CMDB Federation (CMDBf)

Committee Draft

Version 1.0b, 04 January 2008

Authors

Forest Carlisle, CA
Barry Day, Microsoft
Scott Donohoo, IBM
Pratul Dublish, Microsoft
Jacob Eisinger, IBM
Greg Goodman, CA
Andrew Hately, IBM
Mark Johnson, IBM (Editor)
Vincent Kowalski, BMC Software (Editor)
Yannis Labrou, Fujitsu
Kenji Morimoto, Fujitsu
David Snelling, Fujitsu
William Vambenepe, (formerly of) HP (Editor)
Marv Waschke, CA (Editor)
Van Wiles, BMC Software (Editor)
Klaus Wurster, HP

Copyright Notice

THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE SPECIFICATION.

The name and trademarks of the Authors may NOT be used in any manner, including advertising or publicity pertaining to the Specification or its contents without specific, written prior permission. Title to copyright in the Specification will at all times remain with the Authors.

No other rights are granted by implication, estoppel or otherwise.

Abstract

This specification describes the architecture and interactions for federating data repositories together to behave as a data store that satisfies the role of a Configuration Management Database (CMDB). The federation provides an aggregate view of a resource even though the data and underlying repositories are heterogeneous. A query interface is defined for external clients to access these data.

Status

This document is an initial draft still under internal review. A feedback agreement is required before the working group can accept feedback.

At some future date, the contents may be published under another name or under several new specifications, as shall be agreed by the authors and their respective corporations at that time.

Table of Contents

1. Introduction

Many organizations are striving to base IT management on a CMDB (Configuration Management Database). A CMDB contains data describing managed resources like computer systems and application software, process artifacts like incident, problem and change records, and the relationships among these entities. The contents of the CMDB should be managed by a configuration management process and serve as the foundation for other IT management processes, such as change management and availability management.

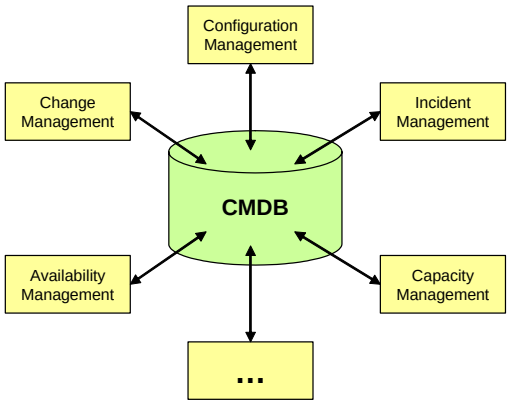


Figure 1 – Role of a CMDB

In practice this goal is challenging because the management data are scattered across repositories that are poorly integrated or coordinated.

The definition of a CMDB in the context of this specification is based on the definition described in the IT Infrastructure Library** (ITIL**): a database that tracks and records configuration items associated with the IT infrastructure and the relationships between them. Strictly speaking, the ITIL CMDB contains a record of the expected configuration of the IT environment, as authorized and controlled through the change management and configuration management processes. The federated CMDB in this specification extends this base definition to federate any management information that complies with the specification's patterns, schema, and interfaces, such as the discovered actual state in addition to the expected state. Typically, an administrator will select the data to be included by configuring the tool that implements the CMDB.

The federated CMDB described in this specification is a collection of services and data repositories that contain configuration and other data records about resources. The term 'resource' includes configuration items (e.g., a computer system, an application, or a router), process artifacts (e.g., an incident record, a change record), and relationships between configuration item(s) and/or process artifact(s). The architecture describes a logical model and does not necessarily reflect a physical manifestation.

1.1 Objectives

1.1.1 Functions

The federated CMDB resulting from using this specification will provide a single aggregate view of the data about an IT resource, even if the data is from different heterogeneous data repositories, as shown in Figure 2. Clients, such as IT processes, management applications, and IT staff will use a query service defined in the specification to access aggregated or non-aggregated views. Data repositories will use the services described in the specification to provide the aggregated view.

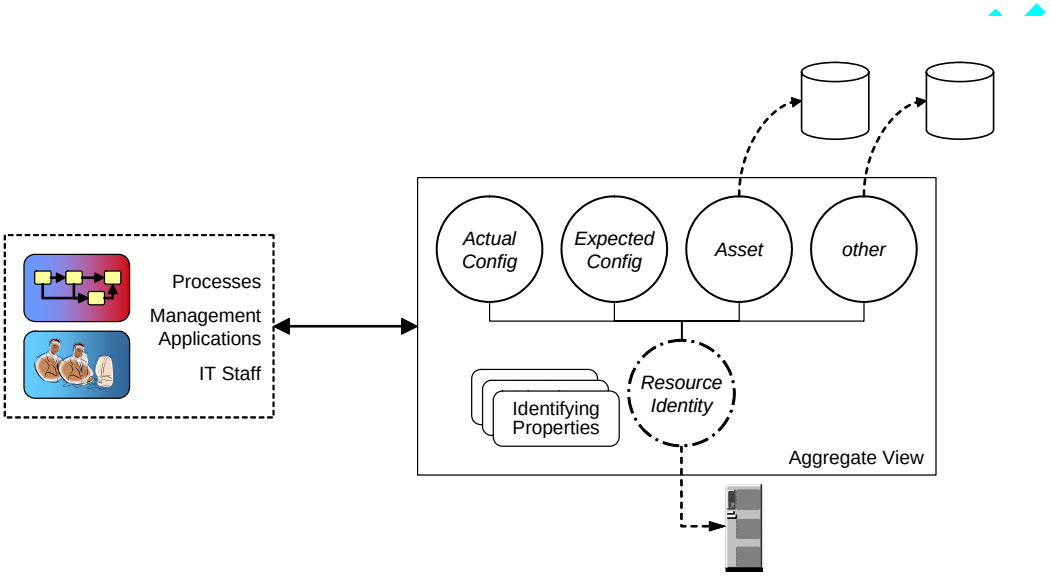


Figure 2 – Aggregate View from Federated Data

The federated CMDB could support the following scenarios (though which scenarios are supported is entirely left to the discretion of each implementation):

- Maintain accurate picture of IT inventory from a combination of asset information (finance) and deployment/configuration
- Reflect changes to IT resources, including asset and licensing data, across all repositories/data sources
- Compare expected configuration vs. actual configuration
- Enable version awareness. Examples:
  - Coordinate planned configuration changes
  - Track change history
- Relate configuration and asset data to other data/data sources, such as incident, problem, and service levels. Examples:
  - Integration of change/incident management with monitoring information
  - SLA incident analysis – use of service desk/incident information in a dependency analysis on both configurations and change records

1.1.2 Target IT Environment

This specification is intended to address requirements in IT environments with the following characteristics

- There are strong requirements to consolidate into one or more databases (logical and/or physical) at least some key data from the many management data repositories so that IT processes can be more effective and efficient.
- IT organizations that implement a CMDB that federates multiple management data repositories will be diverse in terms of their existing tools, process maturity level, usage patterns, and preferred adoption models.
- There are several and possibly many management data repositories (MDRs), each of which may be considered an authoritative source for some set of data.
- The authoritative data for a resource may be dispersed across multiple MDRs.
- It is often neither practical nor desirable for all management data to be kept in one data repository, though it may be practical and desirable to consolidate various subsets of the data into fewer databases.
- Existing management tools will often continue to use their existing data sources. Except over the very long haul, it is not realistic to expect them all to be modified to require and utilize new consolidated databases.

1.1.3 Non-Goals

The following are outside the scope of the specification.

- The mechanisms used by each management data repository to acquire data. For example, the mechanisms could be external instrumentation or proprietary federation and replication function.
- The mechanisms and formats used to store data. The specification is concerned only with the exchange of data. A possible implementation is a relational database that stores data in tables. Another possible implementation is a front-end that accesses the data on demand from an external provider, similar to a commonly used CIMOM/provider pattern.
- The processes used to maintain the data in the federated CMDB. The goal of the specification is to enable IT processes to manage this data, but not to require or dictate specific processes.
- The mechanisms used to change the actual configuration of the IT resources and their relationships. The goal of the specification is to provide means to represent changes after or as they are made, but not to be the agent that makes the change.

1.2 Background Terminology

This section defines terms used throughout this specification. For the most part, these terms are adopted from other sources. The terms are defined here to clarify their usage in this specification and, in some cases, to show their relationship to the use of the terms in other sources. In particular, this specification shares concepts with ITIL (Information Technology Infrastructure Library.) ITIL is not a standard and does not provide normative definitions of terms. However, the ITIL v3 glossary is quoted below as representative of the ITIL position.

Configuration Item (CI) A Configuration Item is a basic tangible or intangible entity in a configuration management solution such as a CMDB. ITIL v3 defines a CI as

> Any Component that needs to be managed in order to deliver an IT Service. Information about each CI is recorded in a Configuration Record within the Configuration Management System and is maintained throughout its Lifecycle by Configuration Management. CIs are under the control of Change Management. CIs typically include IT Services, hardware, software, buildings, people, and formal documentation such as Process documentation and SLAs.

Configuration Management Database (CMDB) ITIL defines a CMDB as

> A database used to store Configuration Records throughout their Lifecycle. The Configuration Management System maintains one or more CMDBs, and each CMDB stores Attributes of CIs, and Relationships with other CIs.

A Configuration Management Database (CMDB) is often implemented using standard database technology and typically persists CI lifecycle data as records (or Configuration Records) in that database. Configuration records are managed according to some data or information model of the IT environment. One of the goals of this specification is to expedite the federated implementation of multiple CMDBs in a single Configuration Management System.

Configuration Record ITIL defines a Configuration Record as

> A Record containing the details of a Configuration Item. Each Configuration Record documents the Lifecycle of a single CI. Configuration Records are stored in a Configuration Management Database.

For the purposes of this specification, a CI is a tangible or intangible entity treated in the abstract by this specification, while a Configuration Record contains concrete data pertaining to a CI. More than one Configuration Record may be associated with a given CI. Often Configuration Records will be from different data sources or document different points in the lifecycle of a CI. It is possible for Configuration Records associated with a single CI to contain data that may appear contradictory and require mediation.

Federated CMDB A federated CMDB is a combination of multiple management data repositories (MDRs), at least one of which federates the others, into an aggregate view of management data. Note that whereas “federated CMDB” refers to the combination of all the data repositories, “Federating CMDB” is a specific role performed by a data repository that federates other MDRs.

Federation The process of combining information from management data repositories (MDRs) into a single representation that can be queried in a consistent manner. Federation is often contrasted with Extract, Transform, and Load (ETL) systems which transfer and store data from one repository to another. This specification does not exclude ETL activities, especially for caching, but the main purpose of the specification is to support systems that minimize or eliminate transferring and storing data from MDRs in federators.

Graph A **graph** is a kind of data structure, specifically an abstract data type, that consists of a set of nodes and a set of edges that establish relationships (connections or links) between the nodes. In this specification the nodes are Items and the edges are Relationships.

Identity The federated CMDB contains data pertaining to real world entities. The identity of each of these real world entities is a set of qualities or characteristics that

distinguish the entity from other entities of the same or different types. This set of qualities may be called the 'identifying properties' of the entity.

ITIL ITIL stands for Information Technology Infrastructure Library and is a framework of best practices for delivering IT services. Two versions of ITIL are currently in use: version 2 released in 2000 and version 3 released in 2007. Since v3 has not yet superseded v2 in practice, both versions have been considered in preparing this specification. A CMDB is a key component in the ITIL best practices.

1.3 Notational Conventions

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

This specification uses the following syntax to define outlines for messages:

- The syntax appears as an XML instance, but values in italics indicate data types instead of literal values.
- Characters are appended to elements and attributes to indicate cardinality:
  - "?" (0 or 1)
  - "*" (0 or more)
  - "+" (1 or more)
  - The absence of any of the above indicates the default (exactly 1)
- The character "|" is used to indicate a choice between alternatives.
- The characters "(" and ")" are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.
- The characters "[" and "]" are used to call out references and property names.
- xs:any and xs:anyAttribute indicate points of extensibility. Additional children and/or attributes MAY be added at the indicated extension points but MUST NOT contradict the semantics of the parent and/or owner, respectively. By default, if a receiver does not recognize an extension, the receiver SHOULD ignore the extension; exceptions to this processing rule, if any, are clearly indicated below.
- Ellipses (i.e., "...") indicate that details are omitted for simplicity, and a further explanation is provided below.
- XML namespace prefixes are used to indicate the namespace of the element being defined or referenced.

2. Technological Assumptions

This specification is based on some very specific assumptions with regard to underlying technology and the context of computing standards that exists at the time of its writing.

2.1 Underlying Technology

2.1.1 Web Services

Although the interface specification contained herein is generic, it assumed that implementations will be based on Web Services. Although interfaces based on programming languages such as Java and C# could be derived from this

specification, such interfaces are considered out of scope and are not addressed here.

2.1.2 Database Management Systems

In general practice CMDBs are implemented using commercially available database technology. Although this is a specification about how one or more CMDBs federate data using a standard mechanism, no assumptions are made about how that federated data is stored or persisted. What is important are the interfaces; their behavior and the data types they convey. Database technology is clearly a needed component in the implementation of this specification, but its use is considered to be a hidden detail of such implementations.

2.2 Standards Basis

This specification builds upon the work of other standards in the area Web Services. The specific standards that this specification is based on are as follows.

- HTTP/1.1
- XML Schema 1.0 Part 1: Structures
- SOAP 1.1
- WSDL 1.1
- WS-I Basic Profile 1.1

3. Architecture

The architecture defines four roles, which implement or use two services. In Figure 3 the roles are (green) shaded shapes with dotted edges and the services are (yellow) shaded rounded boxes with solid edges.

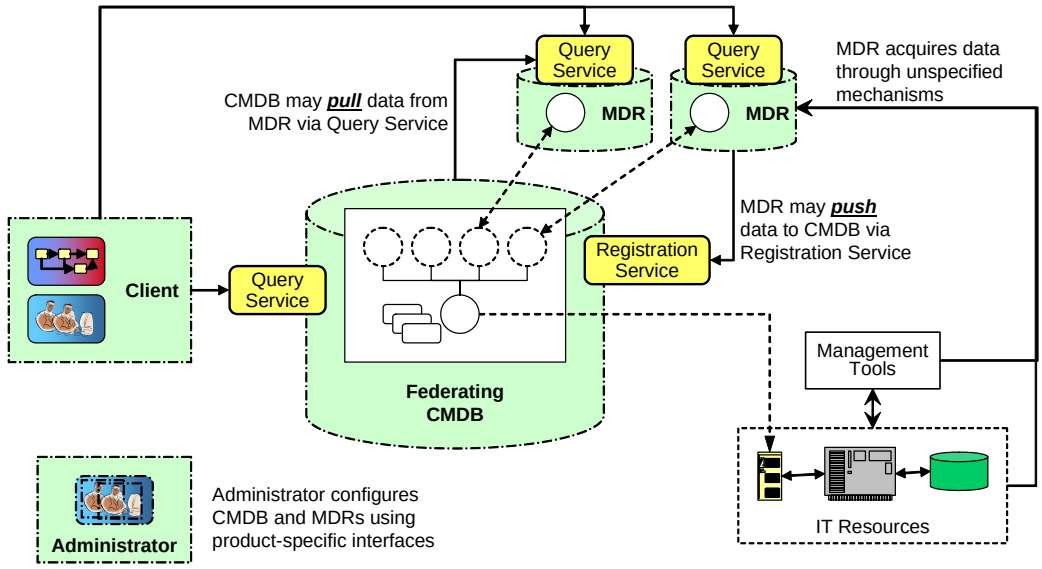


Figure 3 – Roles and Services

3.1 Roles

MDR (Management Data Repository). An MDR provides data about managed resources (e.g., computer systems, application software, and buildings) and/or process artifacts (e.g., incident records and request for change forms), and the relationships between them. In this architecture, managed resources and process artifacts are both called 'items'. The means by which the MDR acquires data is not specified. Examples include direct from instrumented resources or indirectly through management tools.

Federating CMDB. A Federating CMDB federates data from MDRs, and may also contain non-federated data. It provides an aggregate view of an item or relationship, potentially using data from multiple MDRs. A Federating CMDB and all the MDRs together comprise a federated CMDB.

It is possible for one Federating CMDB to have its data federated by a second Federating CMDB. In this case, the first Federating CMDB would appear to the second Federating CMDB to be an MDR. The second Federating CMDB would not be aware of any federation performed by the first Federating CMDB.

Client. A Client is a consumer of management data, either directly from an MDR or an aggregated view from a Federating CMDB. Examples of clients are IT process workflows, management tools, and IT administrators. Clients only read data; there are no provisions for a client to update data through an interface defined in this architecture.

Administrator. An Administrator configures MDRs and Federating CMDBs so they can interact with each other. Administration includes selecting and specifying the data that is federated, describing service endpoints, and describing which data are managed through each endpoint. Administration is done using interfaces that are specific to each tool that acts in the MDR and/or Federating CMDB role.

3.2 Services Overview

The architecture defines two services. There is an implementer of a service and a client (caller) of a service.

Query Service. Both MDRs and Federating CMDBs may implement the Query service to make data available to Clients. Queries may select and return items, relationships, and/or graphs containing items and relationships.

Registration Service. A Federating CMDB may implement the Registration Service. An MDR may call the Registration Service to register data that it has available for federation. A Federating CMDB declares the data types that its Registration service supports. An MDR maps its data to the supported types.

3.2.1 Federation Modes

There are two modes available to federate data. A Federating CMDB must use one or the other mode and MAY use both.

Push Mode. In push mode, the MDR initiates the federation. Typically an administrator configures the MDR by selecting to federate some data types that are supported by both the MDR and the registration service. The MDR notifies the Registration service any time this data is added, updated, or deleted. Depending on the extent of the data types, the registered data may be limited to identification data or it may include many other properties that describe the item or relationship state.

Pull Mode. In pull mode, the Federating CMDB initiates the federation. Typically, an administrator configures the Federating CMDB by selecting the MDR data types that

will be federated. The Federating CMDB queries MDRs for instances of this data. Depending on the implementation, the Federating CMDB may pass through queries to MDRs without maintaining any state, or it may cache some set of MDR data, such as the data used to identify items and relationships.

3.2.2 Usage Profiles

Table 1 lists the service usage profiles for the roles described in section 3.1 that implement or use the services.

