

1 **CMDB Federation (CMDBf)**

2 Committee Draft

3 **Version 1.0b, 04 January 2008**

4 **Authors**

5 Forest Carlisle, CA

6 Barry Day, Microsoft

7 Scott Donohoo, IBM

8 Pratul Dubish, Microsoft

9 Jacob Eisinger, IBM

10 Greg Goodman, CA

11 Andrew Hatley, IBM

12 Mark Johnson, IBM (Editor)

13 Vincent Kowalski, BMC Software (Editor)

14 Yannis Labrou, Fujitsu

15 Kenji Morimoto, Fujitsu

16 David Snelling, Fujitsu

17 William Vambenepe, (formerly of) HP (Editor)

18 Marv Waschke, CA (Editor)

19 Van Wiles, BMC Software (Editor)

20 Klaus Wurster, HP

22 **Copyright Notice**

23 © Copyright 2007 by BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and Microsoft. **All**
24 **Rights Reserved.**

25
26 Permission to copy and display the CMDB Federation Version 1.0 specification, in any medium
27 without fee or royalty is hereby granted, provided that you include the following on ALL copies of
28 the CMDB Federation Version 1.0 specification, or portions thereof, that you make:

- 29 1. A link or URL to the Specification at one of the Authors' websites.
- 30 2. The copyright notice as shown in the Specification.

31
32 BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and Microsoft (collectively, the "Authors")
33 each agree to grant you a royalty-free license, under reasonable, non-discriminatory terms and
34 conditions to their respective patents that they deem necessary to implement the Specification.

35
36 THE SPECIFICATION IS PROVIDED "AS IS," AND THE AUTHORS MAKE NO
37 REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT
38 LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
39 PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE
40 SPECIFICATION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION
41 OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS,
42 TRADEMARKS OR OTHER RIGHTS.

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

47

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

52

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

54 **Abstract**

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

60 **Status**

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

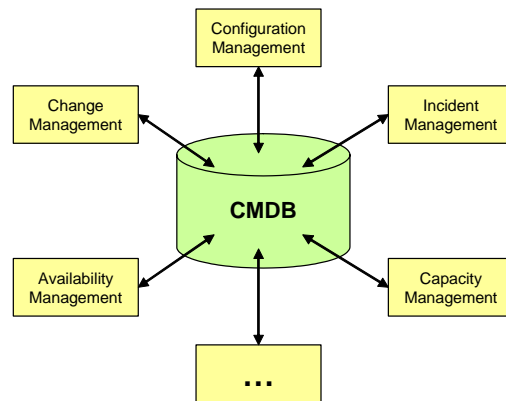
63 At some future date, the contents may be published under another name or under
64 several new specifications, as shall be agreed by the authors and their respective
65 corporations at that time.

66	Table of Contents	
67	CMDB Federation (CMDBf)	1
68	Authors.....	1
69	Copyright Notice.....	1
70	Abstract.....	2
71	Status.....	2
72	Table of Contents.....	3
73	1. Introduction	5
74	1.1 Objectives.....	6
75	1.1.1 Functions.....	6
76	1.1.2 Target IT Environment.....	7
77	1.1.3 Non-Goals.....	7
78	1.2 Background Terminology.....	7
79	1.3 Notational Conventions.....	9
80	2. Technological Assumptions	9
81	2.1 Underlying Technology.....	9
82	2.1.1 Web Services.....	9
83	2.1.2 Database Management Systems.....	10
84	2.2 Standards Basis.....	10
85	3. Architecture	10
86	3.1 Roles.....	11
87	3.2 Services Overview.....	11
88	3.2.1 Federation Modes.....	11
89	3.2.2 Usage Profiles.....	12
90	3.3 Identity Reconciliation.....	12
91	3.4 Data Model Overview.....	13
92	3.4.1 Managed Data.....	13
93	3.4.2 Common data element types.....	15
94	4. Query Service	16
95	4.1 Overview.....	16
96	4.2 Example.....	17
97	4.3 Normative definition.....	19
98	4.3.1 GraphQL.....	19
99	4.3.2 GraphQL Response.....	30
100	4.3.3 GraphQL Faults.....	32
101	4.4 GraphQL Example.....	33
102	5. Registration Service	37
103	5.1 Overview.....	37
104	5.2 Normative definition.....	38
105	5.2.1 Register.....	38
106	5.2.2 Register Response.....	41
107	5.2.3 Register Operation Faults.....	42

108	5.2.4 Deregister	43
109	5.2.5 Deregister Response	43
110	5.2.6 Deregister Operation Faults	44
111	6. Service Metadata	45
112	7. Secure, Reliable, Asynchronous Federation	49
113	7.1 Security	49
114	7.1.1 Authentication	50
115	7.1.2 Authorization	50
116	7.1.3 Confidentiality	50
117	7.1.4 Privacy	50
118	7.1.5 Delegation and Identity Federation	51
119	7.1.6 Integrity	51
120	7.1.7 Availability	51
121	7.1.8 Audit and Compliance	51
122	7.2 Reliability	52
123	7.3 Asynchrony	52
124	8. Acknowledgements	52
125	9. References	53
126	Appendix A Detailed UML Class Diagrams	54
127	Appendix B XML Schema	55
128	CMDBf Data Model Schema	55
129	CMDBf Service Description Schema	67
130	Appendix C WSDL	73
131	Query Service WSDL	73
132	Registration Service WSDL	75
133	Appendix D Fault Binding to SOAP	79
134	Appendix E Sample WSDL Binding (non-normative)	81
135		

136 **1. Introduction**

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



144
145

Figure 1 – Role of a CMDB

146

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

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

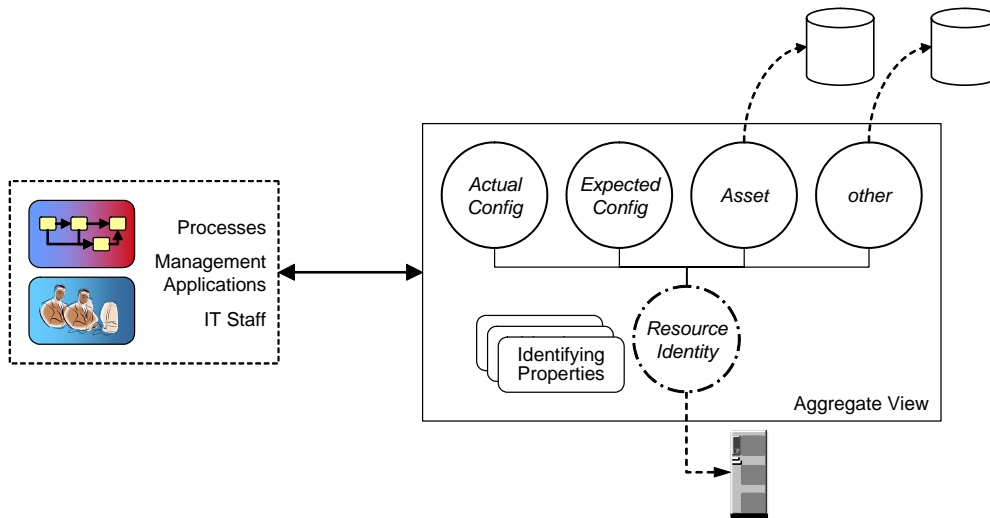
160 The federated CMDB described in this specification is a collection of services and data
161 repositories that contain configuration and other data records about resources. The
162 term 'resource' includes configuration items (e.g., a computer system, an application,
163 or a router), process artifacts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

167

168 **1.1 Objectives**

169 **1.1.1 Functions**

170 The federated CMDB resulting from using this specification will provide a single
171 aggregate view of the data about an IT resource, even if the data is from different
172 heterogeneous data repositories, as shown in Figure 2. Clients, such as IT processes,
173 management applications, and IT staff will use a query service defined in the
174 specification to access aggregated or non-aggregated views. Data repositories will
175 use the services described in the specification to provide the aggregated view.
176



177
178 **Figure 2 – Aggregate View from Federated Data**
179

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

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

197 **1.1.2 Target IT Environment**

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

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

215

216 **1.1.3 Non-Goals**

217 The following are outside the scope of the specification.

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

233

234 **1.2 Background Terminology**

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

242

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

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

252 **Configuration Management Database (CMDB)** ITIL defines a CMDB as

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

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

262 **Configuration Record** ITIL defines a Configuration Record as

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

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

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

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

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

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

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

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

298

299 **1.3 Notational Conventions**

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

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

- 304 • The syntax appears as an XML instance, but values in italics indicate data
305 types instead of literal values.
- 306 • Characters are appended to elements and attributes to indicate cardinality:
 - 307 ○ "?" (0 or 1)
 - 308 ○ "*" (0 or more)
 - 309 ○ "+" (1 or more)
 - 310 ○ The absence of any of the above indicates the default (exactly 1)
- 311 • The character "|" is used to indicate a choice between alternatives.
- 312 • The characters "(" and ")" are used to indicate that contained items are to be
313 treated as a group with respect to cardinality or choice.
- 314 • The characters "[" and "]" are used to call out references and property names.
- 315 • xs:any and xs:anyAttribute indicate points of extensibility. Additional children
316 and/or attributes MAY be added at the indicated extension points but MUST
317 NOT contradict the semantics of the parent and/or owner, respectively. By
318 default, if a receiver does not recognize an extension, the receiver SHOULD
319 ignore the extension; exceptions to this processing rule, if any, are clearly
320 indicated below.
- 321 • Ellipses (i.e., "...") indicate that details are omitted for simplicity, and a
322 further explanation is provided below.
- 323 • XML namespace prefixes are used to indicate the namespace of the element
324 being defined or referenced.

325

326 **2. Technological Assumptions**

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

330 **2.1 Underlying Technology**

331 **2.1.1 Web Services**

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

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

337 2.1.2 Database Management Systems

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

345 2.2 Standards Basis

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

348

349

350

351

352

353

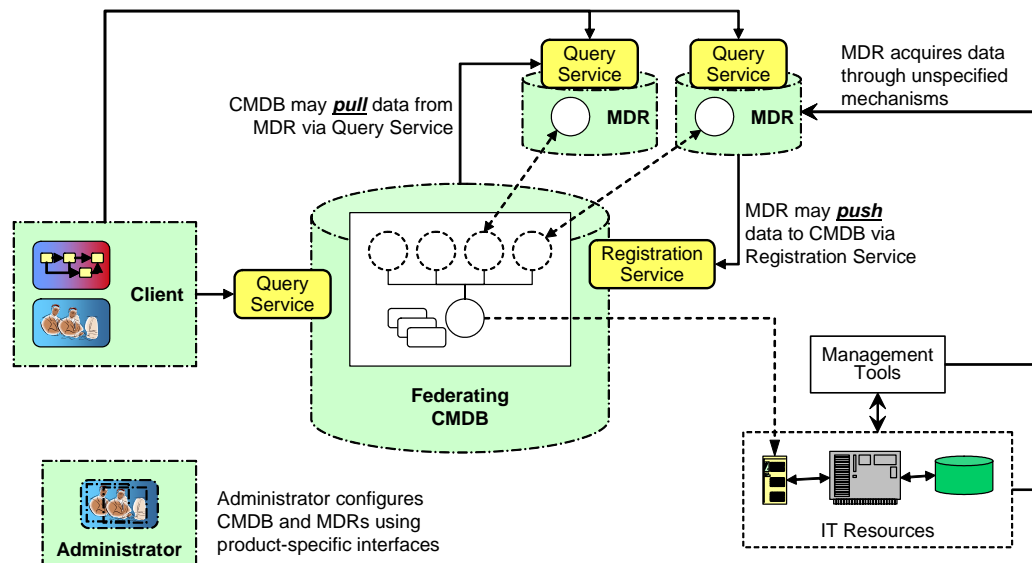
354

355

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

356 3. Architecture

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



360

361

Figure 3 – Roles and Services

362 3.1 Roles

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

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

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

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

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

388

389 3.2 Services Overview

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

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

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

399 3.2.1 Federation Modes

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

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

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

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

414 3.2.2 Usage Profiles

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

417

418 **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

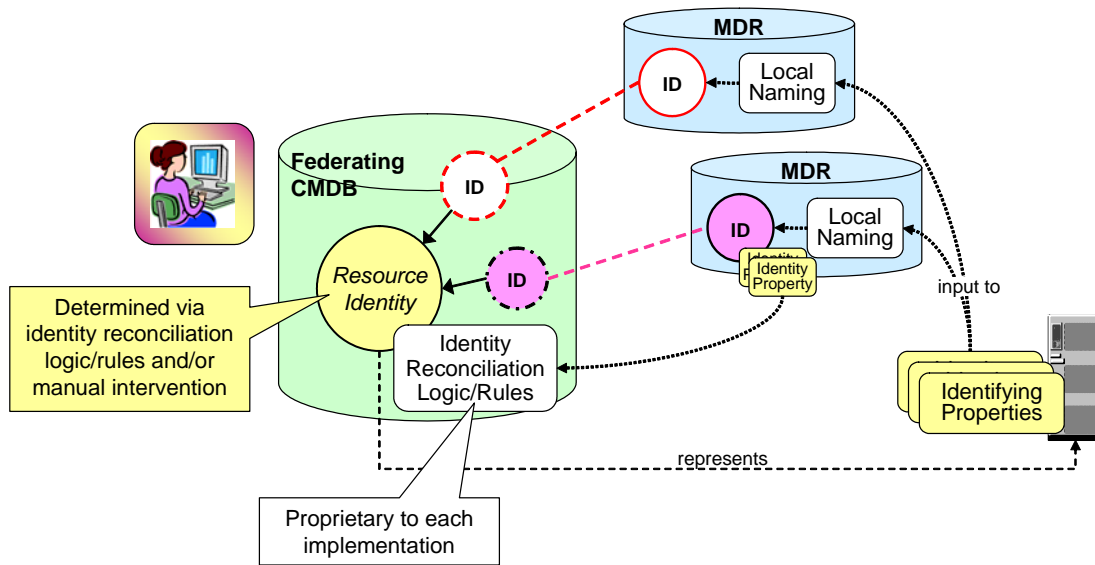
419

420

421 3.3 Identity Reconciliation

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

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



441
442

Figure 4 – Identity Reconciliation

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

453

454 3.4 Data Model Overview

455 3.4.1 Managed Data

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

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

462
463
464
465
466
467

- 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.

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

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

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

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

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

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

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

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

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

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

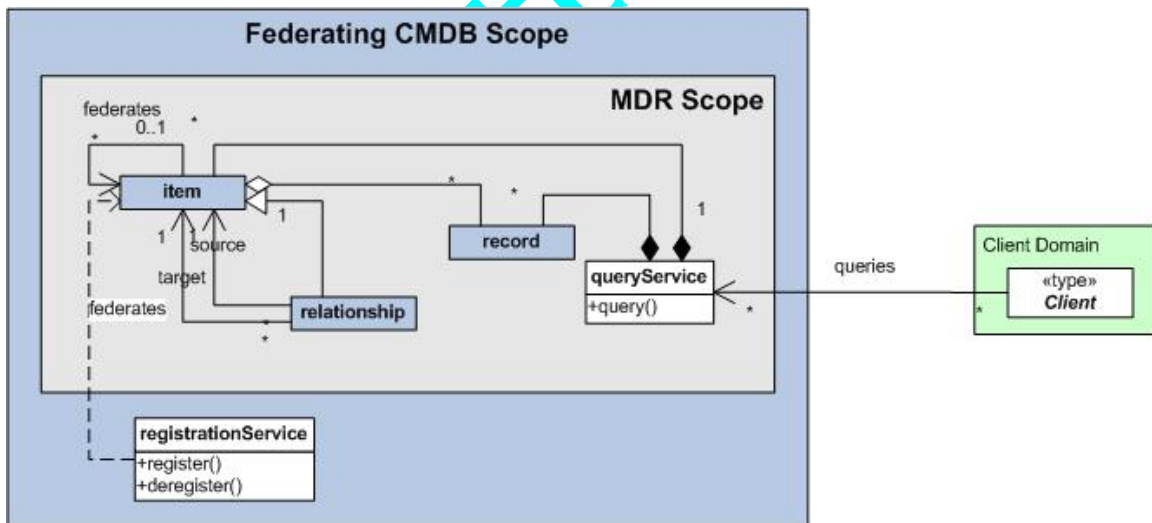
517 • Several records, possibly of various types, MAY be associated to the same
518 item or relationship.

