data domains.

Unfortunately, not all parts of this document are ready for validation or even implementation. Several of the interface specifications cannot be satisfied by existing low-level communication specifications. In particular, the need for CM query and tasking languages, as well as a content retrieval language, are essential to achieving the full use of the model and full implementation of the workflows. We are looking into creating these specifications and have already defined some of the required functionality and major parameters in this document. While the lack of these interface specifications may hinder full use of the model, Appendix A explores what can be accomplished in real world implementations given the existing model. In fact, the state of the art in CM[8] can be achieved along with limited multi-instance communication, hierarchical tiers, additional standardization, and interoperability benefits.

[6] This would be analogous to the Open Vulnerability Assessment Language (OVAL) interpreter that MITRE wrote to both prove the effectiveness of OVAL and to also jumpstart vendor adoption through a provision of open source code (see http://oval.mitre.org/language/interpreter.html).

[7] This would be analogous to the SCAP Validation Program that uses the NIST National Voluntary Laboratory Accreditation Program (NVLAP). See http://scap.nist.gov/validation for more details.

[8] For example, the Department of State's iPost solution can be developed using only the fully specified interfaces (see http://scap.nist.gov/events/2011/cm_workshop/presentations/docx/STREUFERT%20-%20ScoringGuide1%205.docx).

This document is focused on CM and does not discuss remediation. This was done to limit the scope of the work. However, foundational work is being done in the area of remediation automation,[9] separate from this effort. When mature, the emerging remediation specifications can be leveraged by this model to enable both monitoring and remediation.

In summary, the data domain agnostic workflows, subsystem specifications, and interface specifications work together to define an ecosystem in which a variety of interoperable products can be composed together to form CM solutions. If adopted, these specifications will enable teamwork, orchestration, and coordination among CM products that currently work in isolation (or at best aggregate data to a single location or provide coordination only within a single vendor suite). For the computer security domain, this will greatly enhance organizational effectiveness and efficiency in addressing known vulnerabilities and technical policy requirements.

1.4 Document Overview

This report begins in Section 2 with a brief discussion of the subsystem and interface model, including their description and generic purposes. Section 3 describes four workflows that specify necessary functionality that will be used by any implementation of CM regardless of the data domain to be processed (e.g., asset management). Section 4 details the specifications for each subsystem presented in the CM reference model that are necessary for the model to function. Section 5 further describes the interfaces and provides specific requirements for each interface, indicating what events must, or simply may, occur. Section 6 outlines the existing gaps and future work that needs to be performed. Section 7 provides the conclusion. Appendix A discusses implementation options for the CM model. Appendix B summarizes the acronyms used in this report.

2. Subsystem and Interface Model Overview

This section provides an overview of the CM model's subsystems and interfaces.

2.1 Subsystem Overview

There are six distinct subsystems that together compose the CM model. These subsystems are summarized below:

1. **Presentation/Reporting:** This subsystem is the primary user interface for the CM model. It contains a single component called the Dashboard Engine that enables users to create queries, save queries, execute queries, and obtain query results. Executed queries are communicated to the Task Manager for fulfillment. These queries trigger data analysis tasks that compute query results. The queries can also trigger data collection tasks for the collection of new or updated data or even trigger propagation of the query to other CM instances[10]. This latter capability enables querying down a hierarchy of CM instances to retrieve specific data needed at higher level management tiers[11].

[9] http://csrc.nist.gov/publications/drafts/nistir-7670/Draft-NISTIR-7670_Feb2011.pdf
[10] An organization can have one or more CM instances as meets their architectural, technical, and policy needs.
[11] The specifications also allow for queries to be passed between arbitrary instances (e.g. between two instance of the same hierarchical level) but such uses fall outside of the model's scope.

2. **Task Manager:** This subsystem is the central organizer for a CM instance. It orchestrates the operations of the other subsystems in support of queries received from the Presentation/Reporting subsystem as well as queries received from higher-tier CM instance. The orchestration activities include initiating data collection tasks, data analysis tasks, and propagation of queries to other CM instances. The Task Manager is comprised of three components: the Query Orchestrator, the Collection Controller, and the Decision Engine. The Query Orchestrator coordinates query fulfillment among multiple subsystems, thus enabling the CM subsystems to work together as a team in fulfilling specific goals. The Collection Controller receives data collection tasks from the Query Orchestrator and coordinates collection of that data among multiple Collection subsystems. The Decision Engine is a notional component that is used to monitor the implementation of digital policy. The goal is for the Decision Engine to receive digital policy and then monitor compliance against that policy by coordinating collection and analysis of supporting data. The Decision Engine will mature the CM solution beyond responding to human queries and toward full security automation based on digital policy directives.

3. **Collection:** This subsystem detects system state information in accordance with organizational policy by processing data collection tasks. These tasks indicate the policy content to execute (e.g., SCAP benchmark) against a certain set of assets. The policy content may need to be retrieved from the Content subsystem. After collecting the results, the task specifies the required result format and the designated recipient. These data collection tasks may originate locally in order to periodically collect and report on a specific data view. They may also originate from the Collection Controller to enable the CM system to fulfill a specific user query. The results may be sent back through the Collection Controller or directly to the Data Aggregation subsystem.

4. **Data Aggregation:** This subsystem is the central storage repository for the CM system enabling data analysis and query fulfillment. It stores system state information (i.e., raw data), the analysis of raw data compared to policy (findings), the evaluation of findings into numeric metrics (scores), and various types of metadata. This data is stored via four components: the System State Repository, the Asset Repository, the Metrics Repository, and the Metadata Repository. The System State Repository contains raw data (from the Collection subsystems) and findings (cached by the Analysis/Scoring subsystems). The Metrics Repository caches scores generated by the Analysis/Scoring subsystems. The Asset Repository stores standards-based representations of asset data retrieved from Collection subsystems and therefore acts as an aggregation database (it is not itself an asset management tool). Lastly, the metadata repository stores metadata used for de-confliction of overlapping raw data and providing more organizational context to the stored data.

5. **Analysis/Scoring:** This subsystem analyzes data to produce CM users' query results. It does this by processing data analysis tasks arriving from the Query Orchestrator. These tasks indicate the type of analysis to perform (e.g., scoring algorithm) against a specific set of assets. Information on the analysis to be performed may need to be retrieved from

the Content subsystem. Raw data, findings, and scores used as inputs to the analysis will be retrieved from the Data Aggregation subsystem. After analyzing the data, the task will specify the required result format that is to be sent back to the Query Orchestrator. Analysis results are stored in the Data Aggregation subsystem for caching purposes. If any raw data is retrieved during analysis, the subsystem performs data deconfliction services using deconfliction parameters from the Metadata Repository. The distinction between raw data, findings, and scores is essential to the model. The subsystem can compare raw data to policy to create findings (often Boolean values). It then can use findings to create scores (or perhaps more aggregate findings), and then use the aggregated findings (or possibly existing scores) to create new scores.

6. **Content:** This subsystem is a content management system. It stores digital policy and supporting data to enable comparison of system state information against organizational policy and to enable data normalization (enabling correlation of the output of multiple Collection subsystems). The Content subsystem interacts with the Collection subsystems, Analysis/Scoring subsystems, Content subsystems of higher and lower CM tiers, content providers, and content development tools.

A single instance of CM, its subsystems, and associated components are shown in Figure 2. Large organizations may have multiple communicating instances. Extremely large organizations, such as the U.S. Government (USG), may need a large hierarchy of cooperating CM instances to support local operations, higher-level decision makers, and also government-wide security operations (i.e., FISMA, CyberScope).

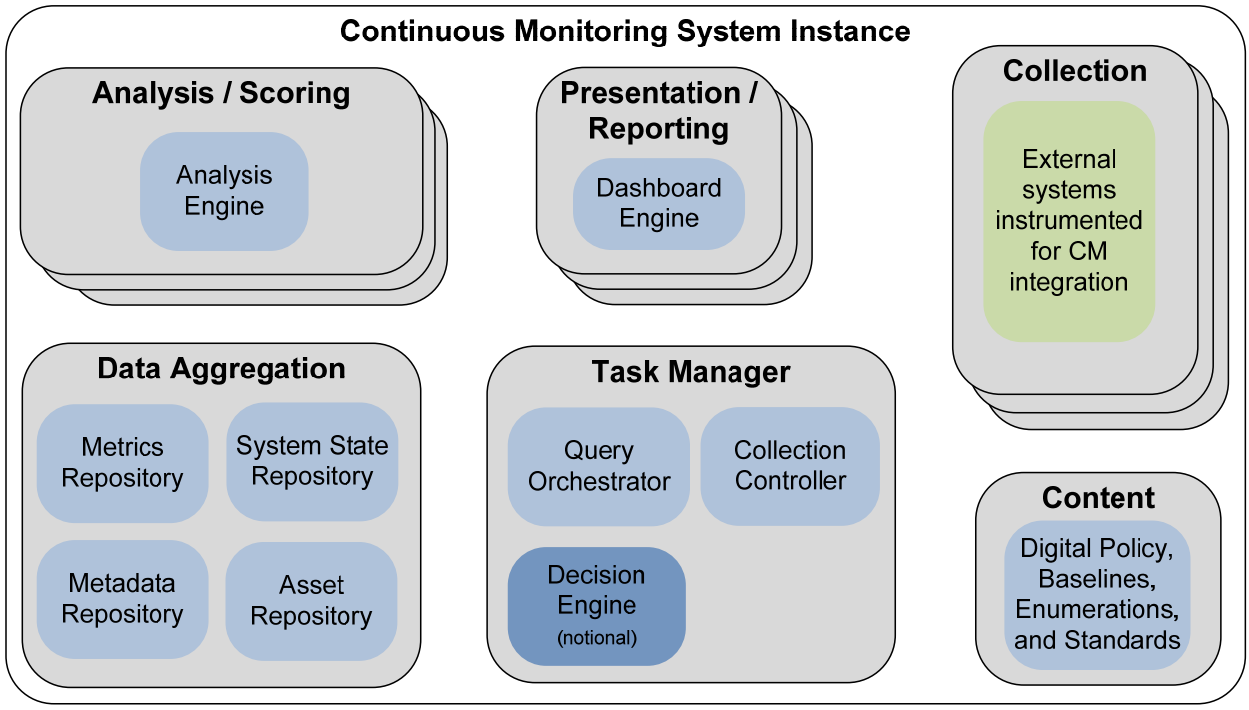


Figure 2. Continuous Monitoring Reference Model Subsystems

2.2 Interface Overview

The CM reference model uses a variety of interfaces (I) to provide communication capabilities between subsystems, both within a CM instance and between multiple CM instances. These interfaces need to be standardized in order to 1) allow for independent development of the subsystems, and 2) for data to be passed successfully throughout the model. Interfaces that are not standardized will need to be addressed through proprietary mechanisms and may result in the necessity to combine multiple subsystems into a single tool or product. The interfaces are numbered in the authors' perception of increasing difficulty of development and standardization. The interfaces with a common integer value and decimal parts (e.g., I2.1 and I2.2) are related. They have similar functionality and often will share communication parameters. The interfaces are summarized below:

- I1 **Result Reporting:** This interface enables reporting of data (e.g., collected raw data or analyzed query results).
- I2 **Content Acquisition:** This interface enables the retrieval of content (digital policy and supporting data) as well as supporting the operations of insertion, modification, and deletion.
  - I2.1 This interface is a subset of I2 that enables content retrieval.
  - I2.2 This interface is a subset of I2 that enables the updating of content in a content repository.
- I3 **Querying and Tasking:** This interface enables both querying and tasking between subsystems.
  - I3.1 This interface is a subset of I3 that enables querying for specified results.
  - I3.2 This interface is a subset of I3 that enables tasking for the collection of specific data (often used to support fulfillment of an I3.1 query).
  - I3.3 This interface is a subset of I3 that enables tasking for the analysis of specific data (often used to support fulfillment of an I3.1 query).
- I4 **Advanced Data Retrieval:** This interface enables the retrieval of data from data repositories using complex descriptors (analogous to a SQL query but without relying on database schemas).

Figure 3 identifies the interfaces used within a CM instance and Figure 4 identifies interfaces used between two CM instances.

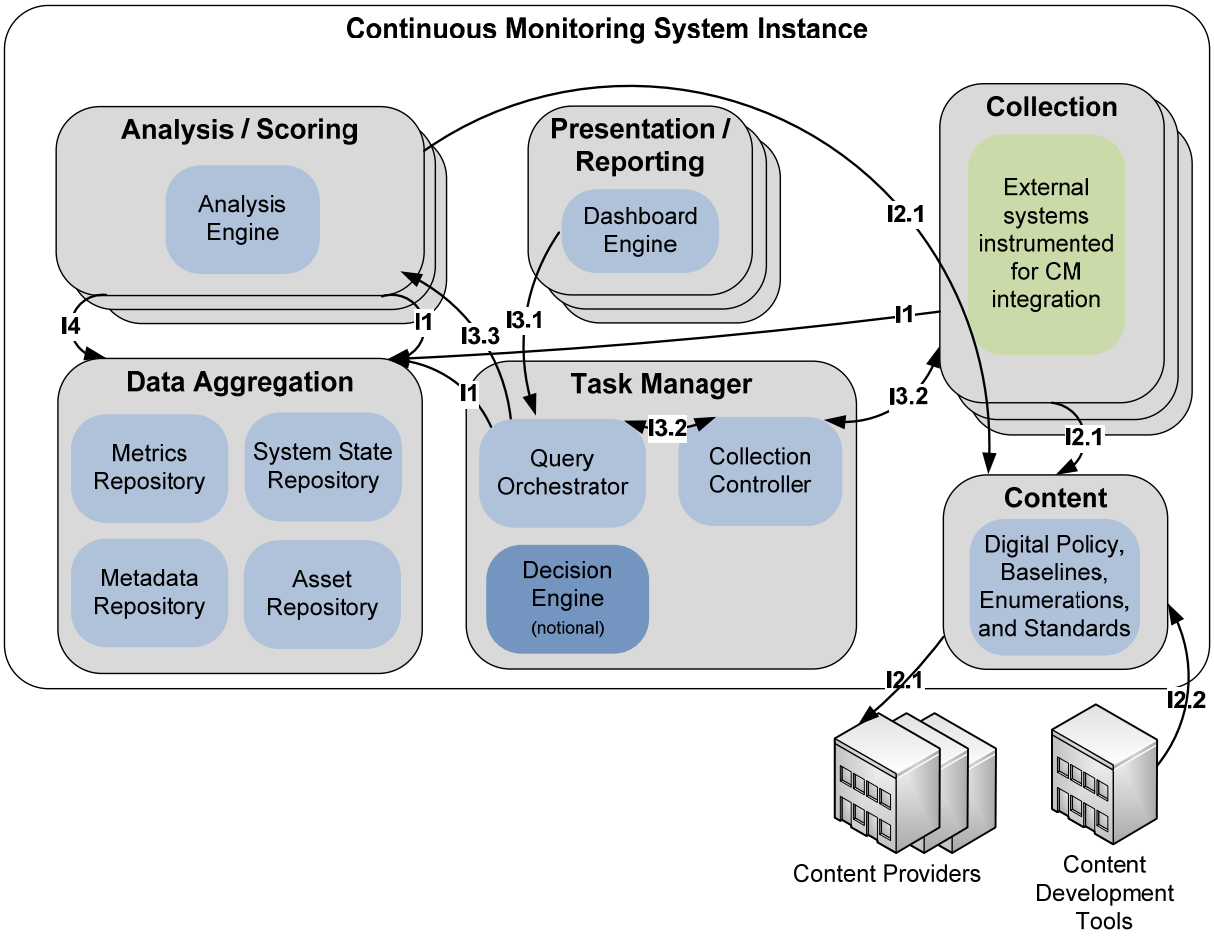


Figure 3. Continuous Monitoring Instance Model with Interfaces

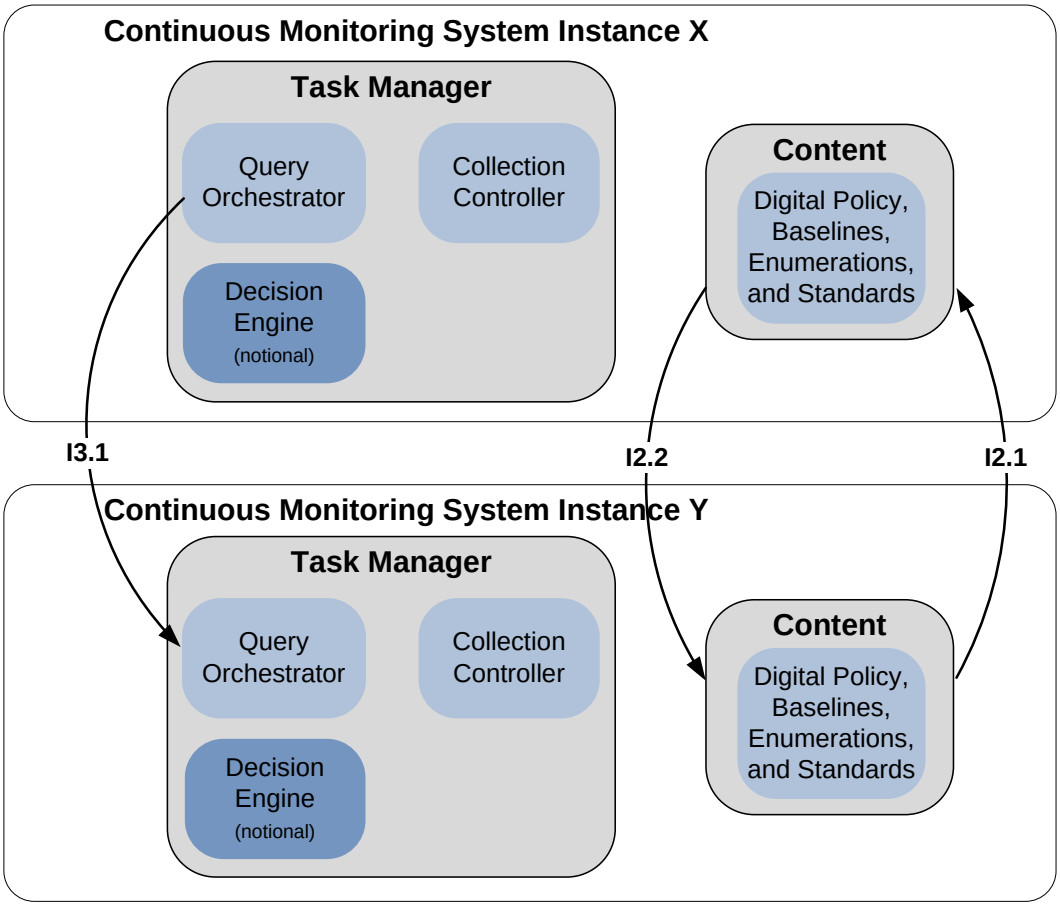


Figure 4. Continuous Monitoring Multi-Instance Model with Interfaces

Below are descriptions of each interface instance depicted in figures 3 and 4. These descriptions, along with the previous summarization of the subsystems, should provide the reader a general understanding of data movement and processing within the CM reference model. The workflows described in section 3 will provide greater specificity regarding this data movement and processing. The subsystem specifications in section 4 and the interface specifications in section 5 will enable actual implementation.

