RESTful Network API for Image Share

Candidate Version 1.0 – 10 Apr 2012

Open Mobile Alliance

OMA-TS-REST_NetAPI_ImageShare-V1_0-20120410-C

Contents

Figures

Tables

1. Scope

This specification defines a RESTful API for Image Share using HTTP protocol bindings.

2. References

2.1 Normative References

[Autho4API_10]	"Authorization Framework for Network APIs", Open Mobile Alliance™, OMA-ER-Autho4API-V1_0, URL: http://www.openmobilealliance.org/
[IETF_ACR_draft]	"The acr URI for anonymous users", S.Jakobsson, K.Smith, July 2011, URL: http://tools.ietf.org/html/draft-uri-acr-extension-03
[IR.79]	"Image Share Interoperability Specification", GSM Association™, URL: http://gsmworld.com/documents
[RC_API_RD]	APIs for Rich Communications Requirements, OMA-RD-RC_API-V1_0, Open Mobile Alliance, URL: http://www.openmobilealliance.org/
[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_NetAPI_NotificationChannel]	"RESTful Network API for Notification Channel", Open Mobile Alliance™, OMA-TS-REST_NetAPI_NotificationChannel-V1_0, URL: http://www.openmobilealliance.org/
[REST_SUP_ImageShare]	"XML schema for the RESTful Network API for Image Share", Open Mobile Alliance™, OMA-SUP-XSD-rest_netapi_imageshare-V1.0, URL:http://www.openmobilealliance.org/
[RFC2045]	"Multipurpose Internet Mail Extensions(MIME) Part One: Format of Internet Message Bodies", N. Freed, November 1996, URL: http://www.ietf.org/rfc/rfc2045.txt
[RFC2119]	"Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt
[RFC2327]	"SDP: Session Description Protocol", M. Handley, April 1998, URL: http://www.ietf.org/rfc/rfc2327.txt
[RFC2616]	"Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding et. al, January 1999, URL:http://www.ietf.org/rfc/rfc2616.txt
[RFC3261]	"SIP: Session Initiation Protocol", J. Rosenberg et al., June 2002, URL: http://www.rfc-editor.org/rfc/rfc3261.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
[RFC4975]	"The Message Session Relay Protocol (MSRP)", B. Campbell et. al, September 2007, URL: http://www.ietf.org/rfc/rfc4975.txt
[RFC5547]	"Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", M. Garcia-Martin, May 2009, URL: http://www.ietf.org/rfc/rfc5547.txt
[SCRRULES]	"SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/
[W3C_URLENC]	HTML 4.01 Specification, Section 17.13.4 Form content types, The World Wide Web Consortium, URL: http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.1
[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 2.8, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_8, 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/
[IMEND]	RCS Release 3 OMA IM Endorsement, GSM Association™, URL: http://www.gsmworld.com/our-work/mobile_lifestyle/rcs/rcs_specification_documents.htm

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

Client-side Notification URL	An HTTP URL exposed by a client, on which it is capable of receiving notifications and that can be used by the client when subscribing to notifications.
Long Polling	A variation of the traditional polling technique, where the server does not reply to a request unless a particular event, status or timeout has occurred. Once the server has sent a response, it closes the connection, and typically the client immediately sends a new request. This allows the emulation of an information push from a server to a client.
Notification Channel	A channel created on the request of the client and used to deliver notifications from a server to a client. The channel is represented as a resource and provides means for the server to post notifications and for the client to receive them via specified delivery mechanisms. For example in the case of Long Polling the channel resource is defined by a pair of URLs. One of the URLs is used by the client as a call-back URL when subscribing for notifications. The other URL is used by the client to retrieve notifications from the Notification Server.
Notification Server	A server that is capable of creating and maintaining Notification Channels.
Originator	The party that initiates an image share session.
Receiver	The party that is invited to an image share session to receive images.
Server-side Notification URL	An HTTP URL exposed by a Notification Server, that identifies a Notification Channel and that can be used by a client when subscribing to notifications.

Additionally, all definitions from the OMA Dictionary apply [OMADICT].

3.3 Abbreviations

ACR	Anonymous Customer Reference
API	Application Programming Interface
CS	Circuit Switch
HTTP	HyperText Transfer Protocol
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
MIME	Multipurpose Internet Mail Extensions
OMA	Open Mobile Alliance
REST	REpresentational State Transfer
SCR	Static Conformance Requirements
SDP	Session Description Protocol
SIP	Session Initiation Protocol
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 Image Share contains HTTP protocol bindings based on the requirements for Image Share defined in [RC_API_RD], 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, JSON, and application/x-www-form-urlencoded).

4.1 Version 1.0

Version 1.0 of this specification supports the following operations:

- Manage subscriptions to image share related event notifications
- Manage image share sessions
- Notify the application about the image share session invitation
- Notify the application about the image share session acceptance
- Notify the application about the image file link
- Notify the application about the image share events

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:Authorization” as a reserved keyword in a resource URL variable that identifies an end user

According [RC_API_RD], there are two types of image sharing:

- image sharing with a CS voice call which is based on [IR.79]
- image sharing without a CS voice call which utilizes OMA file transfer mechanism as defined in [IMEND]

In this specification, the image share session refers to 1-1 image share session which incorporates exactly 2 participants: an Originator and a Receiver.

5. Image Share API definition

This section is organized to support a comprehensive understanding of the Image Share 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 5 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 the detailed specification for each of the resources. Each such subsection defines the resource, the request URL variables that are common for all HTTP commands, the possible HTTP response codes, 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. Application/x-www-form-urlencoded examples are provided in Appendix C, while JSON examples are provided in Appendix D.

Appendix B provides the Static Conformance Requirements (SCR).

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

Appendix F provides a list of all lightweight resources, where applicable.

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

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 Image Share.

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.

The figure below visualizes the resource structure defined by this specification. Note that those nodes in the resource tree which have associated HTTP methods defined in this specification are depicted by solid boxes.