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

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

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

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



542

543 **Figure 5 –Data & Services Overview**

544

545 **3.4.2 Common data element types**

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

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

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

554 This can be abbreviated in a pseudo schema to be:

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

556

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

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

562

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

568 • recordId: the unique Id of the record in the MDR. Required.

569 The <recordMetadata> element MAY also contain these properties:

570 • lastModified: the time/date the record was last modified in ISO 8601 format.
571 The applicable time zone or UTC MUST be indicated.

572 • baselineId: the name or other identifier used to group records into a
573 particular baseline configuration. A value of "0" indicates that this record is
574 not part of any baseline configuration.

575 • snapshotId: the name or other identifier used to group records observed in a
576 configuration snapshot (discovery). A value of "0" indicates that this record is
577 not part of any snapshot configuration.

578 • extensibility elements: additional metadata elements not defined by the
579 specification may also be included

580

581 4. Query Service

582 4.1 Overview

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

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

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

597 4.2 Example

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

```
601  
602 (01) <query>  
603  
604 (02) <itemTemplate id="user">  
605 (03) <recordConstraint>  
606 (04) <recordType namespace="http://example.com/people"  
607 <localName="person"/>  
608 (05) <propertyValue namespace="http://example.com/people"  
609 <localName="state">  
610 (06) <equal>CA</equal>  
611 (07) </propertyValue>  
612 (08) </recordConstraint>  
613 (09) </itemTemplate>  
614  
615 (10) <itemTemplate id="computer">  
616 (11) <recordConstraint>  
617 (12) <recordType namespace="http://example.com/computer"  
618 <localName="computer"/>  
619 (13) </recordConstraint>  
620 (14) </itemTemplate>  
621  
622 (15) <relationshipTemplate id="usage">  
623 (16) <recordConstraint>  
624 (17) <recordType namespace="http://example.com/computer"  
625 <localName="uses"/>  
626 (18) </recordConstraint>  
627 (19) <sourceTemplate ref="user"/>  
628 (20) <targetTemplate ref="computer"/>  
629 (21) </relationshipTemplate>  
630  
631 (22) </query>
```

632
633 The detailed syntax and semantics of the XML elements are described in details in
634 later sections, but here is in summary what items and relationships are returned by
635 this query:

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

- 637 • have a record with a property called "state" (in the
638 http://example.com/people namespace) for which the value is "CA",
- 639 • have a record named "person" (defined in the namespace
640 "http://example.com/people"), and
- 641 • are the source of a relationship that matches the <relationshipTemplate>
642 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

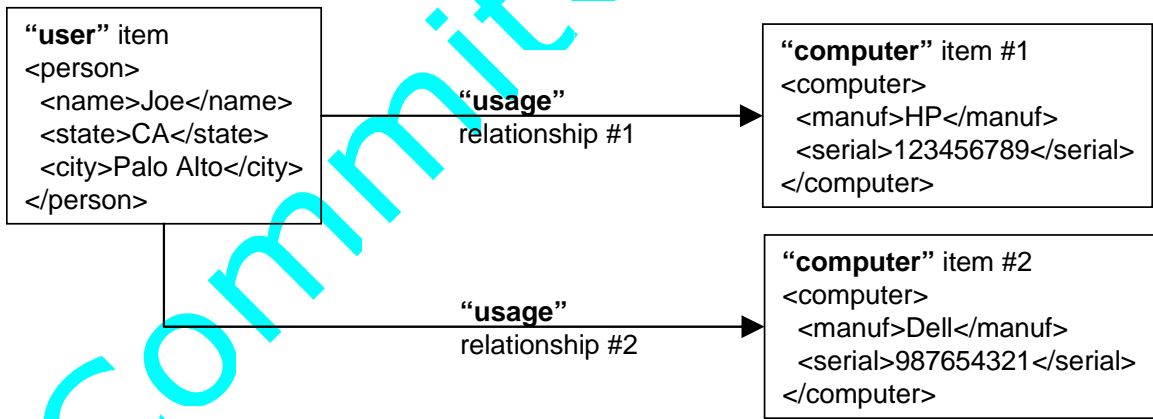
660 the response by three items (one for the user and one for each computer) and two

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

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

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

```
(01) <query>
(02)   <itemTemplate id="user" suppressFromResult="true">
(03)     <recordConstraint>
(04)       <recordType namespace="http://example.com/people"
(05)         localName="person"/>
(06)       <propertyValue namespace="http://example.com/people"
(07)         localName="state">
(08)         <equal>CA</equal>
(09)       </propertyValue>
(10)     </recordConstraint>
(11)   </itemTemplate>
(12)   <itemTemplate id="computer">
(13)     <recordConstraint>
(14)       <recordType namespace="http://example.com/computer"
(15)         localName="computer"/>
(16)     </recordConstraint>
(17)   </itemTemplate>
(18)   <relationshipTemplate id="usage" suppressFromResult="true">
(19)     <recordConstraint>
(20)       <recordType namespace="http://example.com/computer"
(21)         localName="uses"/>
(22)     </recordConstraint>
(23)     <sourceTemplate ref="user"/>
(24)     <targetTemplate ref="computer"/>
(25)   </relationshipTemplate>
(26) </query>
```

715

4.3 Normative definition

716

4.3.1 GraphQL

717

718

719

720

721

As illustrated in the previous example, a GraphQL 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.

722

723

724

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>`.

725

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

```

726 (01) <query>
727 (02)   <itemTemplate id="xs:ID" suppressFromResult="xs:boolean">
728 (03)     (<contentSelector ...>...</contentSelector> ?
729 (04)     <instanceIdConstraint>...</instanceIdConstraint> ?
730 (05)     <recordConstraint>
731 (06)       <recordType ... /> *
732 (07)       <propertyValue ...>...</propertyValue> *
733 (08)     </recordConstraint> *)
734 (09)     |
735 (10)     (<xpathExpression...>...</xpathExpression> *)
736 (11)     xs:any
737 (12)   </itemTemplate> *
738 (13)   <relationshipTemplate id="xs:ID" suppressFromResult="xs:boolean">
739 (14)     (<contentSelector ...>...</contentSelector> ?
740 (15)     <instanceIdConstraint>...</instanceIdConstraint> ?
741 (16)     <recordConstraint>
742 (17)       <recordType>...</recordType> *
743 (18)       <propertyValue>...</propertyValue> *
744 (19)     </recordConstraint> *)
745 (20)     |
746 (21)     (<xpathExpression ...>...</xpathExpression> *)
747 (22)     <sourceTemplate ref="xs:IDREF" minimum="xs:int"?
748 (23)       maximum="xs:int"?/>
749 (24)     <targetTemplate ref="xs:IDREF" minimum="xs:int"?
750 (25)       maximum="xs:int"?/>
751 (26)     <depthLimit ... /> ?
752 (27)     xs:any
753 (28)   </relationshipTemplate> *
754 (29) </query>

```

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

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

762

763 **itemTemplate**

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

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

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

775 <itemTemplate> (unless suppressed by the "suppressFromResult" attribute
776 described above.)

777

778 **relationshipTemplate**

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

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

791 Items cannot match a <relationshipTemplate>.

792

793 **relationshipTemplate/sourceTemplate**

794 **relationshipTemplate/targetTemplate**

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

799 Additionally, <sourceTemplate> and <targetTemplate> elements may have the
800 following optional attributes.

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

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

807

808 **relationshipTemplate/depthLimit**

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

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

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

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

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

826

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

830 **4.3.1.1 Content selection**

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

835

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

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

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

846

847 **contentSelector**

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

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

863

864 **contentSelector/selectedRecordType**

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

867

868 **contentSelector/selectedRecordType/selectedProperty**

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

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

875

```
876 (01) <query>  
877 (02)   <itemTemplate id="user">  
878 (03)     <contentSelector>  
879 (04)       <selectedRecordType namespace="http://example.com/models"  
880 (05)         localName="people">  
881 (06)           <selectedProperty namespace="http://example.com/models/people"  
882 (07)             localName="name"/>  
883 (08)           <selectedProperty namespace="http://example.com/models/people"  
884 (09)             localName="telephone"/>  
885 (10)       </selectedRecordType>  
886 (11)     </contentSelector>  
887 (12)     ...  
888 (13)   </itemTemplate>  
889 (14) </query>
```

890

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

```
894 (01) <record>  
895 (02)   <recordTypeQName>  
896 (03)     <propertyQName>xs:any</propertyQName> *  
897 (04)   </recordTypeQName>  
898 (05)   <recordMetadata>  
899 (06)     <recordId>xs:any</recordId>  
900 (07)     ...  
901 (08)   </recordMetadata>  
902 (09) </record> *
```

903

904 **4.3.1.2 Constraints**

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

907

908 **instanceIdConstraint**

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

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


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

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

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

923

924 **recordConstraint**

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

927 The pseudo-schema of this element is:

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

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

936

937 **recordConstraint/recordType**

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

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

947

948 **recordConstraint/propertyValue**

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

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

957 The pseudo-schema of this element is:

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



```

962 (05) <equal caseSensitive="xs:boolean"? negate="xs:boolean"? >
963 (06)   xs:anySimpleType
964 (07) </equal> *
965 (08) <less negate="xs:boolean"? >xs:anySimpleType</less> ?
966 (09) <lessOrEqual negate="xs:boolean"? >xs:anySimpleType</lessOrEqual> ?
967 (10) <greater negate="xs:boolean"? >xs:anySimpleType</greater> ?
968 (11) <greaterOrEqual negate="xs:boolean"? >
969 (12)   xs:anySimpleType
970 (13) </greaterOrEqual> ?
971 (14) <contains caseSensitive="xs:boolean"? negate="xs:boolean"? >
972 (15)   xs:string
973 (16) </contains> *
974 (17) <like caseSensitive="xs:boolean"? negate="xs:boolean"? >
975 (18)   xs:string
976 (19) </like> *
977 (20) <isNull negate="xs:boolean"? /> ?
978 (21)   xs:any
979 (22) </propertyValue>

```

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

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

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

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

1000 The operators are largely defined in terms of XPath 2.0 [XPath 2.0] comparison
1001 operators. This does not require that an XPath 2.0 implementation be used but
1002 only that the operators be evaluated in a way that is consistent with the XPath
1003 2.0 definitions, as described below.

1004 **recordConstraint/propertyValue/equal**

1005 This operator is defined in terms of the XPath 2.0 value comparison operator
1006 "eq". To evaluate, the left hand operand is the property value from the record
1007 and the right hand operand is the value of the constraint from the query. The
1008 type of the value of the constraint must be interpreted to be of the same type
1009 as the value from the property in the record. This operator is valid for
1010 properties of any simple type. A list of comparison behaviors is available in
1011 the XPath 2.0 Appendix B.2 Operator Mappings.

1012 **recordConstraint/propertyValue/less**

1013 **recordConstraint/propertyValue/lessOrEqual**

1014 **recordConstraint/propertyValue/greater**

1015 **recordConstraint/propertyValue/greaterOrEqual**

1016 These operators are defined in terms of the XPath 2.0 value comparison
1017 operators "lt", "le", "gt", and "ge", respectively. To evaluate, the left hand
1018 operand is the property value from the record and the right hand operand is
1019 the value of the constraint from the query. The type of the value of the
1020 constraint must be interpreted to be of the same type as the value from the
1021 property in the record. This operator is only valid for properties that are
1022 numerals, dates and strings. A list of comparison behaviors is available in the
1023 XPath 2.0 Appendix B.2 Operator Mappings. For example, if a property is of
1024 type date, the operator <less>2000-01-01T00:00:00</less> returns true if
1025 the property value is a date before the year 2000. If the property value was a
1026 string then "2000-01-01T00:00:00" would be interpreted as a string and
1027 compared with the property value using string comparison.

1028 **recordConstraint/propertyValue/contains**

1029 This operator is mapped to the XPath 2.0 function fn:contains(). It is only
1030 valid for properties of type string and used to test if the property value
1031 contains the specified string as a substring. The result of the contains
1032 operator is as if the fn:contains() function was executed with the first
1033 parameter being the property value and the second parameter being the
1034 string specified.

1035 **recordConstraint/propertyValue/like**

1036 This operator is similar in functionality to the SQL LIKE clause. The operator
1037 works like the equal operator with the inclusion of two special characters: the
1038 underscore ("_") acts as a wild card for any single character and the percent
1039 sign ("%") acts as a wild card for zero or more characters. To escape the wild
1040 cards, the backslash("\") can be used. For example,
1041 <like>Joe_Smith%</like> tests whether the property value starts with the
1042 string "Joe_Smith" and would match values such as "Joe_Smith",
1043 "Joe_Smith123" and "Joe_Smith_JR". It would not match "JoeHSmith123". A
1044 double backslash ("\\") represents the single backslash string ("\").

1045 **recordConstraint/propertyValue/isNull**

1046 This operator tests whether the element corresponding to the property is
1047 "nilled". It is equivalent to the result of applying the XPath 2.0 "fn:nilled"
1048 function on the element corresponding to the property.

1049

1050 Additional attributes defined for operator elements:

1051 **@caseSensitive** - equal, contains, and like operators have an optional attribute,
1052 caseSensitive, with a default value of true. If the property value of the record is
1053 an instance of xs:string and the attribute caseSensitive is false, the string
1054 comparison is case-insensitive. More precisely, the result of the comparison is as
1055 if the XPath 2.0 function fn:upper-case() was called on both the property value
1056 and the string value before comparison. If the property value of the record is not
1057 an instance of a xs:string, the caseSensitive attribute has no impact on the
1058 comparison.

1059
1060
1061
1062
1063
1064
1065

@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

1066

1067

Multiple instances of a property:

1068

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:

1069

1070

1071

```
(01) <comp:ComputerConfig xmlns:comp="http://example.com/computers">
```

1072

```
(02)   ...
```

1073

```
(03)   <comp:ip>1.2.3.4</comp:ip>
```

1074

```
(04)   <comp:ip>1.2.3.5</comp:ip>
```

1075

```
(05)   <comp:ip>1.2.3.6</comp:ip>
```

1076

```
(06)   ...
```

1077

```
(07) </comp:ComputerConfig>
```

1078

The following property constraint would return a positive result:

1079

```
(01) <recordConstraint>
```

1080

```
(02)   <propertyValue namespace="http://example.com/computers"
```

1081

```
(03)     localName="ip">
```

1082

```
(04)       <equal>1.2.3.5</equal>
```

1083

```
(05)     </propertyValue>
```

1084

```
(06) </recordConstraint>
```

1085

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:

1086

1087

1088

```

1089 (01) <recordConstraint>
1090 (02)   <propertyValue namespace="http://example.com/computers"
1091 (03)     localName="ip">
1092 (04)     <equal negate="true">1.2.3.5</equal>
1093 (05)   </propertyValue>
1094 (06) </recordConstraint>

```

1095 The property constraint would remove the item above from the result set
1096 because the equality comparison matches one IP address in the list.

1097 Similarly, `<less negate="true">12</less>` is equivalent to
1098 `<greaterOrEqual>12</greaterOrEqual>` if there is only one instance of the
1099 property being tested. But if there is more than one instance of the property,
1100 then the first operator is true if all of the instances have a value of more than 12,
1101 while the second one is true if at least one of the instances has a value of more
1102 than 12.

