RESTful Network API for Capability Discovery

Candidate Version 1.0 – 01 Jul 2013

Open Mobile Alliance

OMA-TS-REST_NetAPI_CapabilityDiscovery-V1_0-20130701-C

Contents

Figures

Tables

1. Scope

This specification defines a RESTful API for Capability Discovery using HTTP protocol bindings.

2. References

2.1 Normative References

[IETF_ACR_draft] "The acr URI for anonymous users", S.Jakobsson, K.Smith, January 2010, URL:http://tools.ietf.org/html/draft-uri-acr-extension-04

[REST_NetAPI_Common] "Common definitions for RESTful Network APIs", Open Mobile Alliance™, OMA-TS-REST_NetAPI_Common-V1_0, URL: http://www.openmobilealliance.org/

[REST_SUP_CapabilityDiscovery] "XML schema for the RESTful Network API for Capability Discovery", Open Mobile Alliance™, OMA-SUP-XSD_rest_netapi_capabilitydiscovery-V1_0, URL: http://www.openmobilealliance.org/

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC2616] "Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding et. al, January 1999, URL:http://www.ietf.org/rfc/rfc2616.txt

[RFC3966] "The tel URI for Telephone Numbers", H.Schulzrinne, December 2004, URL:http://www.ietf.org/rfc/rfc3966.txt

[RFC3986] "Uniform Resource Identifier (URI): Generic Syntax", R. Fielding et. al, January 2005, URL:http://www.ietf.org/rfc/rfc3986.txt

[RFC4627] "The application/json Media Type for JavaScript Object Notation (JSON)", D. Crockford, July 2006, URL: http://www.ietf.org/rfc/rfc4627.txt

[SCRRULES] "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL: http://www.openmobilealliance.org/

[XMLSchema1] W3C Recommendation, XML Schema Part 1: Structures Second Edition, URL:http://www.w3.org/TR/xmlschema-1/

[XMLSchema2] W3C Recommendation, XML Schema Part 2: Datatypes Second Edition, URL:http://www.w3.org/TR/xmlschema-2/

2.2 Informative References

[OMADICT] "Dictionary for OMA Specifications", Version 2.9, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, URL:http://www.openmobilealliance.org/

[REST_WP] "Guidelines for RESTful Network APIs", Open Mobile Alliance™, OMA-WP-Guidelines_for_RESTful_Network_APIs, URL:http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

For the purpose of this TS, all definitions from the OMA Dictionary apply [OMADICT].

Capability Source	An entity that on behalf of a user is managing user’s service capabilities which are valid only a certain time unless they are refreshed. In the context of this specification a Capability Source refers to an instance of service capabilities of the user on the server, which are registered from a particular user application instance (device).
Heavy-weight Resource	A resource which is identified by a resource URL which is then used by HTTP methods to operate on the entire data structure representing the resource
Light-weight Resource	A subordinate resource of a Heavy-weight Resource which is identified by its own resource URL which is then used by HTTP methods to operate on a part of the data structure representing the Heavy-weight Resource. The Light-weight Resource URL can be seen as an extension of the Heavy-weight Resource URL. There could be several levels of Light-weight Resources below the ancestor Heavy-weight Resource, depending on the data structure

3.3 Abbreviations

ACR	Anonymous Customer Reference
API	Application Programming Interface
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
OMA	Open Mobile Alliance
REST	REpresentational State Transfer
SIP	Session Initation Protocol
SCR	Static Conformance Requirements
TS	Technical Specification
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	eXtensible Markup Language
XSD	XML Schema Definition

4. Introduction

The Technical Specification of the RESTful Network API for Capability Discovery contains HTTP protocol bindings for Capability Discovery, using the REST architectural style. The specification provides resource definitions, the HTTP verbs applicable for each of these resources, and the element data structures, as well as support material including flow diagrams and examples using the various supported message body formats (i.e. XML and JSON).

4.1 Version 1.0

Version 1.0 of this specification supports the following operations:

- Register/de-register own service capabilities
- Enable/disable registered own capabilities
- Retrieve service capabilities of a contact
- Discover what type of user is a contact (e.g. RCSe user or not)

In addition, this specification provides:

- Support for scope values used with authorization framework defined in [Autho4API_10]
- Support for Anonymous Customer Reference (ACR) as an end user identifier
- Support for “acr:auth” as a reserved keyword in a resource URL variable that identifies an end user

5. Capability Discovery API definition

This section is organized to support a comprehensive understanding of the Capability Discovery API design. It specifies the definition of all resources, definition of all data structures, and definitions of all operations permitted on the specified resources.

Common data types, naming conventions, fault definitions and namespaces are defined in [REST_NetAPI_Common].

The remainder of this document is structured as follows:

Section **Error! Reference source not found.** starts with a diagram representing the resources hierarchy followed by a table listing all the resources (and their URL) used by this API, along with the data structure and the supported HTTP verbs (section 5.1). What follows are the data structures (section 5.2). A sample of typical use cases is included in section 5.3,described as high level flow diagrams.

Section 6 contains detailed specification for each of the resources. Each such subsection defines the resource, the request URL variables that are common for all HTTP commands, and the supported HTTP verbs. For each supported HTTP verb, a description of the functionality is provided, along with an example of a request and an example of a response. For each unsupported HTTP verb, the returned HTTP error status is specified, as well as what should be returned in the Allow header.

All examples in section 6 use XML as the format for the message body, while JSON examples are provided in Appendix D..

Section 7 contains fault definition details such as Service Exceptions and Policy Exceptions.

Appendix B provides the Static Conformance Requirements (SCR).

Appendix C provides application/x-www-form-urlencoded examples, where applicable.

Appendix E provides the operations mapping to a pre-existing baseline specification, where applicable.

Appendix F provides a list of all Light-Weight resources, where applicable.

Appendix G defines authorization aspects to control access to the resources defined in this specification.

Appendix H defines strings identifying the most common service capabilities, and mappings of the strings to their respective SIP OPTIONS tags or Presence service tuples.

Note: Throughout this document client and application can be used interchangeably.

5.1 Resources Summary

This section summarizes all the resources used by the RESTful Network API for Capability Discovery.

The "apiVersion" URL variable SHALL have the value "v1" to indicate that the API corresponds to this version of the specification. See [REST_NetAPI_Common] which specifies the semantics of this variable.

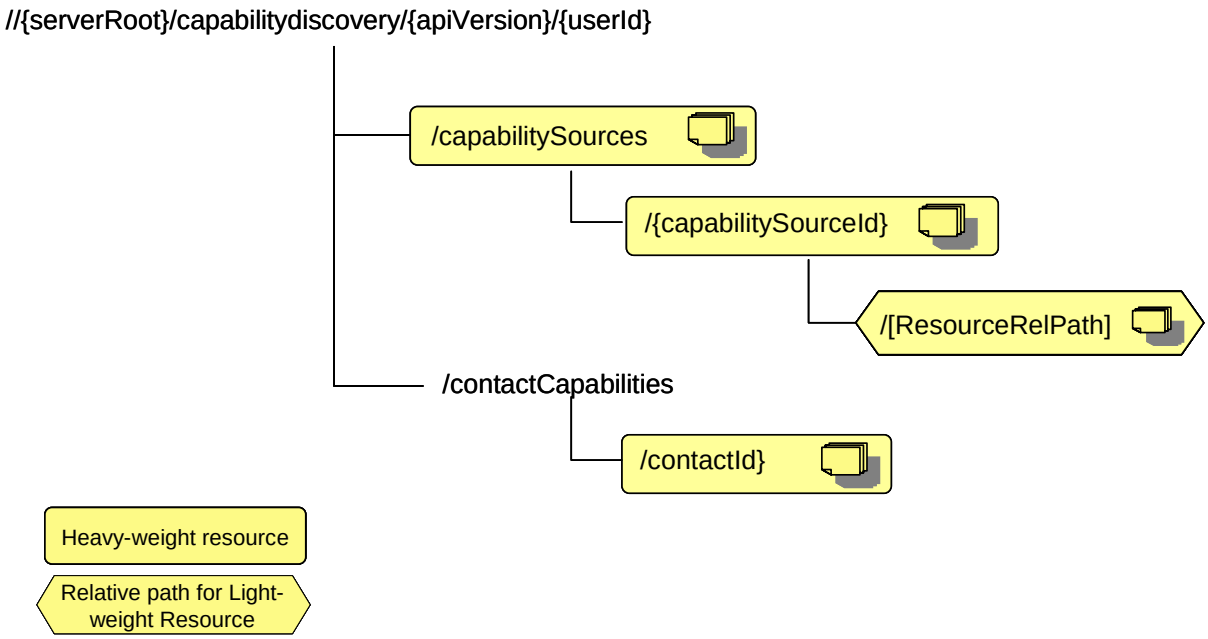

Figure 1 Resource structure defined by this specification

The following tables give a detailed overview of the resources defined in this specification, the data type of their representation and the allowed HTTP methods.

Purpose: To allow application to manage own service capabilities

Resource	URL Base URL: http://{serverRoot}/capabilitydiscovery/{apiVersion}	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
Service Capability Sources	/{userId}/capabilitySources	CapabilitySourceList (used for GET) CapabilitySource (used for POST) common:ResourceReference (OPTIONAL alternative for POST response)	Retrieves all own registered service capabilities with a status Note: Query string parameter can be used to select either enabled or disabled capabilities.	no	Defines (registers) a new own service Capability Source (i.e. registering service capabilities for a new device)	
Individual service Capability Source	/{userId}/capabilitySources/{capabilitySourceId}	CapabilitySource (used for GET and PUT)	Retrieves all own service capabilities with a status for a specified service Capability Source	Updates capability information for a specified Capability Source (i.e. add/remove service capabilities, or enable/disable registered service capabilities) Note: When removing the last service capability registered for a particular Capability	no	Removes (deregisters) all own service capabilities registered for a specified service Capability Source Note that after the completion of this operation the Capability Source will be removed.