Table 1 – Service Usage Profiles

Role	Query service		Registration service	
	Implementation	Client	Implementation	Client
Federating CMDB – Push Mode	REQUIRED	Optional	REQUIRED	N/A
Federating CMDB – Pull Mode	REQUIRED	REQUIRED	N/A	N/A
MDR – Push Mode	Optional	N/A	N/A	REQUIRED
MDR – Pull Mode	REQUIRED	N/A	N/A	N/A
Client (external)	N/A	REQUIRED	N/A	N/A

3.3 Identity Reconciliation

Managed resources are often identified in multiple ways, depending on the management perspective. Examples of management perspectives are a change management process and an availability monitoring tool. Understanding how to identify resources, and reconciling the identifiers across multiple perspectives, is an important capability of a Federating CMDB. The following pattern is used:

- Each MDR identifies a resource based on one or more identifying properties of the resource. Identifying properties are physical or logical properties that distinguish unique instances of resources. Examples are MAC addresses, host names, and serial numbers. Often, more than one property will be necessary to uniquely distinguish a resource, especially when information is incomplete. In addition, when two or more MDRs contain data on a single resource, individual MDRs may choose or have available different identifying properties, which they may use in their resource identifier for the item or relationship.
- Each MDR knows at least one unique and unambiguous identifier for each item or relationship it contains and/or provides access to via the Query service.
- A Federating CMDB attempts to reconcile the item and relationship identification information from each MDR, recognizing when they refer to the same item or relationship.

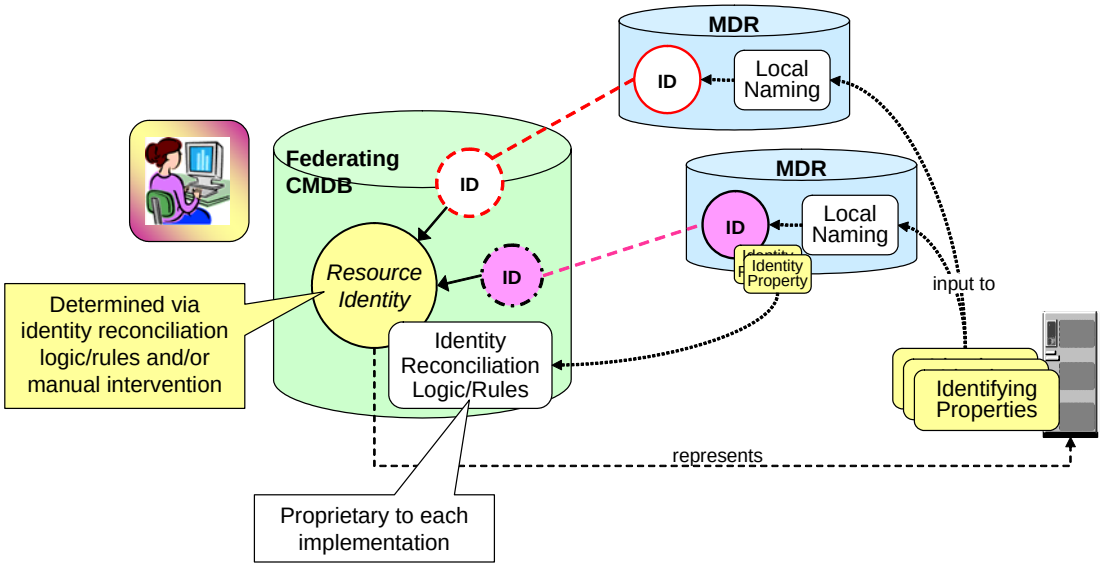


Figure 4 – Identity Reconciliation

The Federating CMDB performs this mapping using any combination of automated analysis and manual input, as shown in Figure 4. In a typical implementation the Federating CMDB analyzes the identifying properties to determine the resource identity. As each item or relationship is registered, the service determines if this item or relationship is already registered or is new. The determination of identity is seldom absolute and often must rely on heuristics because different MDRs typically know about different characteristics of an entity and thus establish different sets of identifying properties which characterize the entities they handle. Further, the determination may change as additional information is discovered and MDRs add, subtract, or change identifying properties as systems evolve.

3.4 Data Model Overview

3.4.1 Managed Data

The architecture defines three elements that each wrap properties specific to the type of item or relationship.

Item. An item represents a managed resource (e.g., computer systems, application software, and buildings) or a process artifact (e.g., incident record and request for change form). With this definition, 'item' is a superset of the 'configuration item' term defined in ITIL. Formally:

- Each item MUST have at least one ID that is unique within the scope of the MDR that contains it and that serves as a key.
- Once an ID has been assigned to an item, it MAY be used in any situation requiring an ID.
- Once an ID has been assigned to an item, it MUST never refer to anything except the original item.

- An Instance ID of an item is the composition of the unique MDR ID and the unique item ID assigned by that MDR. The Instance ID is therefore unique within the group of federated repositories.

Examples of when an item might have multiple IDs include when an item is reconciled across several MDRs and the Federating CMDB knows it by all of the IDs that have been assigned by different MDRs; when two items are thought to be different but are later reconciled to the same item; when an ID changes for any other reason.

Given that each MDR has a unique ID within the group of federated repositories, and that each MDR assigns a unique ID within its own scope, the combination of the MDR ID and the MDR-assigned item ID results in an instance ID that is unique within the group of federated repositories. This instance ID serves two purposes:

- It is an unambiguous identifier for the representation of the item held by the MDR that assigned the instance ID.
- The MDR ID portion of the instance ID identifies the MDR that assigned the instance ID. A client MAY introspect the instance ID to extract the MDR ID. The client may then use the MDR ID to acquire the query service address for this MDR. For example, the MDR ID might be the key in a registry that contains the service addresses for each MDR. The client may then issue a query to this address to retrieve the representation of the item.

When a Federating CMDB federates item data from an MDR, it MAY respond to queries for the representation of the item. It may reuse the instance ID assigned by the MDR as long as the representation that it returns is the same as the representation that would be returned by the MDR that assigned the instance ID. If the Federating CMDB alters the representation, such as overwriting some property values or associating other records to the same item, it must assign a new instance ID using its own MDR ID.

This constraint on reusing IDs does not preclude caching of the MDR data in the Federating CMDB. In particular, at any instant in time a query to the Federating CMDB may return a different representation than the same query to the MDR. This is in recognition of the distributed configuration of the repositories and the absence of any requirements that their data are coherent, such as requiring transactional closure across the repositories for any update.

Relationship. A relationship represents a connection from a source item to a target item. Examples include software 'runs on' operating system, operating system 'installed' on computer system, incident record 'affects' computer system, and service 'uses' (another) service. Formally:

- A relationship links exactly two items and provides information pertaining to that relationship.
- A relationship is a subclass of item (though the relationship XML schema does not formally extend the item XML schema).
- Therefore, like an item, each relationship MUST have an ID that is unique within the scope of the MDR that contains it and that serves as a key
- A reconciled relationship MAY have more than one such ID.

Record. A record contains properties that describe an item or relationship. Formally:

- A record is associated with exactly one item or relationship.
- A record MAY contain properties that are useful to identify the item or relationship, or other properties that describe the item or relationship.

- Several records, possibly of various types, MAY be associated to the same item or relationship.

Records may differ from other records for various reasons, including types of data (e.g., asset vs. configuration), different sets of properties from different providers, different versions, and expected vs. observed data. A record is similar to a row in a SQL view. It is a projection of properties. The same property may appear in multiple records for the same item or relationship. The record may have no properties, in which case it serves as a marker.

Each record has metadata properties that describe the record itself (as opposed to properties that describe the item or relationship).

- An ID that is unique within the scope of its associated item or relationship and that serves as a key. This property is required.
- The date/time the record was last modified (optional).
- A baseline ID that may be used to indicate which of the possibly multiple expected (authorized) configuration baselines this record represents (optional).
- A snapshot ID that may be used to indicate which of the possibly multiple configuration observations this record represents (optional).

The data contained in an MDR or Federating CMDB is a graph where the items are nodes and the relationships are links. The graph is not necessarily connected (there may not be a relationship trail from any item to any other item). The query interface described below allows queries to be constructed based on aspects of the graph (e.g. existence of a relationship between two items) and based on properties of the items and relationships (e.g. requirements for a certain value of a given record property or a certain type for the item/relationship).

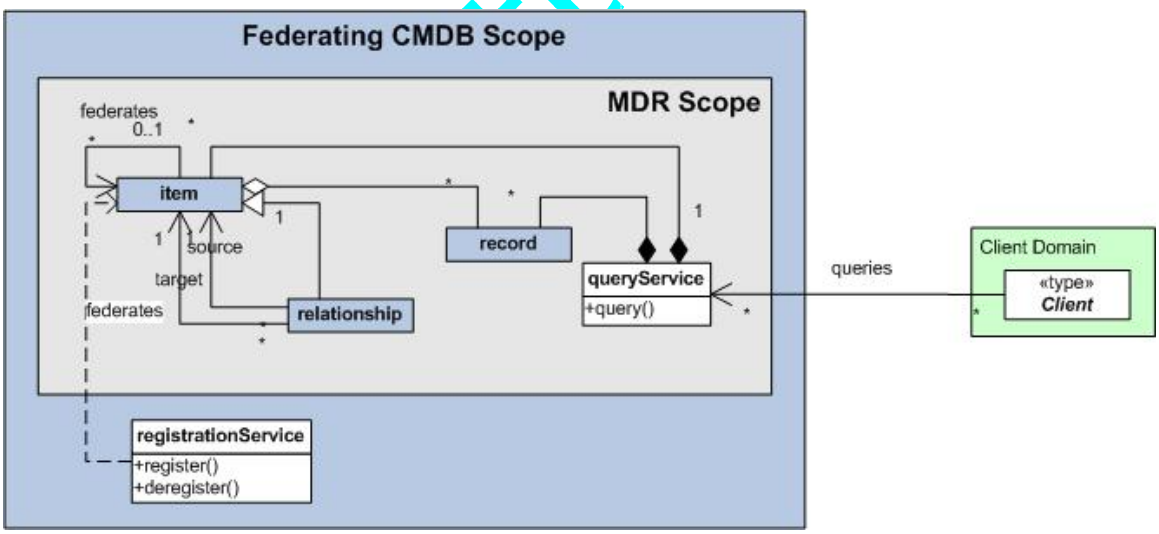


Figure 5 –Data & Services Overview

3.4.2 Common data element types

The cmdbf:MdrScopedIdType is used in several places to identify an item or relationship.

The <instanceId> element is of the type of cmdbf:MdrScopedIdType. The pseudo-schema of the <instanceId> element is:

```
(01) <instanceId>
(02)   <mdrId>xs:anyURI</mdrId>
(03)   <localId>xs:anyURI</localId>
(04) </instanceId>
```

This can be abbreviated in a pseudo schema to be:

```
(01) <instanceId>cmdbf:MdrScopedIdType</instanceId>
```

The cmdbf:MdrScopedIdType is composed of a pair of URIs. The first URI, <mdrId>, is the ID of the MDR that assigned this instance Id to the instance. The second URI, <localId>, is the Id that uniquely identifies the instance within the MDR. The combination of these two URIs identifies the instance in a globally unique way.

There is no expectation that these two URIs are able to be de-referenced.

Every <record> element has exactly one child element of unrestricted content (which is typically used to describe the item or relationship with which the record is associated), followed by a <recordMetadata> element containing common information about the record itself. The <recordMetadata> element contains these properties:

- recordId: the unique Id of the record in the MDR. Required.

The <recordMetadata> element MAY also contain these properties:

- lastModified: the time/date the record was last modified in ISO 8601 format. The applicable time zone or UTC MUST be indicated.
- baselineId: the name or other identifier used to group records into a particular baseline configuration. A value of "0" indicates that this record is not part of any baseline configuration.
- snapshotId: the name or other identifier used to group records observed in a configuration snapshot (discovery). A value of "0" indicates that this record is not part of any snapshot configuration.
- extensibility elements: additional metadata elements not defined by the specification may also be included

4. Query Service

4.1 Overview

The Query service can be provided by MDRs and Federating CMDBs. It provides a way to access the items and relationships that the provider (MDR or Federating CMDB) has access to, whether this provider actually holds the data or federates the source of the data. The Query service contains a GraphQuery operation, that can be used for anything from a simple instance query to a much more complex topological query.

A GraphQuery request describes the items and relationships of interest in the form of a graph. Constraints can be applied to the nodes (items) and edges (relationships) in that graph to further refine them. The GraphQuery response contains the items and relationships that, through their combination, compose a graph that satisfies the constraints of the graph in the query.

The following example and normative definition of the interface provide a more complete description of the request and response messages for the GraphQuery operation.

4.2 Example

Let's assume that an MDR contains two types of items (people and computers) and one type of relationship (a person "uses" a computer). Here is a simple query request to select all computers that are used by a person located in California:

```
(01) <query>

(02)   <itemTemplate id="user">
(03)     <recordConstraint>
(04)       <recordType namespace="http://example.com/people"
                          localName="person"/>
(05)       <propertyValue namespace="http://example.com/people"
                            localName="state">
(06)         <equal>CA</equal>
(07)       </propertyValue>
(08)     </recordConstraint>
(09)   </itemTemplate>

(10)   <itemTemplate id="computer">
(11)     <recordConstraint>
(12)       <recordType namespace="http://example.com/computer"
                          localName="computer"/>
(13)     </recordConstraint>
(14)   </itemTemplate>

(15)   <relationshipTemplate id="usage">
(16)     <recordConstraint>
(17)       <recordType namespace="http://example.com/computer"
                          localName="uses"/>
(18)     </recordConstraint>
(19)     <sourceTemplate ref="user"/>
(20)     <targetTemplate ref="computer"/>
(21)   </relationshipTemplate>

(22) </query>
```

The detailed syntax and semantics of the XML elements are described in details in later sections, but here is in summary what items and relationships are returned by this query:

The <itemTemplate> called "user" (line 02) matches all items that:

- have a record with a property called "state" (in the http://example.com/people namespace) for which the value is "CA",
- have a record named "person" (defined in the namespace "http://example.com/people"), and
- are the source of a relationship that matches the <relationshipTemplate> called "usage" (line 11)

643 The <itemTemplate> called “computer” (line 08) matches all items that:

644 • have a record named “computer” (defined in the namespace
645 “http://example.com/computer”), and

646 • are the target of a relationship that matches the <relationshipTemplate>
647 called “usage” (line 11)

648 The <relationshipTemplate> called “usage” (line 11) matches all relationships that:

649 • have a record named “uses” (defined in the namespace
650 “http://example.com/computer”),

651 • have a source that matches the <itemTemplate> called “user” (line 02), and

652 • have a target that matches the <itemTemplate> called “computer” (line 08).

653 As a result, if a user item does not “use” a computer, it will not be part of the
654 response, whether the user is located in California or not.

655 Here is a graphical representation of the query:

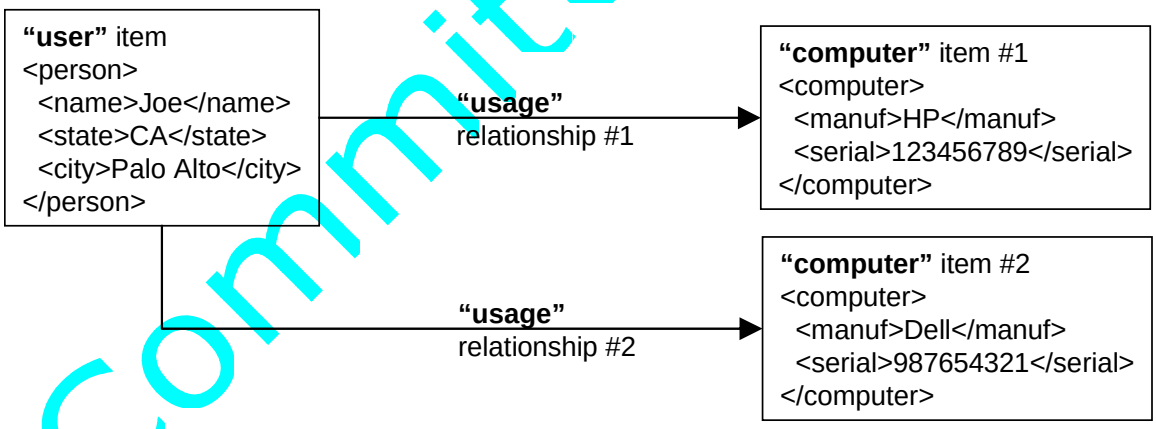
656

657

658

659 If a user located in California happens to “use” two computers, this is represented in
660 the response by three items (one for the user and one for each computer) and two
661 relationships (going from the user to each of his/her computers). Later section will
662 describe the syntax and semantics of the response message in more details. Here is
663 a graphical representation of this response:

664



665

666 In effect, the response contains two graphs, each made of a user, a computer and
667 the relationship between the two, that both meet the constraints of the query graph.
668 In this example, the two graphs in the response happen to overlap (they share the
669 same “user”) but in another example they could be disjoint (e.g. if the second
670 computer was instead “used” by another user also located in California).

671 If the <relationshipTemplate> element (line 11) was not part of the query, the
672 semantics of the query would be very different. The query would return all the items
673 of type “person” that are in California and all the items of type “computer”. It would
674 not return the relationships between users and computers. The existence (or non-
675 existence) of these relationships would have no bearing on what items are returned.

676

The GraphQuery operation can also use relationships to qualify instances, even when the result of the query does not include relationships. In the example above, suppose that all we are interested in are the computers used by people in California, but not the users themselves. We can use the same query as above, but add suppressFromResult="true" to the "user" and "usage" templates. The query result is simply the two computers listed above.

```
(01) <query>

(02)   <itemTemplate id="user" suppressFromResult="true">
(03)     <recordConstraint>
(04)       <recordType namespace="http://example.com/people"
                       localName="person"/>
(05)       <propertyValue namespace="http://example.com/people"
                          localName="state">
(06)         <equal>CA</equal>
(07)       </propertyValue>
(08)     </recordConstraint>
(09)   </itemTemplate>

(10)   <itemTemplate id="computer">
(11)     <recordConstraint>
(12)       <recordType namespace="http://example.com/computer"
                       localName="computer"/>
(13)     </recordConstraint>
(14)   </itemTemplate>

(15)   <relationshipTemplate id="usage" suppressFromResult="true">
(16)     <recordConstraint>
(17)       <recordType namespace="http://example.com/computer"
                       localName="uses"/>
(18)     </recordConstraint>
(19)     <sourceTemplate ref="user"/>
(20)     <targetTemplate ref="computer"/>
(21)   </relationshipTemplate>

(22) </query>
```

4.3 Normative definition

4.3.1 GraphQuery

As illustrated in the previous example, a GraphQuery request consists of a <query> element containing <itemTemplate> and <relationshipTemplate> elements. Templates (of either kind) can contain content selectors and constraints. The same constraints types are used (with the same meaning) inside <itemTemplate> and <relationshipTemplate> elements.

In addition to constraints, <relationshipTemplate> elements also contain a <sourceTemplate> and a <targetTemplate> element. These elements each point (using the xs:ID/xs:IDREF mechanism) to an <itemTemplate>.