I1 Interface Instances:

- The Collection subsystem is depositing data into the Data Aggregation subsystem. This is usually raw data but may be findings (evaluation of raw data against specific policies). Note that this interface covers both security and non-security data collection (including asset inventories).
- The Query Orchestrator component is depositing data into the Data Aggregation subsystem. Typically this is query result data from lower tier CM instances; however, it could also be Collection subsystem data collection results being passed through by the Collection Controller.

- The Analysis Engine component is depositing calculated results into the Data Aggregation subsystem. This caches the calculated results for future use and may include both findings and scores (the evaluation of findings to produce numerical measurements).

I2.1 Interface Instances:

- The Collection subsystem is communicating with the Content subsystem to retrieve digital policy and supporting data (e.g., host configuration baseline or lists of authorized software) that will enable it to fulfill a data collection task.
- The Analysis/Scoring subsystem is communicating with the Content subsystem to retrieve needed digital policy (e.g., scoring algorithms and scoring parameters) that will enable it to fulfill an analysis task. The retrieval of scoring parameters enables customization of the scoring for particular environments, systems, or scenarios.
- The Content subsystem is obtaining content from external Content Providers (possibly to enable it to fulfill a content request from another subsystem). The retrieved content may be needed digital policies, but it may also be supporting data (e.g., vulnerability listings from the National Vulnerability Database).
- The Content subsystem of a lower tier CM instance is acquiring digital policy and supporting content from a higher tier CM instance. Alternately, a higher tier CM instance may be pushing content down to lower tier CM instances.

I2.2 Interface Instance:

- Content Development Tools are inserting, modifying, and deleting digital policies and supporting content in the Content subsystem.

I3.1 Interface Instances:

- The Dashboard Engine component is sending queries to the Query Orchestrator component and is expecting to receive the query results in reply.
- The Query Orchestrator of a higher tier CM instance is sending queries to the Query Orchestrator component of a lower tier CM instance and is expecting to receive the query results in reply.

I3.2 Interface Instances:

- The Query Orchestrator component is sending data collection tasks to the Collection Controller component. In return, it expects to receive status on the data collection activity and it may receive the data collection results depending upon the implementation. These data collection tasks are typically created to support the fulfillment of an I3.1 query.
- The Collection Controller component is sending data collection tasks to the Collection subsystems. In return, it expects to receive status on the data collection activity and it may receive the data collection results depending upon the implementation (note that the I1 interface also enables the Collection subsystem to report data collection results).

I3.3 Interface Instance:

- The Query Orchestrator component is sending data analysis tasking to the Analysis Engine component. In return, it expects to receive the analysis results. These analysis tasks are typically created to support the fulfillment of an I3.1 query.

I4 Interface Instance:

- The Analysis Engine component is issuing data retrieval requests to the Data Aggregation subsystem. The scope of these requests includes raw data, findings, and generated scores as well configuration parameters for data deconfliction activities.

3. Required Workflows

This section describes a required set of CM workflows (WF) that must exist within any conformant CM reference model implementation. These workflows are data domain agnostic in that they apply to any CM data domain being monitored. They cover multi-instance CM implementations but also apply to a single instance implementation.

There are four complementary workflows presented in this section:

WF1. **Data Acquisition**: This workflow describes how raw data is collected and reported to a central repository within a single CM instance.

WF2. **Query Fulfillment**: This workflow describes how query requests are fulfilled in both single and multi-instance CM architectures. Query fulfillment may include propagation of the query to lower level CM instances, data collection activities, and analysis of collected data.

WF3. **Digital Policy Retrieval**: This workflow describes how digital policy and supporting content is acquired or updated from higher tier CM instances and external content repositories.

WF4. **Digital Policy Propagation**: This workflow describes how digital policy and supporting content is propagated from a higher tier CM instance to lower tier CM instances.

These workflows describe requirements for data movement within the entire CM reference model. These workflow requirements then drive the six, lower-level subsystem specifications provided in section 4. The subsystem specifications fulfill and flesh out in more detail the workflow requirements. To ensure coverage of the workflows, the subsystem specifications have their requirements mapped back to the workflow steps. This tight coupling of the subsystem specifications to the workflow steps means that CM product developers need only to implement applicable subsystems specifications to ensure that their tools play their necessary roles within the workflows.

3.1 Data Acquisition

Workflow Name	Data Acquisition
Workflow ID	WF1
Scope	Intra-instance
Primary Actor	Collection subsystem
Secondary Actors	Task Manager, Content, and Data Aggregation subsystems
Brief Description	This workflow describes how raw data is collected and reported to a central repository within a single CM instance. The Collection subsystem is tasked to retrieve data, possibly using a cached local repository, and then sends the data to the Data Aggregation subsystem. If necessary, the Collection subsystem's digital policy and supporting content is updated through communication with the Content subsystem.
Triggers	**Trigger 1**: This workflow may be triggered periodically by the Collection subsystems where they task themselves to regularly collect and update data views for a set of assets. **Trigger 2**: This workflow may be triggered by the Collection Controller sending a collection task to the applicable Collection subsystems to retrieve specific data.
Parameters	**1. Query Identifier**: This parameter describes a CM multi-instance implementation unique name for the query being supported by this activity. **2. Content Descriptor**: This parameter describes the content to use to collect the requested data. 3. **Task Result Descriptor**: This parameter describes the reporting format for the collected data. 4. **Asset Descriptor**: This parameter describes the population of assets to which the query applies. This must be an explicit list of assets and not an abstract asset population (e.g., not "all routers"). Future versions of this workflow may allow for more general asset descriptors depending upon the maturity of the associated interface specifications. 5. **Data Age**: This parameter describes the required freshness of the collected data. 6. **Policy Age**: This parameter describes the required freshness of the applied digital policy and supporting content (originating from the Content subsystem). If set to 0, the digital policy is always updated prior to performing collection activities.

Workflow Name	Data Acquisition
Flow of Events	1) The **Collection** subsystem uses the Content Descriptor parameter to retrieve relevant digital policy (e.g., benchmarks) from the **Content** subsystem if the content is not already available or if the local content is stale according to the Policy Age parameter. 2) The **Collection** subsystem initiates collection of any requested "raw data"[12] that is not available in its local repository (if one exists) or that is considered stale according to the Data Age parameter. To do this, the subsystem uses the Asset Descriptor parameter to determine from what assets to collect the data. If all the requested "raw data" is available in the local repository and that data meets this freshness requirement, then no new data needs to be collected. 3) The **Collection** subsystem sends the collected data to the **Data Aggregation** subsystem using the format specified by the Task Result Descriptor parameter. 4) The **Data Aggregation** subsystem populates the **System State Repository** component and/or **Asset Repository** component with the received data. The data is tagged with the Query Identifier parameter.
Post-Conditions	**Trigger 1**: Data collection that was locally specified in the Collection subsystem has been performed. The data has been stored and is ready for analysis. **Trigger 2**: Data collection, which was initiated by the Collection Controller in response to a user query, has been performed. The data has been stored and is ready for analysis.

Figure 5 shows WF1 and its two triggers. The triggers provide the Collection subsystem a data collection task. The subsequent flow of events are: (1) retrieve the content needed to support the task; (2) retrieve or collect the required data from a local repository; (3) send the resulting data to the Data Aggregation subsystem; and (4) then the Data Aggregation subsystem stores and tags the data in the appropriate repositories. Additionally, please see Appendix C for the WF1 diagram.

[12] The requested data here may be "findings" (results created through the application of specific digital policy to raw data) instead of "raw data". This is not the preferred approach but may be necessary to accommodate limitations of the collection tools.

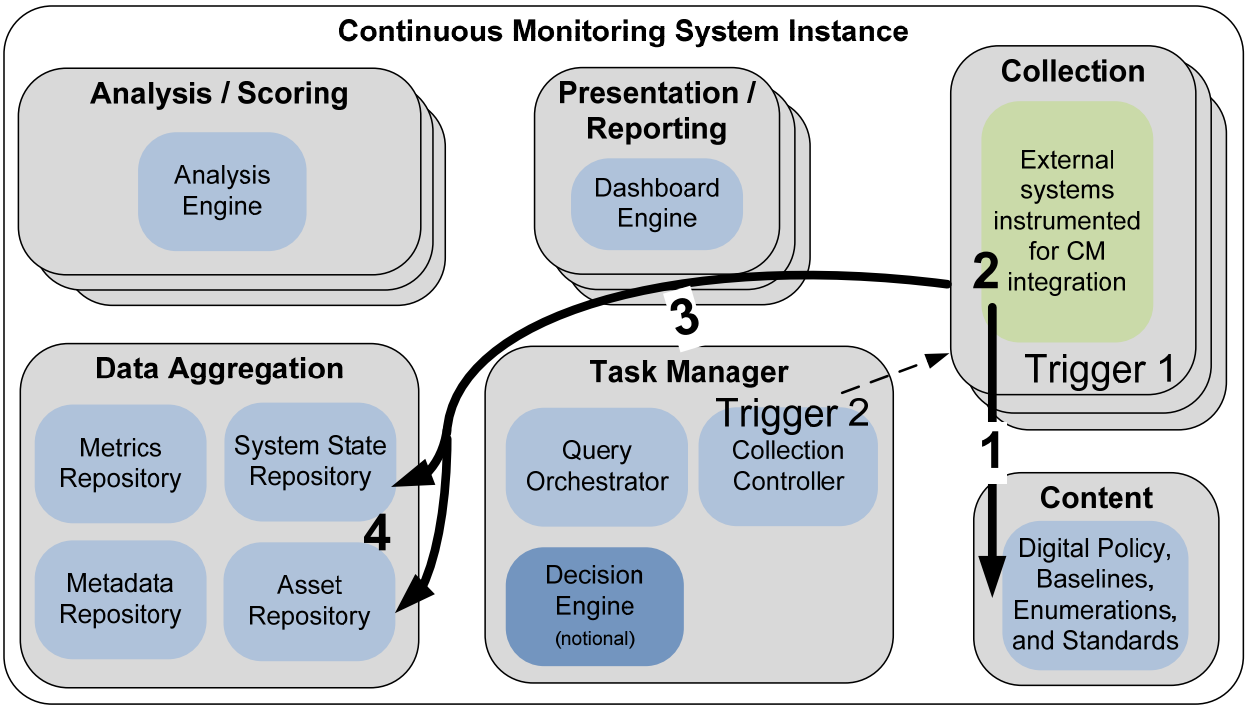


Figure 5. Data Acquisition Workflow

3.2 Query Fulfillment

Workflow Name	Query Fulfillment
Workflow ID	WF2
Scope	Intra-instance and Inter-instance
Primary Actor	Task Manager Subsystem
Secondary Actors	Presentation/Reporting, Analysis/Scoring, Data Aggregation, and Content subsystems along with the Query Orchestrators of higher and lower tiered CM instances
Brief Description	This workflow describes how CM query requests are fulfilled in both single and multi-instance CM architectures. The Query Orchestrator may receive requests from multiple sources and it determines whether or not they are allowed to be executed. If yes, it orchestrates fulfillment of the query. This may include propagating the query to lower tier CM instances (using WF2 recursively), sending collection tasks to its Collection subsystems to gather data (invoking WF1), and sending analysis tasks to the Analysis/Scoring subsystem to evaluate gathered data. Ultimately, it attempts to return the query results to the requestor.

Workflow Name	Query Fulfillment
Parameters	1. **Query Identifier**: This parameter describes a CM multi-instance implementation unique name for the query being supported by this activity. 2. **Task Identifier**: This parameter describes a CM multi-instance implementation unique name for the task being supported by this activity. 3. **Asset Descriptor**: This parameter describes the population of assets to which the query applies. This can be simply a list of assets but may also be a more abstract characterization (e.g., all printers). 4. **Content Descriptor**: This parameter describes the content to use to collect the query information. The content may be provided inline within this parameter or a pointer may be provided to the required content (e.g., to content within the Content subsystem). 5. **Query Result Descriptor**: This parameter describes the query results report. This includes the format of the report as well as the level of detail or abstraction that must be returned. 6. **Task Result Descriptor**: This parameter describes the data collection reports that are used to gather data to support query analysis. This includes the format of the report as well as the level of detail that must be returned. This descriptor must match the required inputs for the analysis algorithm specified in the Analysis Descriptor. 7. **Analysis Descriptor**: This parameter describes the analysis procedure to use in generating the requested query results. This includes specifying the algorithm inputs (e.g., raw data elements), analysis algorithm (i.e., calculations), and output data. It also includes any parameters used to modify the behavior of the analysis algorithm (e.g., modifying the weightings of variables). This parameter can specify all of this data by using a named references (of data stored in the Content subsystem or even hardcoded into the Analysis/Scoring subsystem). The parameter may also contain the data itself as XML blobs. 8. **Instance Depth**: This parameter describes the CM instance depth to which the query should be propagated. This should be set to 0 if the query should not be propagated. A setting of 1 propagates the query down 1 tier. A setting of -1 propagates the query all the way to the leaf CM instances regardless of the hierarchy depth. 9. **Data Age**: This parameter describes the required freshness of the collected data. If the data is not sufficiently fresh, it cannot be used as input to the scoring algorithm and must be first refreshed through a data collection activity. 10. **Policy Age**: This parameter describes the required freshness of the applied content. If the content being used (e.g., by the Collection subsystem) is not sufficiently fresh, it cannot be used for data collection and must be first refreshed through content retrieval (usually from a

Workflow Name	Query Fulfillment
	Content subsystem). Note, if the Content Descriptor contains the content itself, as opposed to referencing content, then this parameter does not apply and is not used. In such cases, the parameter should be set to -1. If the parameter is set to 0, the content must always be updated prior to performing data collection. 11. **Collect Bit**: This parameter describes whether or not to initiate collection of data based on this query. It is binary with 0 meaning that data should not be collected and 1 meaning that data should be collected. If this parameter is set to 0 and data needs to be collected in order to provide the query results, the query will fail to provide the requested results.
Triggers	**Trigger 1**: The Presentation/Reporting subsystem sends a query to the Query Orchestrator component within its own CM instance. **Trigger 2**: The Query Orchestrator of a higher tier CM instance sends a query to the Query Orchestrator of a lower tier CM instance.
Flow of Events	1) The **Query Orchestrator** resolves the Asset Descriptor parameter into a set of specific assets for the current CM instance. It does this by creating an analysis task (that resolves the Asset Descriptor) and then sends it to the **Analysis/Scoring** subsystem for resolution. Jump to the ***Analysis Procedure*** below and return here when it has completed. 2) The **Query Orchestrator** determines whether or not the query will be allowed to execute. Based on local policy, the request may be denied, require human approval, be executed automatically, or require special scheduling. If approval to execute is obtained, then proceed to step 3 according to the scheduling set up in this step (if any). If the request is denied, return an error code and stop this procedure. 3) The **Query Orchestrator** checks to see if the **Analysis/Scoring** subsystem can satisfy the query using data already available in the **Data Aggregation** subsystem. It does this by decomposing the query into an analysis task that will calculate the query results (primarily using the Analysis Descriptor, Query Identifier, and Asset Descriptor parameters) and sends it to the **Analysis/Scoring** subsystem. Jump to the ***Analysis Procedure*** below and return here when it has completed. If the query results are returned then skip to step 7. 4) If the Instance Depth parameter is not 0, then the query must be sent to lower tier CM instances (if any exist):

Workflow Name	Query Fulfillment
	a. The **Query Orchestrator** sends the query to the **Query Orchestrators** of the appropriate continuous monitoring instances with the Instance Depth parameter decremented by 1. This action invokes WF2 recursively. b. The **Query Orchestrator** receives returned query results or error codes from the relevant continuous monitoring instances. c. The **Query Orchestrator** stores the retrieved result data in the **Data Aggregation** subsystem. 5) If the Collect Bit parameter is set to 1, then the **Query Orchestrator** must task the Collection subsystems of the current CM instance to collect the data needed to fulfill the query[13]. To do this, the **Query Orchestrator** derives a data collection task from the query (primarily using the parameters Content Descriptor, Asset Descriptor, Data Age, Policy Age, and Task Result Descriptor). This task is sent to the **Collection Controller**: a. The **Collection Controller** determines which **Collection** subsystems are to be sent data collection tasks and assigns a subtask to each relevant **Collection** subsystem (each subtask is tagged with the supported Task Identifier and Query Identifier). b. The **Collection Controller** sends the subtasks to the identified **Collection** subsystems. This action invokes WF1 trigger 2. c. As data retrieval is completed for all identified **Collection** subsystems, the **Collection** subsystems notify the **Collection Controller** as to the subtasks success or failure[14]. d. The **Collection Controller** notifies the **Query Orchestrator** when the task and derived subtasks[15] are completed or it sends back an error code. 6) The **Query Orchestrator** now must obtain the query results[16]. It does this by decomposing the query into an analysis task that will calculate the query results (primarily using the Analysis Descriptor, Query Identifier, and Asset Descriptor parameters). It then sends the task to the

[13] Steps 4 and 5 can be implemented in parallel as there are no dependencies between them.
[14] The collected data may also be returned depending upon the desired implementation model (see Appendix A).
[15] The collected data may also be returned depending upon the desired implementation model (see Appendix A).
[16] At this point, all the data to analyze the query should be available in the **Data Aggregation** subsystem.