1103 A simple example of using `<propertyValue>`:

1104 In the following example, "Manufacturer" is a property defined in the
1105 "http://example.com/Computer" namespace. The constraint is testing whether
1106 the instance has a record containing this property and where the value of the
1107 property is "HP".

```

1108 (01) <recordConstraint>
1109 (02)   <propertyValue namespace="http://example.com/Computer"
1110 (03)     localName="Manufacturer" >
1111 (04)     <equal>HP</equal>
1112 (05)   </propertyValue>
1113 (06) </recordConstraint>

```

1114 A more complex example:

1115 The `<itemTemplate>` below matches any item that has a CPUCount greater than
1116 or equal to 2, for which the OSName property contains "Linux" (with that exact
1117 mix of upper and lower case) and for which the OSName property also contains
1118 either "ubuntu" or "debian" (irrespective of case).

```

1119 (01) <itemTemplate id="linuxMachine">
1120 (02)   <recordConstraint>
1121 (03)     <propertyValue namespace="http://example.com/computers"
1122 (04)       localName="CPUCount" >
1123 (05)       <greaterOrEqual>2</greaterOrEqual>
1124 (06)     </propertyValue>
1125 (07)     <propertyValue namespace="http://example.com/computers"
1126 (08)       localName="OSName" >
1127 (09)       <contains>Linux</contains>
1128 (10)     </propertyValue>
1129 (11)     <propertyValue namespace="http://example.com/computers"
1130 (12)       localName="OSName"
1131 (13)       matchAny="true" >
1132 (14)       <contains caseSensitive="false">ubuntu</contains>
1133 (15)       <contains caseSensitive="false">debian</contains>
1134 (16)     </propertyValue>
1135 (17)   </recordConstraint/>
1136 (18) </itemTemplate>

```

1137

1138 4.3.1.3 XPath selection and constraints

1139 The <xpathExpression> element provides an alternate mechanism to select content
1140 and filter instances based on the content of their records. The pseudo-schema of this
1141 element is:

```
1142 (01) <xpathExpression dialect="xs:anyURI">  
1143 (02) <prefixMapping prefix="xs:NCName" namespace="xs:anyURI"/> *  
1144 (03) <expression>xs:string</expression>  
1145 (04) </xpathExpression>
```

1146 This element can appear any number of times inside an <itemTemplate> or
1147 <relationshipTemplate> element. The use of the <xpathExpression> element is
1148 mutually exclusive with the usage of the group of <contentSelector>,
1149 <instanceIdConstraint> and <recordConstraint> elements in an <itemTemplate> or
1150 <relationshipTemplate>.

1151 **xpathExpression/@dialect**

1152 The dialect corresponds to a particular version of XPath represented by the URI
1153 value. Values defined in this specification for the dialect attribute:

- 1154 • "http://www.w3.org/TR/1999/REC-xpath-19991116" indicates that the
1155 expression corresponds to an XPath 1 expression.
- 1156 • "http://www.w3.org/TR/2007/REC-xpath-20070123" indicates that the
1157 expression corresponds to an XPath 2 expression.

1158 **xpathExpression/prefixMapping**

1159 Each <prefixMapping> child element of the <xpathExpression> element defines a
1160 namespace declaration for the XPath evaluation. The prefix for this declaration is
1161 provided by the <prefixMapping>/@prefix attribute and the namespace URI is
1162 provided by the <prefixMapping>/@namespace attribute.

1163 **xpathExpression/expression**

1164 The <expression> element contains an XPath expression to be evaluated
1165 according to the chosen dialect/profile (the @dialect attribute).

1166 The results of the XPath evaluation are interpreted as a constraint on the records
1167 of the item or relationship, such that an empty or "false" result from each record
1168 will cause the item or relationship to be excluded from the query response.

1169

1170 The XPath expression is evaluated in the following context:

- 1171 • Context Node: <record> element
- 1172 • Context Position: 1
- 1173 • Context Size: 1
- 1174 • Variable Binding: none
- 1175 • Function Libraries: core function library
- 1176 • Namespace Declarations: each <prefixMapping> child element of the
1177 <xpathExpression> element defines a namespace declaration for the XPath
1178 evaluation and the prefix `cmdbf` mapped to the namespace of this
1179 specification.

1180

1181 The XPath evaluation MUST also be used to select the contents of the result for this
1182 template. Since the return result will be wrapped in a <record> element, the result
1183 MUST NOT be the <record> element. Instead, it MUST be the descendent(s) of the
1184 <record> element or the result of some sort of processing such as `fn:count()`. An

1185 invalid query fault will be returned if the result of processing the XPath is a <record>
1186 element.

1187

1188 In the following example, "name" is a property defined in the
1189 "http://example.com/people" namespace. The constraint is testing whether the
1190 instance has a record containing this property and where the value of the property is
1191 "Pete the Lab Tech". In this example, no metadata is selected by the expression.

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

```
(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>
```

1203

4.3.2 GraphQL Query Response

1204

The pseudo-schema for the query response message is:

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

```
(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>
```



```

1232 (28) <target>
1233 (29) <mdrId>xs:anyURI</mdrId>
1234 (30) <localId>xs:anyURI</localId>
1235 (31) </target>
1236 (32) <record>
1237 (33) xs:any
1238 (34) <recordMetadata>
1239 (35) <recordId>...</recordId>
1240 (36) <lastModified>...</lastModified> ?
1241 (37) <baselineId>...</baselineId> ?
1242 (38) <snapshotId>...</snapshotId> ?
1243 (39) </recordMetadata>
1244 (40) </record> *
1245 (41) <instanceId>
1246 (42) <mdrId>xs:anyURI</mdrId>
1247 (43) <localId>xs:anyURI</localId>
1248 (44) </instanceId> +
1249 (45) <additionalRecordType namespace="xs:anyURI"
1250 (46) localName="xs:NCName" /> *
1251 (47) </relationship> +
1252 (48) </edges> *
1253 (49) </queryResult>

```

1254 Each time an item matches an <itemTemplate>, an <item> element appears inside
1255 a <nodes> element in the <queryResult>. The templateId attribute of this element
1256 contains the same value as the id attribute of the <itemTemplate> in the original
1257 request. If the item matches more than one <itemTemplate>, the <item> will be
1258 contained in the <nodes> for each <itemTemplate> matched by the item (each one
1259 with the appropriate value for its templateId attribute).

1260 Similarly, each time a relationship matches a <relationshipTemplate>, a
1261 <relationship> element appears inside an <edges> element in the <queryResult>.
1262 The templateId attribute of this element contains the same value as the id attribute
1263 of the <relationshipTemplate> in the original request. If the relationship matches
1264 more than one <relationshipTemplate>, the <relationship> will be contained in the
1265 <edges> for each <relationshipTemplate> matched by the relationship (each one
1266 with the appropriate value for its templateId attribute).

1267 If no item is part of the response, there are no <nodes> elements. If no relationship
1268 is part of the response, there are no <edges> elements.

1269 Items and relationships can contain any number of records. Each is represented by a
1270 <record> element. Each record element contains two child elements. The first child
1271 is an element whose QName is a recordType supported by the Query service. The
1272 children of that child are the properties associated with the record. The second child
1273 is a <recordMetadata> element, containing information about the record itself.

1274 Items and relationships MUST contain at least one <instanceId> element. The
1275 instance Id, through a combination of two URIs (<mdrId> to represent the MDR that
1276 assigned the ID and <localId> to uniquely represent the item or relationship inside
1277 this MDR), uniquely and globally identifies the item or relationship. There can be
1278 more than one <instanceId> element, in the case where the item or relationship has
1279 been reconciled from a more fragmented view.

1280 The <source> child element of a relationship identifies the item that is the source of
1281 the relationship. The format of this element matches the format of the <instanceId>
1282 element on the item.

1283 The <target> child element of a relationship identifies the item that is the target of
1284 the relationship. The format of this element matches the format of the <instanceId>
1285 element on the item.

1286 4.3.3 GraphQLy Faults

1287 The faults defined in this section are generated if the condition stated in the
1288 preamble is met. Faults are targeted at a destination endpoint according to the fault
1289 handling rules defined by the Web service binding.

1290 The definitions of faults in this section use the following properties:

1291 [Code] The fault code.

1292 [Subcode] The fault subcode.

1293 [Reason] The English language reason element.

1294 [Detail] The detail element. If absent, no detail element is defined for the fault.

1295 4.3.3.1 Unknown Template ID

1296 This fault occurs when a relationshipTemplate includes an id referring to a
1297 sourceTemplate, targetTemplate, or intermediateItemTemplate that was not included
1298 in the query.

1299 **Properties:**

1300 [Code] Sender

1301 [Subcode] cmdbf:UnkownTemplateID

1302 [Reason] The graph template ID was not declared.

1303 [Detail]

```
1304 <cmdbf:graphId> xs:ID </cmdbf:graphId>
```

1305 4.3.3.2 Property Type Mismatch

1306 This fault occurs when the value in a constraint is invalid for the type of the property
1307 as defined by the schema for the property. For example, this occurs when the
1308 property is a date and the query includes a parameter to compare to the date that is
1309 a string that cannot be cast to a date, such as "foobar."

1310 **Properties:**

1311 [Code] Sender

1312 [Subcode] cmdbf:InvalidPropertyType

1313 [Reason] The property value being compared is not valid.

1314 [Detail]

```
1315 <cmdbf:propertyName namespace="xs:anyURI" localname="xs:NCName" />
```

1316 4.3.3.3 XPath Processing Error

1317 This fault occurs when the XPath expression processing results in an error. See
1318 <http://www.w3.org/TR/xpath20/#id-errors> for details on the cmdbf:xpathErrorCode.

1319 **Properties:**

1320 [Code] Sender

1321 [Subcode] cmdbf:XPathError

1322 [Reason] The XPath expression was not processed successfully.

1323 [Detail]

```
1324 <cmdbf:expression> xs:string </cmdbf:expression>
```

```
1325 <cmdbf:xpathErrorCode> [xpath error code] </cmdbf:xpathErrorCode>
```

1326 4.3.3.4 Unsupported Constraint

1327 A constraint element in the template was specified that is not supported by this MDR.

1328 **Properties:**

1329 [Code] Receiver

1330 [Subcode] cmdbf:UnsupportedConstraint

1331 [Reason] The constraint specified is unsupported.

1332 [Detail]

```
1333 <cmdbf:constraint namespace="xs:anyURI" localname="xs:NCName" />
```

1334 4.3.3.5 Unsupported Selector

1335 A selector element in the template was specified that is not supported by this MDR.

1336 **Properties:**

1337 [Code] Receiver

1338 [Subcode] cmdbf:UnsupportedSelector

1339 [Reason] The selector specified is unsupported.

1340 [Detail]

```
1341 <cmdbf:selector namespace="xs:anyURI" localname="xs:NCName" />
```

1342 4.3.3.6 Query Error

1343 The query was valid, but there was an error while performing the query. When the
1344 query includes an XPath expression, this error may be used to indicate that the
1345 specific XPath dialect is not supported.

1346 **Properties:**

1347 [Code] Receiver

1348 [Subcode] cmdbf:QueryError

1349 [Reason] Error occurred while processing the request.

1350 [Detail]

```
1351 xs:any
```

1352

1353 4.4 GraphQL Example

1354 In this example, the data model contains item records of type ContactInfo and
1355 ComputerConfig and relationship records of type 'administrates'. ComputerConfigs are
1356 related to ContactInfo through the 'administrates' relationship to allow for modeling
1357 logic such as, "UserA administrates ComputerB."

1358 This example queries the graph of the computers which are administrated by Pete
1359 the Lab Tech and returns all items and relationships involved in this graph. The
1360 response shows two computers administrated by one user.

1361 Here are the data we assume the query is executed against.

1362 '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

1363

1364 '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	

1365

1366 'Administrators' 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

1367

1368 **Example "GraphQL" involving a relationship traversal**

1369

1370

1371

1372

1373

1374

1375

1376

1377

1378

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

```
(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
(15)         namespace="http://example.com/computerModel "
(16)         localName="ComputerConfig"/>
(17)     </recordConstraint>
(18)   </itemTemplate>
(19)   <relationshipTemplate id="administers">
(20)     <recordConstraint>
(21)       <recordType
(22)         namespace="http://example.com/computerModel "
(23)         localName="administers"/>
```

```

1392 (22) </recordConstraint>
1393 (23) <sourceTemplate ref="user"/>
1394 (24) <targetTemplate ref="computer"/>
1395 (25) </relationshipTemplate>
1396 (26) </query>

```

Example "GraphQL" response

```

1397 (01) <queryResult>
1398 (02) <nodes templateId="user">
1399 (03) <item>
1400 (04) <record xmlns:hr="http://example.com/people">
1401 (05) <hr:ContactInfo>
1402 (06) <hr:name>Pete the Lab Tech</hr:name>
1403 (07) <hr:phone>111-111-1111</hr:phone>
1404 (08) <hr:employeeNumber>33333</hr:employeeNumber>
1405 (09) </hr:ContactInfo>
1406 (10) <recordMetadata>
1407 (11) <recordId>http://example.com/33333/Current</recordId>
1408 (12) </recordMetadata>
1409 (13) </record>
1410 (14) <instanceId>
1411 (15) <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
1412 (16) <localId>http://example.com/PeteTheLabTech</localId>
1413 (17) </instanceId>
1414 (18) </item>
1415 (19) </nodes>
1416 (20) <nodes templateId="computer">
1417 (21) <item>
1418 (22) <record xmlns:comp="http://example.com/computerModel">
1419 (23) <comp:ComputerConfig>
1420 (24) <comp:CPUType>AMD Athlon 64</comp:CPUType>
1421 (25) <comp:assetTag>XYZ9753</comp:assetTag>
1422 (26) <comp:primaryMACAddress>
1423 (27) 00A4B49D2F41
1424 (28) </comp:primaryMACAddress>
1425 (29) <comp:name>LabMachineA</comp:name>
1426 (30) ...
1427 (31) </comp:ComputerConfig>
1428 (32) <recordMetadata>
1429 (33) <recordId>
1430 (34) http://example.com/machines/XYZ9753/scanned
1431 (35) </recordId>
1432 (36) </recordMetadata>
1433 (37) </record>
1434 (38) <instanceId>
1435 (39) <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
1436 (40) <localId>http://example.com/machines/XYZ9753</localId>
1437 (41) </instanceId>
1438 (42) </item>
1439 (43) </item>
1440

```

```

1441 (44) <record xmlns:comp="http://example.com/computerModel">
1442 (45)   <comp:ComputerConfig>
1443 (46)     <comp:CPUType>AMD Athlon 64</comp:CPUType>
1444 (47)     <comp:assetTag>XYZ9876</comp:assetTag>
1445 (48)     <comp:primaryMACAddress>
1446 (49)       00A4B49D2F42
1447 (50)     </comp:primaryMACAddress>
1448 (51)     <comp:name>LabMachineB</comp:name>
1449 (52)     ...
1450 (53)   </comp:ComputerConfig>
1451 (54)   <recordMetadata>
1452 (55)     <recordId>
1453 (56)       http://example.com/machines/XYZ9876/scanned
1454 (57)     </recordId>
1455 (58)   </recordMetadata>
1456 (59) </record>
1457 (60) <instanceId>
1458 (61)   <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
1459 (62)   <localId>http://example.com/machines/XYZ9876</localId>
1460 (63) </instanceId>
1461 (64) </item>
1462 (65) </nodes>
1463 (66) <edges templateId="administers">
1464 (67)   <relationship>
1465 (68)     <source>
1466 (69)       <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
1467 (70)       <localId>http://example.com/PeteTheLabTech</localId>
1468 (71)     </source>
1469 (72)     <target>
1470 (73)       <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
1471 (74)       <localId>http://example.com/machines/XYZ9876</localId>
1472 (75)     </target>
1473 (76)   <record xmlns:foo="http://example.com/computerModel">
1474 (77)     <foo:administers>
1475 (78)       <foo:adminSupportHours>
1476 (79)         business hours only
1477 (80)       </foo:adminSupportHours>
1478 (81)     </foo:administers>
1479 (82)   <recordMetadata>
1480 (83)     <recordId>adm10001</recordId>
1481 (84)   </recordMetadata>
1482 (85) </record>
1483 (86) <instanceId>
1484 (87)   <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
1485 (88)   <localId>
1486 (89)     http://example.com/administers/PeteTheLabTechToLabMachineB
1487 (90)   </localId>
1488 (91) </instanceId>
1489 (92) </relationship>

```

```

1490 (93) <relationship>
1491 (94) <source>
1492 (95) <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
1493 (96) <localId>http://example.com/PeteTheLabTech</localId>
1494 (97) </source>
1495 (98) <target>
1496 (99) <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
1497 (100) <localId>http://example.com/machines/XYZ9753</localId>
1498 (101) </target>
1499 (102) <record xmlns:foo="http://example.com/computerModel">
1500 (103) <foo:administers>
1501 (104) <foo:adminSupportHours>24/7</foo:adminSupportHours>
1502 (105) </foo:administers>
1503 (106) <recordMetadata>
1504 (107) <recordId>adm10002</recordId>
1505 (108) </recordMetadata>
1506 (109) </record>
1507 (110) <instanceId>
1508 (111) <mdrId>http://testSystem.com/DiscoveryMdr</mdrId>
1509 (112) <localId>
1510 (113) http://example.com/administers/PeteTheLabTechToLabMachineA
1511 (114) </localId>
1512 (115) </instanceId>
1513 (116) </relationship>
1514 (117) </edges>
1515 (118) </queryResult>

```

1516
1517

5. Registration Service

1518
1519

5.1 Overview

1520

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

1521
1522

The fundamentals of push mode federation are:

1523

- 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.