Here is the pseudo-schema of the payload of a GraphQuery request:

```
(01) <query>
(02)   <itemTemplate id="xs:ID" suppressFromResult="xs:boolean">
(03)     (<contentSelector ...>...</contentSelector> ?
(04)     <instanceIdConstraint>...</instanceIdConstraint> ?
(05)     <recordConstraint>
(06)       <recordType ... /> *
(07)       <propertyValue ...>...</propertyValue> *
(08)     </recordConstraint> *)
(09)      |
(10)     (<xpathExpression...>...</xpathExpression> *)
(11)     xs:any
(12)   </itemTemplate> *
(13)   <relationshipTemplate id="xs:ID" suppressFromResult="xs:boolean">
(14)     (<contentSelector ...>...</contentSelector> ?
(15)     <instanceIdConstraint>...</instanceIdConstraint> ?
(16)     <recordConstraint>
(17)       <recordType>...</recordType> *
(18)       <propertyValue>...</propertyValue> *
(19)     </recordConstraint> *)
(20)      |
(21)     (<xpathExpression ...>...</xpathExpression> *)
(22)     <sourceTemplate ref="xs:IDREF" minimum="xs:int"?
          maximum="xs:int"?/>
(23)     <targetTemplate ref="xs:IDREF" minimum="xs:int"?
          maximum="xs:int"?/>
(24)     <depthLimit ... /> ?
(25)     xs:any
(26)   </relationshipTemplate> *
(27) </query>
```

The exact syntax and semantics of each constraint element (<instanceIdConstraint>, <propertyValue>, <recordType>, and <xpathExpression>) will be described in a later section. For now suffice to say that the evaluation of a constraint on an item or relationship returns a Boolean. If the value of the Boolean is “true” then the item or relationship is deemed to satisfy the defined constraint.

Templates are used to identify matching items and relationships to be returned in the graph response.

itemTemplate

An item matches an <itemTemplate> if and only if:

- the item satisfies all the constraints defined by the <itemTemplate> (in effect, there is an implicit AND joining the constraints),
- for every <relationshipTemplate> that points to the <itemTemplate> as its sourceTemplate, there is a relationship matching this <relationshipTemplate> that has the item as its source, and
- for every <relationshipTemplate> that points to the <itemTemplate> as its targetTemplate, there is a relationship matching this <relationshipTemplate> that has the item as its target.

An item can match more than one <itemTemplate> inside a given query. When this is the case, the item appears in the response once for each matching

<itemTemplate> (unless suppressed by the “suppressFromResult” attribute described above.)

relationshipTemplate

A relationship matches a <relationshipTemplate> if and only if:

- the relationship meets all the constraints in the <relationshipTemplate> (in effect, there is an implicit AND joining the constraints),
- the source item of the relationship matches the <itemTemplate> referenced as <sourceTemplate> by the <relationshipTemplate>, and
- the target item of the relationship matches the <itemTemplate> referenced as <targetTemplate> by the <relationshipTemplate>, and
- the cardinality conditions on the <sourceTemplate> and <targetTemplate> elements are satisfied, as defined by the @minimum and @maximum attributes defined below, and
- the depth, or the number of edges between source and target nodes in the graph, satisfies the <depthLimit> condition defined below.

Items cannot match a <relationshipTemplate>.

relationshipTemplate/sourceTemplate

relationshipTemplate/targetTemplate

The <sourceTemplate> and <targetTemplate> elements each refer to an <itemTemplate> element using the required @ref attribute. The value of the @ref attribute must match the value of the @id attribute of an <itemTemplate> element in the query.

Additionally, <sourceTemplate> and <targetTemplate> elements may have the following optional attributes:

@minimum – If n is the value of the @minimum attribute, there must be at least n relationships (including the current one) matching the <relationshipTemplate> that share the same source or target item.

@maximum – If n is the value of the @maximum attribute, there may be at most n relationships (including the current one) matching the <relationshipTemplate> that share the same source or target item.

relationshipTemplate/depthLimit

The <depthLimit> element is used to extend the relationship in the query graph to traverse multiple edges and nodes. For example, this may be used to find all the components of an aggregate system, or all the dependencies of a business service, even if these items are not directly related to the item in question. This extended relationship is called a "relationship chain" below.

The pseudo-schema of the <depthLimit> element is:

```
(01) <depthLimit maxIntermediateItems="xs:positiveInteger"
(02)   intermediateItemTemplate="xs:IDREF" />
```

@maxIntermediateItems – The maximum number of intermediate items in the relationship chain between source and target items. A value of 1 indicates that the <relationshipTemplate> can traverse one intermediate item between the source item and target item.

@intermediateItemTemplate – The value of the intermediateItemTemplate corresponds to the @id attribute of an <itemTemplate> element that is used as a prototype for intermediate items in the relationship chain. The value of the @intermediateItemTemplate attribute is also used to represent the intermediate items in the <nodes> element of the query response.

Content selectors and constraints are defined identically whether they are contained inside of an <itemTemplate> or a <relationshipTemplate> element. In the following sections, we use the term "instance" to mean either an item or a relationship.

4.3.1.1 ***Content selection***

The <contentSelector> element determines how instances matching the template are returned in the response. If a template does not contain a <contentSelector> element (or an <xpathExpression> used for content selection), all matching instances and associated records are returned in the response.

If a template contains a <contentSelector> element, the records and properties returned for the instances that match this template are limited to those explicitly selected. More specifically, only the records and properties that are listed (via their namespace and local name) inside the <contentSelector> element get returned.

The pseudo-schema of the <contentSelector> element is:

```
(01) <contentSelector matchedRecords="xs:boolean">
(02)   <selectedRecordType namespace="xs:anyURI" localName="xs:NCName" >
(03)     <selectedProperty namespace="xs:anyURI" localName="xs:NCName" /> *
(04)   </selectedRecordType> *
(05) </contentSelector>
```

contentSelector

The use of <contentSelector> affects the contents of the matching instances in the response as follows.

- <contentSelector /> (empty element)
  The instances matching this template are returned with no record content in the response. This may be useful if all that is required is the instanceId of instances matching this template.
- <contentSelector matchedRecords="true" />
  Only records of the selected types are returned for each matched instance. For example, if an instance has three records of the selected record type, and one record matches the template, only the matching record is returned. (This is the default behavior if matchedRecords is omitted.)
- <contentSelector matchedRecords="false" />
  All records of the selected types are returned for each matched instance. For example, if an instance has three records of the selected record type, and one record matches the template, all three records are returned.

contentSelector/selectedRecordType

If <selectedRecordType> is used without any <selectedProperty> child elements, the entire record(s) of the selected type are returned in the response.

contentSelector / selectedRecordType / selectedProperty

If <selectedProperty> elements are included in a <selectedRecordType> element, only the selected properties of the selected record types are returned in the response.

In the following example, only the “name” and “telephone” properties in the http://example.com/models/people namespace get returned for the items that match the “user” <itemTemplate>.

```
(01) <query>
(02)   <itemTemplate id="user">
(03)     <contentSelector>
(04)       <selectedRecordType namespace="http://example.com/models"
(05)                           localName="people">
(06)         <selectedProperty namespace="http://example.com/models/people"
(07)                           localName="name"/>
(08)         <selectedProperty namespace="http://example.com/models/people"
(09)                           localName="telephone"/>
(10)       </selectedRecordType>
(11)     </contentSelector>
(12)     ...
(13)   </itemTemplate>
(14) </query>
```

Note: Whether or not individual properties are selected, the contents of an item or relationship in the response will always be in the form of <record> elements as follows.

```
(01) <record>
(02)   <recordTypeQName>
(03)     <propertyQName>xs:any</propertyQName> *
(04)   </recordTypeQName>
(05)   <recordMetadata>
(06)     <recordId>xs:any</recordId>
(07)     ...
(08)   </recordMetadata>
(09) </record> *
```

*4.3.1.2 **Constraints***

Constraints are used to restrict the instances returned based on properties of the instances and associated records.

instanceIdConstraint

The <instanceIdConstraint> element is used to point to specific instances by instance Id. The pseudo-schema of this element is:

```
(01) <instanceIdConstraint>
(02)   <instanceId>cmdbf:MdrScopedIdType</instanceId> +
(03) </instanceIdConstraint>
```

There can be at most one <instanceIdConstraint> in an <itemTemplate> or a <relationshipTemplate> element.

More than one instance Id may be attached to one instance. For example, a Federating CMDB may know, for a given reconciled instance, instance Ids provided by each of the MDRs that have content about the instance, plus possibly an additional instance Id for the instance assigned by the Federating CMDB itself.

The constraint is satisfied if one of the known instance Ids for the instance matches one of the requested values, i.e. if both the <mdrId> and the <localId> match (using string comparison).

recordConstraint

The <recordConstraint> is used to point to specific record types and related properties to be evaluated.

The pseudo-schema of this element is:

```
(01) <recordConstraint>
(02)   <recordType namespace="xs:anyURI"
         localName="xs:NCName"/> *
(03)   <propertyValue> ... <propertyValue/> *
(04)   xs:any
(05) <recordConstraint/>
```

This element can appear any number of times inside an <itemTemplate> or a <relationshipTemplate>.

recordConstraint/recordType

This element can appear any number of times inside a <recordConstraint>.

One way for this constraint to be satisfied is if the instance has a record of that type. More specifically, if the instance contains a record element that has, as first child element, an element in the namespace corresponding to the value of the <recordType>/@namespace attribute and where the local name of that first child element is the value of the <recordType>/@localName attribute. The constraint could also be satisfied by an instance with a record that is an extension of that QName (for example, comp:Linux might be defined as an extension of comp:OperatingSystem).

recordConstraint/propertyValue

Each instance is associated with zero or more records. These records contain properties whose values are accessible through an XML representation of the instance. The <propertyValue> element can only be used on properties that have a type that is a subtype of the xs:anySimpleType type. While the type must be known, it is not required that an XML schema definition of the property be available.

The <propertyValue> element is not applicable to properties that are defined as a complex type.

The pseudo-schema of this element is:

```
(01) <propertyValue namespace="xs:anyURI"
(02)                        localName="xs:NCName"
(03)                        recordMetadata="xs:boolean"
(04)                        matchAny="xs:boolean">
```

```
(05)  <equal caseSensitive="xs:boolean"? negate="xs:boolean"? >
(06)    xs:anySimpleType
(07)  </equal> *
(08)  <less negate="xs:boolean"? >xs:anySimpleType</less> ?
(09)  <lessOrEqual negate="xs:boolean"? >xs:anySimpleType</lessOrEqual> ?
(10)  <greater negate="xs:boolean"? >xs:anySimpleType</greater> ?
(11)  <greaterOrEqual negate="xs:boolean"?>
(12)    xs:anySimpleType
(13)  </greaterOrEqual> ?
(14)  <contains caseSensitive="xs:boolean"? negate="xs:boolean"? >
(15)    xs:string
(16)  </contains> *
(17)  <like caseSensitive="xs:boolean"? negate="xs:boolean"? >
(18)    xs:string
(19)  </like> *
(20)  <isNull negate="xs:boolean"? /> ?
(21)  xs:any
(22) </propertyValue>
```

This element can appear any number of times in <recordConstraint>. Its namespace and localName attributes define the QName of the property being tested. If there are one or more <recordType> elements in the enclosing <recordConstraint>, they define the record types in which to evaluate the constraint. If there are no <recordType> elements the <propertyValue> is evaluated against all record types.

The recordMetadata attribute on <propertyValue> indicates that the property to be evaluated is in the <recordMetadata> element of the record.

The child elements of <propertyValue> are called operators. The matchAny attribute on <propertyValue> defines whether the operators inside that element are logically AND-ed or OR-ed. The default value is false. If the value of the matchAny attribute is false, the constraint returns a positive result for an instance if the instance has a record that contains the property identified by the QName and if the value of that property satisfies *all* the operators in the constraint. If the value of the matchAny attribute is true, the constraint returns a positive result for an instance if the instance has a record that contains the property identified by the QName and if the value of that property satisfies *at least one of* the operators in the constraint.

A <propertyValue> constraint is considered to be satisfied if the operators return a positive (true) result for one or more records associated with the instance.

The operators are largely defined in terms of XPath 2.0 [XPath 2.0] comparison operators. This does not require that an XPath 2.0 implementation be used but only that the operators be evaluated in a way that is consistent with the XPath 2.0 definitions, as described below.

recordConstraint / propertyValue / equal

This operator is defined in terms of the XPath 2.0 value comparison operator “eq”. To evaluate, the left hand operand is the property value from the record and the right hand operand is the value of the constraint from the query. The type of the value of the constraint must be interpreted to be of the same type as the value from the property in the record. This operator is valid for properties of any simple type. A list of comparison behaviors is available in the XPath 2.0 Appendix B.2 Operator Mappings.

recordConstraint/propertyValue/less

recordConstraint/propertyValue/lessOrEqual

recordConstraint/propertyValue/greater

recordConstraint/propertyValue/greaterOrEqual

These operators are defined in terms of the XPath 2.0 value comparison operators “lt”, “le”, “gt”, and “ge”, respectively. To evaluate, the left hand operand is the property value from the record and the right hand operand is the value of the constraint from the query. The type of the value of the constraint must be interpreted to be of the same type as the value from the property in the record. This operator is only valid for properties that are numerals, dates and strings. A list of comparison behaviors is available in the XPath 2.0 Appendix B.2 Operator Mappings. For example, if a property is of type date, the operator <less>2000-01-01T00:00:00</less> returns true if the property value is a date before the year 2000. If the property value was a string then “2000-01-01T00:00:00” would be interpreted as a string and compared with the property value using string comparison.

recordConstraint/propertyValue/contains

This operator is mapped to the XPath 2.0 function fn:contains(). It is only valid for properties of type string and used to test if the property value contains the specified string as a substring. The result of the contains operator is as if the fn:contains() function was executed with the first parameter being the property value and the second parameter being the string specified.

recordConstraint/propertyValue/like

This operator is similar in functionality to the SQL LIKE clause. The operator works like the equal operator with the inclusion of two special characters: the underscore (“_”) acts as a wild card for any single character and the percent sign (“%”) acts as a wild card for zero or more characters. To escape the wild cards, the backslash(“\”) can be used. For example, <like>Joe_Smith%</like> tests whether the property value starts with the string “Joe_Smith” and would match values such as “Joe_Smith”, “Joe_Smith123” and “Joe_Smith_JR”. It would not match “JoeHSmith123”. A double backslash (“\\”) represents the single backslash string (“\”).

recordConstraint/propertyValue/isNull

This operator tests whether the element corresponding to the property is "nilled". It is equivalent to the result of applying the XPath 2.0 "fn:nilled" function on the element corresponding to the property.

Additional attributes defined for operator elements:

@caseSensitive - equal, contains, and like operators have an optional attribute, caseSensitive, with a default value of true. If the property value of the record is an instance of xs:string and the attribute caseSensitive is false, the string comparison is case-insensitive. More precisely, the result of the comparison is as if the XPath 2.0 function fn:upper-case() was called on both the property value and the string value before comparison. If the property value of the record is not an instance of a xs:string, the caseSensitive attribute has no impact on the comparison.

@negate - all operators have an optional attribute, negate, with a default value of false. When the negate attribute is true, the result of the comparison is negated.

As a summary, the following table shows what operators are supported on the various XSD built-in types. Unless explicitly specified, the caseSensitive attribute is not supported.

Built-in Datatypes	equal	isNull	less, lessOrEqual, greater, greaterOrEqual	contains	like
"String-related types" (String, anyURI and types derived from string)	Yes, including optional caseSensitive attribute	Yes	Yes	Yes, including optional caseSensitive attribute	Yes, including optional caseSensitive attribute
"Time-related and numeric types" (duration, dateTime, time, date, gYearMonth, gYear, gMonthDay, gDay, gMonth, float, double, decimals and all types derived from decimals)	Yes	Yes	Yes	No	No
"Others" (boolean, QName, NOTATION, base64Binary, and hexBinary)	Yes	Yes	No	No	No

Multiple instances of a property:

If there is more than one property using the same QName, the comparison only has to hold true for one of the property values. For example, if there is a computer with three IP addresses:

```
(01) <comp:ComputerConfig xmlns:comp="http://example.com/computers">
(02)   ...
(03)   <comp:ip>1.2.3.4</comp:ip>
(04)   <comp:ip>1.2.3.5</comp:ip>
(05)   <comp:ip>1.2.3.6</comp:ip>
(06)   ...
(07) </comp:ComputerConfig>
```

The following property constraint would return a positive result:

```
(01) <recordConstraint>
(02)   <propertyValue namespace="http://example.com/computers"
(03)     localName="ip">
(04)     <equal>1.2.3.5</equal>
(05)   </propertyValue>
(06) </recordConstraint>
```

When the negate attribute is used on a list of properties, the negation is taken after the operator executes. When negating the equal operator, a positive result is returned when none of the properties are equal to the given value. For example, on the same computer with three IP addresses:

```
(01) <recordConstraint>
(02)   <propertyValue namespace="http://example.com/computers"
(03)     localName="ip">
(04)     <equal negate="true">1.2.3.5</equal>
(05)   </propertyValue>
(06) </recordConstraint>
```

The property constraint would remove the item above from the result set because the equality comparison matches one IP address in the list.

Similary, <less negate=”true”>12</less> is equivalent to <greaterOrEqual>12</greaterOrEqual> if there is only one instance of the property being tested. But if there is more than one instance of the property, then the first operator is true if all of the instances have a value of more than 12, while the second one is true if at least one of the instances has a value of more than 12.

A simple example of using <propertyValue>:

In the following example, “Manufacturer” is a property defined in the “http://example.com/Computer” namespace. The constraint is testing whether the instance has a record containing this property and where the value of the property is “HP”.

```
(01) <recordConstraint>
(02)   <propertyValue namespace="http://example.com/Computer"
(03)                         localName="Manufacturer" >
(04)     <equal>HP</equal>
(05)   </propertyValue>
(06) </recordConstraint>
```

A more complex example:

The <itemTemplate> below matches any item that has a CPUCount greater than or equal to 2, for which the OSName property contains “Linux” (with that exact mix of upper and lower case) and for which the OSName property also contains either “ubuntu” or “debian” (irrespective of case).

```
(01) <itemTemplate id=”linuxMachine”>
(02)   <recordConstraint>
(03)     <propertyValue namespace="http://example.com/computers"
(04)                           localName="CPUCount">
(05)       <greaterOrEqual>2</greaterOrEqual>
(06)     </propertyValue>
(07)     <propertyValue namespace="http://example.com/computers"
(08)                           localName="OSName">
(09)       <contains>Linux</contains>
(10)     </propertyValue>
(11)     <propertyValue namespace="http://example.com/computers"
(12)                           localName="OSName"
(13)                           matchAny="true">
(14)       <contains caseSensitive="false">ubuntu</contains>
(15)       <contains caseSensitive="false">debian</contains>
(16)     </propertyValue>
(17)   <recordConstraint/>
(18) </itemTemplate>
```

4.3.1.3 *XPath selection and constraints*

The <xpathExpression> element provides an alternate mechanism to select content and filter instances based on the content of their records. The pseudo-schema of this element is:

```
(01) <xpathExpression dialect="xs:anyURI">
(02)   <prefixMapping prefix="xs:NCName" namespace="xs:anyURI"/> *
(03)   <expression>xs:string</expression>
(04) </xpathExpression>
```

This element can appear any number of times inside an <itemTemplate> or <relationshipTemplate> element. The use of the <xpathExpression> element is mutually exclusive with the usage of the group of <contentSelector>, <instanceIdConstraint> and <recordConstraint> elements in an <itemTemplate> or <relationshipTemplate>.

xpathExpression / @dialect

The dialect corresponds to a particular version of XPath represented by the URI value. Values defined in this specification for the dialect attribute:

- "http://www.w3.org/TR/1999/REC-xpath-19991116" indicates that the expression corresponds to an XPath 1 expression.
- "http://www.w3.org/TR/2007/REC-xpath-20070123" indicates that the expression corresponds to an XPath 2 expression.

xpathExpression / prefixMapping

Each <prefixMapping> child element of the <xpathExpression> element defines a namespace declaration for the XPath evaluation. The prefix for this declaration is provided by the <prefixMapping>/@prefix attribute and the namespace URI is provided by the <prefixMapping>/@namespace attribute.

xpathExpression / expression

The <expression> element contains an XPath expression to be evaluated according to the chosen dialect/profile (the @dialect attribute).