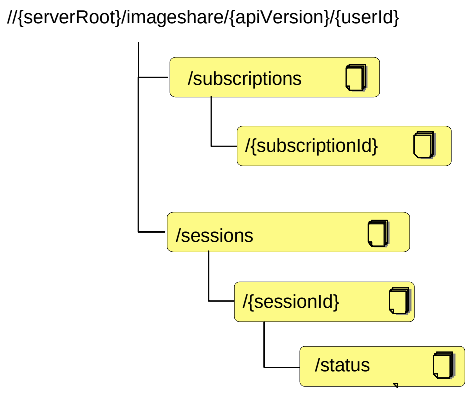

Figure 1 Resource structure defined by this specification

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 client to manage image share notifications subscriptions

Resource	URLBase URL: http://{serverRoot}/ imageshare/{apiVersion}/{userId}	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
All subscriptions to image share notifications	/subscriptions	ImageShareSubscriptionList (used for GET) ImageShareNotificationSubscription (used for POST) common:ResourceReference (optional alternative for POST response)	Read the list of active image share notification subscriptions	no	Create a new subscription for image share notifications	no
Individual subscription to image share notifications	/subscriptions/{subscriptionId}	ImageShareNotificationSubscription	Read an individual subscription	no	no	Cancel subscription and stop corresponding notifications

Purpose: To allow client to manage image share sessions

Resource	URL Base URL: http://{serverRoot}/imageshare/ {apiVersion}/{userId}	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
All image share sessions	/sessions	ImageShareSessionInformation common:ResourceReference (optional alternative for POST response)	no	no	Create a new image share session	no

Resource	URL Base URL: http://{serverRoot}/imageshare/{apiVersion}/{userId}	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
Individual image share session	/sessions/{sessionId}	ImageShareSessionInformation	Read information about an image share session	no	no	Cancel an image share session invitation (Originator) Decline an image share session invitation (Receiver) Terminate an image share session
Individual image share session status	/sessions/{sessionId}/status	ReceiverSessionStatus	no	no	Accept an image share invitation	no

Purpose: To allow server to notify client about image share session status and image file links

Resource	URL Base URL: <Specified by the client>	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
Client notification about image share session invitation	Specified by client when subscription is created or provisioned	SessionInvitationNotification	no	no	Notify client about incoming image share session invitation	no
Client notification about image share session acceptance	Specified by client when subscription is created or provisioned	SessionAcceptanceNotification	no	no	Notify client about image share session acceptance	no

Resource	URL Base URL: <Specified by the client>	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
Client notification about image share session event	Specified by client when subscription is created or provisioned	ImageShareEventNotification	no	no	Notify client about image share session event	No
Client notification about image share file link	Specified by client when subscription is created or provisioned	ImageFileNotification	no	no	Notify client about image share file link	no
Client notification about subscription cancellation	Specified by client when subscription is created or provisioned	SubscriptionCancellationNotification	no	no	Notify client that a subscription has been cancelled (e.g. expired	No

5.2 Data Types

5.2.1 XML Namespaces

The XML namespace for the Image Share API data types is:

urn:oma:xml:rest:netapi:imageshare: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_ImageShare].

5.2.2 Structures

The subsections of this section define the data structures used in the Image Share API.

Some of the structures can be instantiated as so-called root elements.

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: ImageShareSessionInformation

This type represents information about an image share session.

Element	Type	Optional	Description
originatorAddress	xsd:anyURI	No	Address (e.g. 'sip' URI, 'tel' URI, 'acr' URI) of the Originator. When the application acts on behalf of the Originator, the originatorAddress MUST have the same value as the {userId} fragment in the resource URL path if the {userId} is also part of the request URL.
originatorName	xsd:string	Yes	Name of the Originator
receiverAddress	xsd:anyURI	Yes	Address (e.g. 'sip' URI, 'tel' URI, 'acr' URI) of the Receiver. It SHALL be present in request bodies during resource creation in case of image sharing without CS voice call. It SHALL not be present in request bodies during resource creation in case of image sharing with CS voice call. The server can get the receiverAddress using the callObjectRef received in the request bodies during resource creation in case of image sharing with CS voice call. When the application acts on behalf of the Receiver, the receiverAddress MUST have the same value as the {userId} fragment in the resource URL path if the {userId} is also part of the request URL.

callObjectRef	xsd:anyURI	Yes	Reference to the CS voice object (to which the Receiver is linked). It SHALL be present in request bodies during resource creation in case of image sharing with CS voice call. It SHALL not be present in request bodies during resource creation in case of image sharing without CS voice call.
receiverName	xsd:string	Yes	Name of the Receiver. MAY be present when receiverAddress is used.
status	SessionStatus	Yes	Connection status of the image share session. Set by the server (see Table 1 for detailed information). SHALL NOT be present in request bodies during session resource creation by the application of the Originator.
fileInformation	FileInformation	No	A set of image file attributes
fileURL	xsd:anyURI	Yes	The file repository URL from where the file can be retrieved. If it is present in the POST operation during resource creation, the application of the Originator side needs to fetch the image file using this URL. Otherwise, if it is not present in the POST request during resource creation, the image file content is included in the HTTP body. The HTTP body can be represented as multipart/form-data entity bodies, as specified in 5.2.5.
clientCorrelator	xsd:string	Yes	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 re-sending the message 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.
resourceURL	xsd:anyURI	Yes	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 imageShareSessionInformation of type ImageShareSessionInformation 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. The specification [REST_NetAPI_Common] provides a recommendation regarding the generation of the value of this field.