Resource	URL Base URL: http://{serverRoot}/capabilitydiscovery/{apiVersion}	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
				Source, it is up to the server policy whether the Capability Source will be removed or not.		
Individual service capability data	/{userId}/capabilitySources/{capabilitySourceId}/[ResourceRelPath]	The data structure corresponds to an element within the CapabilitySource structure pointed out by the resource URL. (used for PUT/GET)	Retrieves individual own service capability information registered for a specified service Capability Source	Creates or updates capability information for a specified service capability (i.e. add new service capability, or enable/disable registered service capability) Note: Update/refresh of duration time for a specified Capability Source is possible if such option is supported by service provider)	no	Removes (deregisters) specified service capability registered for a specified service Capability Source Note: When removing the last service capability registered for a particular Capability Source, it is up to the server policy whether the Capability Source will be removed or not.

Purpose: To allow application to retrieve service capabilities of a contact

Resource	URL Base URL: http://{serverRoot}/capabilitydiscovery/{apiVers	Data Structures	HTTP verbs

	ion}		GET	PUT	POST	DELETE
Service capabilities for a contact	/{userId}/contactCapabilities/{contactId}	ContactServiceCapabilities	Retrieves list of service capabilities defined for a specified contact (response includes only enabled capabilities) Note: Query string parameters can be used to filter query request, e.g. to query for a specific capability, or to check what type of a user is a contact (e.g. whether the contact is an RCSe user or not)	no	no	no

5.2 Data Types

5.2.1 XML Namespaces

The XML namespace for the Capability Discovery data types is:

urn:oma:xml:rest:netapi:capabilitydiscovery:1

The 'xsd' namespace prefix is used in the present document to refer to the XML Schema data types defined in XML Schema [XMLSchema1, XMLSchema2]. The 'common' namespace prefix is used in the present document to refer to the data types defined in [REST_NetAPI_Common]. The use of namespace prefixes such as 'xsd' is not semantically significant.

The XML schema for the data structures defined in the section below is given in [REST_SUP_CapabilityDiscovery].

5.2.2 Structures

The subsections of this section define the data structures used in the Capability Discovery API.

Some of the structures can be instantiated as so-called root elements, i.e. they define the type of a representation of a so-called Heavy-weight Resource.

The column [ResourceRelPath] in the tables below, if used, includes relative resource paths for Light-weight Resource URLs that are used to access individual elements in the data structure (so-called Light-weight Resources). A string from this column needs to be appended to the corresponding Heavy-weight Resource URL in order to create Light-weight Resource URL for that particular element in the data structure. “Not applicable” means that individual access to that element is not supported. The root element and data type of the resource associated with the [ResourceRelPath] are defined by the Element and Type columns in the row that defines the [ResourceRelPath].

For structures that contain elements which describe a user identifier, the statements in section 6 regarding 'tel', 'sip' and 'acr' URI schemes apply.

5.2.2.1 Type: CapabilitySourceList

This type represents a list of own service Capability Sources.

Element	Type	Optional	Description
capabilitySource	CapabilitySource [0…unbounded]	Yes	A list of Capability Sources
resourceURL	xsd:anyURI	No	Self referring URL

A root element named capabilitySourceList of type CapabilitySourceList is allowed in response bodies.

5.2.2.2 Type: CapabilitySource

This type represents a list of own service capabilities for a particular Capability Source.

Element	Type	Optional	[ResourceRelPath]	Description
serviceCapability	ServiceCapability [0…unbounded]	Yes	{capabilityId}	Array of service capabilities. Sub-element ‘capabilityId’ of the type ServiceCapability is a key property of element serviceCapability and SHALL NOT be altered when accessed as Light-weight Resource.
clientCorrelator	xsd:string	Yes	Not applicable	A correlator that the client can use to tag this particular resource representation

				during a request to create a resource on the server. This element SHOULD be present. Note: this allows the client to recover from communication failures during resource creation and therefore avoids duplicate Capability Source creation in such situations. In case the element is present, the server SHALL not alter its value, and SHALL provide it as part of the representation of this resource. In case the field is not present, the server SHALL NOT generate it.
applicationTag	xsd:string	Yes	Not applicable	A tag that the client MAY use to tag this particular resource on the server. In case the field is present, the server SHALL not alter its value, and SHALL provide it as part of the representation of this resource. In case the field is not present, the server SHALL NOT generate it.
duration	xsd:int	Yes	duration	Specifies the duration of the service Capability Source lifetime in seconds. When this time has elapsed the Capability Source will expire unless it has been refreshed. If the parameter is omitted during Capability Source creation, a default value specified by the server policy will be used. A too low value (including “0”) will result in an error response. What is too low is defined by server policy. A too high requested value may be reduced by the server according to the service policy. In any case the server SHOULD inform the client about the agreed duration time in the response to the resource creation. If the parameter is not included in the response to the resource creation, it is assumed that the duration time for a service capability would be as long as the Capability Source exists**.**

<any element>	< type is defined by the schema which implements the element> [0…unbounded]	Yes	Not applicable	Optional element that can be used to specify additional information relating to a particular Capability Source. It can be any element from any other namespace (schema) than the target namespace. The type of such element is defined by the schema implementing the element. In XML implementations, the element must be qualified with the namespace prefix.
resourceURL	xsd:anyURI	Yes	Not applicable	Self referring URL. The resourceURL SHALL NOT be included in POST requests by the client, but MUST be included in POST requests representing notifications by the server to the client, when a complete representation of the resource is embedded in the notification. The resourceURL MUST also be included in responses to any HTTP method that returns an entity body, and in PUT requests.

A root element named capabilitySource of type CapabilitySource is allowed in request and/or response bodies.

Note that the clientCorrelator is used for purposes of error recovery as specified in [REST_NetAPI_Common], and internal client purposes. The server is NOT REQUIRED to use the clientCorrelator value in any form in the creation of the URL of the resource. [REST_NetAPI_Common] provides a recommendation regarding the generation of the value of this field.

Note that applicationTag is used to enable a particular application instance to pick up a previously created resource (if it exists) and continue to operate on it. A typical usage is that a client will perform a GET on the parent resource and in the response receive a list of previously created resources from where the application is able to find its previously created resource. It is up to the client application how to construct the application tag. Please note that a typical usage of the client correlator is not enough for a stateless application to identify a previously created resource since it is uniquely generated every time a new resource is created.

5.2.2.3 Type: ServiceCapability

This type represents an individual service capability

Element	Type	Optional	[ResourceRelPath]	Description
capabilityId	xsd:token	No	Not applicable	A string identifying a particular service capability (e.g. Chat). SHALL NOT be altered when included in Light-weight Resource URL If capabilityId is also part of the request URL, the two MUST have the same value. **Error! Reference source not found.** defines the strings that are used as identifiers for the most common service capabilities.
status	CapabilityStatus	Yes	{capabilityId}/status	Describes the status of a service capability. During the registration of service capability,

				the application MAY specify the desired initial status of the service capability however it is up to the server policy to accept it or not. If not specified, default status will be “Disabled”. In any case the server SHALL include the accepted capability status in the response to service capability registration.
<any element>	< type is defined by the schema which implements the element> [0…unbounded]	Yes	Not applicable	Optional element that can be used to specify additional information about a service capability. It can be any element from any other namespace (schema) than the target namespace. The type of such element is defined by the schema implementing the element. In XML implementations, the element must be qualified with the namespace prefix.

5.2.2.4 Type: ContactServiceCapabilities

This type represents service capabilities for a contact.

Element	Type	Optional	Description
serviceCapability	ServiceCapability [0…unbounded]	Yes	List of service capabilities for a contact. Not included in response to queries which includes a filter for a user type only.
userType	UserType [0…unbounded]	Yes	Indicates what type of a user is a contact (e.g. RCSe user). Not included in response to queries which include a filter for a service capability only.
resourceURL	xsd:anyURI	No	Self referring URL

A root element named contactServiceCapabilities of type ContactServiceCapabilities is allowed in response bodies.

5.2.3 Enumerations

The subsections of this section define the enumerations used in the Capability Discovery API.

5.2.3.1 Enumeration: CapabilityStatus

This enumeration defines possible values to describe the status of a particular service capability.

Enumeration	Description
Enabled	Indicates that a service capability is visible (discoverable) to other users.
Disabled	Indicates that a service capability SHALL NOT be visible to other users.

5.2.3.2 Enumeration: UserType

This enumeration defines possible values to describe the type of a user.

Enumeration	Description

RCS	Indicates an RCS user.
RCSe	Indicates an RCSe user.

5.2.4 Values of the Link "rel" attribute

The "rel" attribute of the Link element is a free string set by the server implementation, to indicate a relationship between the current resource and an external resource. The following are possible strings (list is non-exhaustive, and can be extended):

- CapabilitySourceList
- CapabilitySource
- ContactServiceCapabilities

These values indicate the kind of resource that the link points to.

5.3 Sequence Diagrams

The following subsections describe the resources, methods and steps involved in typical scenarios.

5.3.1 Retrieving and managing own service capabilities

This figure below shows a scenario for retrieving and managing own service capabilities.

The resources:

- To retrieve a list and status of all own service capabilities (both enabled and disabled) , read resource under

  http://{serverRoot}/capabilitydiscovery/{apiVersion}/{userId}/capabilitiySources

  Note: a query string parameter can be used to filter query request and query for a specific capability status, e.g. for enabled, or disabled capabilities only.