Workflow Name	Query Fulfillment
	Analysis/Scoring subsystem. Jump to the ***Analysis Procedure*** below and return here when it has responded with the query results. 7) The **Query Orchestrator** returns the query results, if any, to the requestor formatted according to the Query Result Descriptor parameter. This includes any results retrieved from lower level CM instances as well as results obtained from the local CM instance. 8) If the requestor is the **Presentation/Reporting** subsystem, then the **Presentation/Reporting** subsystem presents the results to the user in the form requested (e.g., dashboard or report). ***Analysis Procedure***: A.1) The **Analysis/Scoring** subsystem retrieves updated analysis policy and content from the **Content** subsystem (if necessary) based on the Analysis Descriptor parameter. This content will specify the scoring algorithm and any adjustable parameters[17]. A.2) The **Analysis/Scoring** subsystem must determine whether or not the analysis task results are already available in the **Data Aggregation** subsystem (without having to do any additional analysis)[18]. If the results are retrieved, then skip to step A.5. The **Analysis/Scoring** subsystem decomposes the analysis task into a set of sequenced subtasks. These subtasks attempt to create the needed analysis task results from available "raw data" and "findings[19]" in the **Data Aggregation** subsystem. These tasks create "findings" from "raw data" and create "scores[20]" from "findings" as needed to support the generation of the results[21]. If the analysis task is a request for "findings", then there are no "scoring" data elements needed. A.3) The **Analysis/Scoring** subsystem executes the "findings" tasks (creating "findings" from "raw data" or creating new "findings" from existing "findings") in sequenced order as described below: a. The **Analysis Engine** retrieves any needed "raw data" from the **System State Repository** component as well as

[17] These scoring algorithm parameters may be used to adjust the scoring based on the current environment and scoring needs.
[18] This step will cover queries that are simple requests for "raw data" (e.g., lists of assets) and requests for findings and scorings where the data has already been calculated and is considered fresh according to the Data Age parameter.
[19] Findings are the result of applying specific policy to raw data to determine compliance to the policy.
[20] Scores are the results of evaluating findings to create a numerical metric.
[21] This division of data into "raw data", "findings", and "scores" is a core part of the CM model.

Workflow Name	Query Fulfillment
	deconfliction rules from the **Metadata Repository** component. b. The **Analysis Engine** deconflicts and refines any "raw data" to prepare it for analysis. c. The **Analysis Engine** retrieves any needed "findings" from the **System State Repository.** Unlike the raw data, the findings do not need to be deconflicted. d. The **Analysis Engine** creates the required "findings" (either from the raw data, existing findings, or a combination of the two). e. The **Analysis Engine** populates **System State Repository** with "finding" results. "Findings" that will be part of the ultimate analysis task response will persist within the **Analysis/Scoring** subsystem until A.5 when the results are returned. A.4) The **Analysis/Scoring** subsystem executes the "scoring" tasks (if any) in sequenced order as described below: a. The **Analysis Engine** retrieves the required "findings" from the **System State Repository** and "scores" from the **Metrics Repository** that will serve as input to the scoring calculation[22]. b. The **Analysis Engine** calculates the required "scoring" results using the retrieved inputs. c. The **Analysis Engine** populates the **Metrics Repository** with "scoring" results. Scoring calculations that will be part of the ultimate query response will persist within the Analysis/Scoring subsystem until A.5 when the query results are returned. A.5) The **Analysis/Scoring** subsystem returns the analysis task results, if any, to the **Query Orchestrator** in the form specified by the Query Result Descriptor. (RETURN NOW TO CALLING STEP)

[22] Both findings and scorings may be retrieved here because a scoring result can be based not just on findings but on other scoring results.

Workflow Name	Query Fulfillment
Post-Conditions	The query initiator (i.e., a CM user of a Presentation/Reporting subsystem or a higher level CM instance) has received the requested query results.

Figure 6 shows the flow of events for WF2 and its two triggers. Each trigger provides a query to the Task Manager. The subsequent flow of events are: (1) determine the set of applicable assets for this CM instance, (2) make a policy decision whether or not to let the query execute, (3) attempt to generate the query response from existing data, (4) obtain query result data from lower tier CM instances, (5) collect relevant data from the local CM instance, (6) calculate the query results, and (7) provide the results to the requestor. See Appendix C for WF2 and Analysis Procedure diagrams.

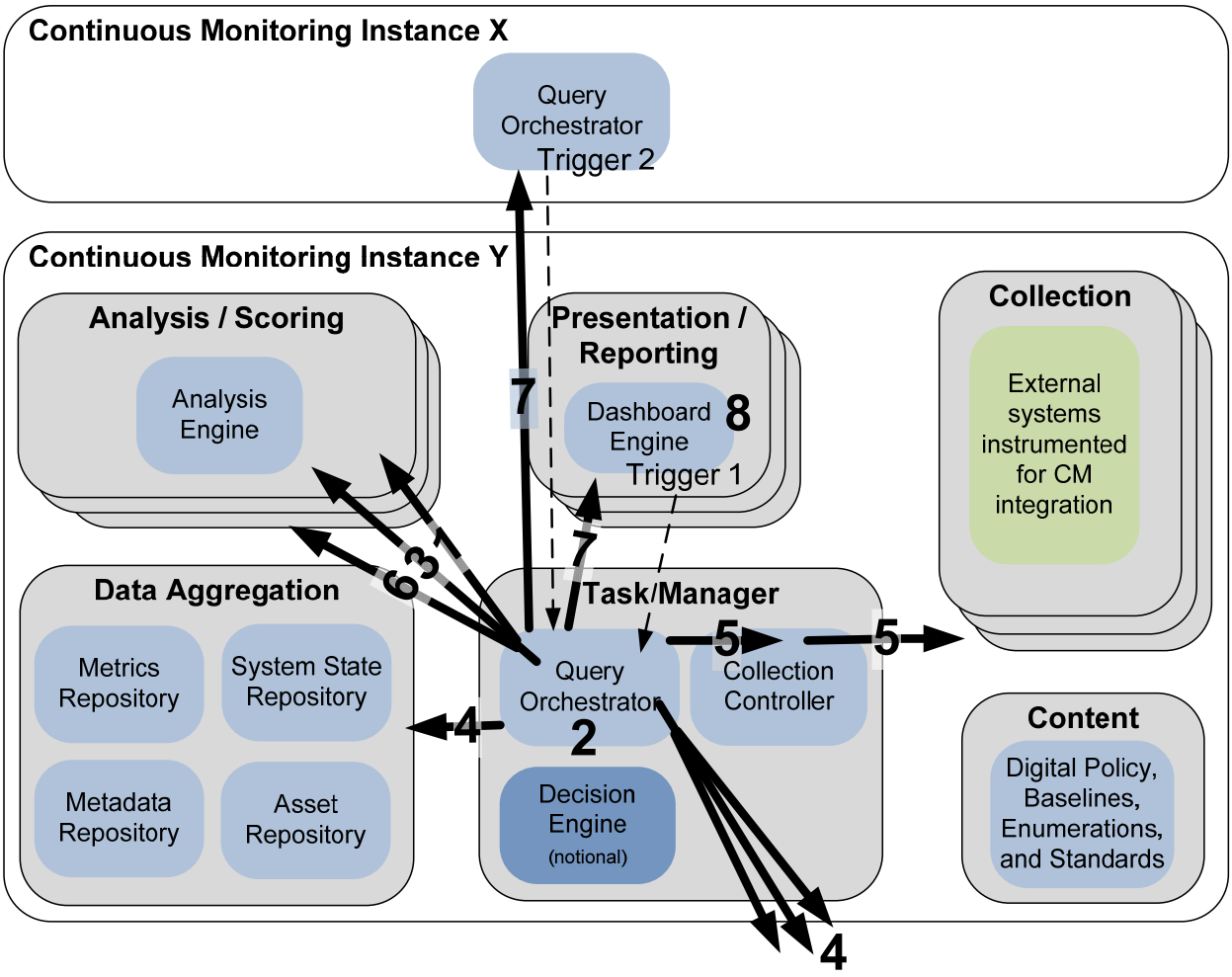


Figure 6. Query Fulfillment Workflow

Figure 7 shows the flow of events for the WF2 Analysis Procedure and its calling step. The calling step provides an analysis task to the Analysis/Scoring subsystem. The subsequent flow of

events are: (A.1) the relevant analysis policy is retrieved, (A.2) attempt to retrieve a pre-calculated result for this query, (A.3) generate the necessary findings from deconflicted raw data and possibly other findings, (A.4) calculate the necessary scores from findings and possibly other scores, and (A.5) return the query results to the requestor.

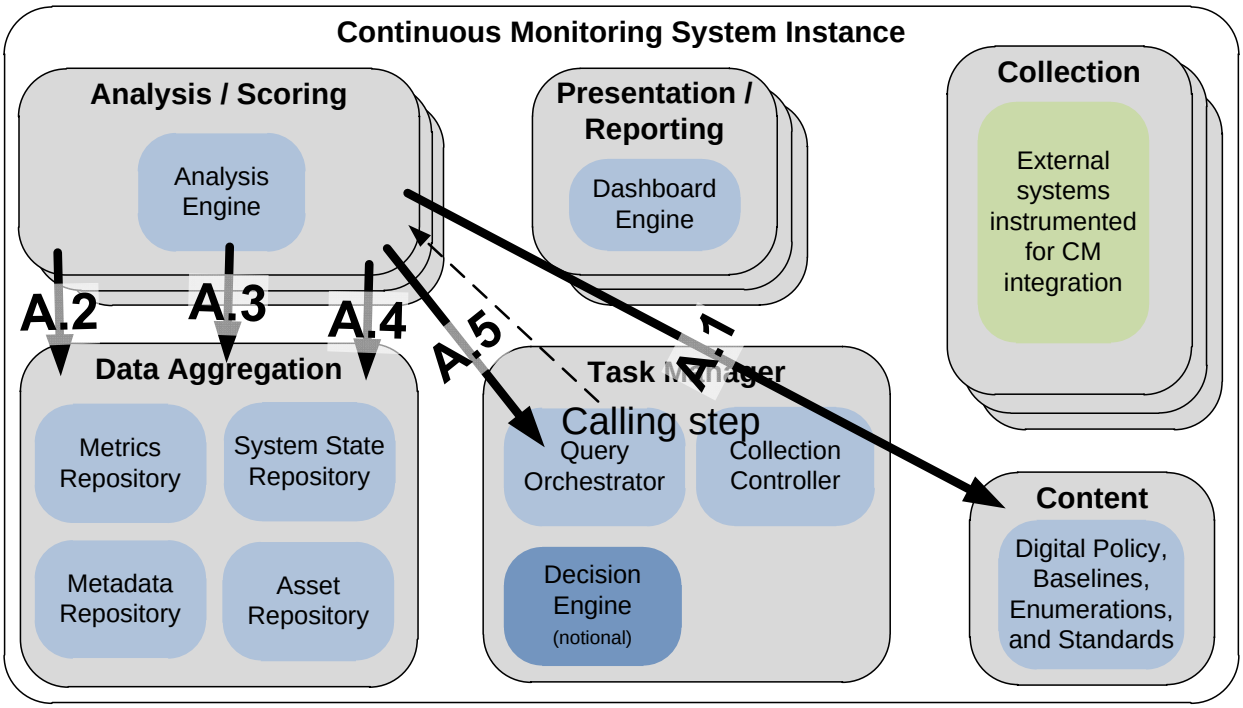


Figure 7. Query Fulfillment Workflow - Analysis Procedure

3.3 Digital Policy Retrieval

Workflow Name	Digital Policy Retrieval
Workflow ID	WF3
Scope	Inter-instance
Primary Actor	Content subsystem
Secondary Actors	Content subsystems of higher and lower tiered CM instances
Brief Description	This workflow describes how a Content subsystem can update itself with new or updated digital policy (and supporting content) from higher tier CM instances and from external content providers. This workflow is applicable to both single instance and multi-instance CM architectures.

Workflow Name	Digital Policy Retrieval
Parameters	1. **Retrieval Identifier**: This parameter describes a CM multi-instance implementation unique name for this retrieval action. 2. **Content Descriptor**: This parameter describes the needed content. 3. **Policy Age**: This parameter describes the required freshness of the applied content. 4. **Instance Depth**: This parameter describes to how many CM instance tiers the request should be propagated. This should be set to 0 if the request is not to be sent to a higher tier CM instance. If it is set to -1, the request is propagated until it reaches a root instance. 5. **Collect Bit**: This parameter describes whether or not to initiate retrieval of the digital policy and supporting content from entities external to the overall CM system. It is binary with 0 meaning that data should not be retrieved and 1 meaning that data should be retrieved.
Triggers	**Trigger 1**: A Content subsystem tasks itself to retrieve (or update) specific content. This will typically be tasking created as a result of a WF2 content request where the Content subsystem doesn't have the needed data. It could also be a periodic content update set up in the Content administration console. **Trigger 2**: A Content subsystem of a lower tier CM instance sends a task to a Content subsystem of a higher tier CM instance to retrieve specific content. This trigger is ONLY used for recursive calls within the workflow.
Flow of Events	1) The **Content** subsystem attempts to retrieve the requested content from the local content repository (using the Content Descriptor). If the content is available and is not stale according to the Policy Age parameter, then skip to step 4. 2) If the Collect bit is 1, the **Content** subsystem attempts to retrieve the requested content from a list of content repositories external to the CM implementation (specified in the **Content** subsystem administration console). **a.** Each external system is sequentially contacted to attempt to retrieve the data (specified by the Content Descriptor and the Policy Age parameters). **b.** If the requested content is returned, store it in the local content repository and skip to step 4. 3) If the Instance Depth is not 0, the **Content** subsystem attempts to

Workflow Name	Digital Policy Retrieval
	retrieve the requested content from the **Content** subsystem in a higher tier CM instance (if one exists). The location of the higher tier CM instance is specified in the Content subsystem administration console. **a.** The subsystem recursively calls WF3 using trigger 2 to request the data from the higher tier. In this call the Instance Depth parameter must be decremented by 1. **b.** If the requested content is returned, store it in the local content repository and skip to step 4. 4) If the content was retrieved, return a success status and the retrieved content to the requestor[23]. Otherwise, return a failure status.
Post-Conditions	The Content subsystem has updated itself with the needed digital policy and/or supporting content.

Figure 8 shows the flow of events for WF3 and its two triggers. Each trigger provides a content acquisition request to the Content subsystem. The subsequent flow of events are: (1) content is retrieved locally, if available, (2) retrieved from a content provider external to the CM instance, (3) retrieved from a higher tier CM instance, and (4) delivered to the requestor. See Appendix C for WF3 diagram.

[23] If this instance of WF3 was instantiated by trigger 2, the requestor will be an instantiation of WF3 executing at a lower level CM instance (waiting on step 2b). If this instance of WF3 was instantiated by trigger 1, the requestor will be the local Content subsystem.

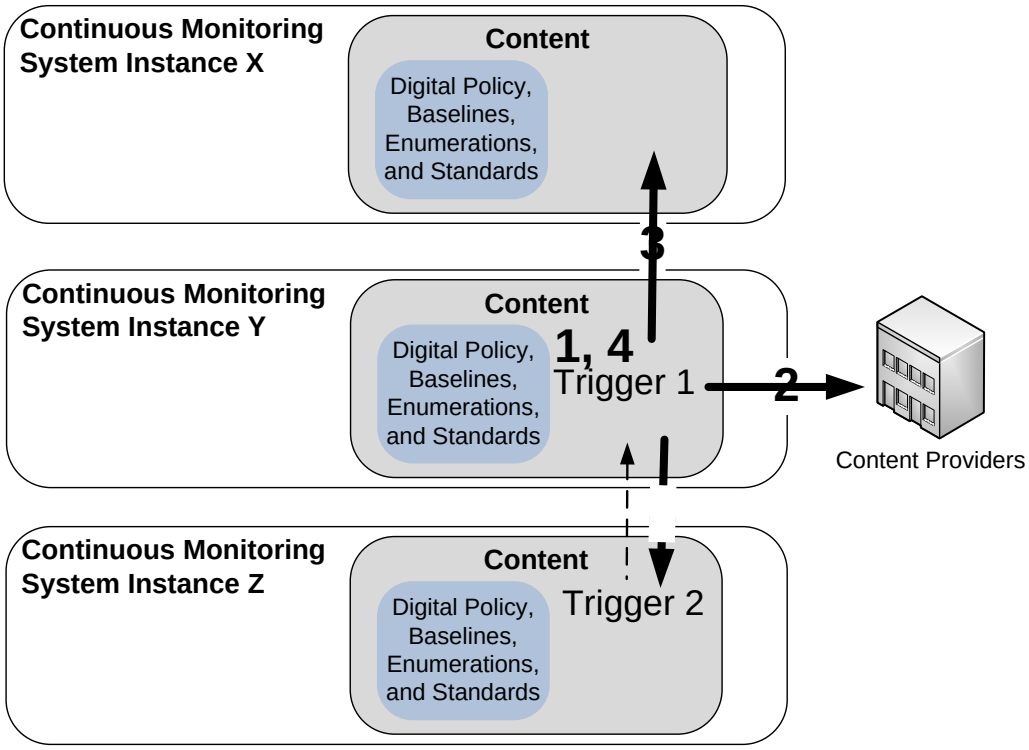


Figure 8. Digital Policy Retrieval Workflow

3.4 Digital Policy Propagation

Workflow Name	Digital Policy Propagation
Workflow ID	WF4
Scope	Inter-instance
Primary Actor	Content subsystem
Secondary Actors	Content subsystems of higher and lower tiered CM instances
Brief Description	This workflow describes how a Content subsystem from a higher tier CM instance can push content to lower tier CM instances. This workflow is applicable only to multi-instance CM architectures.
Parameters	1. **Propagation Identifier**: This parameter describes a CM multi-instance implementation unique name for this content propagation activity. 2. **Content Descriptor**: This parameter contains the actual content data to be propagated. Unlike in WF2 and WF3, it MUST not contain an abstract representation of, or pointers to, the required content.

Workflow Name	Digital Policy Propagation
	3. **Instance Depth**: This parameter describes to how many CM instance tiers the content should be propagated. If set to 0, the content will not be propagated to a lower tier CM instance. If set to -1, the content will be propagated to the CM instance leaf nodes regardless of their depth.
Triggers	**Trigger 1**: A Content subsystem tasks itself to propagate content to lower tier CM instances. This will typically be initiated by an administrator through the Content subsystem administration console. **Trigger 2**: A Content subsystem of a higher tier CM instance propagates content to a Content subsystem of a lower tier CM instance. This trigger is ONLY used for recursive calls within the workflow.
Flow of Events	1) The **Content** subsystem saves the content within the Content Descriptor parameter to its local content repository. 2) If the Instance Depth is not 0, the **Content** subsystem propagates the content to the **Content** subsystems of lower tier CM instances (the locations are specified in the **Content** subsystem administration console). **a.** The subsystem recursively calls WF4 using trigger 2 to propagate content to each of the Content subsystems within lower tier CM instances. When creating the recursive calls, the Instance Depth parameter must be decremented by 1.
Post-Conditions	The specified content has been propagated to the Content subsystems of lower tier CM instances.

Figure 9 shows the flow of events for WF4 and its two triggers. The triggers provide a content propagation activity to the Content subsystem. The subsequent flow of events is: (1) save content to be propagated locally and (2) then send to a lower tier CM instance. See Appendix C for WF4 diagram.