1524

1525

1526

1527

1528

1529

1530

1531

1532

1533

1534

1535

1536

- 1537 ○ If the return status is declined, the Registration service is presumably
- 1538 not maintaining the registration data, and no updates to that data are
- 1539 accepted.
- 1540 • The specification does not stipulate what the Registration service should or
- 1541 must do with the registered data. The semantics of accepted and declined
- 1542 only have meaning with respect to the obligations of the MDR to update the
- 1543 Registration service when the data changes.
- 1544 • The MDR also uses the Register operation to update registered data. An
- 1545 update may consist of any combination of:
 - 1546 ○ Changes to existing data, such as a property value change
 - 1547 ○ Registering an additional record type for this item or relationship
 - 1548 ○ Deregistering a previously registered record type for this item or
 - 1549 relationship
- 1550 • The MDR uses the Deregister operation to remove an existing registration for
- 1551 an item or relationship. For example, if the item or relationship is deleted, the
- 1552 MDR would typically delete its own records and deregister the previous
- 1553 registration. Another example when Deregister would be used is if an
- 1554 administrator decides to stop federating the data about this item or
- 1555 relationship, even though the item or relationship still exists and the MDR still
- 1556 maintains data about it.
- 1557 • The specification does not stipulate what the Registration service should or
- 1558 must do after a Deregister operation. To cite some non-prescriptive
- 1559 examples:
 - 1560 ○ If it has the same data from another MDR that this MDR deregisters, it
 - 1561 might disassociate the data with the deregistering MDR, while
 - 1562 maintaining the existing data.
 - 1563 ○ If it has data from another MDR about the deregistered item or
 - 1564 relationship, it might delete the deregistered data while maintaining
 - 1565 the data from the other MDR.
 - 1566 ○ If it has the same data from another MDR, but it considers the
 - 1567 deregistering MDR the authoritative source, it might mark the item or
 - 1568 relationship as deleted.
 - 1569 ○ If the deregistering MDR is the only source of data about the item or
 - 1570 relationship, it might delete all knowledge of the item or relationship.
 - 1571

1572 5.2 Normative definition

1573 5.2.1 Register

1574 The **Register operation** is used by an MDR to notify a Registration service that new
1575 items have been discovered or updated and data is now available in the MDR.

1576 The outline for the Register operation is as follows.

```

1577 (01) <registerRequest>
1578 (02)   <mdrId>xs:anyURI</mdrId>
1579 (03)   <itemList>
1580 (04)     <item>
1581 (05)       <record>
1582 (06)         xs:any
1583 (07)       <recordMetadata>...</recordMetadata>

```

```

1584      (08)      </record> *
1585      (09)      <instanceId>cmdbf:MdrScopedIdType</instanceId> +
1586      (10)      <additionalRecordType namespace="xs:anyURI "
1587                localName="xs:NCName" /> *
1588      (12)      </item> *
1589      (13)      <itemList> ?
1590      (14)      <relationshipList>
1591      (15)      <relationship>
1592      (16)      <source>cmdbf:MdrScopedIdType</source>
1593      (17)      <target>cmdbf:MdrScopedIdType</target>
1594      (18)      <record>
1595      (19)      xs:any
1596      (20)      <recordMetadata>...</recordMetadata>
1597      (21)      </record> *
1598      (22)      <instanceId>cmdbf:MdrScopedIdType</instanceId> +
1599      (23)      <additionalRecordType namespace="xs:anyURI "
1600                localName="xs:NCName" /> *
1601      (25)      </relationship> *
1602      (26)      <relationshipList> ?
1603      (27) </registerRequest>

```

1604 The following describes additional constraints on the outline listed above:

1605 **mdrId**

1606 The ID of the MDR registering its data. This ID **MUST** be unique among all of the
1607 MDRs and Federating CMDBs that are federated together.

1608 **itemList**

1609 The list of items being registered. The list contains any number of <item>
1610 elements, though if it contains zero <item> elements, including <itemList>
1611 serves no purpose. An <item> **SHOULD NOT** be repeated in the list.

1612 **itemList/item**

1613 Some or all of the contents of an <item>.

1614 **itemList/item/instanceId**

1615 The <instanceId> that serves as a unique key for the <item>. There **MUST** be at
1616 least one for each <item>. The <instanceId> **MUST** contain the values that
1617 would select the <item> in a query using an <instanceIdConstraint>.

1618 **itemList/item/record**

1619 Each <item> contains any number of <record> elements.

1620 The <record> element **MUST** contain exactly one child element of unrestricted
1621 type, followed by a <recordMetadata> element. The namespace and local name
1622 of the first child element together are the record type.

1623 The <record> type **MUST** be supported by the registration service.

1624 The MDR may support queries for <record> types that it chooses to not federate
1625 through the registration service.

1626 There **MAY** be multiple <record> elements. The set of passed elements will be
1627 considered a complete replacement if the registration service already has data
1628 from this MDR about this <item>. For example, if the MDR had previously
1629 registered this <item> with a ComputerConfiguration and ComputerAsset record,
1630 and another registration call is made for the same item with only the

1631 ComputerConfiguration record, then it will be treated as a deletion of the
1632 ComputerAsset record from the federation.

1633 itemList/item/additionalRecordType
1634 An MDR MAY support through its query interface record types for an item that are
1635 not included in the registerRequest message. If so, it MAY indicate the record
1636 types for the item by including one or more <additionalRecordType> elements.
1637 The <additionalRecordType>/@namespace and
1638 additionalRecordType/@localName attributes together represent the record type.
1639 In each <item> the same record type SHOULD NOT appear in both an
1640 <additionalRecordType> and a <record> element.

1641 For example, the MDR may support for queries ComputerIdentification,
1642 ComputerConfiguration, and ComputerAsset records. If the registerRequest
1643 message includes only the ComputerIdentification record contents in the
1644 <record> element, the MDR may provide in <additionalRecordType> elements
1645 the localName and namespace URIs for the ComputerConfiguration and
1646 ComputerAsset records

1647 relationshipList
1648 The list of relationships being registered. The list contains any number of
1649 <relationship> elements, though if it contains zero <relationship> elements,
1650 including <relationshipList> serves no purpose.

1651 relationshipList/relationship
1652 Some or all of the contents of a <relationship>.

1653 relationshipList/relationship/instanceId
1654 The <instanceId> that serves as a unique key for the <relationship>. There
1655 MUST be at least one for each <relationship>. The <instanceId> MUST contain
1656 the values that would select the <relationship> in a query using an
1657 <instanceIdConstraint>.

1658 relationshipList/relationship/source
1659 The <instanceId> that serves as a unique key for the <item> referenced by the
1660 source side of a relationship. There MUST be exactly one for each <relationship>.
1661 The <instanceId> MUST contain one of the values that would select the source
1662 <item> in a query using an <instanceIdConstraint>.

1663 relationshipList/relationship/target
1664 The <instanceId> that serves as a unique key for the <item> referenced by the
1665 target side of a relationship. There MUST be exactly one for each <relationship>.
1666 The <instanceId> MUST contain one of the values that would select the target
1667 <item> in a query using an <instanceIdConstraint>.

1668 relationshipList/relationship/record
1669 Each <relationship> contains any number of <record> elements. The <record>
1670 type MUST be supported by the registration service.

1671 The MDR may support queries for <record> types that it chooses to not federate
1672 through the registration service.

1673 There MAY be multiple <record> elements. The set of passed elements will be
1674 considered a complete replacement if the registration service already has data
1675 from this MDR about this <relationship>. For example, if the MDR had previously
1676 registered this <relationship> with a RunsOn and DependsOn record, and
1677 another registration call is made for the same item with only the RunsOn record,
1678 then it will be treated as a deletion of the DependsOn record from the federation.

1679 **relationshipList/relationship/additionalRecordType**
1680 An MDR MAY support through its query interface more record types for a
1681 relationship than it federates through the registration service. If so, it MAY
1682 indicate the record types per relationship instance by including one or more
1683 <additionalRecordType> elements. The <additionalRecordType>/@namespace
1684 and <additionalRecordType/@localName attributes together represent the record
1685 type. The MDR SHOULD NOT include an <additionalRecordType> if for the same
1686 record type it includes a <record>.
1687

1688 5.2.2 Register Response

1689 The outline for the response to a Register operation is as follows.

```
1690 (01) <registerResponse>  
1691 (02)   <instanceResponse>  
1692 (03)     <instanceId>cmdbf:MdrScopedIdType</instanceId>  
1693 (04)     <accepted>  
1694 (05)       <alternateInstanceId>  
1695 (06)         cmdbf:MdrScopedIdType  
1696 (07)       </alternateInstanceId> *  
1697 (08)     </accepted> ?  
1698 (09)     <declined>  
1699 (10)       <reason>xs:string</reason> *  
1700 (11)     </declined> ?  
1701 (12)   <instanceResponse> *  
1702 (13) </registerResponse>
```

1703 The following describes additional constraints on the outline listed above:

1704 **instanceResponse**

1705 An element that indicates the action taken for one item or relationship in the
1706 Register request. There can be any number of <instanceResponse> elements.
1707 There SHOULD be exactly one <instanceResponse> element per item or
1708 relationship in the Register request.

1709 **instanceResponse/instanceId**

1710 One of the <instanceId> elements from the Register request for an item or
1711 relationship.

1712 **instanceResponse/accepted**

1713 An element that indicates that the item or relationship instance was accepted.
1714 Exactly one of either <accepted> or <declined> MUST be present.

1715 **instanceResponse/accepted/alternateInstanceId**

1716 Zero or more elements that contain other IDs by which the item or relationship is
1717 known, each one of which is acceptable as a key to select the item or relationship
1718 in a query.

1719 **instanceResponse/declined**

1720 An element that indicates that the item or relationship instance was declined.
1721 Exactly one of either <accepted> or <declined> MUST be present.

1722 **instanceResponse/declined/reason**

1723 Zero or more strings that contain reason(s) why the registration was declined.
1724

1725 5.2.3 Register Operation Faults

1726 The faults defined in this section are generated if the condition stated in the
1727 preamble is met. Faults are targeted at a destination endpoint according to the fault
1728 handling rules defined by the Web service binding.

1729 The definitions of faults in this section use the following properties:

1730 [Code] The fault code.

1731 [Subcode] The fault subcode.

1732 [Reason] The English language reason element.

1733 [Detail] The detail element. If absent, no detail element is defined for the fault.

1734 5.2.3.1 Invalid Record

1735 The record does not correspond to the schema specifying the data model. This
1736 occurs when a required property does not exist, an extension property is used when
1737 the data model does not allow for extensions, etc.

1738 **Properties:**

1739 [Code] Sender

1740 [Subcode] cmdbf:InvalidRecord

1741 [Reason] The record is invalid.

1742 [Detail]

1743 `<cmdbf:recordId> xs:anyURI </cmdbf:recordId>`

1744 5.2.3.2 Unsupported Record Type

1745 A record of an unsupported record type was attempted to be registered.

1746 **Properties:**

1747 [Code] Sender

1748 [Subcode] cmdbf:UnsupportedRecordType

1749 [Reason] The record type is not supported.

1750 [Detail]

1751 `<cmdbf:recordType namespace="xs:anyURI" localname="xs:NCName" />`

1752 5.2.3.3 Invalid MDR Id

1753 The MDR Id specified on an item is not recognized.

1754 **Properties:**

1755 [Code] Sender

1756 [Subcode] cmdbf:InvalidMDR

1757 [Reason] The MDR is not registered.

1758 [Detail]

1759 `<cmdbf:mdrId> xs:anyURI </cmdbf:mdrId>`

1760 5.2.3.4 Registration Error

1761 There was a problem with registering the items/relationships.

1762 **Properties:**

1763 [Code] Sender

1764 [Subcode] cmdbf:RegistrationError

1765 [Reason] An error occurred while registering.
1766 [Detail]
1767 <cmdbf:recordId> xs:anyURI </cmdbf:recordId>
1768

1769 5.2.4 Deregister

1770 The Deregister operation is used by an MDR to notify the Registration service that
1771 the data that an MDR has about an item or relationship will no longer be registered.
1772 Each item or relationship only needs to be deregistered once regardless of the
1773 number of <instanceId> elements provided in the register request.

1774 The outline for the Deregister operation is as follows.

```
1775 (01) <deregisterRequest>  
1776 (02)   <mdrId>xs:anyURI</mdrId>  
1777 (03)   <itemIdList>  
1778 (04)     <instanceId>cmdbf:MdrScopedIdType</instanceId> *  
1779 (05)   <itemIdList> ?  
1780 (06)   <relationshipIdList>  
1781 (07)     <instanceId>cmdbf:MdrScopedIdType</instanceId> *  
1782 (08)   <relationshipIdList> ?  
1783 (09) </deregisterRequest>
```

1784 The following describes additional constraints on the outline listed above:

1785 **mdrId**

1786 The ID of the MDR deregistering its data. This ID MUST be the ID used when the
1787 data was registered using the Register request.

1788 **itemIdList**

1789 The list of items being deregistered. The list contains any number of
1790 <instanceId> elements, though if it contains zero <instanceId> elements,
1791 including <itemIdList> serves no purpose.

1792 **itemIdList/instanceId**

1793 The <instanceId> that serves as a key for the <item>. The <instanceId> MUST
1794 be either the <instanceId> from the Register request, or an
1795 <alternateInstanceId> from a <registerResponse>. An <instanceId> SHOULD
1796 NOT be repeated in the list.

1797 **relationshipIdList**

1798 The list of relationships being deregistered. The list contains any number of
1799 <instanceId> elements, though if it contains zero <instanceId> elements,
1800 including <relationshipList> serves no purpose.

1801 **relationshipIdList/instanceId**

1802 The <instanceId> that serves as a key for the <relationship>. The <instanceId>
1803 MUST be either the <instanceId> from the Register request, or an
1804 <alternateInstanceId> from a <registerResponse>. An <instanceId> SHOULD
1805 NOT be repeated in the list.

1806

1807 5.2.5 Deregister Response

1808 The outline for the response to a Deregister operation is as follows.

```
1809 (01) <deregisterResponse>
```

```

1810 (02) <instanceResponse>
1811 (03)   <instanceId>cmdbf:MdrScopedIdType</instanceId>
1812 (04)   <accepted /> ?
1813 (05)   <declined>
1814 (06)     <reason>xs:string</reason> *
1815 (07)   </declined> ?
1816 (08) <instanceResponse> *
1817 (09) </deregisterResponse>

```

1818 The following describes additional constraints on the outline listed above:

1819 **instanceResponse**

1820 An element that indicates the action taken for one item or relationship in the
 1821 Deregister request. There can be any number of <instanceResponse> elements.
 1822 There SHOULD be exactly one <instanceResponse> element per item or
 1823 relationship in the Register request.

1824 **instanceResponse/instanceId**

1825 The <instanceId> from the Deregister request for an item or relationship.

1826 **instanceResponse/accepted**

1827 An element that indicates that the item or relationship instance was accepted.
 1828 Exactly one of either <accepted> or <declined> MUST be present.

1829 **instanceResponse/declined**

1830 An element that indicates that the deregistration of the item or relationship
 1831 instance was declined. An example of when a Deregister request might be
 1832 declined is when the Registration service does not recognize <instanceId> in the
 1833 Deregister request.