- To define (register) a new Capability Source (with its service capabilities), create resource under

  http://{serverRoot}/capabilitydiscovery/{apiVersion}/{userId}/capabilitySources

- To retrieve all service capabilities for a particular Capability Source, read resource under

  http://{serverRoot}/capabilitydiscovery/{apiVersion}/{userId}/capabilitySources/{capabilitySourceId}

- To update created Capability Source (e.g. add/remove service capabilities or enable/disable service capabilities), update resource under

  http://{serverRoot}/capabilitydiscovery/{apiVersion}/{userId}/capabilitySources/{capabilitySourceId}

- To remove (deregister) Capbility Source (including all its service capabilities), delete resource under

  http://{serverRoot}/capabilitydiscovery/{apiVersion}/{userId}/capabilitySources/{capabilitySourceId}

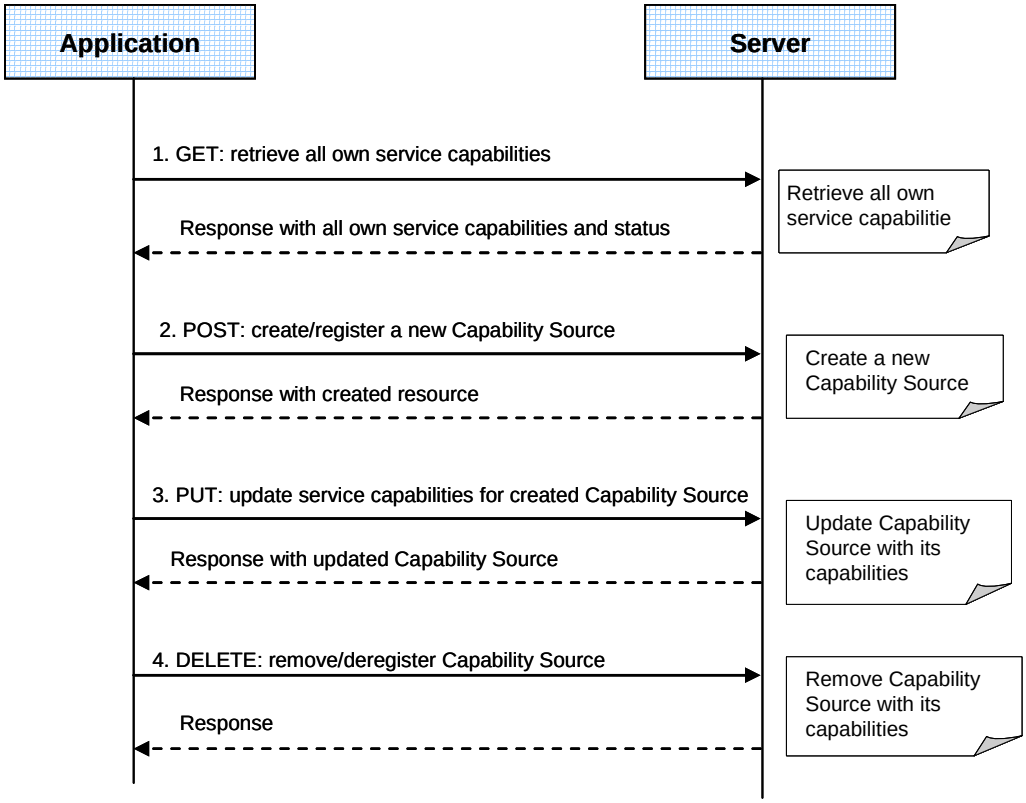

Figure 2 Management of own service capabilities

1. An application on behalf of a user requests a list and status of all own service capabilities by using GET method and receives the list of service capabilities (both disabled and enabled) from all registered Capability Sources for that particular user.
2. An application creates (defines, registers) a new Capability Source (with its service capabilities) for itself by using POST method and receives the resulting resource URL containing the Capability Source Id.
3. An application updates service capabilities for a particular Capability Source (e.g. add/removes service capabilities, or enables/disables service capabilities) by using PUT method and receives response with updated Capability Source.
4. An application deletes (deregisters) a Capability Source with its service capabilities by using DELETE method and receives response with the result of operation. Note that before performing this operation, the application SHOULD ensure that all service capabilities for that particular Capability Source are disabled.

5.3.2 Handling of individual service capability information with Light-weight Resources

This section describes an alternative method for handling of individual own service capability information by using Light-weight Resources.

The resources:

- To update (refresh) duration (lifetime) for a registered Capability Source (subject to service provider policy) the following resource is used:

  http://{serverRoot}/capabilitydiscovery/{apiVersion}/{userId}/capabilitySources/{capabilitySourceId}/[ResourceRelPah]

Where [ResourceRelPath] is a light-weight relative resource URL which shall be replaced with a string “duration” (see column [ResourceRelPath] in data types in section 5.2.2.2).

- For handling an individual service capability for a particular Capability Source the following resource is used:

  http://{serverRoot}/capabilitydiscovery/{apiVersion}/{userId}/capabilitySources/{capabilitySourceId}/[ResourceRelPah]

  Where [ResourceRelPath] is a light-weight relative resource URL which shall be replaced with an appropriate string from the column [ResourceRelPath] for data types in section 5.2.2, as follows:

  - To retrieve data for an individual service capability, replace [ResourceRelPath] with an appropriate capability Id for that particular capability (see variable “{capabilityId}” in section 5.2.2.2).
  - To update status (Enable or Disable) for an individual service capability, replace [ResourceRelPath] with “{capabilityId}/status”, where variable “{capabilityId}” is an identifier of the capability and shall be replaced with a real capability Id.
  - To add or remove an individual service capability, replace [ResourceRelPath] with an appropriate capability Id for that particular capability (see variable “{capabilityId}” in section 5.2.2.2).

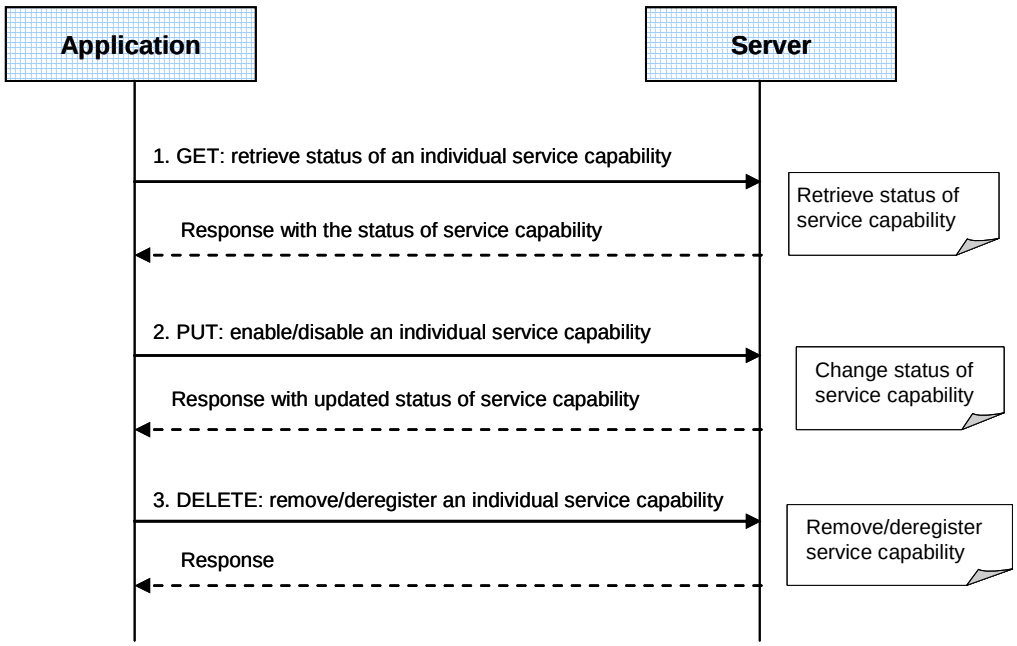

Figure 3 Management of an individual own service capability

1. An application retrieves an individual own service capability by using GET on the Light-weight Resource and receives capability Id along with the status of the capability.
2. The application enables/disables an individual own service capability using PUT on the Light-weight Resource and receives capability Id along with updated status of the capability.
3. The application removes (deregisters) an individual own service capability.

5.3.3 Retrieving service capabilities for a contact

This figure below shows a scenario for retrieving service capabilities registered for a contact (response includes only enabled capabilities)

The resource:

- To retrieve service capabilities for a specified contact, read the resource below with "{contactId}" identifying the targeted contact.
- Note: a query string parameters can be used to filter query request, e.g. to query for a specific capability, or to verify the user type for a contact (e.g. if a contact is an RCSe user).

http://{serverRoot}/capabilitydiscovery/{apiVersion}/{userId}/contactCapabilities/{contactId}

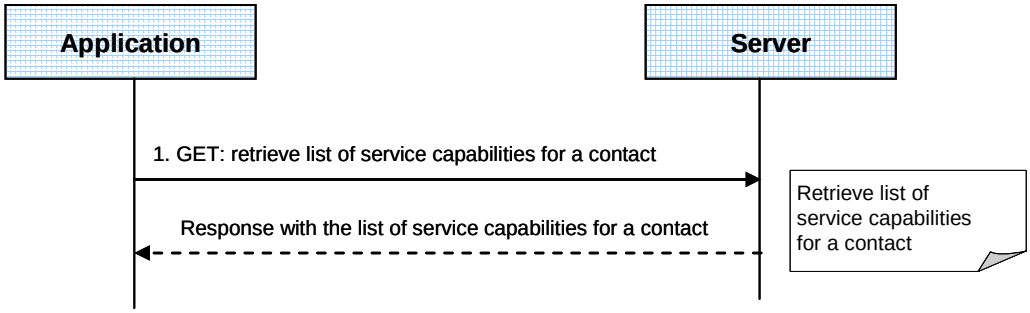

Figure 4 Retrieving service capabilities for a contact

Outline of flow:

1. An application requests service capabilities for a contact identified by "{contactId}" using GET and receives the list of service capabilities (capability Ids) enabled for a contact. To retrieve information on whether the contact has a specific service capability or to verify the user type for a contact (e.g. if the contact is an RCSe user), the application can filter the request by using query string parameters.