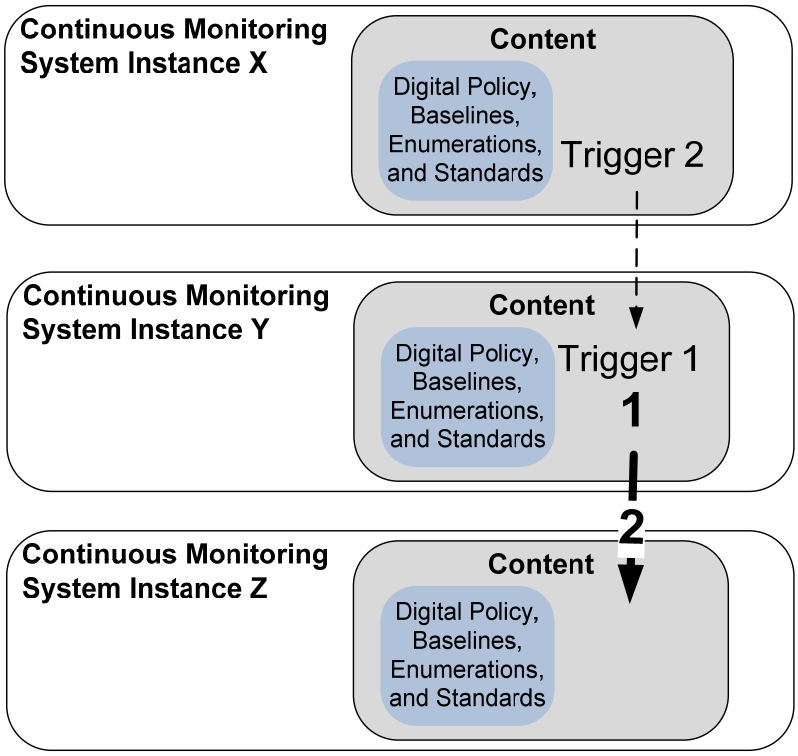


Figure 9. Digital Policy Propagation Workflow

4. Subsystem Specifications

This section provides specifications for each subsystem within the CM reference model. These specifications are data domain agnostic meaning that they apply to any CM implementation regardless of the CM data domain being monitored. An overview of the subsystems is provided in Section 2.1 along with a diagram showing the subsystems and their components within a single CM instance.

These specifications are provided in six independent sections corresponding to each of the six subsystems. This enables developers to implement a subset of the subsystem specifications (perhaps only one) and have their implementation interoperate with implementations of the other subsystems. Developers implementing more than one subsystem in a product are encouraged to preserve the modular design provided in this section. Interoperability is achieved through each subsystem having clients or services that make use of the interface specifications in section 5.

These specifications contain capabilities that define logical parts of the subsystem. They do not require any specific architecture or capability-based modules to be developed within compatible tools.

4.1 Presentation / Reporting Subsystem Specifications

This section presents the Presentation/Reporting Subsystem specifications. The Presentation/Reporting Subsystem is composed of one component for which we provide specifications: the Dashboard Engine.

4.1.1 Dashboard Engine Capabilities

The Dashboard Engine component interacts with the following CM reference model entities:

1. Query Orchestrator component
2. Dashboard and/or user console

In order to implement the necessary functions, the Dashboard Engine MUST provide the following capabilities:

1. **User Request**: The Dashboard Engine provides a mechanism for users to initiate queries.
2. **Query Request and Result Retrieval**: The Dashboard Engine sends out data requests for fulfillment and then retrieves the processed query results.
3. **Query Result Presentation**: The Dashboard Engine presents the results to the user.

Within the Presentation/Reporting subsystem, these capabilities call each other as shown below in Figure 10. These relationships are described in the following subsystem specifications.

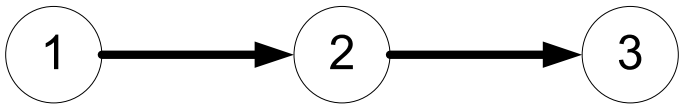


Figure 10: Presentation/Reporting Subsystem Capability Interactions

4.1.1.1 User Request

The User Request capability MUST be able to accept input from authorized users and create queries meeting their specifications. The user MUST have the ability to modify the following query parameters (see the interface 3.1 specifications for details): Asset Descriptor, Content Descriptor, Policy Age, Analysis Descriptor, Query Result Descriptor, Data Age, Collect Bit, and Instance Depth.

When creating a query, users MUST be able to request that the query be saved for future use. When saving a query, the user MUST be required to provide a name that will be used for query retrieval or alternately the subsystem MAY generate a name. Per user direction, the query MUST then to be made available to just that user or to all CM users. The user MAY be provided the ability to specify a select set of users to whom the query will be made available. CM users MUST have the ability to inspect and initiate the saved queries that are available to them. CM users that have saved queries MUST have the ability to delete them. CM administrators MUST have the ability to view all saved queries, see the query creator's username, and delete any saved queries.

All created queries MUST be sent to the Query Request and Result Retrieval capability.

Workflow Steps Supported: WF2, trigger 1

4.1.1.2 Query Request and Result Retrieval

The Query Request and Result Retrieval capability MUST accept queries from the User Request capability. Each received query MUST be sent out for fulfillment (i.e., to the Task Manager's Query Orchestrator). For this interaction the capability acts as a client using the I3.1 interface (see section 5.3.1). Eventually, a response will be received and MUST be forwarded to the Query Result Presentation capability.

Workflow Steps Supported: WF2, trigger 1

4.1.1.3 Query Result Presentation

The Query Results Presentation capability MUST accept interface I3.1 query results (see the interface 3.1 specifications for details). The Query Results Presentation capability MUST present these results to the user and make the original query and its parameters available for inspection. This availability MAY be through a dashboard or as a report depending upon user preference.

The user MUST be able to save query results to a file and retrieve them at a later time.

Workflow Steps Supported: WF2, step 8

4.2 Task Manager Subsystem Specifications

This section presents the Task Manager subsystem specifications. The Task Manager is composed of three components: the Query Orchestrator, the Collection Controller, and the Decision Engine. The Decision Engine is currently notional and will not be addressed further in this specification.

4.2.1 Query Orchestrator Capabilities

The Query Orchestrator component interacts with the following entities within the CM reference model:

1. Presentation/Reporting subsystems
2. Collection Controller component
3. Analysis/Scoring subsystems
4. Data Aggregation subsystem
5. Query Orchestrators of higher and lower tier continuous monitoring instances

In order to implement the necessary interactions, the Query Orchestrator MUST provide the following capabilities:

1. **Query Receipt and Response**: The Query Orchestrator can receive incoming queries for processing and respond with requested results.
2. **Asset Resolution:** The Query Orchestrator can resolve an asset population descriptor into a specific list of assets.
3. **Query Authorization**: The Query Orchestrator can make policy decisions on whether or not to let a query execute and if human approval is required.
4. **Query Fulfillment**: The Query Orchestrator can coordinate query fulfillment through propagating queries, analysis tasks, and collection tasks.
5. **Analysis Task Propagation**: The Query Orchestrator can derive an analysis task from a query and propagate that task to an Analysis/Scoring subsystem to obtain the query results.
6. **Collection Task Propagation**: The Query Orchestrator can derive a data collection task from a query and propagate that task to the Collection controller to gather the data needed to support the query.
7. **Query Propagation**: The Query Orchestrator can forward queries to the appropriate continuous monitoring instances for processing and receive replies containing query results.
8. **Results Publication:** The Query Orchestrator can publish query results for storage.
9. **Query Console**: The Query Orchestrator can implement a console for managing query processing policy and query propagation to other CM instances.

Within the Query Orchestrator, these capabilities call each other as shown below in Figure 11. These relationships are described in the following subsystem specifications.

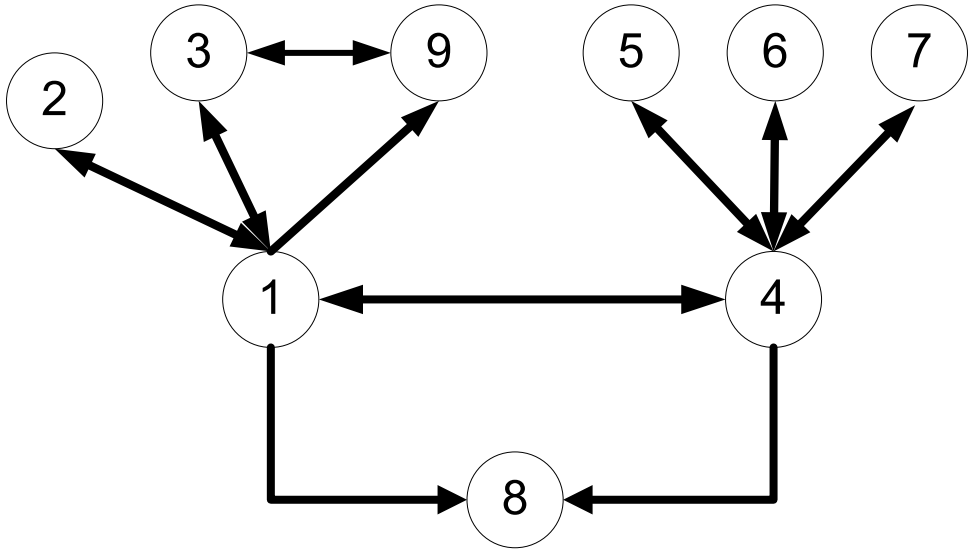


Figure 11: Query Orchestrator Capability Interactions

4.2.1.1 Query Receipt and Response

The Query Receipt and Response capability MUST receive incoming queries for processing and respond with requested results. These queries typically arrive from the Presentation/Reporting subsystem and from the Query Orchestrator component of higher tier CM instances.

To accomplish its objectives, the Query Receipt and Response capability MUST implement a service to accept requests from other subsystems and to provide responses using the I3.1 interface.

When a query is received, it MUST first be passed to the Asset Resolution capability to determine the set of assets within this CM to which the query applies. Next, the query MUST be passed to the Query Authorization capability to check to see if the query is allowed to be processed. If the Asset Resolution capability returns an 'error' or the Query Authorization capability returns a 'denied,' then the capability MUST return an error to the requestor. If the response is 'authorized' then the query MUST be passed to the Query Fulfillment capability. Note, if the Query Authorization capability returns scheduling constraints for the query, these MUST be followed and may result in a delay in passing the query to the Query Fulfillment capability.

Eventually, the Query Receipt and Response capability will receive query result data and/or success or failure status from the Query Fulfillment capability. This Query Receipt and Response capability MUST then respond to the original request with the provided result data (if any) and the success or failure status. If a failure status exists where result data was provided, this indicates a situation where there was partial collection of the requested data.

If the query has been flagged by the Query Authorization capability as needing to be externally published, the Query Receipt and Response capability MUST send the result data to the Results Publication capability.

Workflow Steps Supported: WF2, triggers 1 and 2
Workflow Steps Supported: WF2, steps 1 and 7

4.2.1.2 Asset Resolution

The Asset Resolution capability MUST accept queries from the Query Receipt and Response capability. The Asset Resolution capability MUST take the query's Asset Descriptor parameter and generate a new query to identify the specific list of assets that apply to this CM instance. If the Asset Descriptor parameter contains a general population descriptor (e.g., all routers), then the query MUST resolve this descriptor into a list of assets. If the Asset Descriptor parameter is a list of assets, the query MUST resolve which assets on the list apply to this CM instance. The generated query MUST be sent to the Analysis/Scoring subsystem for resolution. To achieve this, the Asset Resolution capability MUST have a client to send the query over the I3.3 interface.

The response MUST be stored in memory (associated with the query id) to be used by other capabilities. Note that the response must not replace the Asset Descriptor parameter in the query

(at least at this point in the query processing). This is because the query may be passed to other CM instances and the Asset Descriptor parameter was only resolved for this CM instance.

Once a response has been received, the Asset Resolution capability MUST return to the Query Receipt and Response capability with a 'resolved' status along with a pointer to the resolved data. If for some reason, the asset descriptor could not be resolved, the Asset Resolution capability MUST return an 'error.'

Workflow Steps Supported: WF2, step 1

4.2.1.3 Query Authorization

The Query Authorization capability MUST make policy-based decisions on whether or not to let a query execute. The capability will receive queries from the Query Receipt and Response capability.

The Query Authorization capability MUST first decide whether or not the query is allowed to execute. To accomplish this objective, the capability MUST have a policy engine that automatically evaluates incoming queries. Queries MUST be automatically accepted for processing, rejected, or submitted for human review per the policy engine and as configured by the Query Console capability.

If the query is to execute, the Query Authorization capability MUST then decide if there are any scheduling constraints surrounding the execution of the query. Queries MUST be put under scheduling constraints by the policy engine as configured by the Query Console capability.

If the query is to execute, the Query Authorization capability MUST then decide whether or not the query results are to be externally published (apart from providing the results to the requestor). For this, the query MAY contain parameters requesting external publication along with the desired recipients. The Query Console MUST also provide default recipients in case the query external recipient field is empty. The policy engine MUST decide whether or not the query results will be externally published in accordance with policy configured in the Query Console.

The Query Authorization capability MUST then respond to the Query Receipt and Response capability with the 'denied' or 'authorized' status. It MUST also pass along information on scheduling constraints and external publication (if any).

Workflow Steps Supported: WF2, step 2

4.2.1.4 Query Fulfillment

The Query Fulfillment capability MUST coordinate query fulfillment among various Query Orchestrator capabilities and other subsystems. It MUST be able to receive queries from the Query Receipt and Response capability.

Received queries MUST first be sent to the Analysis Task Propagation capability. This action will attempt to retrieve the query results without collecting any new data or retrieving data from

lower tier CM instances. If the query results are obtained, the Query Fulfillment capability MUST pass control back to the Query Receipt and Response capability and provide the query results.

Next, the capability MUST send the query to the Collection Task Propagation capability if the query's Collect Bit parameter is equal to 1. This action will attempt to trigger collection of the data needed for the query results within the current CM instance.

In parallel, the Query Fulfillment capability MUST send the query to the Query Propagation capability if the query Instance Depth parameter is not 0. This action will propagate the query to other CM instances (typically of a lower tier) in an attempt to retrieve their results for this query.

Any result data received from either the Collection Task Propagation capability or the Query Propagation capability MUST be sent to the Results Publication capability. At this point all data should have been collected to support the query.

The Query Fulfillment capability MUST now send the query back to the Analysis Task Propagation capability to get the query results. The response MUST be sent back to the Query Receipt and Response capability.

Workflow Steps Supported: WF2, steps 3, 4, 5, and 6

4.2.1.5 Analysis Task Propagation

The Analysis Task Propagation capability MUST accept queries from the Query Fulfillment capability.

For each received query, the Analysis Task Propagation capability MUST derive an analysis task that will return the requested query results. This is accomplished by taking the query parameters (received over the I3.1 query interface) and creating relevant parameters for the I3.3 analysis task interface (see the Interface section 5 for details).

The Analysis Task Propagation capability MUST implement an I3.3 client. Each derived analysis task MUST be sent out over the I3.3 interface to the Analysis/Scoring subsystem. The response MUST be returned back to the Query Fulfillment capability.

Workflow Steps Supported: WF2, step 3

4.2.1.6 Collection Task Propagation

The Collection Task Propagation capability MUST accept queries from the Query Fulfillment capability.

For each received query, the Collection Task Propagation capability MUST derive a collection task that will trigger collection of the data needed to support query fulfillment. This is accomplished by taking the query parameters (received over the I3.1 query interface) and creating relevant parameters for the I3.2 collection task interface (see the Interface section 5 for

details). This will include replacing the I3.1 Asset Descriptor parameter with the list of assets applicable to this CM instance (as determined previously by the Asset Resolution capability).

The Collection Task Propagation capability MUST implement an I3.2 client. Each derived collection task MUST be sent out over the I3.2 interface to the Collection Controller component. The Collection Controller response MUST be returned back to the Query Fulfillment capability. This response will include overall collection status. It might include returned task results if the Collection subsystems are not sending their results directly to the Data Aggregation subsystem.

Workflow Steps Supported: WF2, step 5

4.2.1.7 Query Propagation

The Query Propagation capability MUST accept queries from the Query Fulfillment capability.

For each received query, the Query Propagation capability MUST forward queries to the appropriate CM instances for processing and then receive replies containing query results. To do this, the Query Propagation capability MUST implement a client that uses the I3.1 interface to send queries. Queries are sent to the Query Orchestrator's of other CM instances (specifically the Query Receipt and Response capability service provided by the other CM instances' Query Orchestrator components).

The Query Propagation capability MUST maintain a list of CM instances to which queries are to be propagated. In a typically hierarchical CM implementation, this will be the set of CM instances at the next lower tier of the hierarchy. This list of propagation instances is maintained by the Query Console capability. Each received query MUST be propagated to every CM instance on the propagation list. Note that the I3.1 "instance depth" parameter within each query controls the extent to which this propagation will occur and is evaluated by the Query Fulfillment capability. When propagating a query, the I3.1 "instance depth" parameter must be decremented by 1.

In some cases, this simplistic algorithm will result in a query being propagated to a CM instance to which the query is irrelevant (e.g., the target assets for the query don't exist in that CM instance). In such cases, the CM instance will simply return an empty result set along with a success status flag since it has no relevant data to be retrieved. More feature-rich propagation algorithms may be explored in the future.

Once responses have been received for all propagations of a query, the Query Propagation capability MUST notify the Query Fulfillment capability as to the overall success or failure of the query propagation activity. If even one propagation failure is received, it results in a failure for the entire propagation activity although useful partial results may still have been retrieved. All retrieved results MUST be sent back to the Query Fulfillment capability.

Workflow Steps Supported: WF2, steps 4

4.2.1.8 Results Publication

The Results Publication capability MUST be able to receive results from other capabilities for publication. To support this, the capability MUST implement a client that can transmit the data over the I1 interface.

If results are received from the Query Fulfillment capability, they MUST be sent to the Data Aggregation subsystem. This data could be query results from other CM instances or collected data from the same CM instance.

If results are received from the Query Receipt and Response capability, they MUST be sent to the approved external publication recipients for that query (see the Query Authorization capability). This enables select query results to be sent to outside entities that implement I1 data receipt services.

Workflow Steps Supported: WF2 step 4c

4.2.1.9 Query Console

The Query Console capability MUST provide a user interface to allow both administrator and user level management of the component. This console MAY be combined with user interfaces from other Task Manager components.

Query Authorization Policy
To accomplish its objective, the user interface MUST provide a mechanism for maintaining policy on the automated acceptance and rejection of received queries in support of the Query Authorization capability. It MUST also enable maintaining policy on whether or not to flag a query for external publication of its results.

The Query Console capability MAY provide a variety of administrator configurable evaluation criteria, but it MUST include the following: the query, query parameters, physical source of the query being processed (e.g., Presentation/Reporting subsystem), an identifier of the user issuing the query, and the asset resolution data for the queries applicability to this CM instance.