The following table gives detailed information about when and what "status" value is set:

Value of "status"	Set in the following occasions:
Initial	Set by server when receiving session resource creation request or when receiving session invitation notification
Connected	Set by server when "status" is set to "Connected" in "receiverSessionStatus " root element when application of Receiver accepts an image share session invitation or Set by server when "status" is set to "Connected" in "receiverSessionStatus" of "sessionAcceptanceNotification" root element when receiving session acceptance notification.
Terminated	Set by server when receiving delete session request or when receiving image share event notifications with eventType of "SessionEnded"/"SessionCancelled"/"Declined"

Table 1: Image share session status

5.2.2.2 Type: FileInformation

This type represents a set of image file attributes.

Element	Type	Optional	Description
fileSelector	FileSelector	No	A tuple of file attributes that the SDP offerer includes in the SDP for the Receiver to decide whether to accept the file or not. File name, size, type and hash as specified in [RFC5547].
fileDescription	xsd:string	Yes	Human-readable short description of the image file (corresponding to 'i=' line in SDP) which could be set by the Originator. See [IR79] & [RFC2327].

5.2.2.3 Type: FileSelector

This type represents the basic information of a file including name, size, type and hash.

Element	Type	Optional	Description
type	xsd:string	No	The MIME type of the file, as concatenated of type, "/" and subtype (e.g. image/jpeg). See [RFC2045].
name	xsd:string	Yes	The name of the file
size	xsd: unsignedLong	Yes	The size of the file in octets

hash	HashInformation	Yes	The file hash information including hash algorithm and hash value

5.2.2.4 Type: HashInformation

This type represents the file hash information.

Element	Type	Optional	Description
algorithm	xsd:string	No	The hash algorithm used (only “sha-1” currently supported)
value	xsd: hexBinary	No	The hash value of the file

5.2.2.5 Type: SessionInvitationNotification

This type represents an image share session invitation notification.

Element	Type	Optional	Description
callbackData	xsd:string	Yes	The ‘callbackData’ element if it was passed by the application in the ‘callbackReference’ element when creating a subscription to image share notifications. See [REST_NetAPI_Common].
link	common:Link [0..unbounded]	Yes	Links to other resources that are in relationship to the notification. The server MUST include links as defined for SessionInvitationNotification (see section 6.6 for detailed information). Further, the server SHOULD include a link to the related subscription.
originatorAddress	xsd:anyURI	No	Address (e.g. 'sip' URI, 'tel' URI, 'acr' URI) of the Originator. When the application acts on behalf of the Originator, the originatorAddress MUST have the same value as the {userId} fragment in the resource URL path if the {userId} is also part of the request URL.
originatorName	xsd:string	Yes	Name of the Originator
receiverAddress	xsd:anyURI	Yes	Address (e.g. 'sip' URI, 'tel' URI, 'acr' URI) of the Receiver. It SHALL be present in notification request bodies in case of image sharing without CS voice call. It MAY be present in case of image sharing with CS voice call. When the application acts on behalf of the Receiver, the receiverAddress MUST have the same value as the {userId} fragment in the resource URL path if the {userId} is also part of the request URL.

callObjectRef	xsd:anyURI	Yes	Reference to the CS voice object (to which the Receiver is linked). It SHALL be present in notification request bodies in case of image sharing with CS voice call. It SHALL not be present in request bodies during resource creation in case of image sharing without CS voice call. The server can get the callObjectRef using receiverAddress in case of image sharing with CS voice call.
receiverName	xsd:string	Yes	Name of the Receiver. It MAY be present when receiverAddress is present.
fileInformation	FileInformation	No	A set of image file attributes

A root element named sessionInvitationNotification of type SessionInvitationNotification is allowed in notification request bodies.

5.2.2.6 Type: ReceiverSessionStatus

This type represents the status of a Receiver in the image share session.

Element	Type	Optional	Description
status	SessionStatus	No	Status of the image share session. To indicate that the Receiver accepts the session invitation, this element MUST be set to “Connected”.
fileAcceptance	xsd:boolean	Yes	Accept (true) or reject (false) the image file by the Receiver. Default is true.

A root element named receiverSessionStatus of type ReceiverSessionStatus is allowed in request bodies.

5.2.2.7 Type: ImageShareEventNotification

This type represents an image share event notification.

Element	Type	Optional	Description
callbackData	xsd:string	Yes	The ‘callbackData’ element if it was passed by the application in the ‘callbackReference’ element when creating a subscription to image share notifications. See [REST_NetAPI_Common].
link	common:Link [0..unbounded]	Yes	Links to other resources that are in relationship to the notification. Depending on the value of eventType, the server

			MUST include links as defined for ImageShareEventNotification (See section 6.9 for detailed information). Further, the server SHOULD include a link to the related subscription.
eventType	EventType	No	Type of event
eventDescription	xsd:string	Yes	Textual description of the event

A root element named imageSharetEventNotification of type ImageShareEventNotification is allowed in notification request bodies.

5.2.2.8 Type: SessionAcceptanceNotification

This type represents an image share session acceptance notification.

Element	Type	Optional	Description
callbackData	xsd:string	Yes	The ‘callbackData’ element if it was passed by the application in the ‘callbackReference’ element when creating a subscription to image share notifications. See [REST_NetAPI_Common].
link	common:Link [0..unbounded]	Yes	Links to other resources that are in relationship to the notification (See section 6.7 for detailed information). The server SHOULD include a link to the related subscription.
receiverAddress	xsd:anyURI	Yes	Address (e.g. 'sip' URI, 'tel' URI, 'acr' URI) of the Receiver. It SHALL be present in notification request bodies in case of image sharing without CS voice call. IT MAY be present in case of image sharing with CS voice call.
callObjectRef	xsd:anyURI	Yes	Reference to the CS voice object (to which the Receiver is linked). It SHALL be present in notification request bodies in case of image sharing with CS voice call. It SHALL not be present in request bodies during resource creation in case of image sharing without CS voice call.
receiverName	xsd:string	Yes	Name of the Receiver. It MAY be present when receiverAddress is present.
receiverSessionStatus	ReceiverSessionStatus	No	The status of the image share session. To indicate that the Receiver accepts the session invitation, this element MUST be set to “Connected”.