6. Detailed specification of the resources

The following applies to all resources defined in this specification regardless of the representation format (i.e. XML and JSON):

- Reserved characters in URL variables (parts of a URL denoted below by a name in curly brackets) MUST be percent-encoded according to [RFC3986]. Note that this always applies, no matter whether the URL is used as a Request URL or inside the representation of a resource (such as in "resourceURL" and "link" elements).
- If a user identifier (e.g. address, userId, etc) of type anyURI is in the form of an MSISDN, it MUST be defined as a global number according to [RFC3966] (e.g. tel:+19585550100) and the use of characters other than digits SHOULD be avoided in order to ensure uniqueness of the resource URL. This applies regardless of whether the user identifier appears in a URL variable or in a parameter in the body of an HTTP message.
- If a user identifier (e.g. address, userId, etc) of type anyURI is in the form of a SIP URI, it MUST be defined according to [RFC3261].
- If a user identifier (e.g. address, userId, etc) of type anyURI is in the form of an Anonymous Customer Reference (ACR), it MUST be defined according to [IETF_ACR_draft], i.e. it MUST include the protocol prefix 'acr:' followed by the ACR.
    - The ACR 'auth' is a supported reserved keyword, and MUST NOT be assigned as an ACR to any particular end user. See G.1.2 for details regarding the use of this reserved keyword.
- For requests and responses that have a body, the following applies: in the requests received, the server SHALL support JSON and XML encoding of the parameters in the body. The Server SHALL return either JSON or XML encoded parameters in the response body, according to the result of the content type negotiation as specified in [REST_NetAPI_Common]. In notifications to the Client, the server SHALL use either XML or JSON encoding, depending on which format the client has specified in the related subscription. The generation and handling of the JSON representations SHALL follow the rules for JSON encoding in HTTP Requests/Responses as specified in [REST_NetAPI_Common].

6.1 Resource: Service Capability Sources

The resource used is:

http://{serverRoot}/capabilitydiscovery/{apiVersion}/{userId}/capabilitySources

This resource is used by a client to retrieve registered own service capabilities from all Capability Sources as well as to create a new Capability Source.

6.1.1 Request URL variables

The following request URL variables are common for all HTTP commands:

Name	Description
serverRoot	Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.1.
userId	Identifier of the user on whose behalf the application acts Examples: tel:+19585550100, acr:pseudonym123

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.1.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Capability Discovery, see section 7.

6.1.3 GET

This operation is used for retrieval of a list with a status of own service capabilities.

Supported parameters in the query string of the Request URL are:

Name	Type/Values	Optional	Description
statusFilter	xsd:string	Yes	Defines the level of information that shall be returned in the body of the GET response. If statusFilter is absent, GET response body SHALL include both enabled and disabled service capabilities. If statusFilter=Enabled, GET response body SHALL include only enabled service capabilities. If statusFilter=Disabled, GET response body SHALL include only disabled service capabilities.

6.1.3.1 Example 1: Retrieve a list with status of all own service capabilities (Informative)

6.1.3.1.1 Request

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources HTTP/1.1
Host: example.com
Accept: application/xml
```

6.1.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySourceList xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
  <capabilitySource>
    <serviceCapability>
      <capabilityId>Chat</capabilityId>
      <status>Disabled</status>
    </serviceCapability>
    <clientCorrelator>123</clientCorrelator>
    <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001</resourceURL>
  </capabilitySource>
 <capabilitySource>
  <serviceCapability>
    <capabilityId>ImageShare</capabilityId>
    <status>Enabled</status>
    </serviceCapability>
    <serviceCapability>
    <capabilityId>FileTransfer</capabilityId>
```



```
    <status>Disabled</status>
   </serviceCapability>
   <clientCorrelator>1234</clientCorrelator>
   <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource002</resourceURL>
  </capabilitySource>
  <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources</resourceURL>
</cd:capabilitySourceList>
```

6.1.3.2 Example 2: Retrieve a list with status of all own service capabilities using a filter (Informative)

6.1.3.2.1 Request

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources?statusFilter=”Enabled” HTTP/1.1
Host: example.com
Accept: application/xml
```

6.1.3.2.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySourceList xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
  <capabilitySource>
   <serviceCapability>
     <capabilityId>ImageShare</capabilityId>
     <status>Enabled</status>
   </serviceCapability>
   <clientCorrelator>1234</clientCorrelator>
   <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource002</resourceURL>
 </capabilitySource>
 <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources</resourceURL>
</cd:capabilitySourceList>
```

6.1.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow:GET, POST’ field in the response as per section 14.7 of [RFC 2616].

6.1.5 POST

This operation is used to create a new service Capability Source.

6.1.5.1 Example 1: Create a new service Capability Source using ‘tel’ URI, response with a copy of created resource (Informative)

6.1.5.1.1 Request

```
POST /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources HTTP/1.1
Host: example.com
```

```
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySource xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
   <serviceCapability>
     <capabilityId>VideoShareDuringACall</capabilityId>
     </serviceCapability>
     <clientCorrelator>12345</clientCorrelator>
</cd:capabilitySource>
```

6.1.5.1.2 Response

```
HTTP/1.1 201 Created
Location: http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003 HTTP/1.1
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySource xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
   <serviceCapability>
     <capabilityId>VideoShareDuringACall</capabilityId>
     <status>Disabled</status>
   </serviceCapability>
   <clientCorrelator>12345</clientCorrelator>
   <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003</resourceURL>
</cd:capabilitySource>
```

6.1.5.2 Example 2: Create a new service Capability Source using ‘acr’ URI, response with a copy of created resource (Informative)

6.1.5.2.1 Request

```
POST /exampleAPI/capabilitydiscovery/v1/acr%3Apseudonym123/capabilitySources HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySource xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
   <serviceCapability>
     <capabilityId>VideoShareDuringACall</capabilityId>
     </serviceCapability>
     <clientCorrelator>12345</clientCorrelator>
</cd:capabilitySource>
```

6.1.5.2.2 Response

```
HTTP/1.1 201 Created
Location: http://exampleAPI/capabilitydiscovery/v1/acr%3Apseudonym123/capabilitySources/capsource003 HTTP/1.1
Date: Thu, 04 Jun 2012 02:51:59 GMT
```



```
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySource xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
   <serviceCapability>
     <capabilityId>VideoShareDuringACall</capabilityId>
     <status>Disabled</status>
   </serviceCapability>
   <clientCorrelator>12345</clientCorrelator>
   <resourceURL>http://exampleAPI/capabilitydiscovery/v1/acr%3Apseudonym123/capabilitySources/capsource003</resourceURL>
</cd:capabilitySource>
```

6.1.5.3 Example 3: Create a new service Capability Source, response with a location of created resource (Informative)

6.1.5.3.1 Request

```
POST /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySource xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
   <serviceCapability>
     <capabilityId>VideoShareDuringACall</capabilityId>
     </serviceCapability>
     <clientCorrelator>12345</clientCorrelator>
</cd:capabilitySource>
```

6.1.5.3.2 Response

```
HTTP/1.1 201 Created
Location: http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003 HTTP/1.1
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:resourceReference xmlns:common="urn:oma:xml:rest:netapi:common:1">
<resourceURL>http://example.com/exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003</reso
urceURL>
</common:resourceReference>
```

6.1.5.4 Example 4: Creating a new Capability Source fails (Informative)

6.1.5.4.1 Request

```
POST /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
```

```
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySource xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
   <serviceCapability>
     <capabilityId>VideoShareDuringACall</capabilityId>
   </serviceCapability>
   <serviceCapability>
     <capabilityId>Chat</capabilityId>
   </serviceCapability>
   <clientCorrelator>12345</clientCorrelator>
</cd:capabilitySource>
```

6.1.5.4.2 Response

```
HTTP/1.1 403 Forbidden
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <link rel="CapabilitySourceList"
    href="http://example.com/exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources"/>
  <policyException>
    <messageId>POL1021</messageId>
    <text>Maximum number of registered Capability Sources is exceeded</text>
  </policyException>
</common:requestError>
```

6.1.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow:GET, POST' field in the response as per section 14.7 of [RFC 2616].

6.2 Resource: Individual service Capability Source

The resource used is:

http://{serverRoot}/capabilitydiscovery/{apiVersion}/{userId}/capabilitySources/{capabilitySourceId}

This resource is used to manage an individual service Capability Source (e.g. retrieve service capabilities, add/remove service capabilities, or enable/disable service capabilities for that particular Capability Source)

6.2.1 Request URL variables

The following request URL variables are common for all HTTP commands:

Name	Description
serverRoot	Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.1.
userId	Identifier of the user on whose behalf the application acts

	Examples: tel:+19585550100, acr:pseudonym123
capabilitySourceId	Identifier of the service Capability Source (e.g. application/device) which is created by the server during the resource creation.

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.2.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Capability Discovery, see section 7.

6.2.3 GET

This operation is used for retrieval of all own service capabilities registered for a particular service Capability Source.

6.2.3.1 Example 1: Retrieve service capabilities registered for a particular Capability Source (Informative)

6.2.3.1.1 Request

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitiySources/capsource001 HTTP/1.1
Host: example.com
Accept: application/xml
```

6.2.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySource xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
    <serviceCapability>
      <capabilityId>Chat</capabilityId>
      <status>Disabled</status>
    </serviceCapability>
    <clientCorrelator>123</clientCorrelator>
    <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001</resourceURL>
  </cd:capabilitySource>
```

6.2.3.2 Example 2: Retrieving service capabilities for non-existent Capability Source (Informative)

This example illustrates the case when trying to retrieve service capabilities for Capability Source that has not been defined yet (non-existent).