The Query Console capability will receive queries from the Query Authorization capability and MUST present these queries to the users. The users MUST be given a mechanism for approving or denying the queries and the response MUST be sent back to the Query Authorization capability.

The users MUST also be provided the ability to schedule when both automatically accepted and human reviewed queries will be processed. This enables the administrator to schedule processing-intensive queries for non-peak usage time. The users MUST be able to view the list of queries that are scheduled for execution and to optionally deny them at any point prior to execution.

Query Propagation Instances

The Query Console capability MUST provide a user interface for configuring the query propagation targets. This is the set of CM instances to which queries will be propagated (see the Query Propagation capability for a description of the propagation algorithm). Typically, this set includes the CM instances that are connected to the current CM instance but that exist at a lower tier in the CM hierarchy. The user interface MUST provide the ability to view the current set of propagation instances, to delete instances, and to add instances.

Results Publication
The Query Console capability MUST enable users to specify default I1 interface services to which select query results will be externally published through the Results Publication capability.

Data Aggregation Location
The capability MUST enable administrators to configure the location of the current CM instance's Data Aggregation subsystem. This information will be used in the I3.2 collection tasking and I3.3 analysis tasking interface parameters (see the Interface Specifications section for details).

Workflow Steps Supported: WF2, steps 1 and 2

4.2.2 Collection Controller Capabilities

The Collection Controller component interacts with the following entities within the CM Reference model:

1. Collection subsystems
2. Query Orchestrator component

In order to implement the necessary interactions, the Collection Controller MUST provide the following capabilities:

1. **Task Processing:** The Collection Controller can receive incoming tasks, manage data collection fulfillment, and respond with completion status.
2. **Subtask Propagation:** The Collection Controller can propagate data collection tasking to the appropriate Collection subsystems and keep track of their completion.
3. **Task Management Console:** The Collection Controller provides a console to manage information available Collection subsystems.

The Collection Controller capabilities call each other as shown below in Figure 12.

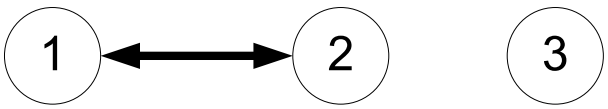


Figure 12: Collection Controller Capability Interactions

4.2.2.1 Task Processing

The Task Processing capability MUST receive incoming tasks, manage data collection fulfillment, and respond with completion status.

Task Receipt
The Task Processing capability MUST accept collection tasks for which data collection will be performed (e.g., from the Query Orchestrator). To accomplish this objective, the capability MUST provide a service that can accept incoming queries using the I3.2 interface.

If the Query Orchestrator and Collection Controller are developed together as a single product this I3.2 interface would exist within the "black box" of the product internals and could be implemented using alternate but functionally equivalent methods.

Data Collection Fulfillment
Each received task MUST be passed to the Subtask Propagation capability. Eventually, a reply will be received indicating either success or failure of the relevant data collection activities. This reply MAY include data supporting the calculation of the task result.

Query Response
Once data collection has been finished, the Task Processing capability MUST respond to received tasks with the task result status and MAY include any received result data.

Workflow Steps Supported: WF2, steps 5 and 5d

4.2.2.2 Subtask Propagation

The Subtask Propagation capability MUST receive tasks from the Task Processing capability.

Task Decomposition
The Subtask Propagation capability then MUST decompose each data collection task into one or more subtasks that are to be sent to the appropriate Collection subsystems. The distinction between tasks and subtasks is that subtasks are tied to directing the collection activities of a particular Collection subsystem. Included with each data collection subtask MUST be the identifier of the task from which it was derived as well as the identifier of the relevant query.

If the Subtask Propagation capability is able to create subtasks only for applicable Collection subsystems, then it MAY do so (e.g., a vulnerability scanning task shouldn't be sent to an asset inventory tool). Otherwise, for each task the Subtask Propagation capability MUST create a subtask corresponding to each Collection subsystem based on Collection Subsystem Recordation (see the Task Management Console below). In this case the Collection subsystem will be responsible for responding back to indicate that the task doesn't apply. Future revisions of this specification may be augmented to require a more intelligent distribution of tasking to the Collection subsystems.

Subtask Propagation
The Subtask Propagation capability MUST send out each subtask to the appropriate Collection subsystem. To do this, the Subtask Propagation capability MUST provide a service that can accept requests for subtasks from a Collection subsystem client. The Subtask Propagation capability MUST also provide a client that can send subtasks to a Collection subsystem service. This ability MUST be implemented using the I3.2 interface. Note that both push and pull capabilities are mandated here in order to support operational needs that require both approaches and to ensure interoperability between subsystems.

The Subtask Propagation capability MUST be able to receive, in response, status for each subtask (including 'success', 'failure', and 'not applicable'). It MAY be able to receive subtask result data. When responses have been received for all subtasks and tasks under a specific query, an overall response MUST be passed to the Task Processing capability.

If all relevant subtasks completed successfully then a success status MUST be sent. If any failed, then a failure status MUST be sent. 'Not applicable' responses MUST be ignored for this calculation unless all the responses are 'not applicable' in which case a failure status MUST be sent. Any received result data MAY be sent along with the status message.

Workflow Steps Supported: WF2, steps 5a, 5b, and 5c

4.2.2.3 Task Management Console

The Task Management Console capability MUST provide user interfaces to manage the Collection Controller component. This console MAY be combined with user interfaces from other Task Manager components.

Collection Subsystem Recordation
The Task Management Console capability MUST provide a user interface for management of the available Collection subsystems. Proprietary solutions for automated discovery of Collection subsystems are encouraged and MAY feed data into this console.

Workflow Steps Supported: WF2, steps 5b and 5c

4.3 Collection Subsystem Specifications

This section presents the Collection subsystem specifications.

4.3.1 Collection Subsystem Capabilities

A Collection subsystem interacts with the following CM reference model entities:

1. Collection Controller component
2. Data Aggregation subsystem
3. Content subsystem

In order to implement the necessary interactions, a Collection subsystem MUST provide the following capabilities:

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

Within the Collection subsystem, these capabilities call each other as shown below in Figure 13. These relationships are described in the following subsystem specifications.

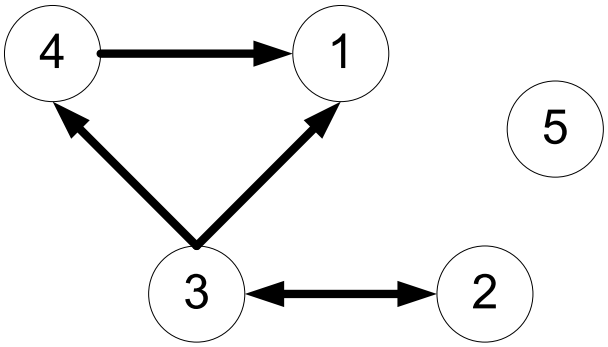


Figure 13: Collection Subsystem Capability Interactions

4.3.1.1 Task Receipt

The Task Receipt capability MUST accept data collection tasks (e.g., from the Collection Controller) for processing and produce responses describing task completion status.

To accomplish its objectives, the capability MUST provide a service that can accept incoming tasks. The capability MUST also provide a client that can retrieve tasks from a task retrieval service. This ability MUST be implemented using the I3.2 interface. Note that both push and pull functionality is mandated here in order to support operational needs that require both approaches. The Collection console will specify which mechanism (push or pull) is active for a particular Collection subsystem instance.

When the Task Receipt capability receives a task, it MUST store the task on a Collection subsystem Task Queue[24]. This makes the task available to other subsystem capabilities for processing.

[24] The exact behavior of the Task Queue is not defined in order to avoid over specification and to allow for multiple approaches.

For each task on the Task Queue, the Task Receipt capability will eventually receive a response from the Data Publication capability. From this response the Task Receipt capability will learn about the success or failure of task collection activities and MUST be able to receive corresponding result data. It might also learn that the task is not applicable to this Collection subsystem. It MUST report the task success, failure, or non-applicability to the task requestor (usually the Collection Controller). If configured to do so in the Collection Console, the Task Receipt capability MUST also report the task result data. If it receives a status notice for a task that it did not receive from a requestor (i.e., it was locally generated by the Collection Console capability) then it MAY ignore the notice.

This capability MUST also be able to report ongoing status of a task (e.g., to the Collection Controller component) over the I3.2 interface whenever asked by the requestor.

Workflow Steps Supported: WF1 trigger 2
Workflow Steps Supported: WF2, steps 5b and 5c

4.3.1.2 Content Retrieval

The Content Retrieval capability MUST accept content retrieval requests from the Data Retrieval capability with the relevant data collection task as a parameter. For each request, the Content Retrieval capability MUST retrieve the appropriate digital policy and supporting content necessary for the subsystem to execute the task. To accomplish this, the capability MUST implement a client that uses the I2.1 interface and MUST use this client to retrieve the needed content (e.g., from the Content subsystem). Upon content retrieval completion, the Content Retrieval capability MUST send a success or failure status to the Data Retrieval capability.

Workflow Steps Supported: WF1, step 1

4.3.1.3 Data Retrieval

The Data Retrieval capability MUST collect and retrieve needed CM data in order to fulfill the data collection tasks on the Task Queue. To support this objective, it SHOULD maintain a local repository of previously collected data that acts as a caching mechanism.

To accomplish its objectives, the capability MUST retrieve tasks from the Task Queue. For each task, the Data Retrieval capability MUST obtain the related digital policy and supporting content by calling the Content Retrieval capability and passing it the task, then awaiting an affirmative response. Next, the Data Retrieval capability MUST determine whether or not the task and retrieved policy content is applicable to this Collection subsystem. If not, it must report a ‘not applicable’ status for the task to the Task Receipt capability.

If the task is applicable, the Data Retrieval capability MUST attempt to retrieve the needed data from its local repository (if one exists). If the requested data is available, the data MUST be sent to the Data Publication capability. If the requested data is not available in the local repository, the data is considered stale according to the task’s Data Age parameter, or the local repository

does not exist, then the Collection subsystem MUST attempt to collect the requested data from its sensors.

To collect the data, the sensors MUST use the retrieved policy content (if there is any content applicable to the collection activity). Collected data SHOULD be stored within the subsystem's local repository. Next, the retrieved or collected data along with the task to which the data responds MUST be sent to the Data Publication capability. An error flag MUST be sent along with the data if some or all of the task data could not be retrieved.

Workflow Steps Supported: WF1, steps 1 and 2

4.3.1.4 Data Publication

The Data Publication capability MUST accept calls from the Data Retrieval capability. These calls will include the following as parameters: the task, the data retrieved to fulfill the task, and any error flags. The Data Publication capability MUST then send the collected data to the appropriate repository (e.g., the Data Aggregation subsystem). If this data was collected as a task originally arriving through the Task Receipt capability, the I3.2 task will have a parameter which includes the location to send the data. If that parameter is empty or the task was generated locally (i.e., from the Collection Console), then the data must be sent to the default location specified in the Collection Console. To do this, the Collection subsystem MUST implement a client using the I1 interface to send data.

Finally, the Data Publication capability MUST notify the Task Receipt capability that the task processing has been completed, report on its success or failure, and pass it the collected result data.

Workflow Steps Supported: WF1, step 3

4.3.1.5 Collection Console

The Collection Console capability MUST provide a user interface that enables administration of the Collection subsystem.

Locally Generated Tasking
The Collection Console capability MUST enable a user to create named sets of data collection tasks and schedule them to be periodically added to the Task Queue. The available periodicities for task execution MUST include options for seconds, minutes, hours, days, weeks, months, and years. The user MUST be able to add named sets of one or more tasks, to delete named sets, and to modify named sets (including changing the periodicity).

Data Publishing Location
The Collection Console capability MUST enable the administrators to enter a default location for publishing collection task result data. This is to be used by the Data Publication capability. This default setting is important because the I3.2 specification has a parameter allowing but *not requiring* a collection task to specify the location for data publication. The Collection Console

capability MUST also enable administrators to specify whether or not task result data will be sent to the task requestor (e.g., the Collection Controller) by the Task Receipt capability.

Task Receipt Configuration
The Collection Console capability MUST enable an administrator to configure, enable, or disable both the push and pull task retrieval mechanisms required within the Task Receipt capability.

Workflow Steps Supported: WF1 triggers 1 and 2
Workflow Steps Supported: WF1 steps 1 - 3
Workflow Steps Supported: WF2 steps 5b and 5c

4.4 Data Aggregation Subsystem

This section presents the Data Aggregation subsystem specifications.

4.4.1 Data Aggregation Subsystem-Wide Capabilities

The Data Aggregation subsystem interacts with the following entities within the CM Reference model:

1. Collection subsystem
2. Analysis/Scoring subsystem
3. Query Orchestrator component

In order to implement the necessary functions and interactions, the Data Aggregation subsystem MUST provide the following capabilities:

1. **Bulk Data Storage**: The Data Aggregation subsystem receives and stores CM data.
2. **Interactive Data Access**: The Data Aggregation subsystem provides an interactive data access and storage service.
3. **Data Maintenance Console**: The Data Aggregation subsystem implements a console for administration and maintenance purposes.

Within the Data Aggregation subsystem, these capabilities call each other as shown below in Figure 14. These relationships are described in the following subsystem specifications.

Figure 14: Data Aggregation Subsystem Capability Interactions

4.4.1.1 Bulk Data Storage

The Bulk Data Storage capability MUST accept and store CM data. To support this, the capability must implement a service that uses the I1 interface. Stored data MUST be tagged to

aid with retrieval. At a minimum, data MUST be tagged with the relevant query identifier that caused the data to be collected.

Workflow Steps Supported: WF1 step 3 and 4
Workflow Steps Supported: WF2 steps 4c, A.3e, A.4c

4.4.1.2 Interactive Data Access

The Interactive Data Access capability MUST provide interactive access for the CM data repositories. To support this, the capability must implement a service that uses the I4 interface. It MUST enable access to data through use of the tags set up by the Bulk Data Storage capability.

Workflow Steps Supported: WF2, step A.3a, A.3c, and A.4a

4.4.1.3 Data Maintenance Console

The Data Maintenance Console capability MUST provide a user interface to allow administrator and user management of the subsystem.

To accomplish its objectives, the capability MUST provide a user interface that enables the Data Administrators to view, insert, and delete data deconfliction rules (used by the Analysis/Scoring subsystem)

Workflow Steps Supported: N/A

4.4.2 Asset Repository

The Bulk Data Storage capability MUST store all asset data in the Asset Repository.

For the CM instance to function, at least one Collection subsystem needs to be set up to periodically report asset data to the Data Aggregation subsystem. This is because query fulfillment relies upon resolving the query's Asset Descriptor parameter (a descriptor of the assets to which the query applies) into a specific list of assets.

4.5 Analysis/Scoring Subsystem Specifications

This section presents the Analysis/Scoring subsystem specifications. The Analysis/Scoring subsystem is composed of just one component: the Analysis Engine.

4.5.1 Analysis Engine Capabilities

The Analysis Engine component interacts with the following entities within the CM Reference model:

1. Query Orchestrator component
2. Content subsystem
3. Data Aggregation subsystem

In order to implement the necessary functionality and interactions, the Analysis Engine component MUST provide the following capabilities:

1. **Analysis Task Receipt and Response:** The Analysis Engine can receive incoming analysis tasks and reply with the requested results.
2. **Analysis Algorithm Retrieval:** The Analysis Engine can retrieve the necessary analysis algorithms and their parameters.
3. **Analysis and Scoring:** The Analysis Engine can generate analysis task result data through analysis and scoring activities.
4. **Data Retrieval:** The Analysis Engine can retrieve data from the Data Aggregation subsystem and deconflict data as necessary.
5. **Data Storage**: The Analysis Engine can publish generated results including both findings and scores (this is used for storing results in the Data Aggregation subsystem).
6. **Analysis Console**: The Analysis Engine has a console that enables administrators to configure the system.

Within the Analysis/Scoring subsystem, these capabilities call each other as shown below in Figure 15. These relationships are described in the following subsystem specification.

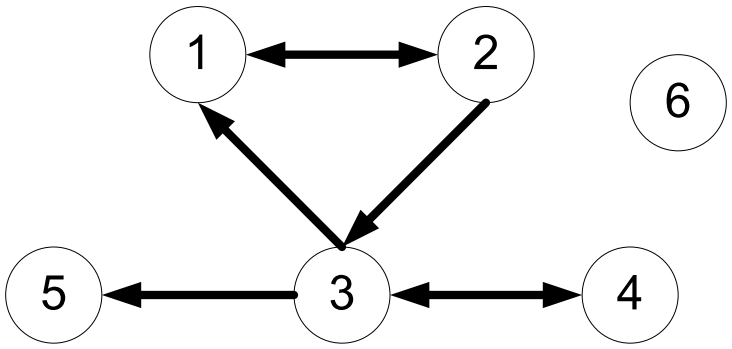


Figure 15: Analysis/Scoring Subsystem Capability Interactions

4.5.1.1 Analysis Task Receipt and Response

The Analysis Task Receipt and Response capability MUST be able to receive incoming analysis tasks for processing and respond with the requested results. The capability MUST send all

received tasks to the Analysis Algorithm Retrieval capability to ensure that the subsystem has the appropriate analysis information to process the task.

To accomplish its objectives, the Analysis Task Receipt and Response capability MUST implement a service to accept analysis tasks using the I3.3 interface. These tasks will typically arrive from the Query Orchestrator component of a Task Manager subsystem.

Eventually, the Analysis Task Receipt and Response capability will receive task result data and/or a success or failure status (from the Analysis and Scoring capability). The Analysis Task Receipt and Response capability MUST then respond to the original task requestor with the provided result data (if any) and the success or failure status (e.g., flag that there are missing data elements).

Workflow Steps Supported: WF2, steps 1, 6, and A.5

4.5.1.2 Analysis Algorithm Retrieval

The Analysis Algorithm Retrieval capability MUST be able to receive analysis tasks from the Analysis Task Receipt and Response capability. Using the analysis task parameters, the Analysis Algorithm Retrieval capability MUST then retrieve the relevant analysis algorithms and their parameters. If the Analysis Algorithm Retrieval capability is unable to retrieve the necessary scoring algorithms, the capability MUST return a failure status to the Analysis Task Receipt and Response capability.

The task will provide a parameter specifying the analysis algorithm to be used. This algorithm MAY be pre-loaded (or even hardcoded) into the subsystem making retrieval a trivial operation. If not, the algorithm MUST be retrieved from the Content subsystem. The task will optionally include specific parameter values to be used within the analysis algorithm (e.g., weighting of input variables for scoring) or a pointer to enable retrieval of those values. If no parameter values or a pointer is provided, the default parameters for the analysis algorithm MUST be used. If a pointer is provided, the parameters MUST be retrieved from the Content subsystem.