The results of the XPath evaluation are interpreted as a constraint on the records of the item or relationship, such that an empty or "false" result from each record will cause the item or relationship to be excluded from the query response.

The XPath expression is evaluated in the following context:

- Context Node: <record> element
- Context Position: 1
- Context Size: 1
- Variable Binding: none
- Function Libraries: core function library
- Namespace Declarations: each <prefixMapping> child element of the <xpathExpression> element defines a namespace declaration for the XPath evaluation and the prefix cmdbf mapped to the namespace of this specification.

The XPath evaluation MUST also be used to select the contents of the result for this template. Since the return result will be wrapped in a <record> element, the result MUST NOT be the <record> element. Instead, it MUST be the descendent(s) of the <record> element or the result of some sort of processing such as fn:count(). An

invalid query fault will be returned if the result of processing the XPath is a <record> element.

In the following example, “name” is a property defined in the “http://example.com/people” namespace. The constraint is testing whether the instance has a record containing this property and where the value of the property is “Pete the Lab Tech”. In this example, no metadata is selected by the expression.

```
(01) <itemTemplate>
(02)   <xpathExpression
(03)       dialect="http://www.w3.org/TR/1999/REC-xpath-19991116">
(04)     <prefixMapping prefix="hr" value="http://example.com/people"/>
(05)     <expression>/cmdbf:record/hr:ContactInfo
(06)       [hr:name = "Pete the Lab Tech"]
(07)     </expression>
(08)   </xpathExpression>
(09) </itemTemplate>
```

4.3.2 GraphQuery Response

The pseudo-schema for the query response message is:

```
(01) <queryResult>
(02)     <nodes templateId="xs:ID">
(03)       <item>
(04)          <record>
(05)            xs:any
(06)            <recordMetadata>
(07)              <recordId>...</recordId>
(08)              <lastModified>...</lastModified> ?
(09)              <baselineId>...</baselineId> ?
(10)              <snapshotId>...</snapshotId> ?
(11)              xs:any
(12)            </recordMetadata>
(13)          </record> *
(14)          <instanceId>
(15)            <mdrId>xs:anyURI</mdrId>
(16)            <localId>xs:anyURI</localId>
(17)          </instanceId> +
(18)          <additionalRecordType namespace="xs:anyURI"
(19)                                localName="xs:NCName"/> *
(20)       </item> +
(21)     </nodes> *
(22)     <edges templateId="xs:ID">
(23)       <relationship>
(24)         <source>
(25)           <mdrId>xs:anyURI</mdrId>
(26)           <localId>xs:anyURI</localId>
(27)         </source>
```

```
(28)         <target>
(29)           <mdrId>xs:anyURI</mdrId>
(30)           <localId>xs:anyURI</localId>
(31)         </target>
(32)           <record>
(33)             xs:any
(34)             <recordMetadata>
(35)               <recordId>...</recordId>
(36)               <lastModified>...</lastModified> ?
(37)               <baselineId>...</baselineId> ?
(38)               <snapshotId>...</snapshotId> ?
(39)             </recordMetadata>
(40)           </record> *
(41)         <instanceId>
(42)           <mdrId>xs:anyURI</mdrId>
(43)           <localId>xs:anyURI</localId>
(44)         </instanceId> +
(45)         <additionalRecordType namespace="xs:anyURI"
(46)                               localName="xs:NCName"/> *
(47)       </relationship> +
(48)     </edges> *
(49) </queryResult>
```

Each time an item matches an <itemTemplate>, an <item> element appears inside a <nodes> element in the <queryResult>. The templateId attribute of this element contains the same value as the id attribute of the <itemTemplate> in the original request. If the item matches more than one <itemTemplate>, the <item> will be contained in the <nodes> for each <itemTemplate> matched by the item (each one with the appropriate value for its templateId attribute).

Similarly, each time a relationship matches a <relationshipTemplate>, a <relationship> element appears inside an <edges> element in the <queryResult>. The templateId attribute of this element contains the same value as the id attribute of the <relationshipTemplate> in the original request. If the relationship matches more than one <relationshipTemplate>, the <relationship> will be contained in the <edges> for each <relationshipTemplate> matched by the relationship (each one with the appropriate value for its templateId attribute).

If no item is part of the response, there are no <nodes> elements. If no relationship is part of the response, there are no <edges> elements.

Items and relationships can contain any number of records. Each is represented by a <record> element. Each record element contains two child elements. The first child is an element whose QName is a recordType supported by the Query service. The children of that child are the properties associated with the record. The second child is a <recordMetadata> element, containing information about the record itself.

Items and relationships MUST contain at least one <instanceId> element. The instance Id, through a combination of two URIs (<mdrId> to represent the MDR that assigned the ID and <localId> to uniquely represent the item or relationship inside this MDR), uniquely and globally identifies the item or relationship. There can be more than one <instanceId> element, in the case where the item or relationship has been reconciled from a more fragmented view.

The <source> child element of a relationship identifies the item that is the source of the relationship. The format of this element matches the format of the <instanceId> element on the item.

The <target> child element of a relationship identifies the item that is the target of the relationship. The format of this element matches the format of the <instanceId> element on the item.

4.3.3 GraphQuery Faults

The faults defined in this section are generated if the condition stated in the preamble is met. Faults are targeted at a destination endpoint according to the fault handling rules defined by the Web service binding.

The definitions of faults in this section use the following properties:

[Code] The fault code.

[Subcode] The fault subcode.

[Reason] The English language reason element.

[Detail] The detail element. If absent, no detail element is defined for the fault.

4.3.3.1 Unknown Template ID

This fault occurs when a relationshipTemplate includes an id referring to a sourceTemplate, targetTemplate, or intermediateItemTemplate that was not included in the query.

Properties:

[Code] Sender

[Subcode] cmdbf:UnkownTemplateID

[Reason] The graph template ID was not declared.

[Detail]

```
<cmdbf:graphId> xs:ID </cdmbf:graphId>
```

4.3.3.2 Property Type Mismatch

This fault occurs when the value in a constraint is invalid for the type of the property as defined by the schema for the property. For example, this occurs when the property is a date and the query includes a parameter to compare to the date that is a string that cannot be cast to a date, such as "foobar."

Properties:

[Code] Sender

[Subcode] cmdbf:InvalidPropertyType

[Reason] The property value being compared is not valid.

[Detail]

```
<cmdbf:propertyName namespace="xs:anyURI" localname="xs:NCName" />
```

4.3.3.3 XPath Processing Error

This fault occurs when the XPath expression processing results in an error. See http://www.w3.org/TR/xpath20/#id-errors for details on the cmdbf:xpathErrorCode.

Properties:

[Code] Sender

[Subcode] cmdbf:XPathError

[Reason] The XPath expression was not processed successfully.

[Detail]

```
<cmdbf:expression> xs:string </cmdbf:expression>
<cmdbf:xpathErrorCode> [xpath error code] </cmdbf:xpathErrorCode>
```

4.3.3.4 Unsupported Constraint

A constraint element in the template was specified that is not supported by this MDR.

Properties:

[Code] Receiver

[Subcode] cmdbf:UnsupportedConstraint

[Reason] The constraint specified is unsupported.

[Detail]

```
<cmdbf:constraint namespace="xs:anyURI" localname="xs:NCName" />
```

4.3.3.5 Unsupported Selector

A selector element in the template was specified that is not supported by this MDR.

Properties:

[Code] Receiver

[Subcode] cmdbf:UnsupportedSelector

[Reason] The selector specified is unsupported.

[Detail]

```
<cmdbf:selector namespace="xs:anyURI" localname="xs:NCName" />
```

4.3.3.6 Query Error

The query was valid, but there was an error while performing the query. When the query includes an XPath expression, this error may be used to indicate that the specific XPath dialect is not supported.

Properties:

[Code] Receiver

[Subcode] cmdbf:QueryError

[Reason] Error occurred while processing the request.

[Detail]

```
xs:any
```

4.4 GraphQuery Example

In this example, the data model contains item records of type ContactInfo and ComputerConfig and relationship records of type ‘administers’. ComputerConfigs are related to ContactInfo through the ‘administers’ relationship to allow for modeling logic such as, "UserA administers ComputerB."

This example queries the graph of the computers which are administrated by Pete the Lab Tech and returns all items and relationships involved in this graph. The response shows two computers administrated by one user.

Here are the data we assume the query is executed against.

‘User’ data:

name	Phone	employeeNumber
Lab Tech	111-111-1111	109
Joe the Manager	111-111-4567	12
Frank the CEO	111-111-9999	1

'Computer' data:

name	primaryMACAddress	CPUType	assetTag	…
LabMachineA	00A4B49D2F41	AMD Athlon 64	XYZ9753	
LabMachineB	00A4B49D2F42	AMD Athlon 64	XYZ9876	
LabMachineC	00A4B49D2H11	Intel Pentium 4	XYZ9900	
LabMachineD	00A4B49D2H53	Intel Pentium 4	XYZ9912	

'Administers' data:

'User' name	'Computer' name	adminSupportHours
Pete the Lab Tech	LabMachineA	24/7
Pete the Lab Tech	LabMachineB	business hours only
Joe the Manager	LabMachineD	24/7

Example "GraphQuery" involving a relationship traversal

```
(01) <query>
(02)   <itemTemplate id="user">
(03)     <recordConstraint>
(04)       <recordType namespace="http://example.com/people"
(05)                   localName="ContactInfo"/>
(06)       <propertyValue namespace="http://example.com/people"
(07)                   localName="name">
(08)         <equal>Pete the Lab Tech</equal>
(09)       </propertyValue>
(10)     </recordConstraint>
(11)   </itemTemplate>
(12)   <itemTemplate id="computer">
(13)     <recordConstraint>
(14)       <recordType
           namespace="http://example.com/computerModel"
(15)       localName="ComputerConfig"/>
(16)     </recordConstraint>
(17)   </itemTemplate>
(18)   <relationshipTemplate id="administers">
(19)     <recordConstraint>
(20)       <recordType
           namespace="http://example.com/computerModel"
(21)       localName="administers"/>
```

```
(22)     </recordConstraint>
(23)     <sourceTemplate ref="user"/>
(24)     <targetTemplate ref="computer"/>
(25)   </relationshipTemplate>
(26) </query>
```

Example "GraphQuery" response

```
(01)    <queryResult>
(02)        <nodes templateId="user">
(03)          <item>
(04)            <record xmlns:hr="http://example.com/people">
(05)              <hr:ContactInfo>
(06)                <hr:name>Pete the Lab Tech</hr:name>
(07)                <hr:phone>111-111-1111</hr:phone>
(08)                <hr:employeeNumber>33333</hr:employeeNumber>
(09)              </hr:ContactInfo>
(10)              <recordMetadata>
(11)                <recordId>http://example.com/33333/Current</recordId>
(12)              </recordMetadata>
(13)            </record>
(14)            <instanceId>
(15)              <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
(16)              <localId>http://example.com/PeteTheLabTech</localId>
(17)            </instanceId>
(18)          </item>
(19)        </nodes>
(20)        <nodes templateId="computer">
(21)          <item>
(22)            <record xmlns:comp="http://example.com/computerModel">
(23)              <comp:ComputerConfig>
(24)                <comp:CPUType>AMD Athlon 64</comp:CPUType>
(25)                <comp:assetTag>XYZ9753</comp:assetTag>
(26)                <comp:primaryMACAddress>
(27)                  00A4B49D2F41
(28)                </comp:primaryMACAddress>
(29)                <comp:name>LabMachineA</comp:name>
(30)                ...
(31)              </comp:ComputerConfig>
(32)              <recordMetadata>
(33)                <recordId>
(34)                  http://example.com/machines/XYZ9753/scanned
(35)                </recordId>
(36)              </recordMetadata>
(37)            </record>
(38)            <instanceId>
(39)              <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
(40)              <localId>http://example.com/machines/XYZ9753</localId>
(41)            </instanceId>
(42)          </item>
(43)          <item>
```

```
(44)            <record xmlns:comp="http://example.com/computerModel">
(45)              <comp:ComputerConfig>
(46)                <comp:CPUType>AMD Athlon 64</comp:CPUType>
(47)                <comp:assetTag>XYZ9876</comp:assetTag>
(48)                <comp:primaryMACAddress>
(49)                  00A4B49D2F42
(50)                </comp:primaryMACAddress>
(51)                <comp:name>LabMachineB</comp:name>
(52)                ...
(53)              </comp:ComputerConfig>
(54)              <recordMetadata>
(55)                <recordId>
(56)                  http://example.com/machines/XYZ9876/scanned
(57)                </recordId>
(58)              </recordMetadata>
(59)            </record>
(60)            <instanceId>
(61)              <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
(62)              <localId>http://example.com/machines/XYZ9876</localId>
(63)            </instanceId>
(64)          </item>
(65)        </nodes>
(66)        <edges templateId="administers">
(67)          <relationship>
(68)            <source>
(69)              <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
(70)              <localId>http://example.com/PeteTheLabTech</localId>
(71)            </source>
(72)            <target>
(73)              <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
(74)              <localId>http://example.com/machines/XYZ9876</localId>
(75)            </target>
(76)            <record xmlns:foo="http://example.com/computerModel">
(77)              <foo:administers>
(78)                <foo:adminSupportHours>
(79)                  business hours only
(80)                </foo:adminSupportHours>
(81)              </foo:administers>
(82)              <recordMetadata>
(83)                <recordId>adm10001</recordId>
(84)              </recordMetadata>
(85)            </record>
(86)            <instanceId>
(87)              <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
(88)              <localId>
(89)            http://example.com/administers/PeteTheLabTechToLabMachineB
(90)              </localId>
(91)            </instanceId>
(92)          </relationship>
```

```
(93)          <relationship>
(94)            <source>
(95)              <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
(96)              <localId>http://example.com/PeteTheLabTech</localId>
(97)            </source>
(98)            <target>
(99)              <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
(100)             <localId>http://example.com/machines/XYZ9753</localId>
(101)           </target>
(102)           <record xmlns:foo="http://example.com/computerModel">
(103)             <foo:administers>
(104)               <foo:adminSupportHours>24/7</foo:adminSupportHours>
(105)             </foo:administers>
(106)             <recordMetadata>
(107)               <recordId>adm10002</recordId>
(108)             </recordMetadata>
(109)           </record>
(110)           <instanceId>
(111)             <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
(112)             <localId>
(113)           http://example.com/administers/PeteTheLabTechToLabMachineA
(114)             </localId>
(115)           </instanceId>
(116)         </relationship>
(117)       </edges>
(118)   </queryResult>
```

5. Registration Service

5.1 Overview

The Registration service is used in push mode federation, as described in section 3.2.1 (Federation Modes).

The fundamentals of push mode federation are:

- The MDR invokes the Register operation for items and/or relationships that it wishes to register. Each item or relationship must be associated with at least one record type supported by the Registration service. The MDR may register a subset of the data records it has about any item or relationship.
- The Registration service responds with the registration status for each item or relationship named in the Register operation. The status is either accepted or declined.
    - If the return status is accepted, the Registration service returns the ID that identifies the item or relationship within the Registration service. For accepted data, the MDR is expected to update the Registration service whenever any of the registered data changes. The specification does not stipulate how soon after the data changes the update must occur – this would typically be determined by local policy.

    - If the return status is declined, the Registration service is presumably not maintaining the registration data, and no updates to that data are accepted.
- The specification does not stipulate what the Registration service should or must do with the registered data. The semantics of accepted and declined only have meaning with respect to the obligations of the MDR to update the Registration service when the data changes.
- The MDR also uses the Register operation to update registered data. An update may consist of any combination of:
    - Changes to existing data, such as a property value change
    - Registering an additional record type for this item or relationship
    - Deregistering a previously registered record type for this item or relationship
- The MDR uses the Deregister operation to remove an existing registration for an item or relationship. For example, if the item or relationship is deleted, the MDR would typically delete its own records and deregister the previous registration. Another example when Deregister would be used is if an administrator decides to stop federating the data about this item or relationship, even though the item or relationship still exists and the MDR still maintains data about it.
- The specification does not stipulate what the Registration service should or must do after a Deregister operation. To cite some non-prescriptive examples:
    - If it has the same data from another MDR that this MDR deregisters, it might disassociate the data with the deregistering MDR, while maintaining the existing data.
    - If it has data from another MDR about the deregistered item or relationship, it might delete the deregistered data while maintaining the data from the other MDR.
    - If it has the same data from another MDR, but it considers the deregistering MDR the authoritative source, it might mark the item or relationship as deleted.
    - If the deregistering MDR is the only source of data about the item or relationship, it might delete all knowledge of the item or relationship.

5.2 Normative definition

5.2.1 Register

The **Register operation** is used by an MDR to notify a Registration service that new items have been discovered or updated and data is now available in the MDR.