6.2.3.2.1 Request

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitiySources/capsource099 HTTP/1.1
Host: example.com
Accept: application/xml
```

6.2.3.2.2 Response

```
HTTP/1.1 404 Not Found
Content-Type: application/xml
Content-Length: nnnn
Date: Thu, 04 Jun 2012 02:51:59 GMT


<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <link rel="capabilitySource"
    href="http://example.com/exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource099"/>
  <serviceException>
    <messageId>SVC1004</messageId>
    <text>Specified Capability Source, %1, is not defined</text>
    <variables>capsource099</variables>
  </serviceException>
</common:requestError>
```

6.2.4 PUT

This operation is used to update registered service Capability Source (e.g. enable/disable service capabilities, or add/remove service capabilities for that particular Capability Source).

6.2.4.1 Example 1: Enable service capabilities for a particular Capability Source (Informative)

6.2.4.1.1 Request

```
PUT /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001 HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySource xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
     <serviceCapability>
       <capabilityId>Chat</capabilityId>
       <status>Enabled</status>
     </serviceCapability>
     <clientCorrelator>123</clientCorrelator>
     <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001</resourceURL>
</cd:capabilitySource>
```

6.2.4.1.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySource xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
```

```
    <serviceCapability>
      <capabilityId>Chat</capabilityId>
      <status>Enabled</status>
    </serviceCapability>
    <clientCorrelator>123</clientCorrelator>
    <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001</resourceURL>
</cd:capabilitySource>
```

6.2.4.2 Example 2: Create (register) a new service capability for a particular Capability Source (Informative)

6.2.4.2.1 Request

```
PUT /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001 HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySource xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
    <serviceCapability>
      <capabilityId>Chat</capabilityId>
      <status>Enabled</status>
    </serviceCapability>
    <serviceCapability>
      <capabilityId>SocialPresenceInfo</capabilityId>
    </serviceCapability>
    <clientCorrelator>123</clientCorrelator>
    <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001</resourceURL>
</cd:capabilitySource>
```

6.2.4.2.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:capabilitySource xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
    <serviceCapability>
      <capabilityId>Chat</capabilityId>
      <status>Enabled</status>
    </serviceCapability>
    <serviceCapability>
      <capabilityId>SocialPresenceInfo</capabilityId>
      <status>Disabled</status>
    </serviceCapability>
    <clientCorrelator>123</clientCorrelator>
    <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001</resourceURL>
</cd:capabilitySource>
```

6.2.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET, PUT, DELETE' field in the response as per section 14.7 of [RFC 2616].

6.2.6 DELETE

This operation is used for deregistration of a particular service Capability Source including all its registered service capabilities. To perform this operation, the application SHOULD ensure that all registered service capabilities for that particular Capability Source are disabled.

6.2.6.1 Example: Deregister service Capability Source (Informative)

6.2.6.1.1 Request

```
DELETE /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001  HTTP/1.1
Host: example.com
Accept: application/xml
```

6.2.6.1.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 04 Jun 2012 02:51:59 GMT
```

6.3 Resource: Individual service capability data

The resource used is:

http://{serverRoot}/capabilitydiscovery/{apiVersion}/{userId}/ownCapabilities/{capabilitySourceId}/[ResourceRelPath]

This resource is used by a client to manage individual data relating to its Capability Source or a service capability. The precondition to use this resource is that the Capability Source has already been registered as described in 6.1.5.

6.3.1 Request URL variables

The following request URL variables are common for all HTTP commands:

Name	Description
serverRoot	Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.1.
userId	Identifier of the user on whose behalf the application acts Examples: tel:+19585550100, acr:pseudonym123
capabilityId	Identifier of the service capability Example: VideoShareDuringACall
[ResourceRelPath]	Relative resource path for a Light-weight Resource, consisting of a relative path down to an element in the data structure. For more information about the applicable values (strings) for this variable, see 6.3.1.1.

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.3.1.1 Light-weight relative resource paths

The following table describes the type of Light-weight Resources that can be accessed by using this resource, applicable methods, and the link to a data structure that contains values (strings) for those relative resource paths.

Light-weight Resource type	Method supported	Description
Individual Capability Source data	GET, PUT, DELETE	Enables access to duration parameter (lifetime) for a particular Capability Source. See column [ResourceRelPath] for element “duration” in section 5.2.2.2 for possible values for the light-weight relative resource path.
Individual service capability data	GET, PUT, DELETE	Enables access to individual service capability data. See column [ResourceRelPath] for elements “serviceCapability” in section 5.2.2.2, and “status” in section 5.2.2.3 for possible values of the light-weight relative resource path.

6.3.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Capability Discovery, see section 7.

6.3.3 GET

This operation is used for retrieval of data for an individual own service capability or retrieval of duration parameter information for a particular Capability Source.

6.3.3.1 Example: Retrieve an individual own service capability data (Informative)

6.3.3.1.1 Request

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003/VideoShareDuringACall
HTTP/1.1
Host: example.com
Accept: application/xml
```

6.3.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:serviceCapability xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
   <capabilityId>VideoShareDuringACall</capabilityId>
   <status>Disabled</status>
</cd:serviceCapability>
```

6.3.4 PUT

This operation is used to register a new own service capability, update the status of a service capability, or update/refresh duration time for aCapability Source.

6.3.4.1 Example 1: Enable an individual own service capability (Informative)

6.3.4.1.1 Request

```
PUT /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003/VideoShareDuringACall/status
HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:status xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">Enabled</cd:status>
```

6.3.4.1.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:status xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">Enabled</cd:status>
```

6.3.4.2 Example 2: Create (register) an individual own service capability (Informative)

6.3.4.2.1 Request

```
PUT /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003/Chat HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:serviceCapability xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
   <capabilityId>Chat</capabilityId>
</cd:serviceCapability>
```

6.3.4.2.2 Response

```
HTTP/1.1 201 Created
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:serviceCapability xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
   <capabilityId>Chat</capabilityId>
   <status>Disabled</status>
</cd:serviceCapability>
```

6.3.4.3 Example 3: Registration of an individual own service capability fails (Informative)

This example illustrates the case when trying to register an individual service capability which is not supported by the server.

6.3.4.3.1 Request

```
PUT /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003/ImageVideoShare HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:serviceCapability xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
   <capabilityId>ImageVideoShare</capabilityId>
</cd:serviceCapability>
```

6.3.4.3.2 Response

```
HTTP/1.1 403 Forbidden
Content-Type: application/xml
Content-Length: nnnn
Date: Thu, 04 Jun 2012 02:51:59 GMT


<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <link rel="serviceCapability"

href="http://example.com/exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003/ImageVideoShare"/>
  <policyException>
    <messageId>POL1022</messageId>
    <text>Specified service capability, %1, is not supported</text>
    <variables>ImageVideoShare</variables>
  </policyException>
</common:requestError>
```

6.3.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET, PUT, DELETE' field in the response as per section 14.7 of [RFC 2616].

6.3.6 DELETE

This operation is used for deregistration of an individual own service capability. To perform this operation, the service capability SHOULD be in status "Disabled".

6.3.6.1 Example: Deregister an own service capability (Informative)

6.3.6.1.1 Request

```
DELETE /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001/SocialPresenceInfo   HTTP/1.1
Host: example.com
Accept: application/xml
```


6.3.6.1.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 04 Jun 2012 02:51:59 GMT
```

6.4 Resource: Service capabilities for a contact

The resource used is:

http://{serverRoot}/capabilitydiscovery/{apiVersion}/{userId}/contactCapabilities/{contactId}

This resource is used for discovery (query) of service capabilities for a contact.

6.4.1 Request URL variables

The following request URL variables are common for all HTTP commands:

Name	Description
serverRoot	Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.1.
userId	Identifier of the user on whose behalf the application acts Examples: tel:+19585550100, acr:pseudonym123
contactId	Identifier of the contact Examples: tel:+19585550101

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.4.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Capability Discovery, see section 7.

6.4.3 GET

This operation is used for retrieval (query) of service capabilities for a contact..

Supported parameters in the query string of the Request URL are:

Name	Type/Values	Optional	Description
capabilityFilter	xsd:token	Yes	Defines for which particular service capability the query is. For capabilityFilter value, any valid capability identifier can be used (e.g. Chat), If a match is found, the response will include the capability Id, otherwise not. **Error! Reference source not found.** defines the strings that are used as capability identifiers for the most common service capabilities.
userTypeFilter	xsd:string	Yes	Defines for which type of users the query is. (For example, if userTypeFilter=RCSe, it indicates request to check whether the specified contact is an RCSe user or not). Supported values for userTypeFilter are listed in the enumeration UserType (section 5.2.3.2). If a match is found, the response will include the user type the query is for, otherwise not

See section 6 for a statement on the escaping of reserved characters in URL.

6.4.3.1 Example 1: Retrieve service capabilities for a contact (Informative)

6.4.3.1.1 Request

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101
HTTP/1.1
Host: example.com
Accept: application/xml
```

6.4.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:contactServiceCapabilities xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
   <serviceCapability>
      <capabilityId>Chat</capabilityId>
  </serviceCapability>
<resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101</resou
rceURL>
</cd:contactServiceCapabilities>
```

6.4.3.2 Example 2: Check whether a contact has a specific service capability (Informative)

6.4.3.2.1 Request

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101?capabilityFilter=Chat
HTTP/1.1
Host: example.com
```



```
Accept: application/xml
```

6.4.3.2.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:contactServiceCapabilities xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
  <serviceCapability>
     <capabilityId>Chat</capabilityId>
  </serviceCapability>
 <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101</resourceURL>
</cd:contactServiceCapabilities>
```

6.4.3.3 Example 3: Check whether a contact is an RCSe user or not, response positive (Informative)

6.4.3.3.1 Request

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101?userTypeFilter=RCSe HTTP/1.1
Host: example.com
Accept: application/xml
```

6.4.3.3.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:contactServiceCapabilities xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
      <userType>RCSe</userType >
 <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101</resourceURL>
</cd:contactServiceCapabilities>
```

6.4.3.4 Example 4: Check whether a contact is an RCSe user or not, response negative (Informative)

6.4.3.4.1 Request

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550109?userTypeFilter=RCSe HTTP/1.1
Host: example.com
Accept: application/xml
```


6.4.3.4.2 Response

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<cd:contactServiceCapabilities xmlns:cd="urn:oma:xml:rest:netapi:capabilitydiscovery:1">
  <resourceURL>http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550109</resourceURL>
</cd:contactServiceCapabilities>
```