To accomplish these objectives, the Analysis Algorithm Retrieval capability MUST implement a client using the I2.1 interface that enables it to request analysis algorithms and parameter data.

Once the analysis algorithm and any associated parameters have been retrieved, the Analysis Algorithm Retrieval capability MUST forward the task to the Analysis and Scoring capability.

Note, the analysis algorithms used MUST conform to a three-phased analysis architecture. First, “raw data” MUST be retrieved and deconflicted. If the task is requesting raw data, then the analysis is complete. Next, the raw data elements MUST be compared against specific policy to produce “findings” (i.e. evaluation results created by comparing a raw data element or set of elements to a required value). Findings are typically true or false and not numeric. These findings then MAY be used to create higher level or more abstract findings. If the task is requesting findings, then the analysis is complete. Lastly, the algorithm MUST use one or more “findings” to create scores (numerical representation of an evaluation of one or more findings).

These scores MAY be used to generate higher level or more abstract scores. Once the scores requested by the task have been generated, the analysis is complete.

Workflow Steps Supported: WF2, step A.1

4.5.1.3 Analysis and Scoring

The Analysis and Scoring capability MUST receive tasks for processing from the Analysis Algorithm Retrieval capability.

First, the Analysis and Scoring capability SHOULD determine whether or not the analysis task results are available and are of the required freshness without any analysis being performed. The task will contain a Data Age parameter which indicates the required freshness of the raw data that was processed to create the task results. This analysis MUST be done interactively with the Data Retrieval capability to provide access to any available result data. If the results are available, they MUST be sent back to the Analysis Task Receipt and Response capability.

If the task results are not already available, the Analysis and Scoring capability MUST determine whether or not it has the data (i.e., raw data, findings, or scoring results) of the required freshness to create the needed task results. This analysis MUST be done interactively with the Data Retrieval capability to provide access to the available data. If the needed underlying data is available, the capability MUST calculate the task results using the analysis algorithm and parameters specified in the task. All final results and intermediary results (findings and scorings) MUST be published by being sent to the Data Storage capability (e.g., the Data Aggregation subsystem). The final results MUST be sent back to the Analysis Task Receipt and Response capability so that they can be delivered directly to the task requestor (usually the Query Orchestrator). If the task result could not be obtained, a failure status for the task MUST be sent back to the Analysis Task Receipt and Response capability.

Workflow Steps Supported: WF2, step A.4b

4.5.1.4 Data Retrieval

The Data Retrieval capability MUST provide a client that enables arbitrary data retrieval from the Data Aggregation subsystem. The location of the Data Aggregation subsystem might be specified in the original I3.3 analysis task parameters. If the location parameter is empty, then the data connection should be made to the default location specified in the Analysis Console capability. To accomplish this, the Data Retrieval capability MUST implement a client using the I4 interface.

Note - this capability is used by the Analysis and Scoring capability to provide dynamic access to the available CM data. If raw data is retrieved, the Data Retrieval capability MUST provide data deconfliction services so that only deconflicted data is provided back to the Analysis and Scoring capability. Configuration parameters for data deconfliction are stored in the Data Aggregation subsystem's Metadata Repository and MUST be retrieved using the I4 interface.

Workflow Steps Supported: WF2, steps A.3a, A.3c, and A.4a

4.5.1.5 Data Storage

The Data Storage capability MUST publish calculated analysis task results so that they can be stored and reused. This includes the final results returned to the task requestor as well as intermediary results (e.g., findings and intermediate scoring).

To support this effort, the Analysis/Scoring subsystem MUST implement a client using the I1 interface to send data to the location specified in the original I3.3 analysis task parameters (this should be the Data Aggregation subsystem). If the location parameter is empty, then the data should be sent to the default location specified in the Analysis Console capability.

Workflow Steps Supported: WF2, steps A.3e and A.4c

4.5.1.6 Analysis Console

The Analysis and Scoring capability MUST have a console that enables administrators to configure the subsystem.

Data Location
The Analysis Console capability MUST enable the administrators to enter a default location for accessing data to be analyzed (usually the Data Aggregation subsystem). This will be the same location at which calculated findings and scorings are cached for future use. This default setting is important because the I3.3 specification has a parameter allowing, but not requiring, an analysis task to specify a location for data retrieval.

Workflow Steps Supported: N/A

4.6 Content Subsystem Specifications

This section presents the Content subsystem specifications.

4.6.1 Content Subsystem Capabilities

The Content subsystem interacts with the following entities within the CM reference model:

1. Collection subsystem
2. Analysis / Scoring subsystem
3. Content Providers
4. Content subsystems of different CM instances

In order to implement the necessary interactions, 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.

Within the Content subsystem, these capabilities call each other as shown below in Figure 16. These relationships are described in the following subsystem specifications.

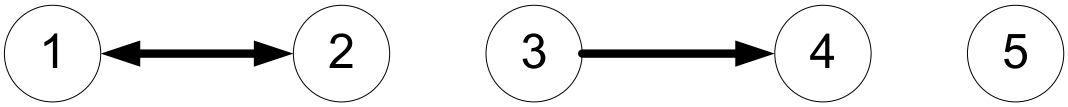


Figure 16: Content Subsystem Capability Interactions

4.6.1.1 Content Provisioning

The Content Provisioning capability MUST accept queries for content (e.g., from the Collection subsystem, Analysis/Scoring subsystem, or another Content subsystem). To accomplish this, the Content subsystem MUST implement a server using the I2.1 interface.

The Content Descriptor parameter will describe the needed content and the Policy Age parameter (if set) will describe the required freshness of the content. If the content is available in the local content repository and is of the required freshness, then it is returned in response to the I2.1 content query.

Otherwise, the Content Provisioning capability MUST pass the content query to the Content Acquisition capability. If the Content Acquisition capability returns the needed content, that

content MUST be returned to the requestor in response to the original I2.1 content query. Otherwise, an error MUST be returned.

Workflow Steps Supported: WF1, step 1
Workflow Steps Supported: WF2, A.1
Workflow Steps Supported: WF3, trigger 1, trigger 2, step 1, and step 4

4.6.1.2 Content Acquisition

The Content Acquisition capability MUST accept content acquisition requests from the Content Provisioning capability. To accomplish this, the capability MUST implement an I2.1 client.

If the content request Collect Bit parameter is 1, then the Content Acquisition capability MUST attempt to retrieve the content from external content repositories. The Content Configuration Console contains an ordered list of the external repositories from which the content MUST be retrieved. Sequential requests to each repository on the list MUST be made using the I2.1 client. Proprietary interfaces MAY also be used. If the content is retrieved and it is of the required freshness according to the Policy Age parameter, then the content is returned to the Content Provisioning capability.

If the content has not yet been received and the Instance Depth parameter is not 0, the Content Acquisition capability MUST attempt to retrieve the content from the next higher tier CM instance (the Content Configuration Console contains the location). If the Instance Depth parameter is not 0, the I2.1 client is used to request the content from the higher tier CM instance with the Instance Depth parameter decremented by 1. If the content is retrieved, then the content is returned to the Content Provisioning capability.

Note that retrieved content MUST be stored in the local repository for caching purposes, be time stamped, and tagged with its source.

Workflow Steps Supported: WF3, steps 2 and 3.

4.6.1.3 Content Persistence

The Content Persistence capability MUST provide an interface for maintaining the content object repository. This interface will enable maintenance by external content management tools. To accomplish this, the capability MUST implement a service using the I2.2 interface.

Local Persistence
As provided by the I2.2 interface, the Content Persistence capability MUST implement the ability to list, retrieve, insert, and delete content objects. All objects in the repository MUST be represented using XML schemas. Each object MUST be tagged with an insertion time stamp and its source. Lastly, the objects MUST be manipulated by the Content subsystem as whole objects without changing their internal data representation. Note that this doesn't preclude external tools from modifying portions of a content blob and the updating the content repository with the new version.

Distributed Persistence
If the I2.2 service receives a request for content to be saved and the Instance Depth parameter is not equal to 0, first the content MUST be stored in the local repository and then the request MUST be passed to the Content Propagation capability.

Workflow Step Supported: WF4, step 1

4.6.1.4 Content Propagation

The Content Propagation capability MUST be able to accept propagation requests from the Content Persistence capability and the Content Configuration Console. Each request will contain the content to be propagated and the Instance Depth parameter. The capability MUST implement an I2.2 client and push the content to the Content subsystem I2.2 services of other CM instances (usually lower tier CM instances) with the Instance Depth parameter decremented by 1. The list of CM instances to which to propagate the content MUST be retrieved from the Content Configuration Console capability (a CM instance leaf node will have an empty list and then no propagation will occur).

Workflow Steps Supported: WF4, trigger 2 and step 2

4.6.1.5 Content Configuration Console

The Content Configuration Console capability MUST provide a console for administrators to configure and control the subsystem.

Content Retrieval Policy
The administrators MUST have the ability to override content request Collect Bit and Instance Depth parameters[25] for all requests in order to disable that functionality or to require human approval on a case by case basis. More sophisticated automated approval policy MAY be developed that enables external content retrieval or retrieval from higher tier CM instances based on a variety of factors.

Locally Initiated Content Acquisition
The capability MUST enable administrators to create content retrieval queries and initiate them immediately (by passing them to the Content Provisioning capability), run them once in the future, or schedule them to run periodically. The ability to periodically retrieve content in increments of hours, days, weeks, and months MUST be supported at a minimum. The Policy Age, Collect Bit, and Instance Depth parameters MUST be supported[26]. The Content Configuration Console capability MUST enable administrators to view the list of scheduled retrieval queries and terminate selected queries.

Locally Initiated Content Propagation
The Content Configuration Console capability MUST enable users to propagate content to other CM instances (usually lower tier instances). The content to be propagated MUST already be

[25] See the previous Content capabilities for an explanation of these parameters.

[26] See the previous Content capabilities for an explanation of these parameters.

stored in the local content repository. To initiate propagation, the Content Configuration Console capability MUST pass the content and an Instance Depth parameter to the Content Propagation capability. The Instance Depth parameter MUST be set to be the user's desired depth of the propagation through the CM instance tree. If the Instance Depth parameter is set to -1 then the content is propagated until it reaches the CM instance leaf nodes.

Local Content Maintenance
The Content Configuration Console capability MUST enable administrators to read, insert, and delete content objects.

Location of Higher Tier Instances Content Repositories
The Content Configuration Console capability MUST enable administrators to define a prioritized list of other Content subsystems from which content MAY be retrieved (usually this is just the higher tier CM instance).

Location of Lower Tier Instances Content Repositories
The Content Configuration Console capability MUST enable administrators to define a prioritized list of other Content subsystems to which content MAY be propagated (usually a list of the lower tier CM instances).

Location of External Content Repositories
The Content subsystem MUST also be able to define a prioritized list of external content repositories from which content MAY be retrieved.

Workflow Steps Supported: WF3, steps 2 and 3
Workflow Steps Supported: WF4, trigger 1, steps 1 and 2.

5. Interface Specifications

The following subsections provide specifications for the four interfaces identified in the continuous monitoring model. Specific requirements are levied on each interface, which are also described with a Web Service Description Language (WSDL) document.

Section 2.2 contains an overview of the interfaces. It also contains diagrams showing where each interface is used for subsystem communication (both within a CM instance and between CM instances) and a short description of each interface instance.

5.1 Result Reporting

The Result Reporting interface (referred to as I1) is intended to enable the sending of asset reports to an endpoint. In the continuous monitoring architecture, it is implemented by the Data Aggregation subsystem and called by various other components to send asset information for storage. I1 MUST be implemented as a Simple Object Access Protocol (SOAP) [SOAP] web service accessible over, at minimum, Hypertext Transfer Protocol (HTTP) [IETF 2616] and HTTP over SSL/TLS (HTTPS) [IETF 2818]. The I1 interface MUST implement the service described by the WSDL found at the URL described below. WS-Security [WS-S] MAY be added to the specified WSDL as necessary. The following request messages MUST be

implemented in the one-way interface as described below. The WSDL describes additional details about the interface.

WSDL location:
http://scap.nist.gov/schema/cmrm/0.1/data_0.1.wsdl
WSDL SHA-256 hash:
B343762A96A405E219586226A35136DC31F96D7395BF176426CEF458F15BB0DD

Request: An ARF report collection. The payload and relationships described in the ARF report collection will be defined in the data domain specific specifications.

5.2 Content Acquisition

The Content Acquisition interface (referred to as I2) describes how to interact with the Content subsystem to retrieve, add, update, and delete content from the Content subsystem. There are two variants of the interface. I2.1 enables only reading content from the Content subsystem. I2.2 enables all four operations described above. These functions are broken into two sub-interfaces to clearly delineate their roles in the CM architecture. Most implementations of I2 in the architecture require only retrieve functionality, fully specified by I2.1, which is a subset of I2.2. I2.1 and I2.2 MAY be implemented as completely separate services or I2.1 MAY be implemented as a restriction of I2.2. In either case, this interface (to include I2.1 and I2.2) MUST be implemented as a SOAP [SOAP] web service accessible over, at minimum, HTTP [IETF 2616] and HTTPS [IETF 2818]. The interface MUST implement the service described by the WSDL found at the URL described below. WS-Security [WS-S] MAY be added to the specified WSDL as necessary. The WSDL describes additional details about the interface.

WSDL location:
http://scap.nist.gov/schema/cmrm/0.1/content_0.1.wsdl
WSDL SHA-256 hash:
C14A72F9B5016F00C1A61012826D336920D444773AE2FF720906BE9475B3C640

The response on each of the four operations is a status code. The possible status codes are:

- SUCCESS – The operation completed successfully
- FAIL – The operation failed to complete for an unspecified reason
- OBJECT_NOT_FOUND – The operation fail because it could not find the targeted object
- PERMISSION_DENIED – The operation failed because the requester is not permitted to execute the request

5.2.1 Interface 2.1

Only the Retrieve operation MUST be implemented. If I2.1 is implemented as a restriction of I2.2, then the Add, Replace, and Delete operations' response messages MUST be a status code of PERMISSION_DENIED. The following request and response messages for each operation MUST be implemented in the interface as described below.

5.2.1.1 Retrieve Operation

Request: The request MUST contain a globally unique ID for the request. In addition, the request MUST contain a *system URI*[27] indicating the nature of the request content, and an *ident* or *complex-ident*. An *ident* is a string that has a known meaning within the system identified. A *complex-ident* is an XML element that has a known meaning with the system identified. All instances of this operation MUST support the system identified by "gov:cm:content:id". That *system URI* indicates that a piece of content is being retrieved by content object ID. When that *system URI* is specified, an *ident* MUST be specified indicating the ID of the content object to retrieve. Data domain specific specification MAY specify additional systems.

Response: A list of content objects or a status code indicating a failure. Valid status codes are FAIL, OBJECT_NOT_FOUND, PERMISSION_DENIED.

5.2.2 Interface 2.2

The following request and response messages for each operation MUST be implemented in the interface as described below.

5.2.2.1 Retrieve Operation

Follow the guidance in Section 5.2.1.1

5.2.2.2 Add Operation

Request: A globally unique ID for the request and an XML element defining the content.

Response: One of the following status codes: SUCCESS, FAIL, PERMISSION_DENIED. If the status is SUCCESS, then the ID of the new content object MUST be returned as well.

5.2.2.3 Replace Operation

Request: A globally unique ID for the request and an XML element defining the new content as well as the content ID to replace.

Response: One of the following status codes: SUCCESS, FAIL, OBJECT_NOT_FOUND, PERMISSION_DENIED

5.2.2.4 Delete Operation

Request: A globally unique ID for the request and the content ID of the content object to delete.

[27] In this context, a *system URI* defines a category of identifiers and it gives context to the identifier. The *system URI* string and definition is defined by some external authority. For example, some external authority could define the URI "http://www.ssa.gov/ssnumber/" to mean that the corresponding *ident* is a Social Security Number (SSN). Therefore, the *system URI* indicates the category of the *ident* (i.e., it is a SSN).

Response: One of the following status codes: SUCCESS, FAIL, OBJECT_NOT_FOUND, PERMISSION_DENIED.

5.3 Querying and Tasking

The Querying and Tasking interface (referred to as I3) will exchange query and tasking information using the Common Tasking Language (CTL). The I3 interface is composed of three sub-interfaces describing the separate functions of this interface. I3.1 enables the sending of a *query* using CTL. I3.2 enables the sending of a *task* related to collection activities using CTL. I3.3 enables the sending of a *task* related to scoring and analysis activities using CTL. All three expose a similar interface and differ only in their CTL payloads.

All I3 interfaces (to include I3.1, I3.2 and I3.3) MUST be implemented as Simple Object Access Protocol (SOAP) [SOAP] web service accessible over, at minimum, Hypertext Transfer Protocol (HTTP) [IETF 2616] and HTTP over SSL/TLS (HTTPS) [IETF 2818]. The I3 interface MUST implement the service described by the WSDL found at the URL described below. WS-Security [WS-S] MAY be added to the specified WSDL as necessary. The request and response messages MUST be implemented in the sub-interfaces interface as described in the following sections. Starred ('*') items are not yet well-defined. The WSDL describes additional details about the interface.

In addition, all I3 interfaces MUST support WS-Addressing [WS-A] and WS-MakeConnection [WS-MC]. Most exchanges on the I3 interface will be asynchronous because query and tasking will often take more time to complete than is reasonable for a synchronous transaction. When a synchronous protocol such as HTTP is leveraged in this situation, [WS-A] provides a way for the recipient of a request to send a response at some later time. In addition, when the sender of a request is not accessible by the recipient for any reason (e.g., a firewall or data guard blocks the connection), [WS-MC] makes provisions to work around that limitation. While [WS-A] MUST be implemented and used in all implementations of I3, [WS-MC] MUST be supported in all implementations, but needs to be used only when one side of the interface cannot connect to an endpoint on the other side.

In situations where both the server and client side of I3 is accessible, the client MUST expose a callback endpoint. The following WS-Addressing fields MUST be populated on the message from the client to the server.

- wsa:Action – MUST match the SoapAction on the HTTP header
- wsa:To – MUST contain the endpoint URL of the destination for this message
- wsa:ReplyTo/wsa:Address – MUST contain the endpoint URL that the server MUST call to provide a response to this message
- wsa:MessageID – A globally unique message ID

The server MUST return an HTTP 202 if it successfully received the message. When the server is prepared to return a response message, it MUST generate a message and send it to the URL specified in the wsa:ReplyTo/wsa:Address field on the request message. It MUST populate the following WS-Addressing fields on the response message:

- wsa:Action – MUST match the SoapAction on the HTTP header
- wsa:To – MUST contain the endpoint URL of the destination for this message
- wsa:MessageID – MUST contain a globally unique message ID
- wsa:RelatesTo – MUST contain the message ID of the request message for which this message is a response

In situations where the server has an accessible endpoint, but the client does not, all of the previous requirements apply, with the following exceptions and additions: WS-MakeConnection MUST be used to enable bi-directional communication; the wsa:ReplyTo on the request message MUST conform to the MakeConnection anonymous URI as defined in [WS-MC] Section 3.1.