A root element named sessionAcceptanceNotification of type SessionAcceptanceNotification is allowed in notification request bodies.

5.2.2.9 Type: ImageShareSubscriptionList

This type represents a list of image share notification subscriptions.

Element	Type	Optional	Description
imageShareNotificationSubscription	ImageShareNotificationSubscription [0..unbounded]	Yes	Array of image share event subscriptions
resourceURL	xsd:anyURI	No	Self referring URL

A root element named imageShareSubscriptionList of type ImageShareSubscriptionList is allowed in response bodies.

5.2.2.10 Type: ImageShareNotificationSubscription

This type represents a subscription to image share notifications, i.e. ImageShareEventNotification, SessionAcceptanceNotification, SessionInvitationNotification,ImageFileNotification targeted at a particular user.

Element	Type	Optional	Description
callbackReference	common:CallbackReference	No	Client's Notification URL and OPTIONAL callbackData
duration	xsd:int	Yes	Period of time (in seconds) notifications are provided for. If set to “0” (zero), a default duration time, which is specified by the service policy, will be used. If the parameter is omitted, the notifications will continue until the maximum duration time, which is specified by the service policy, unless the notifications are stopped by deletion of subscription for notifications. This element MAY be given by the client during resource creation in order to signal the desired lifetime of the subscription. The server SHOULD return in this element the period of time for which the subscription will still be valid.
clientCorrelator	xsd:string	Yes	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 re-sending the message 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.
resourceURL	xsd:anyURI	Yes	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 imageShareNotificationSubscription of type ImageShareNotificationSubscription is allowed in request and/or response bodies.

Regarding the clientCorrelator field, the note in section 5.2.2.11 applies.

5.2.2.11 Type: ImageFileNotification

This type represents a notification delivering an image file URL. After the file has been received, the file URL will be sent in the ImageFileNotification to the application for retrieving the image file.

Element	Type	Optional	Description
callbackData	xsd:string	Yes	The 'callbackData' element if it was passed by the application in the 'callbackReference' element when creating a subscription to image share notifications. See [REST_NetAPI_Common].
link	common:Link [0..unbounded]	Yes	Links to other resources that are in relationship to the notification. The server MUST include links as defined for ImageFileNotification (See section 6.8 for detailed information). Further, the server SHOULD include a link to the related subscription.
fileURL	xsd:anyURI	No	The file repository URL from where the file can be retrieved

A root element named imageFileNotification of type ImageFileNotification is allowed in notification request bodies.

5.2.2.12 Type: SubscriptionCancellationNotification

This type represents a subscription cancellation notification.

Element	Type	Optional	Description
callbackData	xsd:string	Yes	CallbackData if passed by the application in the receiptRequest element during the associated subscription operation. See [REST_NetAPI_Common] for details.
reason	common:ServiceErr or	Yes	Reason notification is being discontinued. SHOULD be present if the reason is different from a regular expiry of the subscription.
link	common:Link[1..un bounded]	No	Link to other resources that are in relationship with the resource. There MUST be a link to the subscription that is cancelled. See section 6.10 for detailed information.

A root element named subscriptionCancellationNotification of type SubscriptionCancellationNotification is allowed in notification request bodies.

5.2.3 Enumerations

The subsections of this section define the enumerations used in the Image Share API.

5.2.3.1 Enumeration: EventType

This enumeration is used in notifications to describe the type of event which the notification is about.

Enumeration	Description
SessionCancelled	The Originator has cancelled the image share session during the invite phase.
SessionEnded	The image share session has ended.
Declined	The Receiver has declined the image share session invitation.
Successful	The image file has been successfully delivered.
Failed	The image share has failed due to errors.
Aborted	The image file delivery was aborted by the Originator.

5.2.3.2 Enumeration: SessionStatus

This enumeration defines the possible values to describe the status of an image share session.

Enumeration	Description
Initial	The image share session is in initial status, the Receiver is being invited to an image share session.
Connected	The image share session is in connected status.
Terminated	The image share session is terminated.

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):

- ImageShareSessionInformation
- ReceiverSessionStatus
- ImageShareNotificationSubscription

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

5.2.5 MIME multipart representation

In Image Share API, the session creation operation can contain actual file content in the HTTP requests. To represent such MIME multipart messages, “multipart/form-data” format is used, where the first entry of the form is the root fields and the second entry of the form is multimedia content. Details about the structure of such messages are defined in [REST_NetAPI_Common].

5.3 Sequence Diagrams

The following sub-sections describe the resources, methods and steps involved in typical scenarios.

A sequence diagram that contains a step that involves delivering a notification, the delivery can be via the HTTP POST method or via an operation labeled “NOTIFY”. The term “NOTIFY” refers to the use of the Notification Channel [REST_NetAPI_NotificationChannel].

Note that some of the following scenarios involve both an application of the Originator and an application of the Receiver, depending on the implementations, if the scenarios involving only the application of one participant (either Originator or Receiver), the following scenarios of the application of that particular participant apply.

5.3.1 Subscription and unsubscription to image share notifications

This figure below shows a scenario for an application subscribing to and unsubscribing from image share notifications.

The notification URL passed by the client during the subscription step can be a Client-side Notification URL, or a Server-side Notification URL. Refer to [REST_NetAPI_NotificationChannel] for sequence flows illustrating the creation of a Notification Channel and obtaining a Server-side Notification URL on the server-side, and its use by the client via Long Polling.