6.4.3.5 Retrieving service capabilities for a contact fails (Informative)

6.4.3.5.1 Request

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101 HTTP/1.1
Host: example.com
Accept: application/xml
```

6.4.3.5.2 Response

```
HTTP/1.1 403 Forbidden
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <link rel="contactServiceCapabilities"
   href="http://example.com/exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101"/>
  <policyException>
    <messageId>POL2005</messageId>
    <text>Maximum number of requests for a given time period is exceeded</text>
  </policyException>
</common:requestError>
```

6.4.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow:GET’ field in the response as per section 14.7 of [RFC 2616].

6.4.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow:GET’ field in the response as per section 14.7 of [RFC 2616].

6.4.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow:GET’ field in the response as per section 14.7 of [RFC 2616].

7. Fault definitions

7.1 Service Exceptions

For common Service Exceptions refer to [REST_NetAPI_Common].

The following additional Service Exception codes are defined for the RESTful Capability Discovery API.

7.1.1 SVC1004: Capability Source does not exist

Name	Description
MessageID	SVC1004
Text	Specified Capability Source, %1, is not defined.
Variables	%1 - Capability Source Id
HTTP status code(s)	404 Not found

7.2 Policy Exceptions

For common Policy Exceptions refer to [REST_NetAPI_Common].

The following additional Policy Exception codes are defined for the RESTful Capability Discovery API.

7.2.1 POL1021: Maximum number of Capability Sources exceeded

Name	Description
MessageID	POL1021
Text	Maximum number of registered Capability Sources is exceeded.
Variables	None
HTTP status code(s)	403 Forbidden

7.2.2 POL1022: Service capability not supported

Name	Description
MessageID	POL1022
Text	Specified service capability, %1, is not supported.
Variables	%1 - service capability Id
HTTP status code(s)	403 Forbidden

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version

A.2 Draft/Candidate Version 1.0 History

Document Identifier	Date	Sections	Description
Draft Versions: REST_NetAPI _CapabilityDiscovery-V1_0	13 Mar 2012	all	Initial baseline. Incorporates input to committee: OMA-ARC-2012-0058-INP_BaselineREST_NetAPI_CapabilityDiscoveryTS
	25 Jun 2012	4, 5, 6, 7, Appendix G	Incorporated: OMA-ARC-REST-CapDis-2012-0001-CR_Resources and Data Structures
	16 Jul 2012	2, 4, 5, 6, Appendix C	Incorporated CRs: OMA-ARC-REST-CapDis-2012-0003R02-CR_TS_updates_for_section_6 OMA-ARC-REST-CapDis-2012-0006R02-CR_TS_updates_for_section_5, OMA-ARC-REST-CapDis-2012-0009R01-CR_TS_removing_form_urlencoded_blueprint
	10 Sep 2012	3.2, 5.1, 5.2, 5.3, 6.1, 6.2	Incorporated CR: OMA-ARC-REST-CapDis-2012-0014R02-CR_CapDis_TS_adding_support_for_multiple_devices
	21 Sep 2012	3, 4, 5, 6, Appendix B, F, G	Incorporated CRs: OMA-ARC-REST-CapDis-2012-0004R02-CR_TS_input_for_Appendix_B OMA-ARC-REST-CapDis-2012-0015R02-CR_TS_adding_light_weight_resources
	31 Oct 2012	Many	Incorporated CR: OMA-ARC-REST-CapDis-2012-0018-CR_TS_CONRR_fixing_editorial_comments
	22 Nov 2012	Many	Incorporated CRs: OMA-ARC-REST-CapDis-2012-0021R01-CR_TS_CONRR_A016_SVC_and_POL_exceptions, OMA-ARC-REST-CapDis-2012-0022R01-CR_TS_CONRR_A014_XML_examples_for_error_responses, OMA-ARC-REST-CapDis-2012-0023R02-CR_TS_CONRR_resolving_Editor_Notes, OMA-ARC-REST-CapDis-2012-0024R01-CR_Appendix_D_with_JSON_examples, OMA-ARC-REST-CapDis-2012-0027-CR_modifying_sequence_diagrams_for_Capability_Discovery, OMA-ARC-REST-CapDis-2012-0028-CR_TS_removing_references_to_Notification_Channel, OMA-ARC-REST-CapDis-2012-0029-CR_TS_CONRR_A011_resolution
	28 Nov 2012		Incorporated CRs: OMA-ARC-REST-CapDis-2012-0031-CR_TS_CONRR_A009_change_string_to_token OMA-ARC-REST-CapDis-2012-0033-CR_implementing_blueprint_changes_for_authorization Small editorial changes such as uppercase/lowercase and grammar.
Candidate Version: REST_NetAPI _CapabilityDiscovery-V1_0	11 Dec 2012	n/a	Status changed to Candidate by TP: Ref# OMA-TP-2012-0453-INP_REST_NetAPI_CapabilityDiscovery_V1.0_ERP_and_ETR_for_Candidate_Approval Editorial chanegs

Document Identifier	Date	Sections	Description
Draft Version: REST_NetAPI _CapabilityDiscovery-V1_0	22 Feb 2013	Appendix H	Status changed to Draft. Incorporated CR: OMA-ARC-REST-NetAPI-2013-0020R01-CR_Feature_Tag_Mapping_Table Editorial change
Candidate Version: REST_NetAPI _CapabilityDiscovery-V1_0	05 Mar 2013	n/a	Status changed to Candidate by TP: Ref# OMA-TP-2013-0080-INP_REST_NetAPI_CapabilityDiscovery_V1.0_ERP_for_Candidate_re_approval
Draft Version OMA-TS-REST_NetAPI_CapabilityDiscovery-V1_0	21 Mar 2013	3.2, 3.3, 5, 5.2.2.3, 6.1.3.1.2, 6.1.3.2.2, 6.1.5.1, 6.1.5.2, 6.1.5.3, 6.1.5.4.1, 6.2.3.1.2, 6.2.3.2, 6.2.3.2.2, 6.2.4.1, 6.2.4.2, 6.3.1, 6.3.3.1, 6.3.4.1.1, 6.3.4.2, 6.3.4.3, 6.3.6.1.1, 6.4.3, 6.4.3.1.2, 6.4.3.2, 6.4.3.4.1, D.1-D.10, D.12-D.18, D.20, Appendix H	Incorporated CRs: OMA-ARC-RCS-NetAPI-2013-0024-CR_CapDis_TS_removing_Editor_note OMA-ARC-RCS-NetAPI-2013-0029-CR_CapDis_TS_fixing_requestError_responses Editorial changes
Candidate Version OMA-TS-REST_NetAPI_CapabilityDiscovery-V1_0	01 Jul 2013	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2013-0213-INP_REST_NetAPI_CapabilityDiscovery_V1_0_ERP_for_Notification

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

B.1 SCR for REST.CapDis Server

Item	Function	Reference	Requirement
REST-CAPDIS-SUPPORT-S-001-M	Support for the RESTful Capability Discovery API	**Error! Reference source not found.**, 6	
REST-CAPDIS-SUPPORT-S-002-M	Support for the XML request & response format	6	
REST-CAPDIS-SUPPORT-S-003-M	Support for the JSON request & response format	6	

B.1.1 SCR for REST.CapDis.Own.CapabilitySources Server

Item	Function	Reference	Requirement
REST-CAPDIS-OWN-CAPSRC-S-001-M	Support for creation and retrieval of own Capability Source data	6.1	
REST-CAPDIS-OWN-CAPSRC-S-002-M	Retrieve own service capabilities from all Capability Sources - GET	6.1.3	
REST-CAPDIS-OWN-CAPSRC-S-003-M	Create (register) Capability Source data – POST (XML or JSON)	6.1.5	

B.1.2 SCR for REST.CapDis.Own.Single.CapabilitySource Server

Item	Function	Reference	Requirement
REST-CAPDIS-OWN-CAPS-S-001-M	Support for management of own service capabilities data from an individual Capability Source	6.2	
REST-CAPDIS-OWN-CAPS-S-002-M	Retrieve own service capabilities from an individual Capability Source - GET	6.2.3	
REST-CAPDIS-OWN-CAPS-S-003-M	Update (e.g. add (register) / remove (deregister), enable/disable) own service capabilities from an individual Capability Source – PUT (XML or JSON)	6.2.4	
REST-CAPDIS-OWN-CAPS-S-004-M	Deregister all own service capabilities for an individual Capability Source - DELETE	6.2.6	

B.1.3 SCR for REST.CapDis.Own.Individual.Capability Server

Item	Function	Reference	Requirement
REST-CAPDIS-OWN-IND-CAP-S-001-O	Support for management of an individual own service capability data	6.3	REST-CAPDIS-IND-OWN-CAP-S-003-O, REST-CAPDIS-OWN-

Item	Function	Reference	Requirement
			IND-CAP-S-004-O
REST-CAPDIS-OWN-IND-CAP-S-002-O	Retrieve data of an individual own service capability - GET	6.3.3	
REST-CAPDIS-OWN-IND-CAP-S-003-O	Register or update status (enable/disable) for individual own service capability – PUT (XML or JSON)	6.3.4	
REST-CAPDIS-OWN-IND-CAP-S-004-O	Deregister an individual own service capability - DELETE	6.3.6	

B.1.4 SCR for REST.CapDis.Contact.Capabilities Server

Item	Function	Reference	Requirement
REST-CAPDIS-CONTACT-CAPS-S-001-M	Support for capability discovery for a contact	6.4	
REST-CAPDIS-CONTACT-CAPS-S-002-M	Discover (retrieve) service capabilities for a specified contact - GET	6.4.3	
REST-CAPDIS-CONTACT-CAPS-S-003-O	Check whether a contact has a certain service capability - GET	6.4.3	
REST-CAPDIS-CONTACT-CAPS-S-004-M	Discover whether a contact belongs to a specified type(s) of users - GET	6.4.3	

Appendix C. Application/x-www-form-urlencoded Request Format for POST Operations (Normative)

This specification does not define any API request based on application/x-www-form-urlencoded MIME type.

Appendix D. JSON examples (Informative)

JSON (JavaScript Object Notation) is a light-weight, text-based, language-independent data interchange format. It provides a simple means to represent basic name-value pairs, arrays and objects. JSON is relatively trivial to parse and evaluate using standard JavaScript libraries, and hence is suited for REST invocations from browsers or other processors with JavaScript engines. Further information on JSON can be found at [RFC 4627].

The following examples show the request and response for various operations using the JSON data format. The examples follow the XML to JSON serialization rules in [REST_NetAPI_Common]. A JSON response can be obtained by using the content type negotiation mechanism specified in [REST_NetAPI_Common].

D.1 For full details on the operations themselves please refer to the section number indicated.Retrieve a list with status of all own service capabilities (section 6.1.3.1)

Request:

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"capabilitySourceList": {
   "capabilitySource": [
      {
         "clientCorrelator": "123",
         "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001",
         "serviceCapability": {
            "capabilityId": "Chat",
            "status": "Disabled"
         }
      },
      {
         "clientCorrelator": "1234",
         "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource002",
         "serviceCapability": [
            {
               "capabilityId": "ImageShare",
               "status": "Enabled"
            },
            {
               "capabilityId": "FileTransfer",
               "status": "Disabled"
            }
         ]
      }
   ],
   "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources"
```

```
}}
```