1834 Exactly one of either <accepted> or <declined> MUST be present.

1835 **instanceResponse/declined/reason**

1836 Zero or more strings that contain reason(s) why the deregistration was declined.

1837 **5.2.6 Deregister Operation Faults**

1838 The faults defined in this section are generated if the condition stated in the
 1839 preamble is met. Faults are targeted at a destination endpoint according to the fault
 1840 handling rules defined by the Web service binding.

1841 The definitions of faults in this section use the following properties:

1842 [Code] The fault code.

1843 [Subcode] The fault subcode.

1844 [Reason] The English language reason element.

1845 [Detail] The detail element. If absent, no detail element is defined for the fault.

1846 **5.2.6.1 Invalid MDR Id**

1847 The MDR Id specified on an item is not recognized.

1848 **Properties:**

1849 [Code] Sender

1850 [Subcode] cmdbf:InvalidMDR

1851 [Reason] The MDR is not registered.

1852 [Detail]

1853 <cmdbf:mdrId> xs:anyURI </cmdbf:mdrId>

1854 5.2.6.2 *Deregistration Error*
1855 There was a problem with deregistering the items/relationships.
1856 **Properties:**
1857 [Code] Sender
1858 [Subcode] cmdbf:DeregistrationError
1859 [Reason] An error occurred while deregistering.
1860 [Detail]
1861 `<cmdbf:recordId> xs:anyURI </cmdbf:recordId>`
1862

1863 6. Service Metadata

1864 The register and query operations defined in this specification have a set of optional
1865 features that MAY be supported by a particular implementation. There are also a
1866 number of extensibility points in the specification that allow for expected variability
1867 in the implementations. One key point of variation is the data models supported for
1868 record types at a given MDR. Prior to sending register or query messages to an MDR,
1869 it may be necessary to inspect the capabilities and data models supported by that
1870 particular MDR. The schema defined in this section, includes two elements,
1871 `<queryServiceMetadata>` and `<registerServiceMetadata>` which can be used to
1872 indicate which optional features and data model(s), or record types, are supported
1873 by a particular implementation. It is RECOMMENDED that each MDR implementation
1874 include an instance of the appropriate `<queryServiceMetadata>` and/or
1875 `<registerServiceMetadata>` element(s) as part of the policies describing the
1876 implementation. An example of how these elements can be incorporated into a WS-
1877 Policy `<policy>` element and then associated with the implementation's WSDL
1878 binding is shown in Appendix D.

1879
1880 Following is a description of the service metadata schema elements
1881 `<queryServiceMetadata>` and `<registerServiceMetadata>` and their contents.

1882
1883 Any MDR supporting the GraphQuery operation MUST support an `<itemTemplate>` with
1884 `<instanceIdConstraint>` query at a minimum. Other query capabilities are optional. The service
1885 metadata for the MDR SHOULD indicate which optional query capabilities are supported.

1886 1887 **queryServiceMetadata**

1888
1889 An instance of the `<queryServiceMetadata>` includes the description of the MDR,
1890 including the ID of the MDR, the supported query capabilities and the supported
1891 records, or data model, for the given implementation being modeled.

1892 The pseudo-schema of the contents of a `<queryServiceMetadata>` element is shown
1893 below:

```
1894  
1895 (01) <queryServiceMetadata>  
1896 (02)   <serviceDescription>  
1897 (03)     <mdrId>xs:anyURI</mdrId>  
1898 (04)     <description>xs:string</description?>  
1899 (05)     xs:any *  
1900 (06)   </serviceDescription>
```

```

1901      (07) <queryCapabilities>
1902      (08)   <relationshipTemplateSupport depthLimit="xs:boolean"
1903      (09)     minimumMaximum="xs:boolean" xs:anyAttribute /> ?
1904      (10)   <contentSelectorSupport recordTypeSelector="xs:boolean"
1905      (11)     propertySelector="xs:boolean" xs:anyAttribute /> ?
1906      (12)   <recordConstraintSupport ...> ... </recordConstraintSupport?>
1907      (13)   <xpathSupport>
1908      (14)     <dialect>xs:anyURI</dialect>*
1909      (15)   </xpathSupport>
1910      (16)   xs:any *
1911      (17) </queryCapabilities>
1912      (18) <recordTypeList>
1913      (19)   <recordTypes namespace="xs:anyURI" schemaLocation="xs:anyURI">
1914      (20)     <recordType localName="xs:NCName" appliesTo="xs:string">
1915      (21)       xs:any *
1916      (22)     </recordType>
1917      (23)   </recordTypes> *
1918      (24) </recordTypeList>
1919      (25)   xs:any *
1920      (26) </queryServiceMetadata>

```

1921

1922 **queryServiceMetadata/serviceDescription**

1923 The required <serviceDescription> element is used to differentiate this particular
 1924 MDR from other MDRs in a particular environment. The <mdrId> is the only
 1925 required element in the <serviceDescription>. The other optional elements in the
 1926 <serviceDescription>, including an extensibility element, allow for further
 1927 description of the query implementation.

1928

1929 **queryServiceMetadata/serviceDescription/mdrId**

1930 The required <mdrId> is the ID of the MDR that assigned this particular
 1931 implementation.

1932

1933 **queryServiceMetadata/serviceDescription/description**

1934 The optional <description> element allows for text description of the instance to
 1935 be incorporated.

1936

1937 **queryServiceMetadata/queryCapabilities**

1938 The <queryCapabilities> describes which query techniques described in this
 1939 specification are supported by this particular implementation of the query
 1940 operation. The <queryCapabilities> includes an extensibility element for
 1941 representing that query extensions beyond the scope of this specification are
 1942 supported by the implementation.

1943

1944 **queryServiceMetadata/queryCapabilities/relationshipTemplateSupport**

1945 When present, the <relationshipTemplateSupport> element indicates that the
 1946 query operation of the implementation supports queries that include
 1947 <relationshipTemplate> elements.

1948 **@depthLimit** – the Boolean value of this attribute indicates if the query service
1949 implementation will process queries with a <depthLimit> element in a
1950 <relationshipTemplate>.
1951 **@minimumMaximum** – the Boolean value of this attribute indicates if the query
1952 implementation will process queries based on the cardinality of
1953 relationships as specified by a @minimum and/or @maximum attribute on a
1954 <sourceTemplate> or <targetTemplate> element of a <relationshipTemplate>.

1955

1956 **queryServiceMetadata/queryCapabilities/contentSelectorSupport**

1957 When present, the <contentSelectorSupport> element indicates that the query
1958 operation of the implementation supports queries that include <contentSelector>
1959 elements.

1960 **@recordTypeSelector** – the Boolean value of this attribute indicates if the query
1961 service implementation will process queries with <selectedRecordType> specified
1962 in the <contentSelector> of an <itemTemplate> or <relationshipTemplate>.

1963 **@propertyTypeSelector** – the Boolean value of this attribute indicates if the
1964 query service implementation will process queries with <selectedProperty>
1965 specified in the <contentSelector> of an <itemTemplate> or
1966 <relationshipTemplate>.

1967

1968 **queryServiceMetadata/queryCapabilities/recordConstraintSupport**

1969 The <recordConstraintSupport> element indicates if the query implementation
1970 will process queries that use constraints in the <itemTemplate> or
1971 <relationshipTemplate>. The complete pseudo-schema of this element is shown
1972 below:

1973

```
1974 (01) <recordConstraintSupport recordTypeConstraint="xs:boolean"  
1975 (02)   propertyValueConstraint="xs:boolean" xs:anyAttribute >  
1976 (03)   <propertyValuesOperators equal="xs:boolean" less="xs:boolean"  
1977 (04)     lessOrEqual="xs:boolean" greater="xs:boolean"  
1978 (05)     greaterOrEqual="xs:boolean" contains="xs:boolean"  
1979 (06)     like="xs:boolean" isNull="xs:boolean" xs:anyAttribute />?  
1980 (07) </recordConstraintSupport>
```

1981

1982 **@recordTypeConstraint** – the Boolean value of this attribute indicates if the
1983 query service implementation will process queries with <recordType> constraints
1984 in an <itemTemplate> or <relationshipTemplate>.

1985

1986 **@propertyValueConstraint** – the Boolean value of this attribute indicates if the
1987 query service implementation will process queries with <propertyValue>
1988 constraints in an <itemTemplate> or <relationshipTemplate>. When
1989 <propertyValue> constraints are supported the metadata SHOULD also indicate
1990 which operators are supported by including the <propertyValueOperators>
1991 element.

1992

1993 **recordConstraintSupport/propertyValueOperators**

1994 The <propertyValueOperators> element is used to indicate which operators are
1995 supported by the query implementation. There is a mandatory attribute for each

1996 operator defined by this specification and an extensibility attribute for other
1997 operators not defined by this specification.

1998

1999 The Boolean value of each of the following attributes indicates if the query
2000 service implementation will process queries with a property value operator of the
2001 same name as the attribute: **@equal**, **@less**, **@lessOrEqual**, **@greater**,
2002 **@greaterOrEqual**, **@contains**, **@like**, **@isNull**.

2003

2004 **queryServiceMetadata/queryCapabilities/xpathSupport**

2005 The <xpathSupport> element is used to indicate that the query implementation
2006 supports the dialects of XPath represented by the contained <dialect> elements.

2007

2008 **queryServiceMetadata/queryCapabilities/xpathSupport/dialect**

2009 The <dialect> elements indicate which dialects of XPath will be processed by the
2010 query implementation. The URI used as the value of the dialect should be one of
2011 the URIs listed in this specification for XPath dialects, or a URI defined by another
2012 specification that is defined to represent an XPath dialect appropriate for use in
2013 the query operation defined in this specification.

2014

2015 **queryServiceMetadata/recordTypeList**

2016 The <recordTypeList> is used to enumerate the elements that are considered
2017 valid for use as records in the implementation of the query service. For
2018 implementations of the query operation that are federating other data stores, this
2019 list of supported record types may change over time and SHOULD be kept current
2020 by the implementation.

2021

2022 **queryServiceMetadata/recordTypeList/recordTypes**

2023 For each different namespace that contains record types supported by the
2024 implementation, a <recordTypes> element should be included in the metadata
2025 that includes the namespace, schemaLocation if appropriate, and the list of the
2026 element names from that namespace which are supported by the implementation
2027 as <recordType> elements.

2028 **@namespace** – This mandatory attribute gives the namespace of the data model
2029 that includes XML elements that correspond to record types supported by the
2030 implementation.

2031 **@schemaLocation** – This optional attribute SHOULD be included when there is a
2032 URI that can be resolved to an XML schema representation of the elements
2033 belonging to the namespace listed in the namespace attribute.

2034

2035 **queryServiceMetadata/recordTypeList/recordTypes/recordType**

2036 This repeating sequence of <recordType> elements indicates which elements are
2037 supported as record types in the implementation. These elements MUST all be
2038 from the same namespace identified in the containing <recordTypes> element.

2039 **@localName** – The value of this attribute corresponds to the localName of a
2040 supported XML element that is a valid record type for the implementation.

2041 **@appliesTo** – This attribute MUST be one of three values indicating whether this
2042 element is valid as a record in a relationship, item, or both. The values for this
2043 attribute are from the enumeration, "relationship", "item" or "both".

2044

2045 **registerServiceMetadata**

2046

2047 An instance of the <registerServiceMetadata> includes the description of the MDR,
2048 including the ID of the MDR, and the supported records, or data model, for the given
2049 implementation being modeled.

2050

2051 The pseudo-schema of the contents of a <registerServiceMetadata> element is
2052 shown below:

2053

```
2054 (01) <registerServiceMetadata>  
2055 (02) <serviceDescription> ... </serviceDescription>  
2056 (03) <recordTypeList> ... </recordTypeList>  
2057 (04) xs:any *  
2058 (05) </registerServiceMetadata>
```

2059

2060 **registerServiceMetadata/serviceDescription**

2061

2062 The definition of the <serviceDescription> element is identical to the definition above
2063 in <queryServiceMetadata> except that the element is used to describe a
2064 registration service.

2065

2066 **registerServiceMetadata /recordTypeList**

2067

2068 The definition of the <recordTypeList> element is identical to the definition above in
2069 <queryServiceMetadata> except that the element is used to enumerate the record
2070 types supported by the registration service.

2071

2072 **7. Secure, Reliable, Asynchronous Federation**

2073

2074 This specification does not address a number of features that will predictably be
2075 required in an operational environment. Such features may be considered largely
2076 orthogonal to the operations defined in this specification and will affect no change to
2077 their definition. As a reference we list here some features which have been
2078 considered by the authors, but have been deemed out of scope. For the convenience
2079 of the reader references to other applicable standards are provided. These could be
2080 composed into the Web Services environment of an implementer needing or desiring
2081 the given functionality.

2082 **7.1 Security**

2083 The Federated Cmdb operates in a closed environment, where some security issues
2084 are less critical than in open access or public systems. Nonetheless there are a
2085 number of security areas that should be considered when implementing this
2086 specification. Although this specification makes no mandatory statements about what
2087 security mechanisms or protocols should be used, implementors should consider the
2088 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 may 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 during 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

2136 this area, but privacy should be provided for by operational policies adhering to local
2137 legislation. This specification sets no requirements with respect to privacy provision.

2138 **7.1.5 Delegation and Identity Federation**

2139 Because in one of the Federated CMDB use cases the CMDB acts as a proxy for other
2140 CMDBs or MDRs, the issues of delegation and/or identity federation need to be
2141 addressed. If security is required, there are three basic approaches available to
2142 implementers. One allows, through a number of mechanisms, the proxy to
2143 impersonate the client when making a request to the underlying CMDB or MDR. This
2144 approach is not recommended by the authors of this specification, as the identity of
2145 the actual requester is lost. In the second approach, the proxy CMDB acts in its own
2146 right when contacting the underlying CMDB or MDR. This approach is considered
2147 acceptable by the authors of this specification and can be implemented by the
2148 specifications listed above. While more complex in implementation, identity
2149 federation, implemented using WS-Trust and WS-Federation for example, is also
2150 considered a valid approach to delegation and federation. This specification sets no
2151 requirements with respect to delegation and identity federation.

2152 **7.1.6 Integrity**

2153 The provision of integrity ensures that data remains complete and unchanged both in
2154 the system and in transit, due to malicious or accidental disruption. With respect to
2155 the facilities provided by this specification, it is sufficient to use mechanisms that
2156 protect the integrity of the data in transit and leave the integrity within the system
2157 as a consideration for implementors. Implementations may consider WS-Security
2158 with XML Signature for message integrity and/or SSL or TLS for some level of "on
2159 the wire" integrity. This specification sets no requirements with respect to data
2160 storage integrity.

2161 **7.1.7 Availability**

2162 In addition to network level threats to availability, e.g. network outages, there is the
2163 potential for malicious or accidental denial of service (DOS) as a result of the open
2164 content nature of some aspects of this specification. Therefore, implementor should
2165 program defensively with respect to parsing and processing XML messages. There
2166 are no specific specifications that provide support in this area. This specification sets
2167 no requirements with respect to availability.

2168 **7.1.8 Audit and Compliance**

2169 Audit and compliance issues are unlikely to arise as a result of implementing this
2170 specification, however, the CMDB itself could be used to implement audit and
2171 compliance processes for other aspects of system operation. There are no specific
2172 specifications that provide support in this area. This specification sets no
2173 requirements with respect to audit and compliance.