When the server has a response message, it MUST place it in a FIFO queue associated with the anonymous endpoint specified in the wsa:ReplyTo of the request. The client MUST periodically send WS-MakeConnection requests as specified in [WS-MC] Section 3.2 to the server. wsmc:MakeConnection/wsmc:Address MUST be populated with anonymous URI for the client. The server MUST return HTTP status code 202 if no messages are available for that endpoint, or it MUST return the first message in the FIFO queue associated with that endpoint. If additional messages are available, then the wsmc:MessagePending/@pending attribute MUST be set to "true" on the response as specified in [WS-MC] Section 3.3.

WSDL location:
http://scap.nist.gov/schema/cmrm/0.1/task_0.1.wsdl
WSDL SHA-256 hash:
28451DD7E37B67921D748942F39822118E7347C5F04072B2277DEDED54252470

5.3.1 Interface 3.1

The instruction MUST be a query in a CTL wrapper with the following parameters.

- **Query Identifier**: This parameter provides a CM multi-instance implementation unique name for the query being specified by this I3.1 request.

- **Asset Descriptor*:** This parameter describes the population of assets to which the query applies. This can be simply a list of assets but may also be a more abstract descriptor (e.g., all printers). The asset list is a list of Asset Identification elements as defined in [Asset Identification]. A more abstract population description language has yet to be well-defined.

- **Content Descriptor***: This parameter describes the content to use to collect the query information. The content may be provided inline within this parameter or a pointer may be provided to the required content (e.g., to content within the Content subsystem). All implementations MUST support retrieving content by unique content object ID. Additional mechanisms for retrieving content MUST be supported as specified in data

domain specific specifications. The format of the content descriptor is not yet well-defined.

- **Policy Age:** This parameter describes the required freshness of the applied content. If the content being used (e.g., by the Collection subsystem) is not sufficiently fresh, it cannot be used for data collection and must be first refreshed through content retrieval (usually from a Content subsystem). Note, if the Content Descriptor contains the content itself, as opposed to referencing content, then this parameter does not apply and is not used. In such cases, the parameter should be set to -1. If the parameter is set to 0, the content must always be updated prior to performing data collection.

- **Analysis Descriptor*:** This parameter describes the analysis procedure to use in generating the requested query results. This includes specifying the algorithm inputs (e.g., raw data elements), analysis algorithm (i.e., calculations), and output data. It also includes any parameters used to modify the behavior of the analysis algorithm (e.g., modifying the weightings of variables). This parameter specifies all of these data by simply pointing to a content identifier within the Content subsystem. This content will be an XML file specifying the analysis algorithm and parameter values. This approach enables the complexity of the analysis algorithm to be hardcoded into the Analysis/Scoring subsystems while the algorithm parameters can be easily modified due to their accessibility in the Content subsystem. The format of an analysis descriptor is not yet well-defined.

- **Task Result Descriptor*:** This parameter describes the data collection reports that are used to gather data to support query analysis. This includes the format of the report as well as the level of detail that must be returned. This descriptor must match the required inputs for the analysis algorithm specified in the Analysis Descriptor. The format of a task result descriptor is not yet well-defined.

- **Query Result Descriptor***: This parameter describes the query results report. This includes the format of the report as well as the level of detail or abstraction that must be returned. The format of a query result descriptor is not yet well-defined.

- **Data Age:** This parameter describes the required freshness of the collected data. If the data is not sufficiently fresh, it cannot be used as input to the scoring algorithm and must be first refreshed through a data collection activity.

- **Collect Bit:** This parameter describes whether or not to initiate collection of data based on this query. It is binary with 0 meaning that data should not be collected and 1 meaning that data should be collected. If this parameter is set to 0 and data needs to be collected in order to provide the query results, the query will fail to provide the requested results.

- **Instance Depth**: This parameter describes the CM instance depth to which the query should be propagated. This should be set to 1 for a single CM instance architecture.

The *result* MUST be a report in for format specified by the query result descriptor wrapped in a CTL wrapper. Possible formats have yet to be well-defined.

5.3.2 Interface 3.2

The instruction MUST be a task in a CTL wrapper with the following parameters.

- **Task Identifier:** This parameter describes CM multi- instance implementation unique name for the I3.2 task.

- **Parent Task Identifier:** This parameter describes CM multi- instance implementation unique name for the I3.2 task that spawned this task. This is populated only when this task is a subtask of another task.

- **Query Identifier**: This parameter provides a CM multi-instance implementation unique name for the I3.1 query being supported by this I3.2 task. If this task is not supporting an I3.2 query, the parameter is set to 0.

- **Asset Descriptor:** This parameter describes the population of assets to which the query applies. This is a list of Asset Identification elements as defined in [Asset Identification]..

- **Content Descriptor***: This parameter describes the content to use to collect the query information. The content may be provided inline within this parameter or a pointer may be provided to the required content (e.g., to content within the Content subsystem). All implementations MUST support retrieving content by content object ID. Additional mechanisms for retrieving content MUST be supported as specified in data domain specific specifications. The format of the content descriptor is not yet well-defined.

- **Policy Age:** This parameter describes the required freshness of the applied content. If the content being used (e.g., by the Collection subsystem) is not sufficiently fresh, it cannot be used for data collection and must be first refreshed through content retrieval (usually from a Content subsystem). Note, if the Content Descriptor contains the content itself, as opposed to referencing content, then this parameter does not apply and is not used. In such cases, the parameter should be set to -1. If the parameter is set to 0, the content must always be updated prior to performing data collection.

- **Task Result Descriptor*:** This parameter describes the data collection reports that are used to gather data to support query analysis. This includes the format of the report as well as the level of detail that must be returned. This descriptor must match the required inputs for the analysis algorithm specified in the query Analysis Descriptor. The format of a task result descriptor is not yet well-defined.

- **Results Target Endpoint:** This parameter specifies a URL that indicates where to send the results once they have been collected. Providing this information on the request allows the results to be sent to a dynamically configurable location of the senders

choosing. The endpoint specified SHOULD expose a web service compliant with Interface 1 (i.e., the URL should point to a Data Aggregation subsystem).

The *result* MUST be either a status of the task, or results of the task, wrapped in a CTL wrapper. Possible formats have yet to be well-defined.

5.3.3 Interface 3.3

The instruction MUST be a task in a CTL wrapper with the following parameters.

- **Task Identifier:** This parameter describes CM multi- instance implementation unique name for the I3.3 task.

- **Query Identifier**: This parameter provides a CM multi-instance implementation unique name for the I3.1 query being supported by this I3.3 task. If this task is not supporting an I4 query, the parameter is set to 0.

- **Asset Descriptor*:** This parameter describes the population of assets to which the query applies. This can be simply a list of assets but may also be a more abstract descriptor (e.g., all printers). The asset list is a list of Asset Identification elements as defined in [Asset Identification]. A more abstract population description language has yet to be well-defined.

- **Policy Age:** This parameter describes the required freshness of the applied content. If the content being used (e.g., by the Collection subsystem) is not sufficiently fresh, it cannot be used for data collection and must be first refreshed through content retrieval (usually from a Content subsystem). Note, if the Content Descriptor contains the content itself, as opposed to referencing content, then this parameter does not apply and is not used. In such cases, the parameter should be set to -1. If the parameter is set to 0, the content must always be updated prior to performing data collection.

- **Analysis Descriptor*:** This parameter describes the analysis procedure to use in generating the requested query results. This includes specifying the algorithm inputs (e.g., raw data elements), analysis algorithm (i.e., calculations), and output data. It also includes any parameters used to modify the behavior of the analysis algorithm (e.g., modifying the weightings of variables). This parameter specifies all of this data by simply pointing to a content identifier within the Content subsystem. This content will be an XML file specifying the analysis algorithm and parameter values. This approach enables the complexity of the analysis algorithm to be hardcoded into the Analysis/Scoring subsystems while the algorithm parameters can be easily modified due to their accessibility in the Content subsystem. The format of an analysis descriptor is not yet well-defined.

- **Query Result Descriptor**: This parameter describes the query results report. This includes the format of the report as well as the level of detail or abstraction that must be returned. The format of a query result descriptor is not yet well-defined.

- **Data Age:** This parameter describes the required freshness of the collected data. If the data is not sufficiently fresh, it cannot be used as input to the scoring algorithm and must be first refreshed through a data collection activity.

- **Data Retrieval Endpoint:** This parameter specifies a URL from which data that is required for analysis should be retrieved. Specifying this parameter allows the location of the data store to be dynamically configured by the sender. The endpoint specified SHOULD expose a web service compliant with Interface 4 (i.e., the URL should point to a Data Aggregation subsystem).

The *result* MUST be either a status of the task, or results of the task, wrapped in a CTL wrapper. Possible formats have yet to be well-defined.

5.4 Advanced Data Retrieval

The Advanced Data Retrieval interface (referred to as I4) is intended to enable arbitrary querying of the Data Aggregation subsystem. It is not yet well-defined, but several requirements are apparent. The interface MUST support a data query language that is robust and flexible. The Data Aggregation subsystem may store information in any format, so the query language SHALL NOT be bound to any particular technology or schema (e.g., SQL), and the types of data in the DA subsystem may change over time, so the query language MUST be flexible to adapt to those varying types of data. The query response MUST be in a standardized format that is consistent with nature of the query, and response MUST be consistent with well-understood semantics of the query. I4 MUST be implemented as a Simple Object Access Protocol (SOAP) [SOAP] web service accessible over, at minimum, Hypertext Transfer Protocol (HTTP) [IETF 2616] and HTTP over SSL/TLS (HTTPS) [IETF 2818]. The interface MUST implement the service described by the WSDL found at the URL described below. WS-Security [WS-S] MAY be added to the specified WSDL as necessary.

WSDL location:

Request and response messages are not yet defined for this interface.

6. Existing Gaps and Future Work

This publication has presented specifications for the CM model presented in NIST IR 7756. This includes the data domain agnostic workflows, subsystem specifications, and interface specifications. These specifications are of sufficient detail to enable implementation of the model. However, in order for such implementations to perform useful work, they must be bound to select data domains. The specifications for such bindings (e.g., for asset, configuration, and vulnerability management) are provided in NIST IR 7800.

Unfortunately, the interface specifications discussed in this publication are incomplete and this limits, but does not prevent, immediate use of the model. In particular, the interfaces for Content Acquisition (I2), Querying and Tasking (I3), and Advanced Data Retrieval (I4) have not been fully specified. For I2 and I3, this publication details necessary functionality and parameters for the related communications. This publication then sets the stage for and defines target goals for

upcoming I2 and I3 specification efforts. For I4, we provide only general goals and challenges because we believe the realization of this specification to be difficult and a long term endeavor.

As far as future work, the authors plan to initiate community efforts (inclusive of both government and industry) to specify I2 and I3 in the near term. This work will be conducted under the umbrella of the SCAP Emerging Specification effort[28]. Work on I4 will wait on the full specification of I2 and I3.

In parallel to filling in these specification gaps, the authors plan to prototype the model. This will enable us to discover and address any development issues that require modification of the specifications. It will also enable us to provide to the community working open source modules that implement the various CM model subsystems. We hope that such an implementation will catalyze vendor participation by providing a transparent proof-of-concept code base. It may even jumpstart vendor participation (especially with small companies) by enabling them to directly adopt some of the modules that implement functionality not currently available in most vendor products (e.g., especially the Task Management and Content subsystems). Our intention is that our proposed prototype will follow the precedent and example set by the MITRE Open Vulnerability Assessment Language (OVAL) interpreter[29] in catalyzing the vendor community's adoption of OVAL.

Given the "in progress" state of many of the interfaces, there may be confusion as to how to use the model in near-term and mid-term implementations. To address this concern, Appendix A discusses how to use the model with and without the availability of specific interfaces. Interestingly, even with the existing interface gaps, the model can still be used to implement current state of the art CM functionality. As the interface gaps are filled, even greater functionality can be achieved eventually enabling full realization of the CM enterprise architecture (EA) model presented in the NIST IR 7756. At that point, organizations should be well equipped to efficiently and cost effectively monitor known issues in a variety of CM domains enabling them to focus scarce resources on more challenging problems.

[28] The website for general information on the scope of the Emerging Specifications effort is http://scap.nist.gov/emerging-specs/index.html. To join the related mailing list, see Emerging Specifications Discussion List at http://scap.nist.gov/community.html.

[29] http://oval.mitre.org/language/interpreter.html

Appendix A – Implementation Maturity Models

NIST IR 7756 describes a CM enterprise architecture and model that is feature rich. It goes beyond standardization of available proprietary CM implementations by adding ideas of plug-and-play components, hierarchical tiers, sensor tasking, multi-tier queries, and digital policy directives. This publication provides specifications for implementing those capabilities. However, as discussed previously, the necessary standard interfaces to achieve these advanced functions may not be available to those implementing a CM system. Either the standards themselves may still be under development or the tools used in a particular implementation may not support the standards.

In such cases, the model may still be suitable for implementation but the resulting architecture will have reduced functionality from the perspective of the full vision of the CM model. A state of "reduced functionality" may actually still result in a robust implementation. For example, it is possible to reproduce in a standardized way a state of the art CM system (e.g., what the Department of State (DOS) has done in their iPost CM implementation[30]) without the availability of the I2, I3, and I4 interfaces. Thus, the I2, I3, and I4 interfaces are necessary only for the more advanced functionality.

The following sections are each focused on a particular interface. They discuss what capabilities are unavailable without that interface and how to mitigate the associated lack of functionality in CM implementations.

Advanced Data Retrieval (I4)

The unavailability of the I4 interface will have the least impact on an implementation of the model compared with the other interfaces. Without I4, the Analysis/Scoring subsystem and the Data Aggregation subsystem will need to be built together as a single product or they will need to use proprietary interconnections. No loss of functionality will occur as the lack of this interface merely limits the ability to fully decompose a CM system into interoperable subsystems.

Querying and Tasking (I3)

The unavailability of the I3 interface will have a substantial impact on the design of a CM system through forcing certain subsystems to be built together, through a loss of envisioned functionality, and through the need to use alternate mechanisms to move data through the model.

Within a single CM instance, the Presentation, Task Manager, and Analysis/Scoring subsystems will need to be built together as a single product or they will need to use proprietary interconnections. Also, the ability to task Collection subsystems (and their related sensors) will not exist. This means that the CM implementation will need to rely on pre-defined views set up within the Collection subsystem's administration console for data collection.

[30] See the DOS scoring guide at http://scap.nist.gov/events/2011/cm_workshop/presentations/index.html.

Within a hierarchical CM implementation, the ability to propagate queries down through multiple tiers will not exist. Thus, the only data available at higher tiers will be that data which is periodically sent up by the lower tier. To push this data up, the Task Manager's Results Publication capability (using the I1 interface) can be used to push the data from a lower tier to the Data Aggregation subsystem of a higher tier. This periodic publication of data would be controlled by queries issued by the Presentation subsystem of the lower tier[31].

Content Acquisition (I2)

The unavailability of the I2 interface will result in lack of functionality in retrieving and propagating digital policy. Overall, this loss will likely have less of an impact on a CM system than the unavailability of the I3 interface.

Within a single CM instance, digital policy and supporting content will need to be manually loaded into the Collection subsystems so that they can collect the data according to organizational policy. The Analysis/Scoring subsystems will either need to have the organization's analysis policies hardcoded into their design (e.g., an organization may choose to use what is already provided by the vendor) or be somehow loaded into the subsystem (probably leveraging a proprietary scripting language). The ability to customize analysis through parameters will have to be implemented through sending those parameters to the Analysis/Scoring subsystem from the Task manager through the I3.3 interface (storing them in the Content subsystem is not an option). Lastly, the ability to automatically retrieve digital content and supporting content from external entities will be unavailable.

Within a hierarchical CM implementation, the mechanism to distribute digital policy and supporting content throughout the enterprise will not exist. This includes the ability to push content down from a higher level tier as well as the ability for a lower level tier to pull content from a higher tier.

Result Reporting (I1)

The unavailability of the I1 interface will have the most impact on implementations of the model relative to the other interfaces. However, this limitation will only impact activity within CM instances and does not affect hierarchical CM instance communication.

Within a CM instance, lack of this interface means that all data reporting to the Data Aggregation subsystem (e.g., from Collection, Task Management, and Analysis/Scoring) will need to be accomplished through proprietary mechanisms.

In summary, each interface provides certain functionality and, in many cases, enables the plug and play subsystem concept. The full vision of the CM model presented in NIST IR 7756 requires the existence and implementation of all four interfaces. However, useful CM implementations may be developed leveraging only the currently available interfaces.

[31] Automating this alternate mechanism to push data up to higher tiers may be a focus of future versions of the model. It could also be addressed through proprietary vendor solutions either in the Presentation subsystem or the Task Manager.