D.2 Retrieve a list with status of all own service capabilities using a filter (section 6.1.3.2)

Request:

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources?statusFilter=”Enabled” HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"capabilitySourceList": {
   "capabilitySource": {
      "clientCorrelator": "1234",
      "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource002",
      "serviceCapability": {
         "capabilityId": "ImageShare",
         "status": "Enabled"
      }
   },
   "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources"
}}
```

D.3 Create a new service Capability Source using ‘tel’ URI, response with a copy of created resource (section 6.1.5.1)

Request:

```
POST /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{"capabilitySource": {
   "clientCorrelator": "12345",
   "serviceCapability": {"capabilityId": "VideoShareDuringACall"}
}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003 HTTP/1.1
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
```

```
Content-Length: nnnn

{"capabilitySource": {
   "clientCorrelator": "12345",
   "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003",
   "serviceCapability": {
      "capabilityId": "VideoShareDuringACall",
      "status": "Disabled"
   }
}}
```

D.4 Create a new service Capability Source using ‘acr’ URI, response with a copy of created resource (section 6.1.5.2)

Request:

```
POST /exampleAPI/capabilitydiscovery/v1/acr%3Apseudonym123/capabilitySources HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{"capabilitySource": {
   "clientCorrelator": "12345",
   "serviceCapability": {"capabilityId": "VideoShareDuringACall"}
}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://exampleAPI/capabilitydiscovery/v1/acr%3Apseudonym123/capabilitySources/capsource003 HTTP/1.1
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"capabilitySource": {
   "clientCorrelator": "12345",
   "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/acr%3Apseudonym123/capabilitySources/capsource003",
   "serviceCapability": {
      "capabilityId": "VideoShareDuringACall",
      "status": "Disabled"
   }
}}
```

D.5 Create a new service Capability Source, response with a location of created resource (section 6.1.5.3)

Request:

```
POST /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
```

```
Content-Length: nnnn

{"capabilitySource": {
    "clientCorrelator": "12345",
    "serviceCapability": {"capabilityId": "VideoShareDuringACall"}
}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003 HTTP/1.1
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"resourceReference": {"resourceURL":
"http://example.com/exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003"}}
```

D.6 Creating a new Capability Source fails (section 6.1.5.4)

Request:

```
POST /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{"capabilitySource": {
    "clientCorrelator": "12345",
    "serviceCapability": [
        {"capabilityId": "VideoShareDuringACall"},
        {"capabilityId": "Chat"}
    ]
}}
```

Response:

```
HTTP/1.1 403 Forbidden
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"requestError": {
    "link": {
        "href": "http://example.com/exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources",
        "rel": "CapabilitySourceList"
    },
    "policyException": {
        "messageId": "POL1021",
        "text": "Maximum number of registered Capability Sources is exceeded"
    }
}}
```

D.7 Retrieve service capabilities registered for a particular Capability Source (section 6.2.3.1)

Request:

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitiySources/capsource001 HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"capabilitySource": {
   "clientCorrelator": "123",
   "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001",
   "serviceCapability": {
      "capabilityId": "Chat",
      "status": "Disabled"
   }
}}
```

D.8 Retrieving service capabilities for non-existent Capability Source (section 6.2.3.2)

Request:

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitiySources/capsource099 HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json
Content-Length: nnnn
Date: Thu, 04 Jun 2012 02:51:59 GMT

{"requestError": {
   "link": {
      "href": "http://example.com/exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource099",
      "rel": "capabilitySource"
   },
   "serviceException": {
      "messageId": "SVC1004",
      "text": "Specified Capability Source, %1, is not defined",
      "variables": "capsource099"
   }
}}
```

D.9 Enable service capabilities for a particular Capability Source (section 6.2.4.1)

Request:

```
PUT /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001 HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{"capabilitySource": {
   "clientCorrelator": "123",
   "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001",
   "serviceCapability": {
      "capabilityId": "Chat",
      "status": "Enabled"
   }
}}
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"capabilitySource": {
   "clientCorrelator": "123",
   "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001",
   "serviceCapability": {
      "capabilityId": "Chat",
      "status": "Enabled"
   }
}}
```

D.10 Create (register) a new service capability for a particular Capability Source (section 6.2.4.2)

Request:

```
PUT /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001 HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{"capabilitySource": {
   "clientCorrelator": "123",
   "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001",
   "serviceCapability": [
      {
         "capabilityId": "Chat",
         "status": "Enabled"
```

```
      },
      {"capabilityId": "SocialPresenceInfo"}
   ]
}}
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"capabilitySource": {
   "clientCorrelator": "123",
   "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001",
   "serviceCapability": [
      {
         "capabilityId": "Chat",
         "status": "Enabled"
      },
      {
         "capabilityId": "SocialPresenceInfo",
         "status": "Disabled"
      }
   ]
}}
```

D.11 Deregister service Capability Source (section 6.2.6.1)

Request:

```
DELETE /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001  HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 04 Jun 2012 02:51:59 GMT
```

D.12 Retrieve an individual own service capability data (section 6.3.3.1)

Request:

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003/VideoShareDuringACall
HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
```

```
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"serviceCapability": {
   "capabilityId": "VideoShareDuringACall",
   "status": "Disabled"
}}
```

D.13 Enable an individual own service capability (section 6.3.4.1)

Request:

```
PUT /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003/ VideoShareDuringACall/status
HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{"status": "Enabled"}
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"status": "Enabled"}
```

D.14 Create (register) an individual own service capability (section 6.3.4.2)

Request:

```
PUT /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003/Chat HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{"serviceCapability": {"Chat"}}
```

Response:

```
HTTP/1.1 201 Created
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"serviceCapability": {
   "capabilityId": "Chat",
```

```
    "status": "Disabled"
}}
```

D.15 Registration of an individual own service capability fails (section 6.3.4.3)

Request:

```
PUT /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003/ImageVideoShare HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{"serviceCapability": {"capabilityId": "ImageVideoShare"}}
```

Response:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
Content-Length: nnnn
Date: Thu, 04 Jun 2012 02:51:59 GMT

{"requestError": {
    "link": {
        "href":
"http://example.com/exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource003/ImageVideoShare",
        "rel": "serviceCapability"
    },
    "policyException": {
        "messageId": "POL1022",
        "text": "Specified service capability, %1, is not supported"
        "variables": "ImageVideoShare"
    }
}}
```

D.16 Deregister an own service capability (section 6.3.6.1)

Request:

```
DELETE /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/capabilitySources/capsource001/SocialPresenceInfo   HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 04 Jun 2012 02:51:59 GMT
```

D.17 Retrieve service capabilities for a contact (section 6.4.3.1)

Request:

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101
HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"contactServiceCapabilities": {
   "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101",
   "serviceCapability": {"capabilityId": "Chat"}
}}
```

D.18 Check whether a contact has a specific service capability (section 6.4.3.2)

Request:

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101?capabilityFilter=Chat
HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"contactServiceCapabilities": {
   "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101",
   "serviceCapability": {"capabilityId": "Chat"}
}}
```

D.19 Check whether a contact is an RCSe user or not, response positive (section 6.4.3.3)

Request:

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101?userTypeFilter=RCSe
HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
```

```
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"contactServiceCapabilities": {
   "resourceURL": "http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101",
   "userType": "RCSe"
}}
```

D.20 Check whether a contact is an RCSe user or not, response negative (section 6.4.3.4)

Request:

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550109?userTypeFilter=RCSe
HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"contactServiceCapabilities": {"resourceURL":
"http://exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550109"}}
```

D.21 Retrieving service capabilities for a contact fails (section 6.4.3.5)

Request:

```
GET /exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101 HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 403 Forbidden
Date: Thu, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"requestError": {
   "link": {
      "href":
"http://example.com/exampleAPI/capabilitydiscovery/v1/tel%3A%2B19585550100/contactCapabilities/tel%3A%2B19585550101",
      "rel": "contactServiceCapabilities"
   },
   "policyException": {
```