The resources:

- To subscribe to image share notifications, create a new resource under **http://{serverRoot}/imageshare/{apiVersion}/{userId}/subscriptions**
- To cancel subscription to image share notifications delete the resource under **http://{serverRoot}/imageshare/{apiVersion}/{userId}/subscriptions/{subscriptionId}**

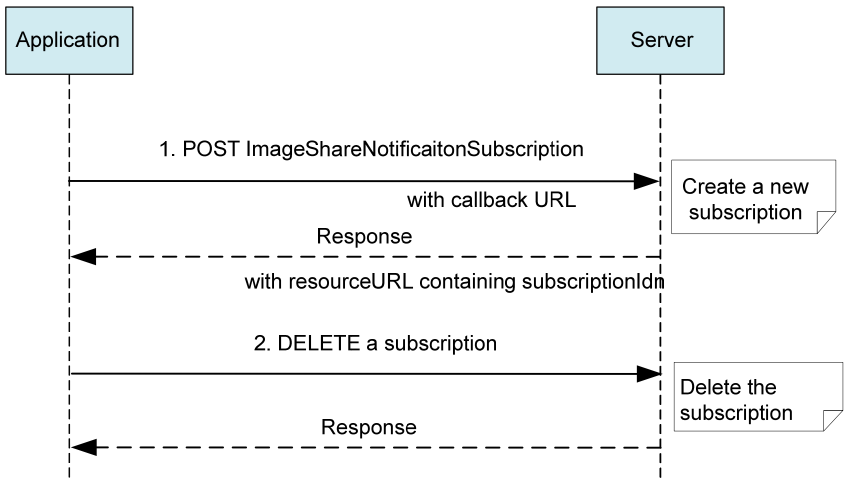
Application
Server
1. POST ImageShareNotificaitonSubscription
with callback URL
Create a new subscription
Response
with resourceURL containing subscriptionIdn
2. DELETE a subscription
Delete the subscription
Response

Figure 2 Subscribing to and unsubscribing from image share notifications

Outline of the flows:

1. An application subscribes to image share notifications using the POST method to submit the ImageShareNotificationSubscription data structure to the resource containing all subscriptions and receives the result resource URL containing the subscriptionId
2. The application stops receiving notifications using DELETE with the resource URL containing the subscriptionId

5.3.2 Normal flow of an image share session

The figure below shows a scenario for an image share session with successful result.

The resources:

- To start an image share session, create a new resource with the ImageShareSessionInformation data structure under **http://{serverRoot}/imageshare/{apiVersion}/{userId}/sessions**
- To accept an image share session invitation update the resource

  http://{serverRoot}/imageshare/{apiVersion}/{userId}/sessions/{sessionId}/status
- To end an image share session delete the resource **http://{serverRoot}/imageshare/{apiVersion}/{userId}/sessions/{sessionId}**

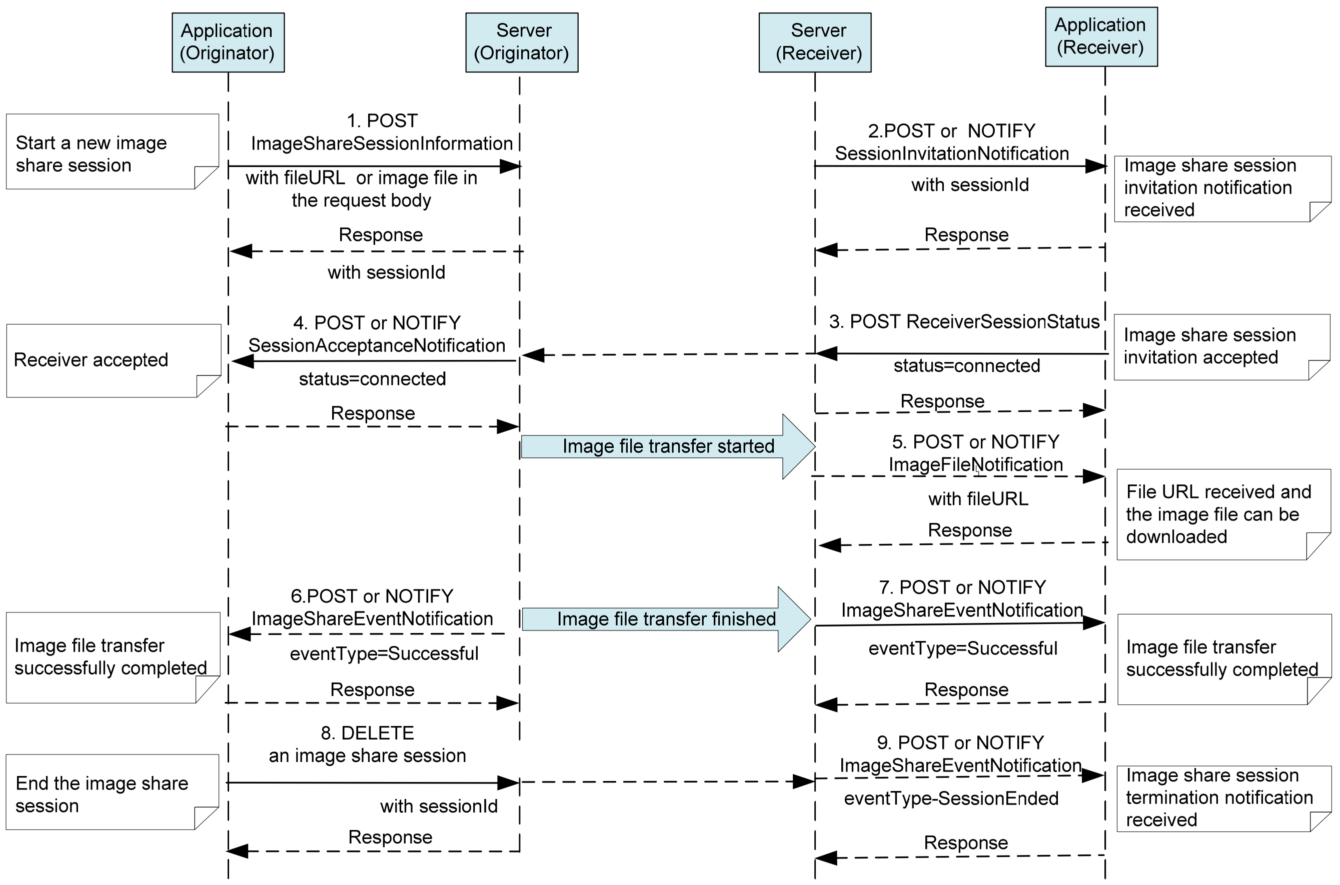
Application (Originator)
Server (Originator)
Server (Receiver)
Application (Receiver)
Start a new image share session
1. POST ImageShareSessionInformation
with fileURL or image file in the request body
Response
with sessionId
2.POST or NOTIFY SessionInvitationNotification
with sessionId
Image share session invitation notification received
Response
Receiver accepted
4. POST or NOTIFY SessionAcceptanceNotification
status=connected
Response
3. POST ReceiverSessionStatus
status=connected
Response
Image share session invitation accepted
Image file transfer started
5. POST or NOTIFY ImageFileNotification
with fileURL
Response
File URL received and the image file can be downloaded
Image file transfer successfully completed
6.POST or NOTIFY ImageShareEventNotification
eventType=Successful
Response
Image file transfer finished
7. POST or NOTIFY ImageShareEventNotification
eventType=Successful
Response
Image file transfer successfully completed
End the image share session
8. DELETE an image share session
with sessionId
Response
9. POST or NOTIFY ImageShareEventNotification
eventType-SessionEnded
Response
Image share session termination notification received