The outline for the Register operation is as follows.

```
(01) <registerRequest>
(02)   <mdrId>xs:anyURI</mdrId>
(03)   <itemList>
(04)     <item>
(05)       <record>
(06)         xs:any
(07)         <recordMetadata>...</recordMetadata>
```

```
(08)       </record> *
(09)       <instanceId>cmdbf:MdrScopedIdType</instanceId> +
(10)       <additionalRecordType namespace="xs:anyURI"
(11)                             localName="xs:NCName"/> *
(12)     </item> *
(13)   <itemList> ?
(14)   <relationshipList>
(15)     <relationship>
(16)       <source>cmdbf:MdrScopedIdType</source>
(17)       <target>cmdbf:MdrScopedIdType</target>
(18)       <record>
(19)         xs:any
(20)         <recordMetadata>...</recordMetadata>
(21)       </record> *
(22)       <instanceId>cmdbf:MdrScopedIdType</instanceId> +
(23)       <additionalRecordType namespace="xs:anyURI"
(24)                             localName="xs:NCName"/> *
(25)     </relationship> *
(26)   <relationshipList> ?
(27) </registerRequest>
```

The following describes additional constraints on the outline listed above:

mdrId

The ID of the MDR registering its data. This ID MUST be unique among all of the MDRs and Federating CMDBs that are federated together.

itemList

The list of items being registered. The list contains any number of <item> elements, though if it contains zero <item> elements, including <itemList> serves no purpose. An <item> SHOULD NOT be repeated in the list.

itemList/item

Some or all of the contents of an <item>.

itemList/item/instanceId

The <instanceId> that serves as a unique key for the <item>. There MUST be at least one for each <item>. The <instanceId> MUST contain the values that would select the <item> in a query using an <instanceIdConstraint>.

itemList/item/record

Each <item> contains any number of <record> elements.

The <record> element MUST contain exactly one child element of unrestricted type, followed by a <recordMetadata> element. The namespace and local name of the first child element together are the record type.

The <record> type MUST be supported by the registration service.

The MDR may support queries for <record> types that it chooses to not federate through the registration service.

There MAY be multiple <record> elements. The set of passed elements will be considered a complete replacement if the registration service already has data from this MDR about this <item>. For example, if the MDR had previously registered this <item> with a ComputerConfiguration and ComputerAsset record, and another registration call is made for the same item with only the

ComputerConfiguration record, then it will be treated as a deletion of the ComputerAsset record from the federation.

itemList/item/additionalRecordType

An MDR MAY support through its query interface record types for an item that are not included in the registerRequest message. If so, it MAY indicate the record types for the item by including one or more <additionalRecordType> elements. The <additionalRecordType>/@namespace and additionalRecordType/@localName attributes together represent the record type. In each <item> the same record type SHOULD NOT appear in both an <additionalRecordType> and a <record> element.

For example, the MDR may support for queries ComputerIdentification, ComputerConfiguration, and ComputerAsset records. If the registerRequest message includes only the ComputerIdentification record contents in the <record> element, the MDR may provide in <additionalRecordType> elements the localName and namespace URIs for the ComputerConfiguration and ComputerAsset records

relationshipList

The list of relationships being registered. The list contains any number of <relationship> elements, though if it contains zero <relationship> elements, including <relationshipList> serves no purpose.

relationshipList/relationship

Some or all of the contents of a <relationship>.

relationshipList/relationship/instanceId

The <instanceId> that serves as a unique key for the <relationship>. There MUST be at least one for each <relationship>. The <instanceId> MUST contain the values that would select the <relationship> in a query using an <instanceIdConstraint>.

relationshipList/relationship/source

The <instanceId> that serves as a unique key for the <item> referenced by the source side of a relationship. There MUST be exactly one for each <relationship>. The <instanceId> MUST contain one of the values that would select the source <item> in a query using an <instanceIdConstraint>.

relationshipList/relationship/target

The <instanceId> that serves as a unique key for the <item> referenced by the target side of a relationship. There MUST be exactly one for each <relationship>. The <instanceId> MUST contain one of the values that would select the target <item> in a query using an <instanceIdConstraint>.

relationshipList/relationship/record

Each <relationship> contains any number of <record> elements. The <record> type MUST be supported by the registration service.

The MDR may support queries for <record> types that it chooses to not federate through the registration service.

There MAY be multiple <record> elements. The set of passed elements will be considered a complete replacement if the registration service already has data from this MDR about this <relationship>. For example, if the MDR had previously registered this <relationship> with a RunsOn and DependsOn record, and another registration call is made for the same item with only the RunsOn record, then it will be treated as a deletion of the DependsOn record from the federation.

relationshipList/relationship/additionalRecordType

An MDR MAY support through its query interface more record types for a relationship than it federates through the registration service. If so, it MAY indicate the record types per relationship instance by including one or more <additionalRecordType> elements. The <additionalRecordType>/@namespace and <additionalRecordType/@localName attributes together represent the record type. The MDR SHOULD NOT include an <additionalRecordType> if for the same record type it includes a <record>.

5.2.2 Register Response

The outline for the response to a Register operation is as follows.

```
(01) <registerResponse>
(02)   <instanceResponse>
(03)     <instanceId>cmdbf:MdrScopedIdType</instanceId>
(04)       <accepted>
(05)         <alternateInstanceId>
(06)           cmdbf:MdrScopedIdType
(07)         </alternateInstanceId> *
(08)       </accepted> ?
(09)       <declined>
(10)         <reason>xs:string</reason> *
(11)       </declined> ?
(12)    <instanceResponse> *
(13) </registerResponse>
```

The following describes additional constraints on the outline listed above:

instanceResponse

An element that indicates the action taken for one item or relationship in the Register request. There can be any number of <instanceResponse> elements. There SHOULD be exactly one <instanceResponse> element per item or relationship in the Register request.

instanceResponse/instanceId

One of the <instanceId> elements from the Register request for an item or relationship.

instanceResponse/accepted

An element that indicates that the item or relationship instance was accepted.

Exactly one of either <accepted> or <declined> MUST be present.

instanceResponse/accepted/alternateInstanceId

Zero or more elements that contain other IDs by which the item or relationship is known, each one of which is acceptable as a key to select the item or relationship in a query.

instanceResponse/declined

An element that indicates that the item or relationship instance was declined.

Exactly one of either <accepted> or <declined> MUST be present.

instanceResponse/declined/reason

Zero or more strings that contain reason(s) why the registration was declined.

5.2.3 Register Operation Faults

The faults defined in this section are generated if the condition stated in the preamble is met. Faults are targeted at a destination endpoint according to the fault handling rules defined by the Web service binding.

The definitions of faults in this section use the following properties:

[Code] The fault code.

[Subcode] The fault subcode.

[Reason] The English language reason element.

[Detail] The detail element. If absent, no detail element is defined for the fault.

5.2.3.1 Invalid Record

The record does not correspond to the schema specifying the data model. This occurs when a required property does not exist, an extension property is used when the data model does not allow for extensions, etc.

Properties:

[Code] Sender

[Subcode] cmdbf:InvalidRecord

[Reason] The record is invalid.

[Detail]

```
<cmdbf:recordId> xs:anyURI </cmdbf:recordId>
```

5.2.3.2 Unsupported Record Type

A record of an unsupported record type was attempted to be registered.

Properties:

[Code] Sender

[Subcode] cmdbf:UnsupportedRecordType

[Reason] The record type is not supported.

[Detail]

```
<cmdbf:recordType namespace="xs:anyURI" localname="xs:NCName" />
```

5.2.3.3 Invalid MDR Id

The MDR Id specified on an item is not recognized.

Properties:

[Code] Sender

[Subcode] cmdbf:InvalidMDR

[Reason] The MDR is not registered.

[Detail]

```
<cmdbf:mdrId> xs:anyURI </cmdbf:mdrId>
```

5.2.3.4 Registration Error

There was a problem with registering the items/relationships.

Properties:

[Code] Sender

[Subcode] cmdbf:RegistrationError

[Reason] An error occurred while registering.

[Detail]

```
<cmdbf:recordId> xs:anyURI </cmdbf:recordId>
```

5.2.4 Deregister

The Deregister operation is used by an MDR to notify the Registration service that the data that an MDR has about an item or relationship will no longer be registered. Each item or relationship only needs to be deregistered once regardless of the number of <instanceId> elements provided in the register request.

The outline for the Deregister operation is as follows.

```
(01) <deregisterRequest>
(02)   <mdrId>xs:anyURI</mdrId>
(03)   <itemIdList>
(04)     <instanceId>cmdbf:MdrScopedIdType</instanceId> *
(05)   <itemIdList> ?
(06)   <relationshipIdList>
(07)     <instanceId>cmdbf:MdrScopedIdType</instanceId> *
(08)   <relationshipIdList> ?
(09) </deregisterRequest>
```

The following describes additional constraints on the outline listed above:

mdrId
The ID of the MDR deregistering its data. This ID MUST be the ID used when the data was registered using the Register request.

itemIdList
The list of items being deregistered. The list contains any number of <instanceId> elements, though if it contains zero <instanceId> elements, including <itemIdList> serves no purpose.

itemIdList/instanceId
The <instanceId> that serves as a key for the <item>. The <instanceId> MUST be either the <instanceId> from the Register request, or an <alternateInstanceId> from a <registerResponse>. An <instanceId> SHOULD NOT be repeated in the list.

relationshipIdList
The list of relationships being deregistered. The list contains any number of <instanceId> elements, though if it contains zero <instanceId> elements, including <relationshipList> serves no purpose.

relationshipIdList/instanceId
The <instanceId> that serves as a key for the <relationship>. The <instanceId> MUST be either the <instanceId> from the Register request, or an <alternateInstanceId> from a <registerResponse>. An <instanceId> SHOULD NOT be repeated in the list.

5.2.5 Deregister Response

The outline for the response to a Deregister operation is as follows.

```
(01) <deregisterResponse>
```

```
(02)   <instanceResponse>
(03)     <instanceId>cmdbf:MdrScopedIdType</instanceId>
(04)     <accepted /> ?
(05)     <declined>
(06)       <reason>xs:string</reason> *
(07)     </declined> ?
(08)   <instanceResponse> *
(09) </deregisterResponse>
```

The following describes additional constraints on the outline listed above:

instanceResponse

An element that indicates the action taken for one item or relationship in the Deregister request. There can be any number of <instanceResponse> elements. There SHOULD be exactly one <instanceResponse> element per item or relationship in the Register request.

instanceResponse / instanceId

The <instanceId> from the Deregister request for an item or relationship.

instanceResponse / accepted

An element that indicates that the item or relationship instance was accepted.

Exactly one of either <accepted> or <declined> MUST be present.

instanceResponse / declined

An element that indicates that the deregistration of the item or relationship instance was declined. An example of when a Deregister request might be declined is when the Registration service does not recognize <instanceId> in the Deregister request.

Exactly one of either <accepted> or <declined> MUST be present.

instanceResponse / declined / reason

Zero or more strings that contain reason(s) why the deregistration was declined.

5.2.6 Deregister Operation Faults

The faults defined in this section are generated if the condition stated in the preamble is met. Faults are targeted at a destination endpoint according to the fault handling rules defined by the Web service binding.

The definitions of faults in this section use the following properties:

[Code] The fault code.

[Subcode] The fault subcode.

[Reason] The English language reason element.

[Detail] The detail element. If absent, no detail element is defined for the fault.

5.2.6.1 Invalid MDR Id

The MDR Id specified on an item is not recognized.

Properties:

[Code] Sender

[Subcode] cmdbf:InvalidMDR

[Reason] The MDR is not registered.

[Detail]

```
<cmdbf:mdrId> xs:anyURI </cmdbf:mdrId>
```

5.2.6.2 Deregistration Error

There was a problem with deregistering the items/relationships.

Properties:

[Code] Sender

[Subcode] cmdbf:DeregistrationError

[Reason] An error occurred while deregistering.

[Detail]

```
<cmdbf:recordId> xs:anyURI </cmdbf:recordId>
```

6. Service Metadata

The register and query operations defined in this specification have a set of optional features that MAY be supported by a particular implementation. There are also a number of extensibility points in the specification that allow for expected variability in the implementations. One key point of variation is the data models supported for record types at a given MDR. Prior to sending register or query messages to an MDR, it may be necessary to inspect the capabilities and data models supported by that particular MDR. The schema defined in this section, includes two elements, <queryServiceMetadata> and <registerServiceMetadata> which can be used to indicate which optional features and data model(s), or record types, are supported by a particular implementation. It is RECOMMENDED that each MDR implementation include an instance of the appropriate <queryServiceMetadata> and/or <registerServiceMetadata> element(s) as part of the policies describing the implementation. An example of how these elements can be incorporated into a WS-Policy <policy> element and then associated with the implementation's WSDL binding is shown in Appendix D.

Following is a description of the service metadata schema elements <queryServiceMetadata> and <registerServiceMetadata> and their contents.

Any MDR supporting the GraphQuery operation MUST support an <itemTemplate> with <instanceIdConstraint> query at a minimum. Other query capabilities are optional. The service metadata for the MDR SHOULD indicate which optional query capabilities are supported.

queryServiceMetadata

An instance of the <queryServiceMetadata> includes the description of the MDR, including the ID of the MDR, the supported query capabilities and the supported records, or data model, for the given implementation being modeled.

The pseudo-schema of the contents of a <queryServiceMetadata> element is shown below:

```
(01) <queryServiceMetadata>
(02)   <serviceDescription>
(03)     <mdrId>xs:anyURI</mdrId>
(04)     <description>xs:string</description>?
(05)     xs:any *
(06)   </serviceDescription>
```

```
(07)  <queryCapabilities>
(08)    <relationshipTemplateSupport depthLimit="xs:boolean"
(09)       minimumMaximum="xs:boolean" xs:anyAttribute /> ?
(10)    <contentSelectorSupport recordTypeSelector="xs:boolean"
(11)       propertySelector="xs:boolean" xs:anyAttribute /> ?
(12)    <recordConstraintSupport ...> ... </recordConstraintSupport>?
(13)    <xpathSupport>
(14)      <dialect>xs:anyURI</dialect>*
(15)    </xpathSupport>
(16)    xs:any *
(17)  </queryCapabilities>
(18)  <recordTypeList>
(19)    <recordTypes namespace="xs:anyURI" schemaLocation="xs:anyURI">
(20)      <recordType localName="xs:NCName" appliesTo="xs:string">
(21)        xs:any *
(22)      </recordType>
(23)    </recordTypes> *
(24)  </recordTypeList>
(25)  xs:any *
(26) </queryServiceMetadata>
```

queryServiceMetadata/serviceDescription

The required <serviceDescription> element is used to differentiate this particular MDR from other MDRs in a particular environment. The <mdrId> is the only required element in the <serviceDescription>. The other optional elements in the <serviceDescription>, including an extensibility element, allow for further description of the query implementation.

queryServiceMetadata/serviceDescription/mdrId

The required <mdrId> is the ID of the MDR that assigned this particular implementation.

queryServiceMetadata/serviceDescription/description

The optional <description> element allows for text description of the instance to be incorporated.

queryServiceMetadata/queryCapabilities

The <queryCapabilities> describes which query techniques described in this specification are supported by this particular implementation of the query operation. The <queryCapabilities> includes an extensibility element for representing that query extensions beyond the scope of this specification are supported by the implementation.

queryServiceMetadata/queryCapabilities/relationshipTemplateSupport

When present, the <relationshipTemplateSupport> element indicates that the query operation of the implementation supports queries that include <relationshipTemplate> elements.

@depthLimit – the Boolean value of this attribute indicates if the query service implementation will process queries with a <depthLimit> element in a <relationshipTemplate>.

@minimumMaximum – the Boolean value of this attribute indicates if the query service implementation will process queries based on the cardinality of relationships as specified by a @minimum and/or @maximum attribute on a <sourceTemplate> or <targetTemplate> element of a <relationshipTemplate>.

queryServiceMetadata/queryCapabilities/contentSelectorSupport

When present, the <contentSelectorSupport> element indicates that the query operation of the implementation supports queries that include <contentSelector> elements.

@recordTypeSelector – the Boolean value of this attribute indicates if the query service implementation will process queries with <selectedRecordType> specified in the <contentSelector> of an <itemTemplate> or <relationshipTemplate>.

@propertyTypeSelector – the Boolean value of this attribute indicates if the query service implementation will process queries with <selectedProperty> specified in the <contentSelector> of an <itemTemplate> or <relationshipTemplate>.

queryServiceMetadata/queryCapabilities/recordConstraintSupport

The <recordConstraintSupport> element indicates if the query implementation will process queries that use constraints in the <itemTemplate> or <relationshipTemplate>. The complete pseudo-schema of this element is shown below:

```
(01) <recordConstraintSupport recordTypeConstraint="xs:boolean"
(02)    propertyValueConstraint="xs:boolean" xs:anyAttribute >
(03)    <propertyValuesOperators equal="xs:boolean" less="xs:boolean"
(04)       lessOrEqual="xs:boolean" greater="xs:boolean"
(05)       greaterOrEqual="xs:boolean" contains="xs:boolean"
(06)       like="xs:boolean" isNull="xs:boolean" xs:anyAttribute />?
(07) </recordConstraintSupport>
```

@recordTypeConstraint – the Boolean value of this attribute indicates if the query service implementation will process queries with <recordType> constraints in an <itemTemplate> or <relationshipTemplate>.

@propertyValueConstraint – the Boolean value of this attribute indicates if the query service implementation will process queries with <propertyValue> constraints in an <itemTemplate> or <relationshipTemplate>. When <propertyValue> constraints are supported the metadata SHOULD also indicate which operators are supported by including the <propertyValueOperators> element.

recordConstraintSupport/propertyValueOperators

The <propertyValueOperators> element is used to indicate which operators are supported by the query implementation. There is a mandatory attribute for each

operator defined by this specification and an extensibility attribute for other operators not defined by this specification.

The Boolean value of each of the following attributes indicates if the query service implementation will process queries with a property value operator of the same name as the attribute: **@equal**, **@less**, **@lessOrEqual**, **@greater**, **@greaterOrEqual**, **@contains**, **@like**, **@isNull**.

queryServiceMetadata / queryCapabilities / xpathSupport

The <xpathSupport> element is used to indicate that the query implementation supports the dialects of XPath represented by the contained <dialect> elements.

queryServiceMetadata / queryCapabilities / xpathSupport / dialect

The <dialect> elements indicate which dialects of XPath will be processed by the query implementation. The URI used as the value of the dialect should be one of the URIs listed in this specification for XPath dialects, or a URI defined by another specification that is defined to represent an XPath dialect appropriate for use in the query operation defined in this specification.

queryServiceMetadata / recordTypeList

The <recordTypeList> is used to enumerate the elements that are considered valid for use as records in the implementation of the query service. For implementations of the query operation that are federating other data stores, this list of supported record types may change over time and SHOULD be kept current by the implementation.

queryServiceMetadata / recordTypeList / recordTypes

For each different namespace that contains record types supported by the implementation, a <recordTypes> element should be included in the metadata that includes the namespace, schemaLocation if appropriate, and the list of the element names from that namespace which are supported by the implementation as <recordType> elements.

@namespace – This mandatory attribute gives the namespace of the data model that includes XML elements that correspond to record types supported by the implementation.

@schemaLocation – This optional attribute SHOULD be included when there is a URI that can be resolved to an XML schema representation of the elements belonging to the namespace listed in the namespace attribute.

queryServiceMetadata / recordTypeList / recordTypes / recordType

This repeating sequence of <recordType> elements indicates which elements are supported as record types in the implementation. These elements MUST all be from the same namespace identified in the containing <recordTypes> element.

@localName – The value of this attribute corresponds to the localName of a supported XML element that is a valid record type for the implementation.