2174 The reader is referred to the following standards:

- 2175 • XML Signature Syntax and Processing
- 2176 • XML Encryption Syntax and Processing
- 2177 • WS-Security 1.0
- 2178 • WS-SecureConversation 1.0
- 2179 • WS-Basic Security Profile 1.0

2180 **7.2 Reliability**

2181 Reliability is the ability for a sender of a given SOAP message to know that his or her
2182 message will be delivered to the correct receiver(s) with no loss of data. This is
2183 feature is addressed by the following Web Services standards and specifications:

- 2184 • WS-ReliableMessaging 1.0, 1.1
- 2185 • WS-I Reliable Secure Profile (in development)

2186 **7.3 Asynchrony**

2187 An asynchronous Web Service is one in which a request is made, but a response may
2188 not be given until some later time. During this intervening time the requestor is
2189 freed to do other operations. In this sense we consider asynchronous Web Services
2190 to be of a non-blocking nature. Asynchrony is addressed in the following Web
2191 Services standards and specifications:

- 2192 • WS-Addressing 1.0

2193
2194

2195 **8. Acknowledgements**

2196 The authors would like to acknowledge the contributions of the CMDB Federation
2197 Business Committee whose members included:

- 2198 Mike Baskey, IBM
- 2199 Tom Bishop, BMC Software
- 2200 Josh Cohen, Microsoft
- 2201 Rob Orr, IBM
- 2202 Jim Saliba, CA
- 2203 William Vambenepe, (formerly of) HP
- 2204 John Van Son, IBM
- 2205 Yoshinari Abe, Fujitsu

2206

2207 The authors would also like to acknowledge the CMDB Federation Use Case Working
2208 Group whose members included:

- 2209 Mark Johnson, IBM
- 2210 Pam Molennor, CA
- 2211 Mike Oitzman, (formerly of) BMC Software
- 2212 Klaus Wurster, HP

2213

2214 Finally, the authors would like to acknowledge people who have had some
2215 involvement in the discussion of the specification at various times during its
2216 development, including:

- 2217 Dale Clark, CA
- 2218 Ken Huang, BMC Software
- 2219 Stefan Negritoiu, Microsoft
- 2220 Tim van Ash, HP
- 2221 Marshall Whatley, HP
- 2222 Boris Yanishpolsky, Microsoft

2223

2224 **9. References**

2225

2226 **[RFC 2119]**

2227 S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC
2228 2119, Harvard University, March 1997. (See <http://www.ietf.org/rfc/rfc2119.txt>.)

2229 **[XPath 2.0]**

2230 "XML Path Language (XPath) 2.0", W3C Recommendation, January 2007 (See
2231 <http://www.w3.org/TR/xpath20/>.)

2232

2233 ****ITIL® is a Registered Trade Mark, and a Registered Community Trade Mark of the Office**
2234 **of Government Commerce, and is Registered in the U.S. Patent and Trademark Office.**

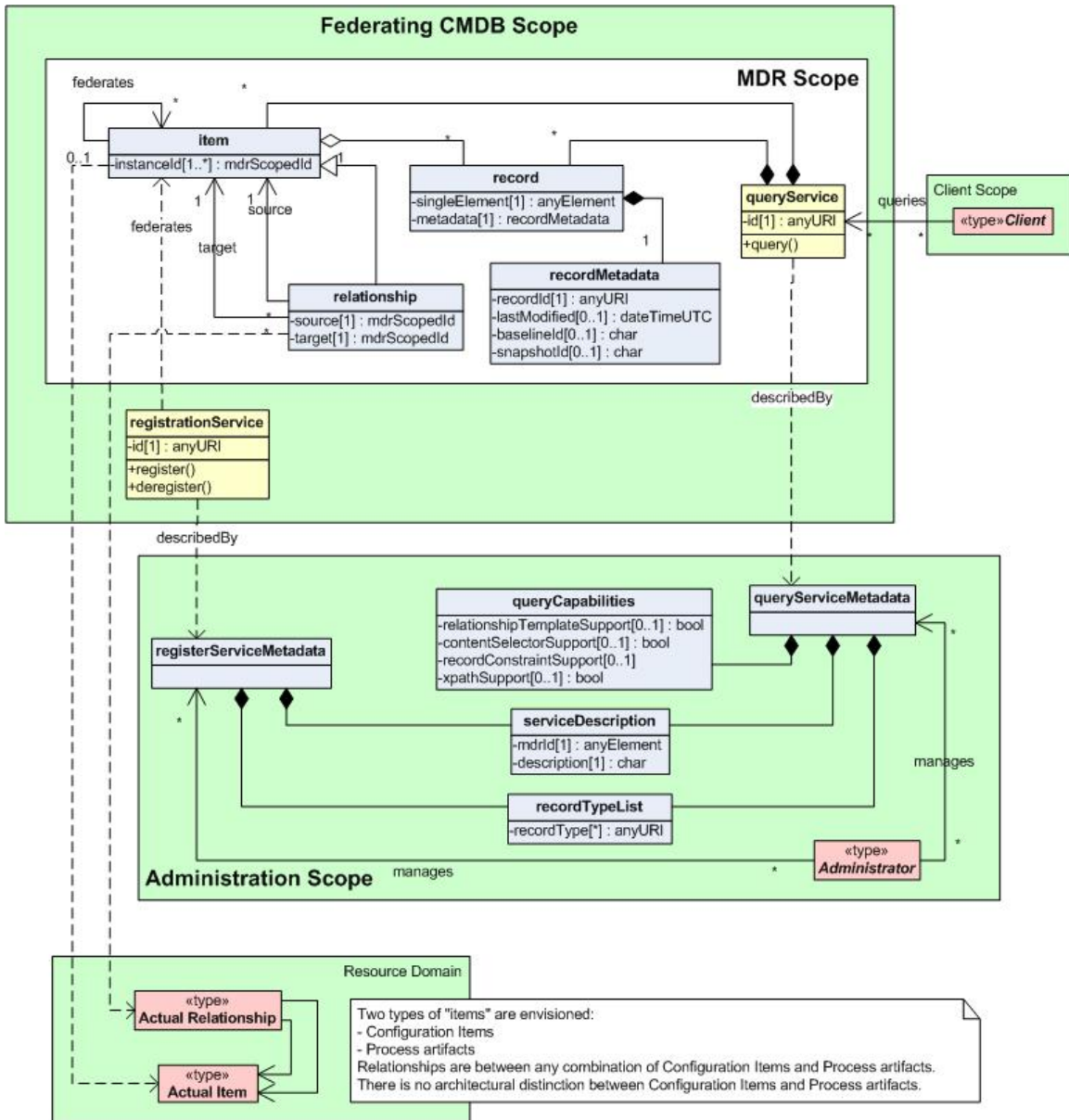
2235

2236

Committee Draft

2237
2238
2239

Appendix A Detailed UML Class Diagrams



2240

2241 **Figure 6 – Overall Class Diagrams**

2242
2243
2244

2245 **Appendix B XML Schema**

2246

2247 A normative copy of the XML Schema [[XML Schema Part 1](#), [Part 2](#)] description for
2248 this specification can be retrieved from the following addresses:

2249 <http://cmdbf.org/schema/1-0-0/cmdbfDatamodel.xsd>

2250 <http://cmdbf.org/schema/1-0-0/cmdbfServiceMetadata.xsd>

2251 A non-normative copy of the XML Schema description for the data model is listed
2252 below for convenience.

2253 **CMDBf Data Model Schema**



```
2254 <?xml version="1.0" encoding="UTF-8" ?>
```

2255

```
2256 <!--
```

```
2257   Copyright Notice
```

```
2258   (c) 2007 by BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and  
2259   Microsoft. All rights reserved.
```

2260

```
2261   Permission to copy and display this portion of the CMDB Federation  
2262   specification, in any medium without fee or royalty is hereby granted,  
2263   provided that you include the following on ALL copies of the CMDB  
2264   Federation specification, or portions thereof, that you make:
```

- ```
2265 1. A link or URL to the Specification at the website of one of the
2266 authors.
2267 2. The copyright notice as shown in the Specification.
```

2268

```
2269 BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and Microsoft
2270 (collectively, the "Authors") each agree to grant you a royalty-free
2271 license, under reasonable, non-discriminatory terms and conditions to
2272 their respective patents that they deem necessary to implement the
2273 Specification.
```

2274

```
2275 THE "CMDB FEDERATION" SPECIFICATION IS PROVIDED "AS IS,"
2276 AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
2277 IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY,
2278 FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT
2279 THE CONTENTS OF THE "CMDB FEDERATION" SPECIFICATION ARE SUITABLE FOR
2280 ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT
2281 INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER
2282 RIGHTS.
```

2283

```
2284 THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL,
2285 INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY
2286 USE OR DISTRIBUTION OF THE "CMDB FEDERATION" SPECIFICATION.
```

2287

2288 The name and trademarks of the Authors may NOT be used in any manner,  
2289 including advertising or publicity pertaining to the "CMDB  
2290 FEDERATION" Specification or its contents without specific, written  
2291 prior permission. Title to copyright in the "CMDB FEDERATION"  
2292 Specification will at all times remain with the Authors.  
2293  
2294 No other rights are granted by implication, estoppel or otherwise.  
2295 -->  
2296  
2297 <xs:schema targetNamespace="http://cmdbf.org/schema/1-0-0/datamodel"  
2298 elementFormDefault="qualified" blockDefault="#all"  
2299 xmlns:xs="http://www.w3.org/2001/XMLSchema"  
2300 xmlns:cmdbf="http://cmdbf.org/schema/1-0-0/datamodel">  
2301  
2302 <!-- Message Global Element Declarations -->  
2303  
2304 <xs:element name="query" type="cmdbf:QueryType" />  
2305 <xs:element name="queryResult" type="cmdbf:QueryResultType" />  
2306 <xs:element name="registerRequest"  
2307 type="cmdbf:RegisterRequestType" />  
2308 <xs:element name="registerResponse"  
2309 type="cmdbf:RegistrationServiceResponseType" />  
2310 <xs:element name="deregisterRequest"  
2311 type="cmdbf:DeregisterRequestType" />  
2312 <xs:element name="deregisterResponse"  
2313 type="cmdbf:RegistrationServiceResponseType" />  
2314  
2315 <!-- Query Fault Message Global Element Declarations -->  
2316  
2317 <xs:element name="UnkownTemplateIDFault">  
2318 <xs:complexType>  
2319 <xs:sequence>  
2320 <xs:element name="graphId" type="xs:ID" />  
2321 </xs:sequence>  
2322 </xs:complexType>  
2323 </xs:element>  
2324 <xs:element name="InvalidPropertyTypeFault">  
2325 <xs:complexType>  
2326 <xs:sequence>  
2327 <xs:element name="propertyName">  
2328 <xs:complexType>  
2329 <xs:attribute name="localName"  
2330 type="xs:NCName" use="required" />  
2331 <xs:attribute name="namespace"  
2332 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" />

```

```

2378 </xs:sequence>
2379 </xs:complexType>
2380 </xs:element>
2381
2382 <!-- Registration Message Global Element Declarations -->
2383
2384 <xs:element name="InvalidRecordFault">
2385 <xs:complexType>
2386 <xs:sequence>
2387 <xs:element name="recordId" type="xs:anyURI" />
2388 </xs:sequence>
2389 </xs:complexType>
2390 </xs:element>
2391 <xs:element name="UnsupportedRecordTypeFault">
2392 <xs:complexType>
2393 <xs:sequence>
2394 <xs:element name="recordType">
2395 <xs:complexType>
2396 <xs:attribute name="localName"
2397 type="xs:NCName" use="required" />
2398 <xs:attribute name="namespace"
2399 type="xs:anyURI" use="required" />
2400 </xs:complexType>
2401 </xs:element>
2402 </xs:sequence>
2403 </xs:complexType>
2404 </xs:element>
2405 <xs:element name="InvalidMDRFault">
2406 <xs:complexType>
2407 <xs:sequence>
2408 <xs:element name="mdrId" type="xs:anyURI" />
2409 </xs:sequence>
2410 </xs:complexType>
2411 </xs:element>
2412 <xs:element name="RegistrationErrorFault">
2413 <xs:complexType>
2414 <xs:sequence>
2415 <xs:element name="recordId" type="xs:anyURI" />
2416 </xs:sequence>
2417 </xs:complexType>
2418 </xs:element>
2419 <xs:element name="DeregistrationErrorFault">
2420 <xs:complexType>
2421 <xs:sequence>
2422 <xs:element name="recordId" type="xs:anyURI" />

```

```

2423 </xs:sequence>
2424 </xs:complexType>
2425 </xs:element>
2426
2427 <!-- Query Request Definitions -->
2428 <xs:complexType name="QueryType">
2429 <xs:sequence>
2430 <xs:element name="itemTemplate"
2431 type="cndbf:ItemTemplateType" minOccurs="0"
2432 maxOccurs="unbounded" />
2433 <xs:element name="relationshipTemplate"
2434 type="cndbf:RelationshipTemplateType" minOccurs="0"
2435 maxOccurs="unbounded" />
2436 </xs:sequence>
2437 </xs:complexType>
2438
2439 <xs:complexType name="ItemTemplateType">
2440 <xs:sequence>
2441 <xs:choice>
2442 <xs:group ref="cndbf:Constraints" />
2443 <xs:element name="xpathExpression"
2444 type="cndbf:XPathExpressionType" minOccurs="0"
2445 maxOccurs="unbounded" />
2446 </xs:choice>
2447 <xs:any minOccurs="0" maxOccurs="unbounded"
2448 namespace="##other" processContents="lax" />
2449 </xs:sequence>
2450 <xs:attribute name="id" type="xs:ID" use="required" />
2451 <xs:attribute name="suppressFromResult" type="xs:boolean"
2452 use="optional" default="false"/>
2453 </xs:complexType>
2454
2455 <xs:group name="Constraints" >
2456 <xs:sequence>
2457 <xs:element name="contentSelector"
2458 type="cndbf:ContentSelectorType" minOccurs="0"
2459 maxOccurs="1" />
2460 <xs:element name="instanceIdConstraint"
2461 type="cndbf:InstanceIdConstraintType"
2462 minOccurs="0" maxOccurs="1" />
2463 <xs:element name="recordConstraint"
2464 type="cndbf:RecordConstraintType" minOccurs="0"
2465 maxOccurs="unbounded" />
2466 </xs:sequence>
2467 </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="cndbf: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="cndbf: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="cndbf:QNameType" minOccurs="0"
2539 maxOccurs="unbounded" />
2540 <xs:element name="propertyValue"
2541 type="cndbf: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="cndbf:EqualOperatorType"
2551 minOccurs="0" maxOccurs="unbounded" />
2552 <xs:element name="less" type="cndbf:ComparisonOperatorType"
2553 minOccurs="0" maxOccurs="1" />
2554 <xs:element name="lessOrEqual"
2555 type="cndbf:ComparisonOperatorType" minOccurs="0"
2556 maxOccurs="1" />
2557 <xs:element name="greater"

```



```

2558 type="cndbf:ComparisonOperatorType" minOccurs="0"
2559 maxOccurs="1" />
2560 <xs:element name="greaterOrEqual"
2561 type="cndbf:ComparisonOperatorType" minOccurs="0"
2562 maxOccurs="1" />
2563 <xs:element name="contains" type="cndbf:StringOperatorType"
2564 minOccurs="0" maxOccurs="unbounded" />
2565 <xs:element name="like" type="cndbf:StringOperatorType"
2566 minOccurs="0" maxOccurs="unbounded" />
2567 <xs:element name="isNull" type="cndbf: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

```

```

2603 <xs:complexType name="EqualOperatorType">
2604 <xs:simpleContent>
2605 <xs:extension base="xs:anySimpleType">
2606 <xs:attribute name="caseSensitive" type="xs:boolean"
2607 use="optional" default="true" />
2608 <xs:attribute name="negate" type="xs:boolean"
2609 use="optional" default="false" />
2610 </xs:extension>
2611 </xs:simpleContent>
2612 </xs:complexType>
2613
2614 <xs:complexType name="NullOperatorType">
2615 <xs:attribute name="negate" type="xs:boolean" use="optional"
2616 default="false" />
2617 </xs:complexType>
2618
2619 <xs:complexType name="XPathExpressionType">
2620 <xs:sequence>
2621 <xs:element name="prefixMapping" type="cndbf:PrefixMappingType"
2622 minOccurs="1" maxOccurs="unbounded" />
2623 <xs:element name="expression" type="xs:string" />
2624 </xs:sequence>
2625 <xs:attribute name="dialect" type="xs:anyURI" use="required" />
2626 </xs:complexType>
2627
2628 <xs:complexType name="PrefixMappingType">
2629 <xs:attribute name="prefix" type="xs:NCName" use="required" />
2630 <xs:attribute name="namespace" type="xs:anyURI"
2631 use="required" />
2632 </xs:complexType>
2633
2634 <!-- Query Response definition -->
2635 <xs:complexType name="QueryResultType">
2636 <xs:sequence>
2637 <xs:element name="nodes" type="cndbf:NodesType"
2638 minOccurs="0" maxOccurs="unbounded" />
2639 <xs:element name="edges" type="cndbf:EdgesType"
2640 minOccurs="0" maxOccurs="unbounded" />
2641 </xs:sequence>
2642 </xs:complexType>
2643
2644 <xs:complexType name="NodesType">
2645 <xs:sequence>
2646 <xs:element ref="cndbf:item" minOccurs="1"
2647 maxOccurs="unbounded" />

```