Figure 3 Image share session with successful result

Outline of the flows:

1. An application of the Originator starts an image share session using the POST method to submit the ImageShareSessionInformation data structure to the resource containing all image share sessions. Thereby the creation of a new image share session resource is triggered and the application of the Originator receives the resulting resource URL containing the sessionId.
2. An application of the Receiver receives an image share session invitation notification.

3. The application of the Receiver accepts the image share session invitation using the POST method to submit the ReceiverSessionStatus data structure to the resource containing the session status. The status MUST be set to “Connected”.
4. The application of the Originator receives a notification with SessionAcceptanceNotification data structure indicating the Receiver has accepted the invitation. The application of the Originator starts to transfer the image file.
5. When the image file is ready for retrieval, the server of the Receiver notifies the application of the Receiver using ImageFileNotification data structure containing the fileURL. The application of the Receiver can start downloading the image file using the file URL received.

Note: Depending on the implementations, the notification of the file URL can be sent after the first chunk of data is received or when the complete image file has been received (i.e. after step 7).

Note: How the application retrieves the image file using the file URL is out of scope.

6. After the image file transfer is completed, the application of the Originator receives a notification containing the ImageShareEventNotification data structure indicating that the image file transfer is successful.
7. At the mean time, the application of the Receiver also receives a notification containing the ImageShareEventNotification data structure indicating that the image file transfer is successful.
8. The application of the Originator ends the image share session using DELETE method on the resource URL of the session with sessionId

Note: Both the application of the Originator and application of the Receiver can initiate ending the image share session.

9. The application of the Receiver receives a notification containing the ImageShareEventNotification data structure indicating that the session has been ended.

Note: In case of the application of the Receiver ends the image share session, the application of the Originator receives a notification containing the ImageShareEventNotification data structure indicating that the session has been ended.

5.3.3 Image share session failure

There are different causes which may lead to image share session failed, following are some options (not exclusive list):

a. The application of the Originator cancels the image share session.
b. The application of the Receiver reject or decline the image share session invitation
c. The application of the Originator aborts the image file transfer.
d. The image share fails due to errors

5.3.3.1 Cancelling an image share invitation

The figure below shows a scenario for an application of the Originator to cancel an image share session invitation.

The resources:

- To cancel an image share session invitation delete the session resource **http://{serverRoot}/imageshare/{apiVersion}/{userId}/sessions/{sessionId}**

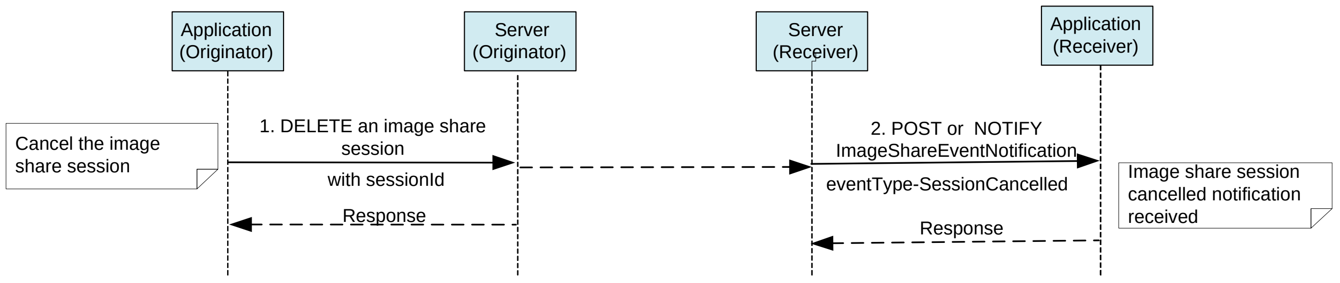
Application (Originator)
Server (Originator)
Server (Receiver)
Application (Receiver)
Cancel the image share session
1. DELETE an image share session
with sessionId
Response
2. POST or NOTIFY ImageShareEventNotification
eventType-SessionCancelled
Response
Image share session cancelled notification received

Figure 4 Cancelling an image share invitation

Outline of the flows:

An application of the Originator has created an image share session resource triggering an image share invitation sent to the application of the Receiver (Refer to step 1 and step 2 in 5.3.2). Subsequently:

1. The application of the Originator cancels an image share session invitation using the DELETE method on the resource URL of the session with sessionId and receives a response weather the request was successfully initiated.
2. An ImageShareEventNotification is sent to the application of the Receiver when the image share session has been cancelled, then the session is turned down.

Note that cancelling a session only works before the Receiver has accepted the image share invitation.