@appliesTo – This attribute MUST be one of three values indicating whether this element is valid as a record in a relationship, item, or both. The values for this attribute are from the enumeration, “relationship”, “item” or “both”.

registerServiceMetadata

An instance of the <registerServiceMetadata> includes the description of the MDR, including the ID of the MDR, and the supported records, or data model, for the given implementation being modeled.

The pseudo-schema of the contents of a <registerServiceMetadata> element is shown below:

```
(01) <registerServiceMetadata>
(02)   <serviceDescription> ... </serviceDescription>
(03)   <recordTypeList> ... </recordTypeList>
(04)   xs:any *
(05) </registerServiceMetadata>
```

registerServiceMetadata/serviceDescription

The definition of the <serviceDescription> element is identical to the definition above in <queryServiceMetadata> except that the element is used to describe a registration service.

registerServiceMetadata /recordTypeList

The definition of the <recordTypeList> element is identical to the definition above in <queryServiceMetadata> except that the element is used to enumerate the record types supported by the registration service.

7. Secure, Reliable, Asynchronous Federation

This specification does not address a number of features that will predictably be required in an operational environment. Such features may be considered largely orthogonal to the operations defined in this specification and will affect no change to their definition. As a reference we list here some features which have been considered by the authors, but have been deemed out of scope. For the convenience of the reader references to other applicable standards are provided. These could be composed into the Web Services environment of an implementer needing or desiring the given functionality.

7.1 Security

The Federated CMDB operates in a closed environment, where some security issues are less critical than in open access or public systems. Nonetheless there are a number of security areas that should be considered when implementing this specification. Although this specification makes no mandatory statements about what security mechanisms or protocols should be used, implementors should consider the following areas and should, as far as reasonable, adhere to well known security

2089 standards in order to promote better interoperability. The following sections outline
2090 the key issues to be considered when implementing this specification.
2091

2092 **7.1.1 Authentication**

2093 Authentication of the sender of request messages, e.g. Query, Register, and in later
2094 versions of this specification posted notifications, needs to be addressed to the level
2095 required by operational authentication policy and the authorization requirements of
2096 the recipient. In many cases the requester will be a client operating under the
2097 identity of a human operator; in other cases some automatic system will be
2098 providing or requesting information. In the latter case either some form
2099 of identity delegation or federation is taking place, or the recipient requires only
2100 authorization assertions. Because of the Web Services context of this specification,
2101 recommended specifications for authentication include WS-Security 1.0, WS-Basic
2102 Security Profile 1.0, however this specification sets no requirements with respect to
2103 authentication.
2104

2105 **7.1.2 Authorization**

2106 At the level of this specification (i.e. interface definition), authorization is limited to
2107 the answering the question, "Is this requester allowed to make this request and
2108 therefore receive a reply?" That is does the requester, based on identity or role
2109 assertions carried with the message, have access to the information requested, e.g.
2110 Access Control? In the wider context of a CMDB, e.g. beyond the scope of the
2111 interface, there may be cases where the nature of the request or the type of
2112 information requested may also play a role in the authorization process. For
2113 example, the requester may have access to only some data available on the CMDB,
2114 but not other data. For this specification we recommend that implementations
2115 provide, at a minimum, authorization for read-only, write-only, read-write, or
2116 administrative access to an entity implementing one of the interfaces specified
2117 herein. Furthermore, implementations mat want to provide authorization at the data
2118 item level within the CMDB. It is expected that authorization tokens will be passed
2119 using SOAP based headers such as those specified by the SAML 2.0 specification.
2120 There are also standards available for the description of authorization policy, e.g.
2121 XACML. This specification sets no requirements with respect to authorization.
2122

2123 **7.1.3 Confidentiality**

2124 In general the confidentiality of possibly sensitive information needs to be protected
2125 from unauthorized access while held within systems and uring transmission between
2126 systems. Stored information can be protected through authorization techniques, e.g.
2127 Access Control), and the establishment of policies within the organization. With
2128 respect to protection during transmission, WS-Security with XML Encryption, SSL,
2129 and TLS all provide confidentiality "on the wire". This specification sets no
2130 requirements with respect to confidentiality.

2131 **7.1.4 Privacy**

2132 Although most data managed by a CMBF will not be of a personal nature, there are
2133 some items of personal information that may prove valuable in the CMDB context,
2134 e.g. contact information for on-call staff. This data is subject to various privacy laws,
2135 which vary across jurisdictions. There are no specific standards providing support in

this area, but privacy should be provided for by operational policies adhering to local legislation. This specification sets no requirements with respect to privacy provision.

7.1.5 Delegation and Identity Federation

Because in one of the Federated CMDB use cases the CMDB acts as a proxy for other CMDBs or MDRs, the issues of delegation and/or identity federation need to be addressed. If security is required, there are three basic approaches available to implementers. One allows, though a number of mechanisms, the proxy to impersonate the client when making a request to the underlying CMDB or MDR. This approach is not recommended by the authors of this specification, as the identity of the actual requester is lost. In the second approach, the proxy CMDB acts in its own right when contacting the underlying CMDB or MDR. This approach is considered acceptable by the authors of this specification and can be implemented by the specifications listed above. While more complex in implementation, identity federation, implemented using WS-Trust and WS-Federation for example, is also considered a valid approach to delegation and federation. This specification sets no requirements with respect to delegation and identity federation.

7.1.6 Integrity

The provision of integrity ensures that data remains complete and unchanged both in the system and in transit, due to malicious or accidental disruption. With respect to the facilities provided by this specification, it is sufficient to use mechanisms that protect the integrity of the data in transit and leave the integrity within the system as a consideration for implementors. Implementations may consider WS-Security with XML Signature for message integrity and/or SSL or TLS for some level of "on the wire" integrity. This specification sets no requirements with respect to data storage integrity.

7.1.7 Availability

In addition to network level threats to availability, e.g. network outages, there is the potential for malicious or accidental denial of service (DOS) as a result of the open content nature of some aspects of this specification. Therefore, implementor should program defensively with respect to parsing and processing XML messages. There are no specific specifications that provide support in this area. This specification sets no requirements with respect to availability.

7.1.8 Audit and Compliance

Audit and compliance issues are unlikely to arise as a result of implementing this specification, however, the CMDB itself could be used to implement audit and compliance processes for other aspects of system operation. There are no specific specifications that provide support in this area. This specification sets no requirements with respect to audit and compliance.

The reader is referred to the following standards:

- XML Signature Syntax and Processing
- XML Encryption Syntax and Processing
- WS-Security 1.0
- WS-SecureConversation 1.0
- WS-Basic Security Profile 1.0

7.2 Reliability

Reliability is the ability for a sender of a given SOAP message to know that his or her message will be delivered to the correct receiver(s) with no loss of data. This is feature is addressed by the following Web Services standards and specifications:

- WS-ReliableMessaging 1.0, 1.1
- WS-I Reliable Secure Profile (in development)

7.3 Asynchrony

An asynchronous Web Service is one in which a request is made, but a response may not be given until some later time. During this intervening time the requestor is freed to do other operations. In this sense we consider asynchronous Web Services to be of a non-blocking nature. Asynchrony is addressed in the following Web Services standards and specifications:

- WS-Addressing 1.0

8. Acknowledgements

The authors would like to acknowledge the contributions of the CMDB Federation Business Committee whose members included:

Mike Baskey, IBM
Tom Bishop, BMC Software
Josh Cohen, Microsoft
Rob Orr, IBM
Jim Saliba, CA
William Vambenepe, (formerly of) HP
John Van Son, IBM
Yoshinari Abe, Fujitsu

The authors would also like to acknowledge the CMDB Federation Use Case Working Group whose members included:

Mark Johnson, IBM
Pam Molennor, CA
Mike Oitzman, (formerly of) BMC Software
Klaus Wurster, HP

Finally, the authors would like to acknowledge people who have had some involvement in the discussion of the specification at various times during its development, including:

Dale Clark, CA
Ken Huang, BMC Software
Stefan Negritoiu, Microsoft
Tim van Ash, HP
Marshall Whatley, HP
Boris Yanishpolsky, Microsoft

9. References

[RFC 2119]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997. (See http://www.ietf.org/rfc/rfc2119.txt.)

[XPath 2.0]
"XML Path Language (XPath) 2.0", W3C Recommendation, January 2007 (See http://www.w3.org/TR/xpath20/.)

****ITIL ® is a Registered Trade Mark, and a Registered Community Trade Mark of the Office of Government Commerce, and is Registered in the U.S. Patent and Trademark Office.**

Appendix A Detailed UML Class Diagrams

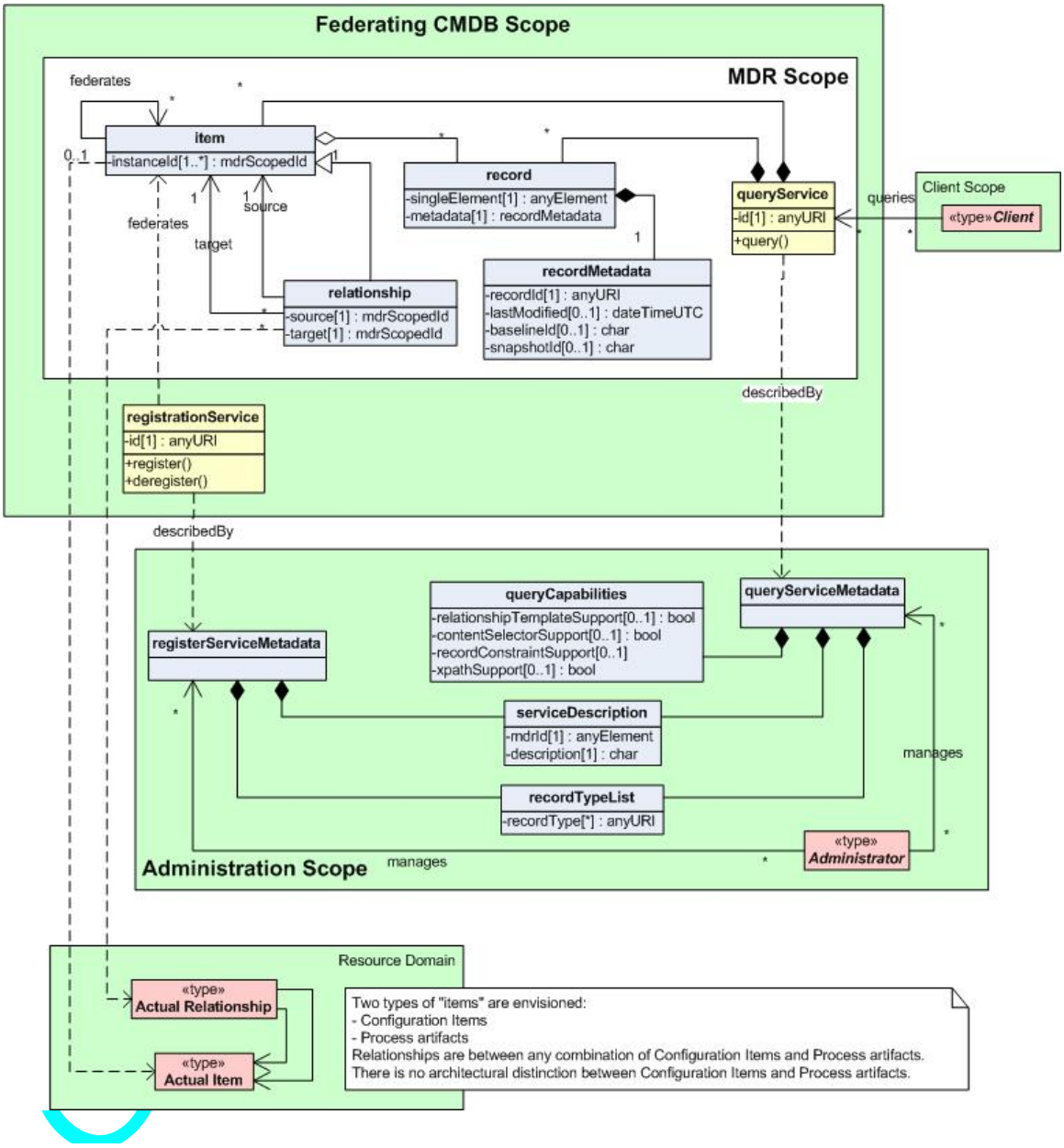


Figure 6 – Overall Class Diagrams

Appendix B XML Schema

A normative copy of the XML Schema [XML Schema Part 1, Part 2] description for this specification can be retrieved from the following addresses:

http://cmdbf.org/schema/1-0-0/cmdbfDatamodel.xsd

http://cmdbf.org/schema/1-0-0/cmdbfServiceMetadata.xsd

A non-normative copy of the XML Schema description for the data model is listed below for convenience.

CMDBf Data Model Schema

```
<?xml version="1.0" encoding="UTF-8" ?>

<!--
  Copyright Notice
  (c) 2007 by BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and
  Microsoft.   All rights reserved.

  Permission to copy and display this portion of the CMDB Federation
  specification, in any medium without fee or royalty is hereby granted,
  provided that you include the following on ALL copies of the CMDB
  Federation specification, or portions thereof, that you make:
  1. A link or URL to the Specification at the website of one of the
     authors.
  2. The copyright notice as shown in the Specification.

  BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and Microsoft
  (collectively, the "Authors") each agree to grant you a royalty-free
  license, under reasonable, non-discriminatory terms and conditions to
  their respective patents that they deem necessary to implement the
  Specification.

  THE "CMDB FEDERATION" SPECIFICATION IS PROVIDED "AS IS,"
  AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
  IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY,
  FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT
  THE CONTENTS OF THE "CMDB FEDERATION" SPECIFICATION ARE SUITABLE FOR
  ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT
  INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER
  RIGHTS.

  THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL,
  INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY
  USE OR DISTRIBUTION OF THE "CMDB FEDERATION" SPECIFICATION.