Appendix B – Acronyms

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

ARF Asset Reporting Format specification

CM Continuous Monitoring
CTL Common Tasking Language

DOS Department of State

EA Enterprise Architecture

FIFO First-In-First-Out
FISMA Federal Information Security Management Act

HTTP Hypertext Transfer Protocol
HTTPS Hypertext Transfer Protocol Secure

I Interface
ID Identifier
IETF Internet Engineering Task Force
ISIMC Information Security and Identity Management Committee
IT Information Technology
ITL Information Technology Laboratory

NIST National Institute of Standards and Technology
NIST IR National Institute of Standards and Technology Interagency Report
NVLAP National Voluntary Laboratory Accreditation Program

OVAL Open Vulnerability Assessment Language

SCAP Security Content Automation Protocol
SOAP Simple Object Access Protocol[32]
SQL Structured Query Language
SSL Secure Sockets Layer
SSN Social Security Number

TLS Transport Layer Security
TM Task Manager subsystem

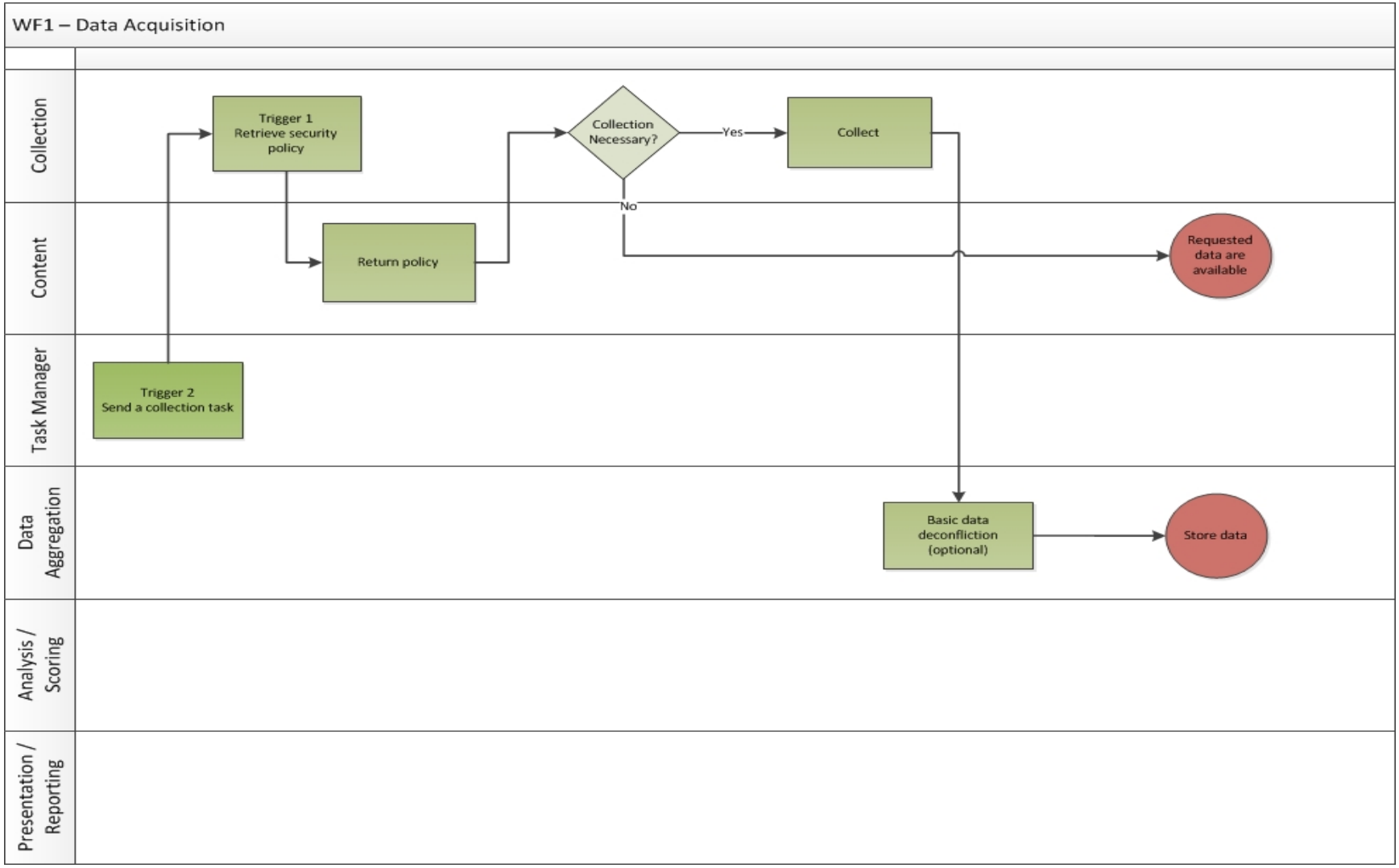
URI Uniform Resource Identifier
URL Uniform Resource Locator
USG United States Government

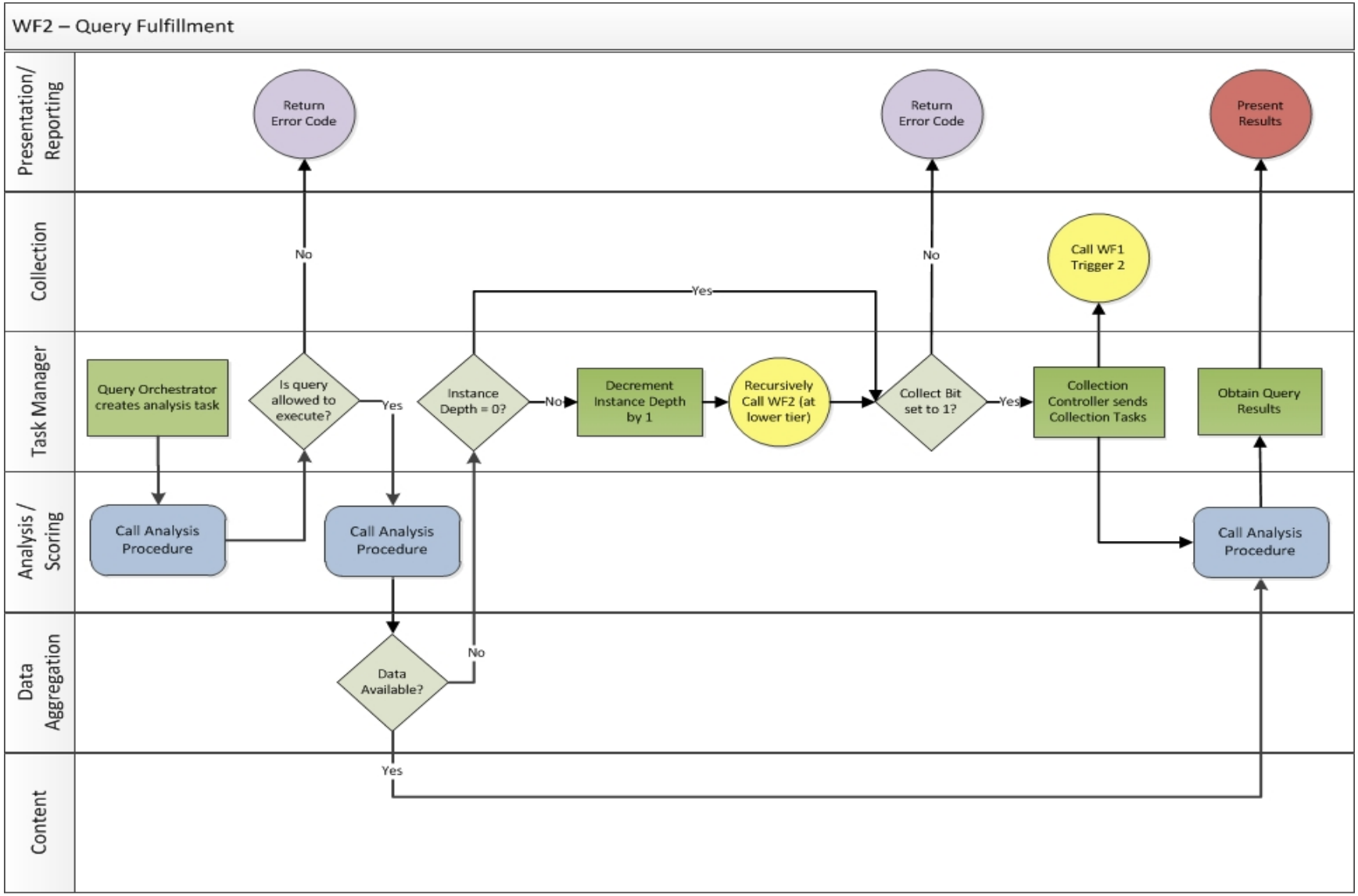
[32] In version 1.2 in 2007, this expansion of the SOAP acronym was dropped but no replacement was provided.

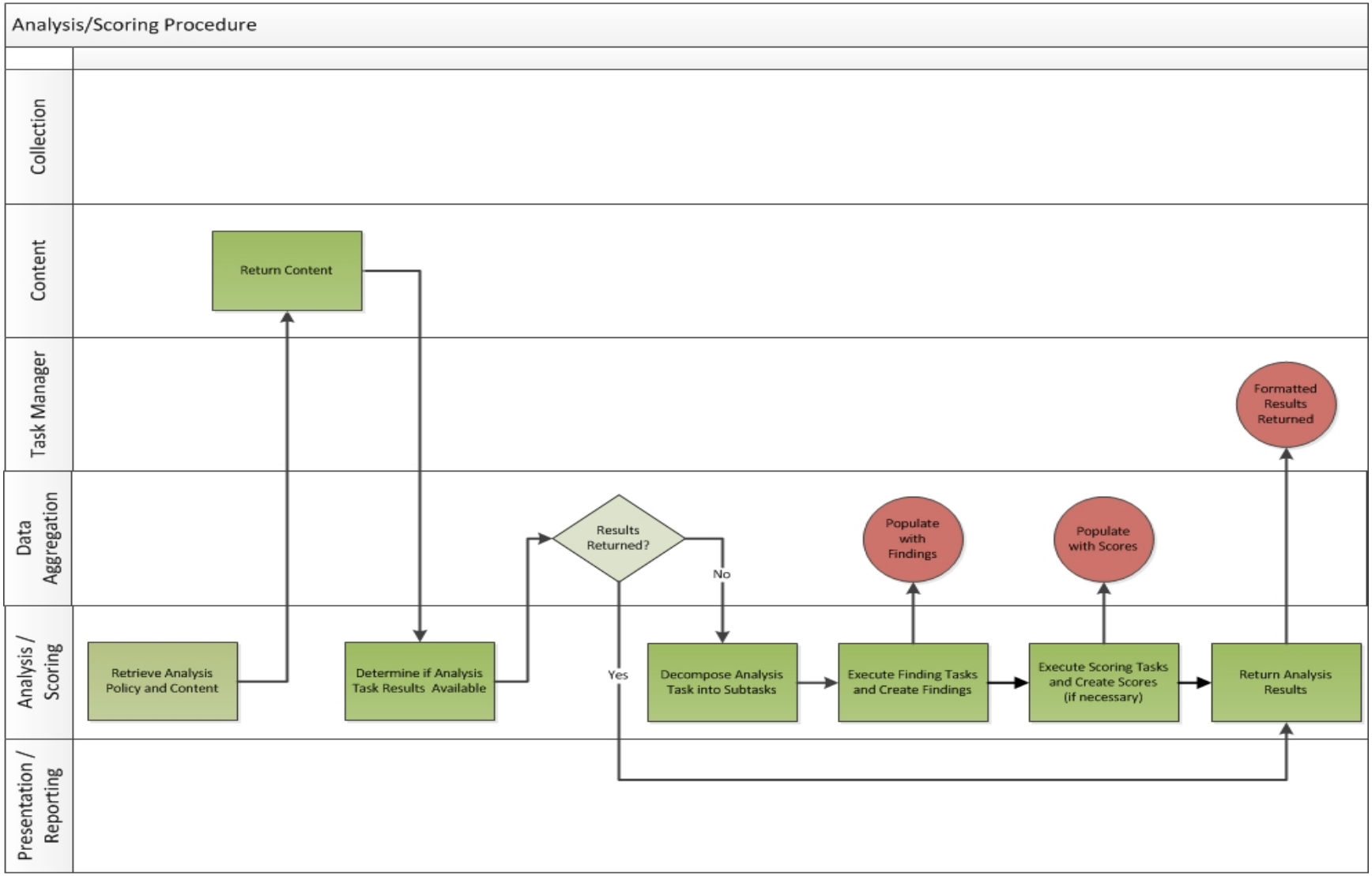
WF	Workflow
WS-A	WS-Addressing
WSDL	Web Services Description Language
WS-MC	WS-MakeConnection
WS-S	WS-Security
XML	Extensible Markup Language

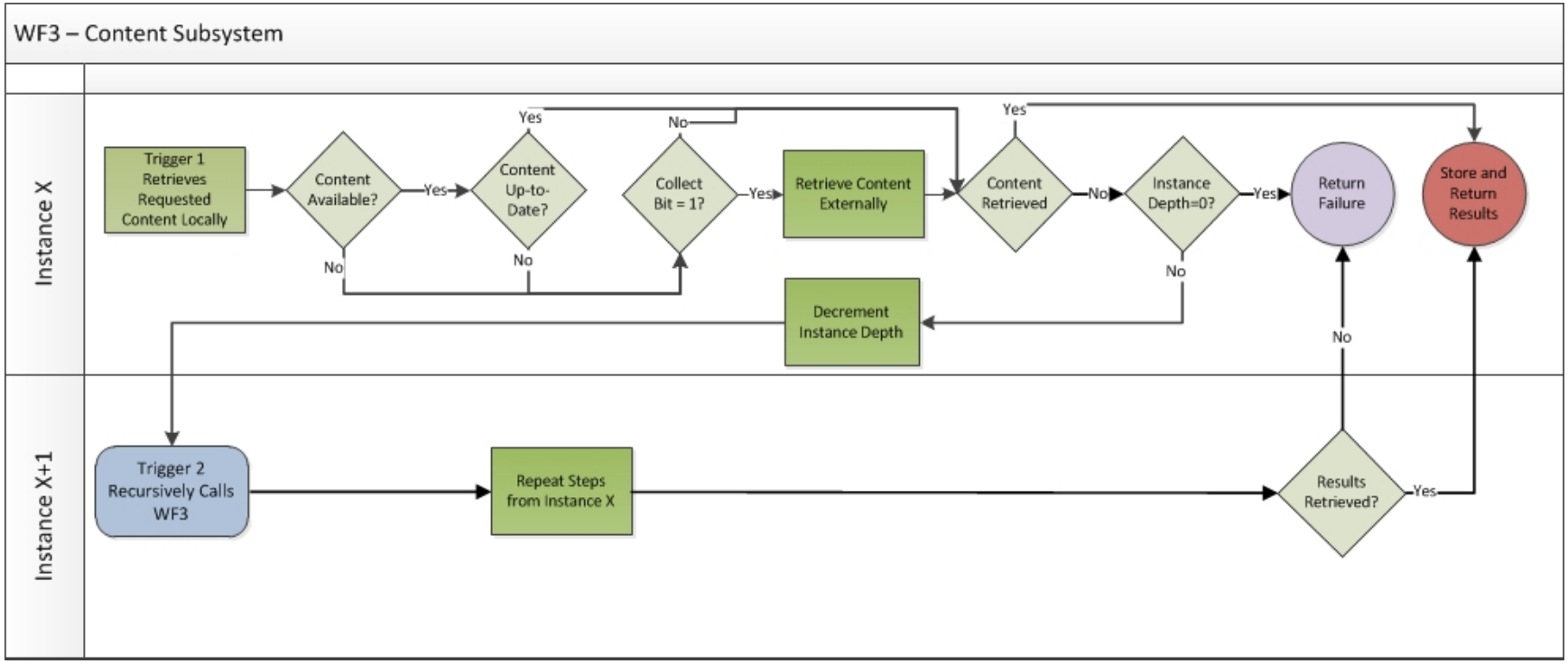
Appendix C – Workflow Diagrams

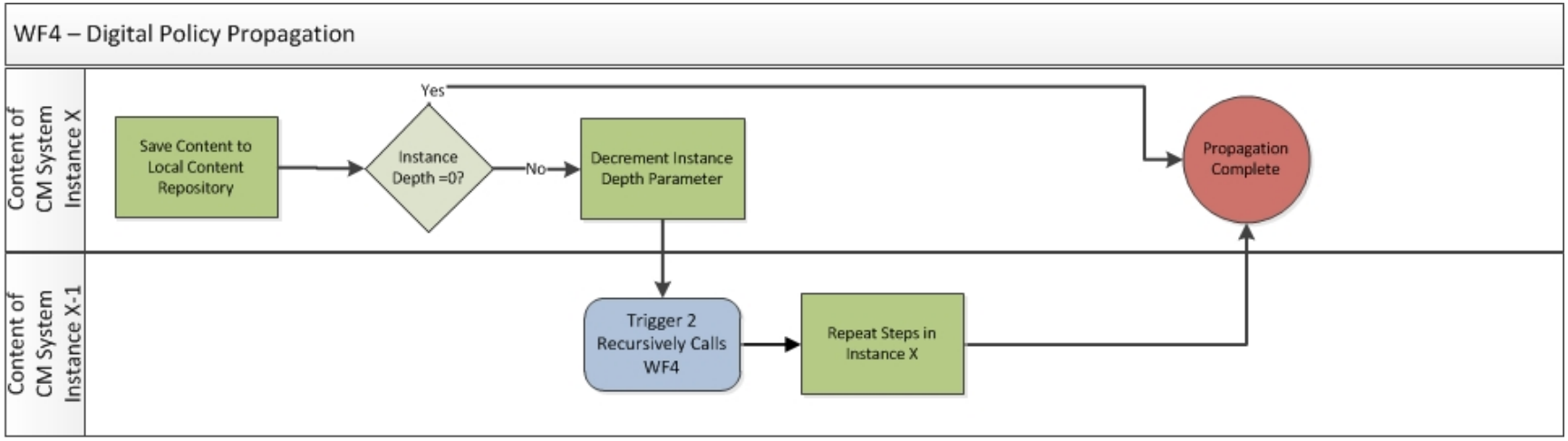
Below are the Workflow Diagrams that illustrate the CM Workflows described in the Section 3.




WF2 – Query Fulfillment
Presentation/ Reporting
Collection
Task Manager
Analysis / Scoring
Data Aggregation
Content
Return Error Code
Return Error Code
Present Results
No
No
Yes
Call WF1 Trigger 2
Query Orchestrator creates analysis task
Is query allowed to execute?
Yes
Instance Depth = 0?
No
Decrement Instance Depth by 1
Recursively Call WF2 (at lower tier)
Collect Bit set to 1?
Yes
Collection Controller sends Collection Tasks
Obtain Query Results
Call Analysis Procedure
Call Analysis Procedure
Call Analysis Procedure
Data Available?
No
Yes


Analysis/Scoring Procedure
Collection
Content
Task Manager
Data Aggregation
Analysis / Scoring
Presentation / Reporting
Return Content
Retrieve Analysis Policy and Content
Determine if Analysis Task Results Available
Results Returned?
No
Yes
Decompose Analysis Task into Subtasks
Execute Finding Tasks and Create Findings
Populate with Findings
Execute Scoring Tasks and Create Scores (if necessary)
Populate with Scores
Return Analysis Results
Formatted Results Returned


WF3 – Content Subsystem
Instance X
Instance X+1
Trigger 1 Retrieves Requested Content Locally
Content Available?
Yes
No
Content Up-to-Date?
Yes
No
Collect Bit = 1?
Yes
No
Retrieve Content Externally
Content Retrieved
Yes
No
Instance Depth=0?
Yes
No
Return Failure
Store and Return Results
Decrement Instance Depth
Trigger 2 Recursively Calls WF3
Repeat Steps from Instance X
Results Retrieved?
No
Yes


WF4 – Digital Policy Propagation
Content of CM System Instance X
Content of CM System Instance X-1
Save Content to Local Content Repository
Instance Depth =0?
Yes
No
Decrement Instance Depth Parameter
Propagation Complete
Trigger 2 Recursively Calls WF4
Repeat Steps in Instance X