```

2648 </xs:sequence>
2649 <xs:attribute name="templateId" type="xs:ID" use="required" />
2650 </xs:complexType>
2651
2652 <xs:complexType name="EdgesType">
2653 <xs:sequence>
2654 <xs:element ref="cndbf:relationship" minOccurs="1"
2655 maxOccurs="unbounded" />
2656 </xs:sequence>
2657 <xs:attribute name="templateId" type="xs:ID" use="required" />
2658 </xs:complexType>
2659
2660 <!-- Registration Service -->
2661 <xs:complexType name="RegisterRequestType">
2662 <xs:sequence>
2663 <xs:element name="mdrId" type="xs:anyURI" />
2664 <xs:element name="itemList" type="cndbf:ItemListType"
2665 minOccurs="0" maxOccurs="1" />
2666 <xs:element name="relationshipList"
2667 type="cndbf:RelationshipListType" minOccurs="0"
2668 maxOccurs="1" />
2669 </xs:sequence>
2670 </xs:complexType>
2671
2672 <xs:complexType name="ItemListType">
2673 <xs:sequence>
2674 <xs:element ref="cndbf:item" minOccurs="1"
2675 maxOccurs="unbounded" />
2676 </xs:sequence>
2677 </xs:complexType>
2678 <xs:complexType name="RelationshipListType">
2679 <xs:sequence>
2680 <xs:element ref="cndbf:relationship" minOccurs="1"
2681 maxOccurs="unbounded" />
2682 </xs:sequence>
2683 </xs:complexType>
2684
2685 <xs:complexType name="DeregisterRequestType">
2686 <xs:sequence>
2687 <xs:element name="mdrId" type="xs:anyURI" />
2688 <xs:element name="itemIdList"
2689 type="cndbf:MdrScopedIdListType" minOccurs="0"
2690 maxOccurs="1" />
2691 <xs:element name="relationshipIdList"
2692 type="cndbf:MdrScopedIdListType" minOccurs="0"

```

```

2693 maxOccurs="1" />
2694 </xs:sequence>
2695 </xs:complexType>
2696
2697 <xs:complexType name="MdrScopedIdListType">
2698 <xs:sequence>
2699 <xs:element ref="cmdbf:instanceId" minOccurs="1"
2700 maxOccurs="unbounded" />
2701 </xs:sequence>
2702 </xs:complexType>
2703
2704 <xs:complexType name="RegistrationServiceResponseType">
2705 <xs:sequence>
2706 <xs:element name="instanceResponse"
2707 type="cmdbf:InstanceResponseType" minOccurs="0"
2708 maxOccurs="unbounded" />
2709 </xs:sequence>
2710 </xs:complexType>
2711
2712 <xs:complexType name="InstanceResponseType">
2713 <xs:sequence>
2714 <xs:element name="instanceId" type="cmdbf:MdrScopedIdType"
2715 minOccurs="1" maxOccurs="1" />
2716 <xs:element name="accepted" type="cmdbf:AcceptedType"
2717 maxOccurs="1" minOccurs="0" />
2718 <xs:element name="declined" type="cmdbf:DeclinedType"
2719 maxOccurs="1" minOccurs="0" />
2720 </xs:sequence>
2721 </xs:complexType>
2722
2723 <xs:complexType name="AcceptedType">
2724 <xs:sequence>
2725 <xs:element name="alternateInstanceId"
2726 type="cmdbf:MdrScopedIdType" maxOccurs="unbounded"
2727 minOccurs="0" />
2728 </xs:sequence>
2729 </xs:complexType>
2730
2731 <xs:complexType name="DeclinedType">
2732 <xs:sequence>
2733 <xs:element name="reason" type="xs:string"
2734 maxOccurs="unbounded" minOccurs="0" />
2735 </xs:sequence>
2736 </xs:complexType>
2737

```

```

2738 <!-- Shared elements definition -->
2739 <xs:element name="item" type="cmdbf:ItemType" />
2740 <xs:complexType name="ItemType">
2741 <xs:sequence>
2742 <xs:element ref="cmdbf:record" minOccurs="0"
2743 maxOccurs="unbounded" />
2744 <xs:element ref="cmdbf:instanceId" minOccurs="1"
2745 maxOccurs="unbounded" />
2746 <xs:element name="additionalRecordType"
2747 type="cmdbf:QNameType" minOccurs="0"
2748 maxOccurs="unbounded" />
2749 </xs:sequence>
2750 </xs:complexType>
2751
2752 <xs:element name="relationship" type="cmdbf:RelationshipType" />
2753 <xs:complexType name="RelationshipType">
2754 <xs:sequence>
2755 <xs:element name="source" type="cmdbf:MdrScopedIdType"
2756 minOccurs="1" maxOccurs="1" />
2757 <xs:element name="target" type="cmdbf:MdrScopedIdType"
2758 minOccurs="1" maxOccurs="1" />
2759 <xs:element ref="cmdbf:record" minOccurs="0"
2760 maxOccurs="unbounded" />
2761 <xs:element ref="cmdbf:instanceId" minOccurs="1"
2762 maxOccurs="unbounded" />
2763 <xs:element name="additionalRecordType"
2764 type="cmdbf:QNameType" maxOccurs="unbounded"
2765 minOccurs="0" />
2766 </xs:sequence>
2767 </xs:complexType>
2768
2769 <xs:element name="record" type="cmdbf:RecordType" />
2770 <xs:complexType name="RecordType">
2771 <xs:sequence>
2772 <xs:any namespace="##other" processContents="skip" />
2773 <xs:element name="recordMetadata" >
2774 <xs:complexType>
2775 <xs:sequence>
2776 <xs:element name="recordId" type="xs:anyURI" />
2777 <xs:element name="lastModified" type="xs:dateTime"
2778 minOccurs="0" />
2779 <xs:element name="baselineId" type="xs:string"
2780 minOccurs="0" />
2781 <xs:element name="snapshotId" type="xs:string"
2782 minOccurs="0" />

```

```

2783 <xs:any namespace="##other" processContents="lax"
2784 minOccurs="0" maxOccurs="unbounded" />
2785 </xs:sequence>
2786 </xs:complexType>
2787 </xs:element>
2788 </xs:sequence>
2789 </xs:complexType>
2790
2791 <xs:element name="instanceId" type="cmdbf:MdrScopedIdType" />
2792 <xs:complexType name="MdrScopedIdType">
2793 <xs:sequence>
2794 <xs:element name="mdrId" type="xs:anyURI" minOccurs="1"
2795 maxOccurs="1" />
2796 <xs:element name="localId" type="xs:anyURI" minOccurs="1"
2797 maxOccurs="1" />
2798 </xs:sequence>
2799 </xs:complexType>
2800
2801 <xs:complexType name="QNameType">
2802 <xs:attribute name="namespace" type="xs:anyURI"
2803 use="required" />
2804 <xs:attribute name="localName" type="xs:NCName"
2805 use="required" />
2806 </xs:complexType>
2807
2808 </xs:schema>

```

2809 A non-normative copy of the XML Schema description for the service description  
2810 meta-model is listed below for convenience.

## 2811 **CMDBf Service Description Schema**

```

2812 <?xml version="1.0" encoding="UTF-8" ?>
2813
2814 <!--
2815 Copyright Notice
2816 (c) 2007 by BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and
2817 Microsoft. All rights reserved.
2818
2819 Permission to copy and display this portion of the CMDB Federation
2820 specification, in any medium without fee or royalty is hereby granted,
2821 provided that you include the following on ALL copies of the CMDB
2822 Federation specification, or portions thereof, that you make:
2823 1. A link or URL to the Specification at the website of one of the
2824 authors.
2825 2. The copyright notice as shown in the Specification.
2826

```

2827 BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and Microsoft  
2828 (collectively, the "Authors") each agree to grant you a royalty-free  
2829 license, under reasonable, non-discriminatory terms and conditions to  
2830 their respective patents that they deem necessary to implement the  
2831 Specification.  
2832  
2833 THE "CMDB FEDERATION" SPECIFICATION IS PROVIDED "AS IS,"  
2834 AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR  
2835 IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY,  
2836 FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT  
2837 THE CONTENTS OF THE "CMDB FEDERATION" SPECIFICATION ARE SUITABLE FOR  
2838 ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT  
2839 INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER  
2840 RIGHTS.  
2841  
2842 THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL,  
2843 INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY  
2844 USE OR DISTRIBUTION OF THE "CMDB FEDERATION" SPECIFICATION.  
2845  
2846 The name and trademarks of the Authors may NOT be used in any manner,  
2847 including advertising or publicity pertaining to the "CMDB  
2848 FEDERATION" Specification or its contents without specific, written  
2849 prior permission. Title to copyright in the "CMDB FEDERATION"  
2850 Specification will at all times remain with the Authors.  
2851  
2852 No other rights are granted by implication, estoppel or otherwise.  
2853 -->  
2854  
2855 <xs:schema  
2856 targetNamespace="http://cmdbf.org/schema/1-0-0/serviceMetadata"  
2857 elementFormDefault="qualified" blockDefault="#all"  
2858 xmlns:xs="http://www.w3.org/2001/XMLSchema"  
2859 xmlns:cmdbfmd="http://cmdbf.org/schema/1-0-0/serviceMetadata">  
2860  
2861 <!-- Common elements -->  
2862 <xs:element name="serviceDescription">  
2863 <xs:complexType>  
2864 <xs:sequence>  
2865 <xs:element name="mdrId" type="xs:anyURI" />  
2866 <xs:element name="description" type="xs:string"  
2867 minOccurs="0" maxOccurs="1" />  
2868 <xs:any minOccurs="0" maxOccurs="unbounded"  
2869 namespace="##other" processContents="lax" />  
2870 </xs:sequence>  
2871 </xs:complexType>

```

2872 </xs:element>
2873
2874 <xs:element name="recordTypeList">
2875 <xs:complexType>
2876 <xs:sequence>
2877 <xs:element ref="cmdbfmd:recordTypes"
2878 minOccurs="0" maxOccurs="unbounded" />
2879 </xs:sequence>
2880 </xs:complexType>
2881 </xs:element>
2882
2883 <xs:element name="recordTypes">
2884 <xs:complexType>
2885 <xs:sequence>
2886 <xs:element ref="cmdbfmd:recordType"
2887 minOccurs="0" maxOccurs="unbounded" />
2888 </xs:sequence>
2889 <xs:attribute name="namespace" type="xs:anyURI"
2890 use="required" />
2891 <xs:attribute name="schemaLocation" type="xs:anyURI"
2892 use="optional" />
2893 </xs:complexType>
2894 </xs:element>
2895
2896 <xs:element name="recordType">
2897 <xs:complexType>
2898 <xs:sequence>
2899 <xs:any minOccurs="0" maxOccurs="unbounded"
2900 namespace="##other" processContents="lax" />
2901 </xs:sequence>
2902 <xs:attribute name="localName" type="xs:NCName"
2903 use="required" />
2904 <xs:attribute name="appliesTo" use="required">
2905 <xs:simpleType>
2906 <xs:restriction base="xs:string">
2907 <xs:enumeration value="item" />
2908 <xs:enumeration value="relationship" />
2909 <xs:enumeration value="both" />
2910 </xs:restriction>
2911 </xs:simpleType>
2912 </xs:attribute>
2913 </xs:complexType>
2914 </xs:element>
2915
2916

```



```

2917 <!-- Query Service metadata definition -->
2918 <xs:element name="queryServiceMetadata">
2919 <xs:complexType>
2920 <xs:sequence>
2921 <xs:element ref="cmdbfmd:serviceDescription" />
2922 <xs:element ref="cmdbfmd:queryCapabilities" />
2923 <xs:element ref="cmdbfmd:recordTypeList" />
2924 <xs:any minOccurs="0" maxOccurs="unbounded"
2925 namespace="##other" processContents="lax" />
2926 </xs:sequence>
2927 </xs:complexType>
2928 </xs:element>
2929
2930 <xs:element name="queryCapabilities">
2931 <xs:complexType>
2932 <xs:sequence>
2933 <xs:element name="relationshipTemplateSupport"
2934 type="cmdbfmd:RelationshipTemplateType"
2935 minOccurs="0" maxOccurs="1" />
2936 <xs:element name="contentSelectorSupport"
2937 type="cmdbfmd:ContentSelectorType"
2938 minOccurs="0" maxOccurs="1" />
2939 <xs:element name="recordConstraintSupport"
2940 type="cmdbfmd:RecordConstraintType"
2941 minOccurs="0" maxOccurs="1" />
2942 <xs:element name="xpathSupport"
2943 type="cmdbfmd:XPathType"
2944 minOccurs="0" maxOccurs="1" />
2945 <xs:any minOccurs="0" maxOccurs="unbounded"
2946 namespace="##other" processContents="lax" />
2947 </xs:sequence>
2948 </xs:complexType>
2949 </xs:element>
2950
2951 <xs:complexType name="RelationshipTemplateType">
2952 <xs:attribute name="depthLimit" type="xs:boolean" use="required" />
2953 <xs:attribute name="minimumMaximum"
2954 type="xs:boolean" use="required" />
2955 <xs:anyAttribute namespace="##other" processContents="lax" />
2956 </xs:complexType>
2957
2958 <xs:complexType name="ContentSelectorType">
2959 <xs:attribute name="recordTypeSelector"
2960 type="xs:boolean" use="required" />
2961 <xs:attribute name="propertySelector"

```

```

2962 type="xs:boolean" use="required" />
2963 <xs:anyAttribute namespace="##other" processContents="lax" />
2964 </xs:complexType>
2965
2966 <xs:complexType name="RecordConstraintType">
2967 <xs:sequence>
2968 <xs:element name="propertyValueOperators"
2969 type="cmdbfmd:PropertyValueOperatorsType"
2970 minOccurs="0" maxOccurs="1" />
2971 </xs:sequence>
2972 <xs:attribute name="recordTypeConstraint"
2973 type="xs:boolean" use="required" />
2974 <xs:attribute name="propertyValueConstraint"
2975 type="xs:boolean" use="required" />
2976 <xs:anyAttribute namespace="##other" processContents="lax" />
2977 </xs:complexType>
2978
2979 <xs:complexType name="PropertyValueOperatorsType">
2980 <xs:attribute name="equal" type="xs:boolean" use="required" />
2981 <xs:attribute name="less" type="xs:boolean" use="required" />
2982 <xs:attribute name="lessOrEqual" type="xs:boolean" use="required"/>
2983 <xs:attribute name="greater" type="xs:boolean" use="required" />
2984 <xs:attribute name="greaterOrEqual"
2985 type="xs:boolean" use="required" />
2986 <xs:attribute name="contains" type="xs:boolean" use="required" />
2987 <xs:attribute name="like" type="xs:boolean" use="required" />
2988 <xs:attribute name="isNull" type="xs:boolean" use="required" />
2989 <xs:anyAttribute namespace="##other" processContents="lax" />
2990 </xs:complexType>
2991
2992 <xs:complexType name="XPathType">
2993 <xs:sequence>
2994 <xs:element name="dialect" type="xs:anyURI"
2995 minOccurs="0" maxOccurs="unbounded" />
2996 </xs:sequence>
2997 </xs:complexType>
2998
2999
3000 <!-- Registration Service metadata definition -->
3001 <xs:element name="registrationServiceMetadata">
3002 <xs:complexType>
3003 <xs:sequence>
3004 <xs:element ref="cmdbfmd:serviceDescription" />
3005 <xs:element ref="cmdbfmd:recordTypeList" />
3006 <xs:any minOccurs="0" maxOccurs="unbounded"

```

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

Committee Draft

## 3013 **Appendix C WSDL**

3014

3015 A normative copy of the WSDL [[WSDL 1.1](#)] description for this specification can be  
3016 retrieved from the following addresses:

3017 <http://cmdbf.org/schema/1-0-0/cmdbfQuery.wsdl>

3018 <http://cmdbf.org/schema/1-0-0/cmdbfRegistration.wsdl>

3019

3020 A non-normative copy of the WSDL descriptions are listed below for convenience.

### 3021 **Query Service WSDL**