```

```
  The name and trademarks of the Authors may NOT be used in any manner,
  including advertising or publicity pertaining to the "CMDB
  FEDERATION" Specification or its contents without specific, written
  prior permission. Title to copyright in the "CMDB FEDERATION"
  Specification will at all times remain with the Authors.

  No other rights are granted by implication, estoppel or otherwise.
-->

<xs:schema  targetNamespace="http://cmdbf.org/schema/1-0-0/datamodel"
  elementFormDefault="qualified" blockDefault="#all"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:cmdbf="http://cmdbf.org/schema/1-0-0/datamodel">

  <!-- Message Global Element Declarations -->

  <xs:element name="query" type="cmdbf:QueryType" />
  <xs:element name="queryResult" type="cmdbf:QueryResultType" />
  <xs:element name="registerRequest"
    type="cmdbf:RegisterRequestType" />
  <xs:element name="registerResponse"
    type="cmdbf:RegistrationServiceResponseType" />
  <xs:element name="deregisterRequest"
    type="cmdbf:DeregisterRequestType" />
  <xs:element name="deregisterResponse"
    type="cmdbf:RegistrationServiceResponseType" />

  <!-- Query Fault Message Global Element Declarations -->

  <xs:element name="UnkownTemplateIDFault">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="graphId" type="xs:ID" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="InvalidPropertyTypeFault">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="propertyName">
          <xs:complexType>
            <xs:attribute name="localName"
              type="xs:NCName" use="required" />
            <xs:attribute name="namespace"
              type="xs:anyURI" use="required" />
```

```
2333          </xs:complexType>
2334        </xs:element>
2335      </xs:sequence>
2336    </xs:complexType>
2337  </xs:element>
2338  <xs:element name="XPathErrorFault">
2339    <xs:complexType>
2340      <xs:sequence>
2341        <xs:element name="expression" type="xs:string" />
2342        <xs:element name="xpathErrorCode" type="xs:string" />
2343      </xs:sequence>
2344    </xs:complexType>
2345  </xs:element>
2346  <xs:element name="UnsupportedConstraintFault">
2347    <xs:complexType>
2348      <xs:sequence>
2349        <xs:element name="constraint">
2350          <xs:complexType>
2351            <xs:attribute name="localName"
2352              type="xs:NCName" use="required" />
2353            <xs:attribute name="namespace"
2354              type="xs:anyURI" use="required" />
2355          </xs:complexType>
2356        </xs:element>
2357      </xs:sequence>
2358    </xs:complexType>
2359  </xs:element>
2360  <xs:element name="UnsupportedSelectorFault">
2361    <xs:complexType>
2362      <xs:sequence>
2363        <xs:element name="selector">
2364          <xs:complexType>
2365            <xs:attribute name="localName"
2366              type="xs:NCName" use="required" />
2367            <xs:attribute name="namespace"
2368              type="xs:anyURI" use="required" />
2369          </xs:complexType>
2370        </xs:element>
2371      </xs:sequence>
2372    </xs:complexType>
2373  </xs:element>
2374  <xs:element name="QueryErrorFault">
2375    <xs:complexType>
2376      <xs:sequence>
2377        <xs:any namespace="##any" processContents="lax" />
```

```
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Registration Message Global Element Declarations -->

  <xs:element name="InvalidRecordFault">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="recordId" type="xs:anyURI" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="UnsupportedRecordTypeFault">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="recordType">
          <xs:complexType>
            <xs:attribute name="localName"
              type="xs:NCName" use="required" />
            <xs:attribute name="namespace"
              type="xs:anyURI" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="InvalidMDRFault">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="mdrId" type="xs:anyURI" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="RegistrationErrorFault">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="recordId" type="xs:anyURI" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="DeregistrationErrorFault">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="recordId" type="xs:anyURI" />
```

```
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Query Request Definitions -->
  <xs:complexType name="QueryType">
    <xs:sequence>
      <xs:element name="itemTemplate"
        type="cmdbf:ItemTemplateType" minOccurs="0"
        maxOccurs="unbounded" />
      <xs:element name="relationshipTemplate"
        type="cmdbf:RelationshipTemplateType" minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ItemTemplateType">
    <xs:sequence>
      <xs:choice>
        <xs:group ref="cmdbf:Constraints" />
        <xs:element name="xpathExpression"
          type="cmdbf:XPathExpressionType" minOccurs="0"
          maxOccurs="unbounded" />
      </xs:choice>
      <xs:any minOccurs="0" maxOccurs="unbounded"
        namespace="##other" processContents="lax" />
      </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required" />
    <xs:attribute name="suppressFromResult" type="xs:boolean"
                  use="optional" default="false"/>
  </xs:complexType>

  <xs:group name="Constraints" >
      <xs:sequence>
        <xs:element name="contentSelector"
          type="cmdbf:ContentSelectorType" minOccurs="0"
          maxOccurs="1" />
        <xs:element name="instanceIdConstraint"
          type="cmdbf:InstanceIdConstraintType"
          minOccurs="0" maxOccurs="1" />
        <xs:element name="recordConstraint"
          type="cmdbf:RecordConstraintType" minOccurs="0"
          maxOccurs="unbounded" />
    </xs:sequence>
  </xs:group>
```

```
2468
2469    <xs:complexType name="RelationshipTemplateType">
2470      <xs:sequence>
2471        <xs:choice>
2472          <xs:group ref="cmdbf:Constraints" />
2473          <xs:element name="xpathExpression"
2474            type="cmdbf:XPathExpressionType" minOccurs="0"
2475            maxOccurs="unbounded" />
2476          </xs:choice>
2477        <xs:element name="sourceTemplate"
2478                    type="cmdbf:RelationshipRefType"
2479          minOccurs="0" maxOccurs="1" />
2480        <xs:element name="targetTemplate"
2481                    type="cmdbf:RelationshipRefType"
2482          minOccurs="0" maxOccurs="1" />
2483        <xs:element name="depthLimit" type="cmdbf:DepthLimitType"
2484          minOccurs="0" maxOccurs="1" />
2485        <xs:any minOccurs="0" maxOccurs="unbounded"
2486          namespace="##other" processContents="lax" />
2487      </xs:sequence>
2488      <xs:attribute name="id" type="xs:ID" use="required" />
2489      <xs:attribute name="suppressFromResult" type="xs:boolean"
2490                    use="optional" default="false"/>
2491    </xs:complexType>
2492
2493    <xs:complexType name="RelationshipRefType">
2494      <xs:attribute name="ref" type="xs:IDREF" use="required" />
2495      <xs:attribute name="minimum" type="xs:int" />
2496      <xs:attribute name="maximum" type="xs:int" />
2497    </xs:complexType>
2498
2499    <xs:complexType name="DepthLimitType">
2500      <xs:attribute name="maxIntermediateItems"
2501                    type="xs:positiveInteger" />
2502      <xs:attribute name="intermediateItemTemplate" type="xs:IDREF" />
2503    </xs:complexType>
2504
2505    <xs:complexType name="ContentSelectorType">
2506      <xs:sequence>
2507        <xs:element name="selectedRecordType"
2508              type="cmdbf:SelectedRecordTypeType" minOccurs="0"
2509              maxOccurs="unbounded" />
2510        <xs:any minOccurs="0" maxOccurs="unbounded"
2511                namespace="##other" processContents="lax" />
2512      </xs:sequence>
```

```
2513      <xs:attribute name="matchedRecords" type="xs:boolean"
2514                    use="optional" default="true"/>
2515    </xs:complexType>
2516
2517    <xs:complexType name="SelectedRecordTypeType">
2518      <xs:sequence>
2519        <xs:element name="selectedProperty" type="cmdbf:QNameType"
2520          minOccurs="0" maxOccurs="unbounded" />
2521      </xs:sequence>
2522      <xs:attribute name="namespace" type="xs:anyURI"
2523        use="required" />
2524      <xs:attribute name="localName" type="xs:NCName"
2525        use="required" />
2526    </xs:complexType>
2527
2528    <xs:complexType name="InstanceIdConstraintType">
2529      <xs:sequence>
2530        <xs:element ref="cmdbf:instanceId" maxOccurs="unbounded"
2531          minOccurs="1" />
2532      </xs:sequence>
2533    </xs:complexType>
2534
2535    <xs:complexType name="RecordConstraintType">
2536      <xs:sequence>
2537        <xs:element name="recordType"
2538          type="cmdbf:QNameType" minOccurs="0"
2539          maxOccurs="unbounded" />
2540        <xs:element name="propertyValue"
2541          type="cmdbf:PropertyValueType" minOccurs="0"
2542          maxOccurs="unbounded" />
2543        <xs:any minOccurs="0" maxOccurs="unbounded"
2544          namespace="##other" processContents="lax" />
2545      </xs:sequence>
2546    </xs:complexType>
2547
2548    <xs:complexType name="PropertyValueType">
2549      <xs:sequence>
2550        <xs:element name="equal" type="cmdbf:EqualOperatorType"
2551          minOccurs="0" maxOccurs="unbounded" />
2552        <xs:element name="less" type="cmdbf:ComparisonOperatorType"
2553          minOccurs="0" maxOccurs="1" />
2554        <xs:element name="lessOrEqual"
2555          type="cmdbf:ComparisonOperatorType" minOccurs="0"
2556          maxOccurs="1" />
2557        <xs:element name="greater"
```

```
2558          type="cmdbf:ComparisonOperatorType" minOccurs="0"
2559          maxOccurs="1" />
2560        <xs:element name="greaterOrEqual"
2561          type="cmdbf:ComparisonOperatorType" minOccurs="0"
2562          maxOccurs="1" />
2563        <xs:element name="contains" type="cmdbf:StringOperatorType"
2564          minOccurs="0" maxOccurs="unbounded" />
2565        <xs:element name="like" type="cmdbf:StringOperatorType"
2566          minOccurs="0" maxOccurs="unbounded" />
2567        <xs:element name="isNull" type="cmdbf:NullOperatorType"
2568          minOccurs="0" maxOccurs="1" />
2569        <xs:any minOccurs="0" maxOccurs="unbounded"
2570          namespace="##other" processContents="lax" />
2571      </xs:sequence>
2572      <xs:attribute name="namespace" type="xs:anyURI"
2573        use="required" />
2574      <xs:attribute name="localName" type="xs:NCName"
2575        use="required" />
2576      <xs:attribute name="recordMetadata" type="xs:boolean"
2577        use="optional" default="false" />
2578      <xs:attribute name="matchAny" type="xs:boolean"
2579        use="optional" default="false" />
2580    </xs:complexType>
2581
2582    <!-- property value operators -->
2583    <xs:complexType name="ComparisonOperatorType">
2584      <xs:simpleContent>
2585        <xs:extension base="xs:anySimpleType">
2586          <xs:attribute name="negate" type="xs:boolean"
2587            use="optional" default="false" />
2588        </xs:extension>
2589      </xs:simpleContent>
2590    </xs:complexType>
2591
2592    <xs:complexType name="StringOperatorType">
2593      <xs:simpleContent>
2594        <xs:extension base="xs:string">
2595          <xs:attribute name="caseSensitive" type="xs:boolean"
2596            use="optional" default="true" />
2597          <xs:attribute name="negate" type="xs:boolean"
2598            use="optional" default="false" />
2599        </xs:extension>
2600      </xs:simpleContent>
2601    </xs:complexType>
2602
```

```
  <xs:complexType name="EqualOperatorType">
    <xs:simpleContent>
      <xs:extension base="xs:anySimpleType">
        <xs:attribute name="caseSensitive" type="xs:boolean"
          use="optional" default="true" />
        <xs:attribute name="negate" type="xs:boolean"
          use="optional" default="false" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="NullOperatorType">
    <xs:attribute name="negate" type="xs:boolean" use="optional"
      default="false" />
  </xs:complexType>

  <xs:complexType name="XPathExpressionType">
    <xs:sequence>
      <xs:element name="prefixMapping" type="cmdbf:PrefixMappingType"
                  minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="expression" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="dialect" type="xs:anyURI" use="required" />
  </xs:complexType>

  <xs:complexType name="PrefixMappingType">
    <xs:attribute name="prefix" type="xs:NCName" use="required" />
    <xs:attribute name="namespace" type="xs:anyURI"
      use="required" />
  </xs:complexType>

  <!-- Query Response definition -->
  <xs:complexType name="QueryResultType">
    <xs:sequence>
      <xs:element name="nodes" type="cmdbf:NodesType"
        minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="edges" type="cmdbf:EdgesType"
        minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="NodesType">
    <xs:sequence>
      <xs:element ref="cmdbf:item" minOccurs="1"
        maxOccurs="unbounded" />
```

```
    </xs:sequence>
    <xs:attribute name="templateId" type="xs:ID" use="required" />
  </xs:complexType>

  <xs:complexType name="EdgesType">
    <xs:sequence>
      <xs:element ref="cmdbf:relationship" minOccurs="1"
        maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="templateId" type="xs:ID" use="required" />
  </xs:complexType>

  <!-- Registration Service -->
  <xs:complexType name="RegisterRequestType">
    <xs:sequence>
      <xs:element name="mdrId" type="xs:anyURI" />
      <xs:element name="itemList" type="cmdbf:ItemListType"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="relationshipList"
        type="cmdbf:RelationshipListType" minOccurs="0"
        maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ItemListType">
    <xs:sequence>
      <xs:element ref="cmdbf:item" minOccurs="1"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="RelationshipListType">
    <xs:sequence>
      <xs:element ref="cmdbf:relationship" minOccurs="1"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="DeregisterRequestType">
    <xs:sequence>
      <xs:element name="mdrId" type="xs:anyURI" />
      <xs:element name="itemIdList"
        type="cmdbf:MdrScopedIdListType" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="relationshipIdList"
        type="cmdbf:MdrScopedIdListType" minOccurs="0"
```

```
        maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="MdrScopedIdListType">
    <xs:sequence>
      <xs:element ref="cmdbf:instanceId" minOccurs="1"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="RegistrationServiceResponseType">
    <xs:sequence>
      <xs:element name="instanceResponse"
        type="cmdbf:InstanceResponseType" minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="InstanceResponseType">
    <xs:sequence>
      <xs:element name="instanceId" type="cmdbf:MdrScopedIdType"
        minOccurs="1" maxOccurs="1" />
      <xs:element name="accepted" type="cmdbf:AcceptedType"
        maxOccurs="1" minOccurs="0" />
      <xs:element name="declined" type="cmdbf:DeclinedType"
        maxOccurs="1" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="AcceptedType">
    <xs:sequence>
      <xs:element name="alternateInstanceId"
        type="cmdbf:MdrScopedIdType" maxOccurs="unbounded"
        minOccurs="0" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="DeclinedType">
    <xs:sequence>
      <xs:element name="reason" type="xs:string"
        maxOccurs="unbounded" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>

```

```
<!-- Shared elements definition -->
<xs:element name="item" type="cmdbf:ItemType" />
<xs:complexType name="ItemType">
  <xs:sequence>
    <xs:element ref="cmdbf:record" minOccurs="0"
      maxOccurs="unbounded" />
    <xs:element ref="cmdbf:instanceId" minOccurs="1"
      maxOccurs="unbounded" />
    <xs:element name="additionalRecordType"
      type="cmdbf:QNameType" minOccurs="0"
      maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<xs:element name="relationship" type="cmdbf:RelationshipType" />
<xs:complexType name="RelationshipType">
  <xs:sequence>
    <xs:element name="source" type="cmdbf:MdrScopedIdType"
      minOccurs="1" maxOccurs="1" />
    <xs:element name="target" type="cmdbf:MdrScopedIdType"
      minOccurs="1" maxOccurs="1" />
    <xs:element ref="cmdbf:record" minOccurs="0"
      maxOccurs="unbounded" />
    <xs:element ref="cmdbf:instanceId" minOccurs="1"
      maxOccurs="unbounded" />
    <xs:element name="additionalRecordType"
      type="cmdbf:QNameType" maxOccurs="unbounded"
      minOccurs="0" />
  </xs:sequence>
</xs:complexType>

<xs:element name="record" type="cmdbf:RecordType" />
<xs:complexType name="RecordType">
  <xs:sequence>
    <xs:any namespace="##other" processContents="skip" />
    <xs:element name="recordMetadata" >
       <xs:complexType>
         <xs:sequence>
         <xs:element name="recordId" type="xs:anyURI" />
         <xs:element name="lastModified" type="xs:dateTime"
                    minOccurs="0" />
         <xs:element name="baselineId" type="xs:string"
                    minOccurs="0" />
         <xs:element name="snapshotId" type="xs:string"
                    minOccurs="0" />
```

```
            <xs:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>

    <xs:element name="instanceId" type="cmdbf:MdrScopedIdType" />
    <xs:complexType name="MdrScopedIdType">
      <xs:sequence>
        <xs:element name="mdrId" type="xs:anyURI" minOccurs="1"
          maxOccurs="1" />
        <xs:element name="localId" type="xs:anyURI" minOccurs="1"
          maxOccurs="1" />
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="QNameType">
      <xs:attribute name="namespace" type="xs:anyURI"
        use="required" />
      <xs:attribute name="localName" type="xs:NCName"
        use="required" />
    </xs:complexType>

</xs:schema>
```

A non-normative copy of the XML Schema description for the service description meta-model is listed below for convenience.

CMDBf Service Description Schema

```
<?xml version="1.0" encoding="UTF-8" ?>

<!--
  Copyright Notice
  (c) 2007 by BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and
  Microsoft.   All rights reserved.

  Permission to copy and display this portion of the CMDB Federation
  specification, in any medium without fee or royalty is hereby granted,
  provided that you include the following on ALL copies of the CMDB
  Federation specification, or portions thereof, that you make:
  1. A link or URL to the Specification at the website of one of the
     authors.
  2. The copyright notice as shown in the Specification.

```

```
  BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and Microsoft
  (collectively, the "Authors") each agree to grant you a royalty-free
  license, under reasonable, non-discriminatory terms and conditions to
  their respective patents that they deem necessary to implement the
  Specification.

  THE "CMDB FEDERATION" SPECIFICATION IS PROVIDED "AS IS,"
  AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
  IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY,
  FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT
  THE CONTENTS OF THE "CMDB FEDERATION" SPECIFICATION ARE SUITABLE FOR
  ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT
  INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER
  RIGHTS.

  THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL,
  INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY
  USE OR DISTRIBUTION OF THE "CMDB FEDERATION" SPECIFICATION.

  The name and trademarks of the Authors may NOT be used in any manner,
  including advertising or publicity pertaining to the "CMDB
  FEDERATION" Specification or its contents without specific, written
  prior permission. Title to copyright in the "CMDB FEDERATION"
  Specification will at all times remain with the Authors.

  No other rights are granted by implication, estoppel or otherwise.
-->

<xs:schema
  targetNamespace="http://cmdbf.org/schema/1-0-0/serviceMetadata"
  elementFormDefault="qualified" blockDefault="#all"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:cmdbfmd="http://cmdbf.org/schema/1-0-0/serviceMetadata">

  <!-- Common elements -->
  <xs:element  name="serviceDescription">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="mdrId" type="xs:anyURI" />
      <xs:element name="description" type="xs:string"
         minOccurs="0" maxOccurs="1" />
      <xs:any minOccurs="0" maxOccurs="unbounded"
        namespace="##other" processContents="lax" />
    </xs:sequence>
  </xs:complexType>
```

```
  </xs:element>

  <xs:element name="recordTypeList">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="cmdbfmd:recordTypes"
           minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="recordTypes">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="cmdbfmd:recordType"
          minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="namespace" type="xs:anyURI"
        use="required" />
      <xs:attribute name="schemaLocation" type="xs:anyURI"
        use="optional" />
    </xs:complexType>
  </xs:element>

  <xs:element name="recordType">
    <xs:complexType>
      <xs:sequence>
        <xs:any minOccurs="0" maxOccurs="unbounded"
                namespace="##other" processContents="lax" />
      </xs:sequence>
      <xs:attribute name="localName" type="xs:NCName"
                    use="required" />
      <xs:attribute name="appliesTo" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="item" />
              <xs:enumeration value="relationship" />
              <xs:enumeration value="both" />
            </xs:restriction>
          </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
```

```
  <!-- Query Service metadata definition -->
  <xs:element name="queryServiceMetadata">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="cmdbfmd:serviceDescription" />
        <xs:element ref="cmdbfmd:queryCapabilities" />
        <xs:element ref="cmdbfmd:recordTypeList" />
        <xs:any minOccurs="0" maxOccurs="unbounded"
          namespace="##other" processContents="lax" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="queryCapabilities">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="relationshipTemplateSupport"
                    type="cmdbfmd:RelationshipTemplateType"
                    minOccurs="0" maxOccurs="1" />
        <xs:element name="contentSelectorSupport"
                    type="cmdbfmd:ContentSelectorType"
                    minOccurs="0" maxOccurs="1" />
        <xs:element name="recordConstraintSupport"
                    type="cmdbfmd:RecordConstraintType"
                    minOccurs="0" maxOccurs="1" />
        <xs:element name="xpathSupport"
                    type="cmdbfmd:XPathType"
                    minOccurs="0" maxOccurs="1" />
        <xs:any minOccurs="0" maxOccurs="unbounded"
                namespace="##other" processContents="lax" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="RelationshipTemplateType">
    <xs:attribute name="depthLimit" type="xs:boolean" use="required" />
    <xs:attribute name="minimumMaximum"
                  type="xs:boolean" use="required" />
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>

  <xs:complexType name="ContentSelectorType">
    <xs:attribute name="recordTypeSelector"
                  type="xs:boolean" use="required" />
    <xs:attribute name="propertySelector"
```

```
                type="xs:boolean" use="required" />
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>

  <xs:complexType name="RecordConstraintType">
    <xs:sequence>
      <xs:element name="propertyValueOperators"
                type="cmdbfmd:PropertyValueOperatorsType"
                minOccurs="0" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="recordTypeConstraint"
                type="xs:boolean" use="required" />
    <xs:attribute name="propertyValueConstraint"
                type="xs:boolean" use="required" />
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>

  <xs:complexType name="PropertyValueOperatorsType">
    <xs:attribute name="equal" type="xs:boolean" use="required" />
    <xs:attribute name="less" type="xs:boolean" use="required" />
    <xs:attribute name="lessOrEqual" type="xs:boolean" use="required"/>
    <xs:attribute name="greater" type="xs:boolean" use="required" />
    <xs:attribute name="greaterOrEqual"
                type="xs:boolean" use="required" />
    <xs:attribute name="contains" type="xs:boolean" use="required" />
    <xs:attribute name="like" type="xs:boolean" use="required" />
    <xs:attribute name="isNull" type="xs:boolean" use="required" />
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>

  <xs:complexType name="XPathType">
    <xs:sequence>
      <xs:element name="dialect" type="xs:anyURI"
                minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>


  <!-- Registration Service metadata definition -->
  <xs:element name="registrationServiceMetadata">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="cmdbfmd:serviceDescription" />
        <xs:element ref="cmdbfmd:recordTypeList" />
        <xs:any minOccurs="0" maxOccurs="unbounded"
```

```
3007            namespace="##other" processContents="lax" />
3008        </xs:sequence>
3009      </xs:complexType>
3010    </xs:element>
3011
3012  </xs:schema>
```

Appendix C WSDL

A normative copy of the WSDL [WSDL 1.1] description for this specification can be retrieved from the following addresses:

```
http://cmdbf.org/schema/1-0-0/cmdbfQuery.wsdl
http://cmdbf.org/schema/1-0-0/cmdbfRegistration.wsdl
```

A non-normative copy of the WSDL descriptions are listed below for convenience.

Query Service WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  Copyright Notice
  (c) 2007 by BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and
  Microsoft.   All rights reserved.

  Permission to copy and display this portion of the CMDB Federation
  specification, in any medium without fee or royalty is hereby granted,
  provided that you include the following on ALL copies of the CMDB
  Federation specification, or portions thereof, that you make:
  1. A link or URL to the Specification at the website of one of the
     authors.
  2. The copyright notice as shown in the Specification.

  BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and Microsoft
  (collectively, the "Authors") each agree to grant you a royalty-free
  license, under reasonable, non-discriminatory terms and conditions to
  their respective patents that they deem necessary to implement the
  Specification.

  THE "CMDB FEDERATION" SPECIFICATION IS PROVIDED "AS IS,"
  AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
  IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY,
  FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT
  THE CONTENTS OF THE "CMDB FEDERATION" SPECIFICATION ARE SUITABLE FOR
  ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT
  INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER
  RIGHTS.

  THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL,
  INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY
  USE OR DISTRIBUTION OF THE "CMDB FEDERATION" SPECIFICATION.

  The name and trademarks of the Authors may NOT be used in any manner,
```

```
3056    including advertising or publicity pertaining to the "CMDB
3057    FEDERATION" Specification or its contents without specific, written
3058    prior permission. Title to copyright in the "CMDB FEDERATION"
3059    Specification will at all times remain with the Authors.
3060
3061    No other rights are granted by implication, estoppel or otherwise.
3062  -->
3063
3064  <wsdl:definitions
3065    targetNamespace="http://cmdbf.org/schema/1-0-0/query"
3066    xmlns:tns="http://cmdbf.org/schema/1-0-0/query"
3067    xmlns:cmdbf="http://cmdbf.org/schema/1-0-0/datamodel"
3068    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
3069    xmlns:xs="http://www.w3.org/2001/XMLSchema">
3070
3071    <wsdl:types>
3072      <xs:schema
3073        targetNamespace="http://cmdbf.org/schema/1-0-0/datamodel">
3074        <xs:include
3075          schemaLocation=
3076                   "http://cmdbf.org/schema/1-0-0/cmdbfDatamodel.xsd" />
3077      </xs:schema>
3078    </wsdl:types>
3079
3080    <wsdl:message name="QueryRequest">
3081      <wsdl:part name="body" element="cmdbf:query" />
3082    </wsdl:message>
3083
3084    <wsdl:message name="QueryResponse">
3085      <wsdl:part name="body" element="cmdbf:queryResult" />
3086    </wsdl:message>
3087
3088    <wsdl:message name="UnkownTemplateIDFault">
3089      <wsdl:part name="Detail" element="cmdbf:UnkownTemplateIDFault"/>
3090    </wsdl:message>
3091    <wsdl:message name="InvalidPropertyTypeFault">
3092      <wsdl:part name="Detail" element="cmdbf:InvalidPropertyTypeFault"/>
3093    </wsdl:message>
3094
3095    <wsdl:message name="XPathErrorFault">
3096      <wsdl:part name="Detail" element="cmdbf:XPathErrorFault"/>
3097    </wsdl:message>
3098
3099    <wsdl:message name="UnsupportedConstraintFault">
3100      <wsdl:part name="Detail"
```

```
                element="cmdbf:UnsupportedConstraintFault"/>
  </wsdl:message>

  <wsdl:message name="UnsupportedSelectorFault">
    <wsdl:part name="Detail" element="cmdbf:UnsupportedSelectorFault"/>
  </wsdl:message>

  <wsdl:message name="QueryErrorFault">
    <wsdl:part name="Detail" element="cmdbf:QueryErrorFault"/>
  </wsdl:message>

  <wsdl:portType name="QueryPortType">
    <wsdl:operation name="GraphQuery">
      <wsdl:input message="tns:QueryRequest" />
      <wsdl:output message="tns:QueryResponse" />
      <wsdl:fault name="UnkownTemplateID"
                  message="tns:UnkownTemplateIDFault"/>
      <wsdl:fault name="InvalidPropertyType"
                  message="tns:InvalidPropertyTypeFault"/>
      <wsdl:fault name="XPathError"
                  message="tns:XPathErrorFault"/>
      <wsdl:fault name="UnsupportedConstraint"
                  message="tns:UnsupportedConstraintFault"/>
      <wsdl:fault name="UnsupportedSelector"
                  message="tns:UnsupportedSelectorFault"/>
      <wsdl:fault name="QueryError"
                  message="tns:QueryErrorFault"/>
    </wsdl:operation>
  </wsdl:portType>

</wsdl:definitions>
```

Registration Service WSDL

```
<?xml version='1.0' encoding='UTF-8' ?>
<!--
  Copyright Notice
  (c) 2007 by BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and
  Microsoft.   All rights reserved.

  Permission to copy and display this portion of the CMDB Federation
  specification, in any medium without fee or royalty is hereby granted,
  provided that you include the following on ALL copies of the CMDB
  Federation specification, or portions thereof, that you make:
  1. A link or URL to the Specification at the website of one of the
```

```
3145      authors.
3146    2. The copyright notice as shown in the Specification.
3147
3148    BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and Microsoft
3149    (collectively, the "Authors") each agree to grant you a royalty-free
3150    license, under reasonable, non-discriminatory terms and conditions to
3151    their respective patents that they deem necessary to implement the
3152    Specification.
3153
3154    THE "CMDB FEDERATION" SPECIFICATION IS PROVIDED "AS IS,"
3155    AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
3156    IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY,
3157    FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT
3158    THE CONTENTS OF THE "CMDB FEDERATION" SPECIFICATION ARE SUITABLE FOR
3159    ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT
3160    INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER
3161    RIGHTS.
3162
3163    THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL,
3164    INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY
3165    USE OR DISTRIBUTION OF THE "CMDB FEDERATION" SPECIFICATION.
3166
3167    The name and trademarks of the Authors may NOT be used in any manner,
3168    including advertising or publicity pertaining to the "CMDB
3169    FEDERATION" Specification or its contents without specific, written
3170    prior permission. Title to copyright in the "CMDB FEDERATION"
3171    Specification will at all times remain with the Authors.
3172
3173    No other rights are granted by implication, estoppel or otherwise.
3174  -->
3175
3176  <wsdl:definitions
3177    targetNamespace="http://cmdbf.org/schema/1-0-0/registration"
3178    xmlns:tns="http://cmdbf.org/schema/1-0-0/registration"
3179    xmlns:cmdbf="http://cmdbf.org/schema/1-0-0/datamodel"
3180    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
3181    xmlns:xs="http://www.w3.org/2001/XMLSchema">
3182
3183    <wsdl:types>
3184      <xs:schema
3185        targetNamespace="http://cmdbf.org/schema/1-0-0/datamodel">
3186        <xs:include
3187          schemaLocation=
3188                   "http://cmdbf.org/schema/1-0-0/cmdbfDatamodel.xsd" />
3189      </xs:schema>
```

```
  </wsdl:types>

  <wsdl:message name="RegisterRequest">
    <wsdl:part name="body" element="cmdbf:registerRequest" />
  </wsdl:message>

  <wsdl:message name="RegisterResponse">
    <wsdl:part name="body" element="cmdbf:registerResponse" />
  </wsdl:message>

  <wsdl:message name="DeregisterRequest">
    <wsdl:part name="body" element="cmdbf:deregisterRequest" />
  </wsdl:message>

  <wsdl:message name="DeregisterResponse">
    <wsdl:part name="body" element="cmdbf:deregisterResponse" />
  </wsdl:message>

  <wsdl:message name="InvalidRecordFault">
    <wsdl:part name="Detail" element="cmdbf:InvalidRecordFault"/>
  </wsdl:message>

  <wsdl:message name="UnsupportedRecordTypeFault">
    <wsdl:part name="Detail"
               element="cmdbf:UnsupportedRecordTypeFault"/>
  </wsdl:message>

  <wsdl:message name="InvalidMDRFault">
    <wsdl:part name="Detail" element="cmdbf:InvalidMDRFault"/>
  </wsdl:message>

  <wsdl:message name="RegistrationErrorFault">
    <wsdl:part name="Detail" element="cmdbf:RegistrationErrorFault"/>
  </wsdl:message>

  <wsdl:message name="DeregistrationErrorFault">
    <wsdl:part name="Detail" element="cmdbf:DeregistrationErrorFault"/>
  </wsdl:message>

  <wsdl:portType name="RegistrationPortType">
    <wsdl:operation name="Register">
      <wsdl:input message="tns:RegisterRequest" />
      <wsdl:output message="tns:RegisterResponse" />
      <wsdl:fault name="InvalidRecordFault"
                  message="tns:InvalidRecordFault"/>
```

```
      <wsdl:fault name="UnsupportedRecordTypeFault"
                  message="tns:UnsupportedRecordTypeFault"/>
      <wsdl:fault name="InvalidMDRFault"
                  message="tns:InvalidMDRFault"/>
      <wsdl:fault name="RegistrationErrorFault"
                  message="tns:RegistrationErrorFault"/>
    </wsdl:operation>

    <wsdl:operation name="Deregister">
      <wsdl:input message="tns:DeregisterRequest" />
      <wsdl:output message="tns:DeregisterResponse" />
      <wsdl:fault name="InvalidMDRFault"
                  message="tns:InvalidMDRFault"/>
      <wsdl:fault name="DeregistrationErrorFault"
                  message="tns:DeregistrationErrorFault"/>
    </wsdl:operation>
  </wsdl:portType>

</wsdl:definitions>
```

Appendix D Fault Binding to SOAP

Faults are may be generated for any CMDBf operation. The bindings of faults for both SOAP 1.1 and SOAP 1.2 are described in this section.

The definitions of faults use the following properties:

[Code] The fault code.

[Subcode] The fault subcode.

[Reason] A language localized readable description of the error.

[Detail] Optional detail element(s). If more than one detail element is defined for a fault, implementations MUST include the elements in the order that they are specified.

Services that generate CMDBf faults MUST set the [Code] property to either "Sender" or "Receiver". These properties are serialized into text XML as follows:

SOAP Version	Sender	Receiver
SOAP 1.1	S11:Client	S11:Server
SOAP 1.2	S:Sender	S:Receiver

The properties above bind to a SOAP 1.2 fault as follows:

```
<S:Envelope>
 <S:Header>
   <wsa:Action>
     http://cmdbf.org/schema/1-0-0/fault
   </wsa:Action>
   <!-- Headers elided for brevity.  -->
 </S:Header>
 <S:Body>
  <S:Fault>
   <S:Code>
     <S:Value> [Code] </S:Value>
     <S:Subcode>
      <S:Value> [Subcode] </S:Value>
     </S:Subcode>
   </S:Code>
   <S:Reason>
     <S:Text xml:lang="en"> [Reason] </S:Text>
   </S:Reason>
   <S:Detail>
     [Detail]
     ...
   </S:Detail>
  </S:Fault>
 </S:Body>
</S:Envelope>
```

The properties bind to a SOAP 1.1 fault as follows when the fault is generated as a result of processing a CMDBf request message:

```
<S11:Envelope>
 <S11:Header>
   <cmdbf:fault>
     <cmdbf:faultCode> [Subcode] </cmdbf:faultCode>
     <cmdbf:detail> [Detail] </cmdbf:detail>
     ...
   </cmdbf:fault>
   <!-- Headers elided for brevity.  -->
 </S11:Header>
 <S11:Body>
  <S11:Fault>
   <S11:faultcode> [Code] </S11:faultcode>
   <S11:faultstring> [Reason] </S11:faultstring>
  </S11:Fault>
 </S11:Body>
</S11:Envelope>
```

When a binding to a CMDBf operation that supports WS-Addressing, the fault message MUST include the following action URI defined below as the [action] property.

`http://cmdbf.org/schema/1-0-0/fault`

Fault handling rules for operations using WS-Addressing are defined in section 6 of WS-Addressing SOAP Binding.

Appendix E Sample WSDL Binding (non-normative)

The following example illustrates how the interfaces defined in this specification should be described in a Web service binding that implements the interfaces. This example also illustrates how the CMDBf service metadata should be associated with a particular implementation of a CMDBf interface.

As shown below, this query implementation uses SOAP 1.1 over HTTP as the protocol and supports the use of WS-Addressing if the message sender uses WS-Addressing for an asynchronous request/response. Since this specification does not define specific WS-Addressing actions, the action header values for WS-Addressing are determined according to the defaults described in the WS-Addressing 1.0 WSDL Binding specification at http://www.w3.org/TR/ws-addr-wsdl/#defactionwsdl11

The queryServiceMetadata element is included in a WS-Policy expression which is included by reference in the WSDL binding to the query port type. This particular sample is of a query service that supports the complete set of record constraint and selector operators defined in the specification. The metadata in the sample also shows that XPath1 and XPath 2 are supported by the service.

The metadata for the service also includes the two record types that may be queried at this service, a “R_ComputerSystem” data type, and a “CIM_CommonDatabase” data type.

The approach to including metadata as a policy in the WSDL is a recommended approach to creating the WSDL documentation for the binding implementation as it allows for the file containing the WSDL binding to completely describe the interface to the service and the options allowed by this specification.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!--
  Copyright Notice
  (c) 2007 by BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and
  Microsoft.   All rights reserved.

  Permission to copy and display this portion of the CMDB Federation
  specification, in any medium without fee or royalty is hereby granted,
  provided that you include the following on ALL copies of the CMDB
  Federation specification, or portions thereof, that you make:
  1. A link or URL to the Specification at the website of one of the
     authors.
  2. The copyright notice as shown in the Specification.

  BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and Microsoft
  (collectively, the "Authors") each agree to grant you a royalty-free
  license, under reasonable, non-discriminatory terms and conditions to
  their respective patents that they deem necessary to implement the
  Specification.

  THE "CMDB FEDERATION" SPECIFICATION IS PROVIDED "AS IS,"
  AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
```

```
3367    IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY,
3368    FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT
3369    THE CONTENTS OF THE "CMDB FEDERATION" SPECIFICATION ARE SUITABLE FOR
3370    ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT
3371    INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER
3372    RIGHTS.
3373
3374    THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL,
3375    INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY
3376    USE OR DISTRIBUTION OF THE "CMDB FEDERATION" SPECIFICATION.
3377
3378    The name and trademarks of the Authors may NOT be used in any manner,
3379    including advertising or publicity pertaining to the "CMDB
3380    FEDERATION" Specification or its contents without specific, written
3381    prior permission. Title to copyright in the "CMDB FEDERATION"
3382    Specification will at all times remain with the Authors.
3383
3384    No other rights are granted by implication, estoppel or otherwise.
3385  -->
3386
3387  <wsdl:definitions
3388    targetNamespace="http://cmdbf.org/schema/1-0-0/binding"
3389    xmlns:cmdbfPort="http://cmdbf.org/schema/1-0-0/query"
3390    xmlns:cmdbfBind="http://cmdbf.org/schema/1-0-0/binding"
3391    xmlns:cmdbfMetadata="http://cmdbf.org/schema/1-0-0/serviceMetadata"
3392    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
3393    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
3394    xmlns:wsp="http://www.w3.org/ns/ws-policy"
3395    xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
3396    xmlns:xs="http://www.w3.org/2001/XMLSchema">
3397
3398    <wsdl:import location="cmdbfQuery.wsdl"
3399      namespace="http://cmdbf.org/schema/1-0-0/query">
3400    </wsdl:import>
3401
3402    <!-- Subject supports WS-Addressing -->
3403    <wsp:Policy xml:Id="SupportsWSAddressing">
3404      <wsam:Addressing wsp:Optional="true">
3405        <wsp:Policy />
3406      </wsam:Addressing>
3407    </wsp:Policy>
3408
3409
3410    <!-- Subject supports the referenced data model in the operations -->
3411    <wsp:Policy xml:Id="SupportedMetadata">
```

```
    <queryServiceMetadata
              xmlns="http://cmdbf.org/schema/1-0-0/serviceMetadata">
      <serviceDescription>
        <mdrId>CMDBf12345</mdrId>
      </serviceDescription>
      <queryCapabilities>
        <contentSelectorSupport propertySelector="true"
                                recordTypeSelector="true" />
        <recordConstraintSupport recordTypeConstraint="true"
                                 propertyValueConstraint="true">
          <propertyValueOperators equal="true" less="true"
                                  greater="true" lessOrEqual="true"
                                  greaterOrEqual="true"
                                  contains="true"
                                  like="false"
                                  isNull="false" />
        </recordConstraintSupport>
        <xpathSupport>
          <dialect>
            http://www.w3.org/TR/1999/REC-xpath-19991116
          </dialect>
          <dialect>
            http://www.w3.org/TR/2007/REC-xpath-20070123
          </dialect>
        </xpathSupport>
      </queryCapabilities>

      <recordTypeList>
         <recordTypes namespace="http://cmdbf.org"
                schemaLocation="http://cmdbf.org/common_schemas/R_ComputerSystem.xsd">
           <recordType localName="R_ComputerSystem" />
         </recordTypes>
         <recordTypes
    namespace="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_CommonDatabase"
    schemaLocation="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_CommonDatabase.xsd">
           <recordType localName="CIM_CommonDatabase" />
         </recordTypes>
      </recordTypeList>

    </queryServiceMetadata>
  </wsp:Policy>

  <!-- Sample Binding for SOAP 1.1 with WS-Addressing support
  -->
  <wsdl:binding name="QueryBinding" type="cmdbfPort:QueryPortType">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http" />
```

```
    <wsp:PolicyReference URI="SupportsWSAddressing" />
    <wsp:PolicyReference URI="SupportedMetadata" />
    <wsdl:operation name="GraphQuery">
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
      <wsdl:fault name="UnkownTemplateID">
        <soap:fault name="UnkownTemplateID" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="InvalidPropertyType">
        <soap:fault name="InvalidPropertyType" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="XPathError">
        <soap:fault name="XPathError" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="UnsupportedConstraint">
        <soap:fault name="UnsupportedConstraint" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="UnsupportedSelector">
        <soap:fault name="UnsupportedSelector" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="QueryError">
        <soap:fault name="QueryError" use="literal" />
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

</wsdl:definitions>
```

Appendix F Editorial Corrections

Following is a list of changes that are editorial in nature, such as typographical errors or other obvious errors.

Version & Date	Editor	List of Changes
Version 1.0b 04 January 2008	Mark Johnson	Line 3: Updated the version suffix and date Lines 895-897: Italicized to indicate that these are *types* of elements Line 2047-2048: Replaced <queryServiceMetadata> with <registerServiceMetadata> and deleted an extraneous reference to query capabilities Line 2250: Corrected the spelling of cmdbfServiceMetadata.xsd Lines 2445,2475: Corrected the schema to indicate that the xpathExpression element may appear an unbounded number of times. This change is also reflected in the associated xsd file. Lines 2500-2502: Corrected the schema to make maxIntermediateItems and intermediateItemTemplate attributes rather than elements. This change is also reflected in the associated xsd file. Lines 2576-2577: Added the missing recordMetadata attribute. This change is also reflected in the associated xsd file. Line 2776: Removed an extraneous blank space before the trailing double quote. This change is also reflected in the associated xsd file. Lines 3492+: Added Appendix F Editorial Corrections