5.3.3.2 Declining an image share session invitation

The figure below shows a scenario for an application to decline an image share session invitation.

The resources:

- To decline an image share session invitation delete the session resource **http://{serverRoot}/image share/{apiVersion}/{userId}/sessions/{sessionId}**

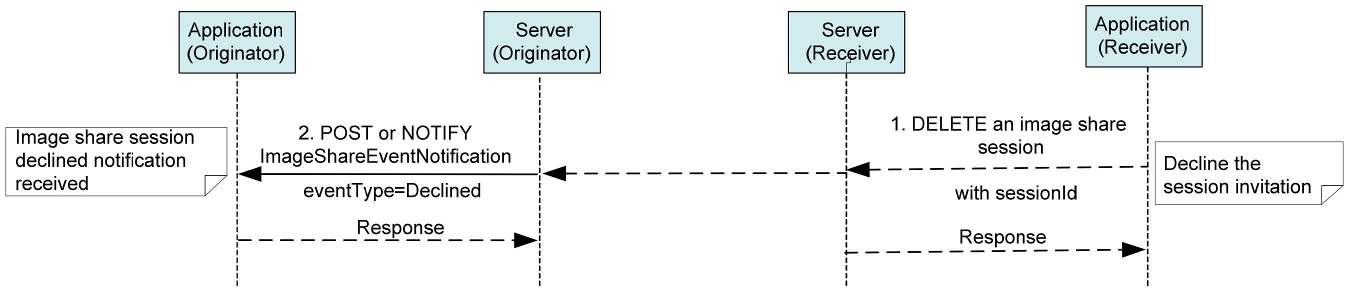
Application (Originator)
Server (Originator)
Server (Receiver)
Application (Receiver)
Image share session declined notification received
2. POST or NOTIFY ImageShareEventNotification
eventType=Declined
Response
1. DELETE an image share session
with sessionId
Response
Decline the session invitation

Figure 5 Declining an image share session invitation

Outline of the flows:

An application of the Originator has created an image share session resource triggering an image share invitation sent to the application of the Receiver (Refer to step 1 and step 2 in 5.3.2). Subsequently:

1. The application of the Receiver declines the image share session invitation using the DELETE method on the session resource including the sessionId

2. The application of the Originator receives a notification containing the ImageShareEventNotification data structure indicating that the Receiver has declined the invitation, then the session is turned down.

Note that declining a session only works before the Receiver has accepted the image share invitation.

5.3.3.3 Image file transfer aborted

The figure below shows a scenario for image file transfer aborted.

The resources:

- To abort an image share session delete the resource **http://{serverRoot}/imageshare/{apiVersion}/{userId}/sessions/{sessionId}**

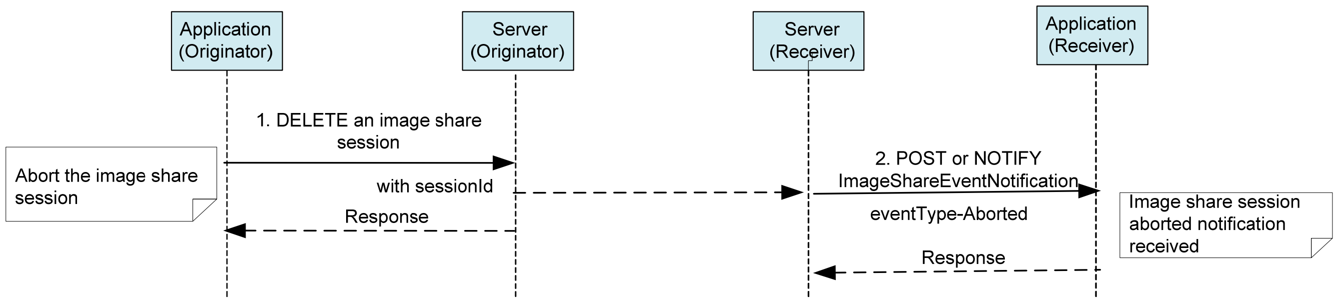
Application (Originator)
Server (Originator)
Server (Receiver)
Application (Receiver)
1. DELETE an image share session
with sessionId
Abort the image share session
Response
2. POST or NOTIFY ImageShareEventNotification
eventType-Aborted
Image share session aborted notification received
Response

Figure 6 Image file transfer aborted

Outline of the flows:

After an application of the Originator creates an image share session resource and the application of the Receiver accepts the image share session invitation (Refer to step 1 to step 4 in 5.3.2), the image file transfer is started, subsequently:

1. The application of the Originator can abort the image file transfer using DELETE method on the resource URL of the session with sessionId
2. The application of the Receiver receives an ImageShareEventNotification data structure indicating the image file transfer has been aborted and the session is turned down.

Note that aborting the image file transfer only works before the image file has been completely transferred. After that, the DELETE method leads to a normal ending of the session.

Note: When the image file transfer has been aborted, the application of the Receiver may already received the file URL and started to fetch the file, the server of the Receiver should cancel any HTTP request downloading the image file using the file URL and disable the file URL. How the server implements this is out of scope.

5.3.3.4 Image share fails

The figure below shows a scenario for the failure of an image share session.