```
3022 <?xml version="1.0" encoding="utf-8"?>
```

```
3023 <!--
```

```
3024 Copyright Notice
```

```
3025 (c) 2007 by BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and
3026 Microsoft. All rights reserved.
```

```
3027
```

```
3028 Permission to copy and display this portion of the CMDB Federation
3029 specification, in any medium without fee or royalty is hereby granted,
3030 provided that you include the following on ALL copies of the CMDB
3031 Federation specification, or portions thereof, that you make:
```

- ```
3032 1. A link or URL to the Specification at the website of one of the  
3033 authors.  
3034 2. The copyright notice as shown in the Specification.
```

```
3035
```

```
3036 BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and Microsoft  
3037 (collectively, the "Authors") each agree to grant you a royalty-free  
3038 license, under reasonable, non-discriminatory terms and conditions to  
3039 their respective patents that they deem necessary to implement the  
3040 Specification.
```

```
3041
```

```
3042 THE "CMDB FEDERATION" SPECIFICATION IS PROVIDED "AS IS,"  
3043 AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR  
3044 IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY,  
3045 FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT  
3046 THE CONTENTS OF THE "CMDB FEDERATION" SPECIFICATION ARE SUITABLE FOR  
3047 ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT  
3048 INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER  
3049 RIGHTS.
```

```
3050
```

```
3051 THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL,  
3052 INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY  
3053 USE OR DISTRIBUTION OF THE "CMDB FEDERATION" SPECIFICATION.
```

```
3054
```

```
3055 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"

```

3101         element="cmdbf:UnsupportedConstraintFault"/>
3102     </wsdl:message>
3103
3104     <wsdl:message name="UnsupportedSelectorFault">
3105         <wsdl:part name="Detail" element="cmdbf:UnsupportedSelectorFault"/>
3106     </wsdl:message>
3107
3108     <wsdl:message name="QueryErrorFault">
3109         <wsdl:part name="Detail" element="cmdbf:QueryErrorFault"/>
3110     </wsdl:message>
3111
3112     <wsdl:portType name="QueryPortType">
3113         <wsdl:operation name="GraphQL">
3114             <wsdl:input message="tns:QueryRequest" />
3115             <wsdl:output message="tns:QueryResponse" />
3116             <wsdl:fault name="UnkownTemplateID"
3117                 message="tns:UnkownTemplateIDFault"/>
3118             <wsdl:fault name="InvalidPropertyType"
3119                 message="tns:InvalidPropertyTypeFault"/>
3120             <wsdl:fault name="XPathError"
3121                 message="tns:XPathErrorFault"/>
3122             <wsdl:fault name="UnsupportedConstraint"
3123                 message="tns:UnsupportedConstraintFault"/>
3124             <wsdl:fault name="UnsupportedSelector"
3125                 message="tns:UnsupportedSelectorFault"/>
3126             <wsdl:fault name="QueryError"
3127                 message="tns:QueryErrorFault"/>
3128         </wsdl:operation>
3129     </wsdl:portType>
3130
3131 </wsdl:definitions>

```

Registration Service WSDL

```

3134 <?xml version='1.0' encoding='UTF-8' ?>
3135 <!--
3136     Copyright Notice
3137     (c) 2007 by BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and
3138     Microsoft. All rights reserved.
3139
3140     Permission to copy and display this portion of the CMDB Federation
3141     specification, in any medium without fee or royalty is hereby granted,
3142     provided that you include the following on ALL copies of the CMDB
3143     Federation specification, or portions thereof, that you make:
3144     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>


```

3190 </wsdl:types>
3191
3192 <wsdl:message name="RegisterRequest">
3193   <wsdl:part name="body" element="cmdbf:registerRequest" />
3194 </wsdl:message>
3195
3196 <wsdl:message name="RegisterResponse">
3197   <wsdl:part name="body" element="cmdbf:registerResponse" />
3198 </wsdl:message>
3199
3200 <wsdl:message name="DeregisterRequest">
3201   <wsdl:part name="body" element="cmdbf:deregisterRequest" />
3202 </wsdl:message>
3203
3204 <wsdl:message name="DeregisterResponse">
3205   <wsdl:part name="body" element="cmdbf:deregisterResponse" />
3206 </wsdl:message>
3207
3208 <wsdl:message name="InvalidRecordFault">
3209   <wsdl:part name="Detail" element="cmdbf:InvalidRecordFault"/>
3210 </wsdl:message>
3211
3212 <wsdl:message name="UnsupportedRecordTypeFault">
3213   <wsdl:part name="Detail"
3214     element="cmdbf:UnsupportedRecordTypeFault"/>
3215 </wsdl:message>
3216
3217 <wsdl:message name="InvalidMDRFault">
3218   <wsdl:part name="Detail" element="cmdbf:InvalidMDRFault"/>
3219 </wsdl:message>
3220
3221 <wsdl:message name="RegistrationErrorFault">
3222   <wsdl:part name="Detail" element="cmdbf:RegistrationErrorFault"/>
3223 </wsdl:message>
3224
3225 <wsdl:message name="DeregistrationErrorFault">
3226   <wsdl:part name="Detail" element="cmdbf:DeregistrationErrorFault"/>
3227 </wsdl:message>
3228
3229 <wsdl:portType name="RegistrationPortType">
3230   <wsdl:operation name="Register">
3231     <wsdl:input message="tns:RegisterRequest" />
3232     <wsdl:output message="tns:RegisterResponse" />
3233     <wsdl:fault name="InvalidRecordFault"
3234       message="tns:InvalidRecordFault"/>

```

```
3235     <wsdl:fault name="UnsupportedRecordTypeFault"
3236             message="tns:UnsupportedRecordTypeFault" />
3237     <wsdl:fault name="InvalidMDRFault"
3238             message="tns:InvalidMDRFault" />
3239     <wsdl:fault name="RegistrationErrorFault"
3240             message="tns:RegistrationErrorFault" />
3241 </wsdl:operation>
3242
3243 <wsdl:operation name="Deregister">
3244     <wsdl:input message="tns:DeregisterRequest" />
3245     <wsdl:output message="tns:DeregisterResponse" />
3246     <wsdl:fault name="InvalidMDRFault"
3247             message="tns:InvalidMDRFault" />
3248     <wsdl:fault name="DeregistrationErrorFault"
3249             message="tns:DeregistrationErrorFault" />
3250 </wsdl:operation>
3251 </wsdl:portType>
3252
3253 </wsdl:definitions>
3254
```

Committee

3255 **Appendix D Fault Binding to SOAP**

3256 Faults may be generated for any CMDBf operation. The bindings of faults for both
3257 SOAP 1.1 and SOAP 1.2 are described in this section.

3258 The definitions of faults use the following properties:

3259 [Code] The fault code.

3260 [Subcode] The fault subcode.

3261 [Reason] A language localized readable description of the error.

3262 [Detail] Optional detail element(s). If more than one detail element is defined for
3263 a fault, implementations MUST include the elements in the order that they are
3264 specified.

3265 Services that generate CMDBf faults MUST set the [Code] property to either "Sender"
3266 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

3267

3268 The properties above bind to a SOAP 1.2 fault as follows:

```
3269 <S:Envelope>
3270   <S:Header>
3271     <wsa:Action>
3272       http://cmdbf.org/schema/1-0-0/fault
3273     </wsa:Action>
3274     <!-- Headers elided for brevity. -->
3275   </S:Header>
3276   <S:Body>
3277     <S:Fault>
3278       <S:Code>
3279         <S:Value> [Code] </S:Value>
3280         <S:Subcode>
3281           <S:Value> [Subcode] </S:Value>
3282         </S:Subcode>
3283       </S:Code>
3284       <S:Reason>
3285         <S:Text xml:lang="en"> [Reason] </S:Text>
3286       </S:Reason>
3287       <S:Detail>
3288         [Detail]
3289         ...
3290       </S:Detail>
3291     </S:Fault>
3292   </S:Body>
3293 </S:Envelope>
```

3294 The properties bind to a SOAP 1.1 fault as follows when the fault is generated as a
3295 result of processing a CMDBf request message:

```
3296 <S11:Envelope>  
3297   <S11:Header>  
3298     <cmdbf:fault>  
3299       <cmdbf:faultCode> [Subcode] </cmdbf:faultCode>  
3300       <cmdbf:detail> [Detail] </cmdbf:detail>  
3301       ...  
3302     </cmdbf:fault>  
3303     <!-- Headers elided for brevity. -->  
3304   </S11:Header>  
3305   <S11:Body>  
3306     <S11:Fault>  
3307       <S11:faultcode> [Code] </S11:faultcode>  
3308       <S11:faultstring> [Reason] </S11:faultstring>  
3309     </S11:Fault>  
3310   </S11:Body>  
3311 </S11:Envelope>
```

3312
3313 When a binding to a CMDBf operation that supports WS-Addressing, the fault
3314 message MUST include the following action URI defined below as the [action]
3315 property.

3316
3317 <http://cmdbf.org/schema/1-0-0/fault>

3318
3319 Fault handling rules for operations using WS-Addressing are defined in section 6 of
3320 WS-Addressing SOAP Binding.

3321 **Appendix E Sample WSDL Binding (non-normative)**

3322 The following example illustrates how the interfaces defined in this specification
3323 should be described in a Web service binding that implements the interfaces. This
3324 example also illustrates how the CMDBf service metadata should be associated with
3325 a particular implementation of a CMDBf interface.

3326 As shown below, this query implementation uses SOAP 1.1 over HTTP as the protocol
3327 and supports the use of WS-Addressing if the message sender uses WS-Addressing
3328 for an asynchronous request/response. Since this specification does not define
3329 specific WS-Addressing actions, the action header values for WS-Addressing are
3330 determined according to the defaults described in the WS-Addressing 1.0 WSDL
3331 Binding specification at <http://www.w3.org/TR/ws-addr-wsdl/#defactionwsdl11>

3332 The queryServiceMetadata element is included in a WS-Policy expression which is
3333 included by reference in the WSDL binding to the query port type. This particular
3334 sample is of a query service that supports the complete set of record constraint and
3335 selector operators defined in the specification. The metadata in the sample also
3336 shows that XPath1 and XPath 2 are supported by the service.

3337 The metadata for the service also includes the two record types that may be queried
3338 at this service, a "R_ComputerSystem" data type, and a "CIM_CommonDatabase"
3339 data type.

3340 The approach to including metadata as a policy in the WSDL is a recommended
3341 approach to creating the WSDL documentation for the binding implementation as it
3342 allows for the file containing the WSDL binding to completely describe the interface
3343 to the service and the options allowed by this specification.

3344

```
3345 <?xml version='1.0' encoding='UTF-8' ?>
```

```
3346 <!--
```

```
3347 Copyright Notice
```

```
3348 (c) 2007 by BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and  
3349 Microsoft. All rights reserved.
```

```
3350
```

```
3351 Permission to copy and display this portion of the CMDB Federation  
3352 specification, in any medium without fee or royalty is hereby granted,  
3353 provided that you include the following on ALL copies of the CMDB  
3354 Federation specification, or portions thereof, that you make:
```

- ```
3355 1. A link or URL to the Specification at the website of one of the
3356 authors.
3357 2. The copyright notice as shown in the Specification.
```

```
3358
```

```
3359 BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, and Microsoft
3360 (collectively, the "Authors") each agree to grant you a royalty-free
3361 license, under reasonable, non-discriminatory terms and conditions to
3362 their respective patents that they deem necessary to implement the
3363 Specification.
```

```
3364
```

```
3365 THE "CMDB FEDERATION" SPECIFICATION IS PROVIDED "AS IS,"
```

```
3366 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">

```

3412 <queryServiceMetadata
3413 xmlns="http://cmdbf.org/schema/1-0-0/serviceMetadata">
3414 <serviceDescription>
3415 <mdrId>CMDBf12345</mdrId>
3416 </serviceDescription>
3417 <queryCapabilities>
3418 <contentSelectorSupport propertySelector="true"
3419 recordTypeSelector="true" />
3420 <recordConstraintSupport recordTypeConstraint="true"
3421 propertyValueConstraint="true">
3422 <propertyValueOperators equal="true" less="true"
3423 greater="true" lessOrEqual="true"
3424 greaterOrEqual="true"
3425 contains="true"
3426 like="false"
3427 isNull="false" />
3428 </recordConstraintSupport>
3429 <xpathSupport>
3430 <dialect>
3431 http://www.w3.org/TR/1999/REC-xpath-19991116
3432 </dialect>
3433 <dialect>
3434 http://www.w3.org/TR/2007/REC-xpath-20070123
3435 </dialect>
3436 </xpathSupport>
3437 </queryCapabilities>
3438
3439 <recordTypeList>
3440 <recordTypes namespace="http://cmdbf.org"
3441 schemaLocation="http://cmdbf.org/common_schemas/R_ComputerSystem.xsd">
3442 <recordType localName="R_ComputerSystem" />
3443 </recordTypes>
3444 <recordTypes
3445 namespace="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_CommonDatabase"
3446 schemaLocation="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_CommonDatabase.xsd">
3447 <recordType localName="CIM_CommonDatabase" />
3448 </recordTypes>
3449 </recordTypeList>
3450
3451 </queryServiceMetadata>
3452 </wsp:Policy>
3453
3454 <!-- Sample Binding for SOAP 1.1 with WS-Addressing support
3455 -->
3456 <wsdl:binding name="QueryBinding" type="cmdbfPort:QueryPortType">
3457 <soap:binding style="document"
3458 transport="http://schemas.xmlsoap.org/soap/http" />

```

```
3459 <wsp:PolicyReference URI="SupportsWSAddressing" />
3460 <wsp:PolicyReference URI="SupportedMetadata" />
3461 <wsdl:operation name="GraphQL">
3462 <wsdl:input>
3463 <soap:body use="literal" />
3464 </wsdl:input>
3465 <wsdl:output>
3466 <soap:body use="literal" />
3467 </wsdl:output>
3468 <wsdl:fault name="UnkownTemplateID">
3469 <soap:fault name="UnkownTemplateID" use="literal" />
3470 </wsdl:fault>
3471 <wsdl:fault name="InvalidPropertyType">
3472 <soap:fault name="InvalidPropertyType" use="literal" />
3473 </wsdl:fault>
3474 <wsdl:fault name="XPathError">
3475 <soap:fault name="XPathError" use="literal" />
3476 </wsdl:fault>
3477 <wsdl:fault name="UnsupportedConstraint">
3478 <soap:fault name="UnsupportedConstraint" use="literal" />
3479 </wsdl:fault>
3480 <wsdl:fault name="UnsupportedSelector">
3481 <soap:fault name="UnsupportedSelector" use="literal" />
3482 </wsdl:fault>
3483 <wsdl:fault name="QueryError">
3484 <soap:fault name="QueryError" use="literal" />
3485 </wsdl:fault>
3486 </wsdl:operation>
3487 </wsdl:binding>
3488
3489 </wsdl:definitions>
3490
3491
```





3492  
3493  
3494  
3495

## 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<br>04 January 2008 | Mark Johnson | <p>Line 3: Updated the version suffix and date</p> <p>Lines 895-897: Italicized to indicate that these are <i>types</i> of elements</p> <p>Line 2047-2048: Replaced &lt;queryServiceMetadata&gt; with &lt;registerServiceMetadata&gt; and deleted an extraneous reference to query capabilities</p> <p>Line 2250: Corrected the spelling of cmdbfServiceMetadata.xsd</p> <p>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.</p> <p>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.</p> <p>Lines 2576-2577: Added the missing recordMetadata attribute. This change is also reflected in the associated xsd file.</p> <p>Line 2776: Removed an extraneous blank space before the trailing double quote. This change is also reflected in the associated xsd file.</p> <p>Lines 3492+: Added Appendix F Editorial Corrections</p> |
|                                 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|                                 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|                                 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

3496  
3497