```
        "messageId": "POL2005",
        "text": "Maximum number of requests for a given time period is exceeded"
    }
}}
```

Appendix E. Operations mapping to a pre-existing baseline specification (Informative)

As this specification does not have a baseline specification, this appendix is empty.

Appendix F. Light-weight resources (Informative)

The following table lists all Capability Discovery data structure elements that can be accessed individually as light-weight resources. For each light-weight resource, the following information is provided: corresponding root element name, root element type and [ResourceRelPath] string.

Type of light-weight resources (and references to data structures)	Element/attribute that can be accessed as light-weight resource	Root element name for the light-weight resource	Root element type for the light-weight resource	[ResourceRelPath] string that needs to be appended to the corresponding heavy-weight resource URL
Capability Source information (5.2.2.2)	duration	duration	xsd:int	duration
Individual service capability information (5.2.2.2, and 5.2.2.3)	serviceCapability	serviceCapability	ServiceCapability	{capabilityId}
	status	status	CapabilityStatus	{capabilityId}/status

Note: When appending [ResourceRelPath] string to its Heavy-weight Resource URL, variable within curly brackets "{}" such is: {capabilityId} has to be replaced with its real value.

Appendix G. Authorization aspects (Normative)

This appendix specifies how to use the RESTful Capability Discovery API in combination with some authorization frameworks.

G.1 Use with OMA Authorization Framework for Network APIs

The RESTful Capability Discovery API MAY support the authorization framework defined in [Autho4API_10].

A RESTful Capability Discovery API supporting [Autho4API_10]:

- SHALL conform to section D.1 of [REST_NetAPI_Common];
- SHALL conform to this section G.1.

G.1.1 Scope values

G.1.1.1 Definitions

In compliance with [Autho4API_10], an authorization server serving clients requests for getting authorized access to the resources exposed by the RESTful Capability Discovery API:

- SHALL support the scope values defined in the table below;
- MAY support scope values not defined in this specification.

Scope value	Description	For one-time access token
oma_rest_capabilitydiscovery.all_{apiVersion}	Provide access to all defined operations on the resources in this version of the API. The {apiVersion} part of this identifier SHALL have the same value as the “apiVersion” URL variable which is defined in section 5.1. This scope value is the union of the other scope values listed in next rows of this table.	No
oma_rest_capabilitydiscovery_own	Provide access to all operations defined for management of own service capabilities.	No
oma_rest_capabilitydiscovery_contact	Provide access to all operations defined for checking of service capabilities for a contact.	No

Table 1 Scope values for RESTful Capability Discovery API

G.1.1.2 Downscoping

In the case where the client requests authorization for “oma_rest_capabilitydiscovery.all_{apiVersion}” scope, the authorization server and/or resource owner MAY restrict the granted scope to some of the following scope values:

- “oma_rest_capabilitydiscovery_own”
- “oma_rest_capabilitydiscovery_contact”

G.1.1.3 Mapping with resources and methods

Tables in this section specify how the scope values defined in section G.1.1.1for the RESTful Capability Discovery API map to the REST resources and methods of this API. In these tables, the root “oma_rest_capabilitydiscovery.” of scope values is omitted for readability reasons

Resource	URL Base URL: http://{serverRoot}/capabilitydiscovery/{apiVersion}	Section reference	HTTP verbs			
			GET	PUT	POST	DELETE
Service Capability Sources	/{userId}/capabilitySources	6.1	**all_{apiVersion}**, or **own**	**all_{apiVersion}**, or **own**	**all_{apiVersion}**, or **own**	**all_{apiVersion}**, or **own**
Individual service Capability Source	/{userId}/capabilitySources/{capabilitySourceId}	6.3	**all_{apiVersion}**, or **own**	**all_{apiVersion}**, or **own**	n/a	**all_{apiVersion}**, or **own**
Individual service capability data	/{userId}/capabilitySources/{capabilitySourceId}/[ResourceRelPath]	6.3	**all_{apiVersion}**, or **own**	**all_{apiVersion}**, or **own**		**all_{apiVersion}**, or **own**

Table 2 Required scope values for: Management of own service capabilities

Resource	URL Base URL: http://{serverRoot}/capabilitydiscovery/{apiVersion}	Section reference	HTTP verbs			
			GET	PUT	POST	DELETE
Service capabilities for a contact	/{userId}/contactCapabilities/{contactId}	6.4	**all_{apiVersion}**, or **contact**	n/a	n/a	n/a

Table 3 Required scope values for: Retrieving of service capabilities for a contact

G.1.2 Use of ‘acr:auth’

This section specifies the use of ‘acr:auth’ in place of an end user identifier in a resource URL path.

An ‘acr’ URI of the form ‘acr:auth’, where ‘auth’ is a reserved keyword MAY be used to avoid exposing a real end user identifier in the resource URL path.

A client MAY use ‘acr:auth’ in a resource URL in place of the{userId} resource URL variable in the resource URL path, when the RESTful Capability Discovery API is used in combination with [Autho4API_10].

In the case the RESTful Capability Discovery API supports [Autho4API_10], the server:

- SHALL accept ‘acr:auth’ as a valid value for the resource URL variable {userId}.

SHALL conform to [REST_Common_TS] section 5.8.1.1 regarding the processing of ‘acr:auth’

Appendix H. Feature Tag Mapping Table (Normative)

The following table defines the strings that are used to identify most common service capabilities, and how these strings (shown in first column) SHALL be mapped on the server to their respective SIP OPTIONS tags or Presence service tuples.

Feature Tag Mapping Table		
String identifying the Service at API level	**Tag**	**SIP OPTIONS Tag**
	Service Tuple	**Presence Service Tuple**
StandaloneMessaging	Tag	+g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.oma.cpm.msg; urn%3Aurn-7%3A3gpp-service.ims.icsi.oma.cpm.largemsg"
	Service Tuple	Service-id: org.3gpp.urn:urn-7:3gpp-application.ims.iari.rcs.sm Version: 1.0
Chat	Tag	+g.3gpp.iari-ref="urn%3Aurn-7%3A3gpp-application.ims.iari.rcse.im"
	Service Tuple	Service-id: org.openmobilealliance:IM-session Version : 1.0
StoreAndForwardGroupChat	Tag	+g.3gpp.iari-ref="urn%3Aurn-7%3A3gpp-application.ims.iari.rcs.fullsfgroupchat"
	Service Tuple	Service-Id: org.3gpp.urn:urn-7:3gpp-application.ims.iari.rcs.fullsfgroupchat Version: 1.0
FileTransfer	Tag	+g.3gpp.iari-ref="urn%3Aurn-7%3A3gpp-application.ims.iari.rcse.ft"
	Service Tuple	Service-id: org.openmobilealliance:File-Transfer Version : 1.0
FileTransferThumbnail	Tag	+g.3gpp.iari-ref="urn%3Aurn-7%3A3gpp-application.ims.iari.rcs.ftthumb"
	Service Tuple	Service-id: org.3gpp.urn:urn-7:3gpp-application.ims.iari.rcs.ftthumb Version : 1.0
FileTransferStoreAndForward	Tag	+g.3gpp.iari-ref="urn%3Aurn-7%3A3gpp-application.ims.iari.rcs.ftstandfw"
	Service Tuple	Service-id: org.openmobilealliance:File-Transfer Version : 2.0
FileTransferViaHTTP	Tag	+g.3gpp.iari-ref="urn%3Aurn-7%3A3gpp-application.ims.iari.rcs.fthttp"
	Service Tuple	Service-id: org.openmobilealliance:File-Transfer-HTTP Version : 1.0
ImageShare	Tag	+g.3gpp.iari-ref="urn%3Aurn-7%3A3gpp-application.ims.iari.gsma-is"
	Service Tuple	Service-id: org.gsma.imageshare Version: 1.0
VideoShareDuringACall	Tag	+g.3gpp.cs-voice
	Service Tuple	Service-id: org.gsma.videoshare Version: 1.0
VideoShareOutsideOfAVoiceCall	Tag	+g.3gpp.iari-ref=”urn:urn-7:3gpp-application.ims.iari.gsma-vs"
	Service Tuple	Service-id: org.gsma.videoshare Version: 2.0
SocialPresenceInfo	Tag	+g.3gpp.iari-ref="urn%3Aurn-7%3A3gpp-application.ims.iari.rcse.sp"

	Service Tuple	Service-id: org.3gpp.urn:urn-7:3gpp-application.ims.iari.rcse.sp Version: 1.0
CapabilityDiscoveryViaPresence	Tag	+g.3gpp.iari-ref="urn%3Aurn-7%3A3gpp-application.ims.iari.rcse.dp"
	Service Tuple	Service-id: org.3gpp.urn:urn-7:3gpp-application.ims.iari.rcse.dp Version: 1.0
IPVoiceCall	Tag	+g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.mmtel"
	Service Tuple	Service-id: org.3gpp.urn:urn-7:3gpp-service.ims.icsi.mmtel Version: 1.0 Media capabilities: audio, duplex
IPVideoCall	Tag	+g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.mmtel";video
	Service Tuple	Service-id: org.3gpp.urn:urn-7:3gpp-service.ims.icsi.mmtel Version: 1.0 Media capabilities: audio, video, duplex
GeolocationPull	Tag	+g.3gpp.iari-ref="urn%3Aurn-7%3A3gpp-application.ims.iari.rcs.geopull"
	Service Tuple	Service-id: org.3gpp.urn:urn-7:3gpp-application.ims.iari.rcs.geopull Version: 1.0
GeolocationPullUsingFileTransfer	Tag	+g.3gpp.iari-ref="urn%3Aurn-7%3A3gpp-application.ims.iari.rcs.geopullft"
	Service Tuple	Service-id: org.3gpp.urn:urn-7:3gpp-application.ims.iari.rcs.geopullft Version: 1.0
GeolocationPush	Tag	+g.3gpp.iari-ref="urn%3Aurn-7%3A3gpp-application.ims.iari.rcs.geopush"
	Service Tuple	Service-id: org.3gpp.urn:urn-7:3gpp-application.ims.iari.rcs.geopush Version: 1.0

Table 4 Feature Tag Mapping Table

Note that in addition to the strings listed in the above table, deployments MAY also support other strings.