The resources:

- applications supplied

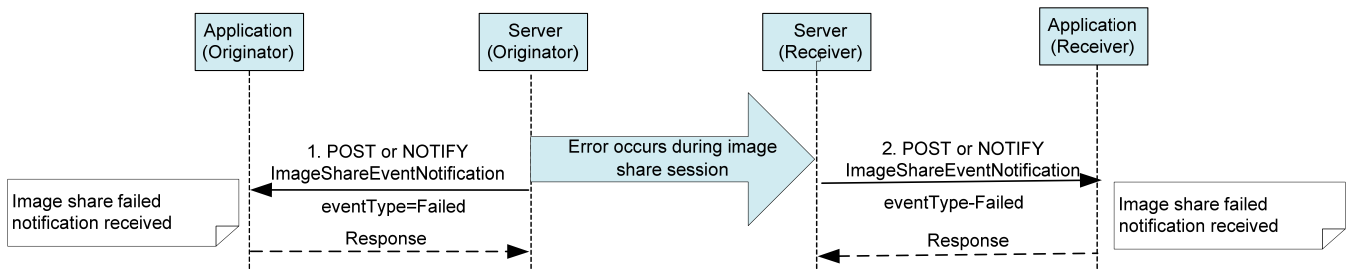
Application (Originator)
Server (Originator)
Server (Receiver)
Application (Receiver)
1. POST or NOTIFY ImageShareEventNotification
eventType=Failed
Response
Error occurs during image share session
2. POST or NOTIFY ImageShareEventNotification
eventType-Failed
Response
Image share failed notification received
Image share failed notification received

Figure 7 Image share fails

Outline of the flows:

1. When error occurs during an image share session, the application of the Originator receives a notification containing the ImageShareEventNotification data structure indicating that the image share has failed.
2. At the mean time, the application of the Receiver also receives a notification containing the ImageShareEventNotification data structure indicating that the image share has failed.

Note: When error occurs during the image file transfer, the application of the Receiver may already received the file URL and started to fetch the file, the server of the Receiver should cancel any HTTP request downloading the image file using the file URL and disable the file URL. How the server implements this is out of scope.

6. Detailed specification of the resources

The following applies to all resources defined in this specification regardless of the representation format (i.e. XML, JSON, application /x-www-form-urlencoded):

- 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 and the leading "+" sign 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.
    - o The ACR 'Authorization' 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, and MAY support application/www-form-urlencoded 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: All subscriptions to image share notifications

The resource used is:

http://{serverRoot}/imageshare/{apiVersion}/{userId}/subscriptions

This resource is used to manage subscriptions to image share notifications. Note that there is one subscription per client instance.

This resource can be used in conjunction with a Client-side Notification URL, or in conjunction with a Server-side Notification URL. In this latter case, the application MUST first create a Notification Channel (see [REST_NetAPI_NotificationChannel]) before creating a subscription.

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 the RESTful image share API, see section 7.

6.1.3 GET

This operation is used for reading the list of active image share notification subscriptions.

6.1.3.1 Example 1: Reading all active image share notification subscriptions (Informative)

6.1.3.1.1 Request

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

6.1.3.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Thu, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareSubscriptionList xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
    <imageShareNotificationSubscription>
        <callbackReference>
            <notifyURL>http://application.example.com/imageshare/notifications/77777</notifyURL>
            <callbackData>abcd</callbackData>
        </callbackReference>
        <duration>7200</duration>
        <clientCorrelator>12345</clientCorrelator>
        <resourceURL>http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001</resourceURL>
    </imageShareNotificationSubscription>
    <resourceURL>http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions</resourceURL>
</is:imageShareSubscriptionList>
```

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 [RFC2616].

6.1.5 POST

This operation is used to create a new subscription for image share notifications..

The notifyURL in the callbackReference either contains the Client-side Notification URL (as defined by the client) or the Server-side Notification URL (as obtained during the creation of the Notification Channel [REST_NetAPI_NotificationChannel]).

6.1.5.1 Example 1: Creating a new subscription to image share notifications using tel URI (Informative)

6.1.5.1.1 Request

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

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareNotificationSubscription xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
    <callbackReference>
        <notifyURL>http://application.example.com/imageshare/notifications/77777</notifyURL>
        <callbackData>abcd</callbackData>
    </callbackReference>
    <duration>7200</duration>
    <clientCorrelator>12345</clientCorrelator>
</is:imageShareNotificationSubscription>
```

6.1.5.1.2 Response

```
HTTP/1.1 201 Created
Content-Type: application/xml
Content-Length: nnnn
Location: http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001
Date: Thu, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareNotificationSubscription xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
   <callbackReference>
      <notifyURL>http://application.example.com/imageshare/notifications/77777</notifyURL>
      <callbackData>abcd</callbackData>
   </callbackReference>
   <duration>7200</duration>
   <clientCorrelator>12345</clientCorrelator>
   <resourceURL>http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001</resourceURL>
</is:imageShareNotificationSubscription>
```

Note that alternatively to returning a copy of the created resource, the location of created resource could be returned using the common:resourceReference root element (see section 6.1.5.2.2).

6.1.5.2 Example 2: Creating a new subscription to image share notifications using ACR (Informative)

6.1.5.2.1 Request

```
POST  /exampleAPI/imageshare/v1/acr%3Apseudonym123/subscriptions/ HTTP/1.1
Content-Type: application/xml
```

```
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareNotificationSubscription xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
    <callbackReference>
        <notifyURL>http://application.example.com/imageshare/notifications/77777</notifyURL>
        <callbackData>abcd</callbackData>
    </callbackReference>
    <duration>7200</duration>
    <clientCorrelator>12345</clientCorrelator>
</is:imageShareNotificationSubscription>
```

6.1.5.2.2 Response

```
HTTP/1.1 201 Created
Content-Type: application/xml
Content-Length: nnnn
Location: http://example.com/exampleAPI/imageshare/v1/acr%3Apseudonym123/subscriptions/sub001
Date: Thu, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<common:resourceReference xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <resourceURL>http://example.com/exampleAPI/imageshare/v1/acr%3Apseudonym123/subscriptions/sub001</resourceURL>
</common:resourceReference>
```

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 [RFC2616].

6.2 Resource: Individual subscription to image share notifications

The resource used is:

http://{serverRoot}/imageshare/{apiVersion}/{userId}/subscriptions/{subscriptionId}

This resource represents an individual subscription to image share notifications.

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 clients want 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
subscriptionId	Identifier of the subscription

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 the RESTful image share API, see section 7.

6.2.3 GET

This operation is used for reading an individual subscription.

6.2.3.1 Example: Reading an individual subscription (Informative)

This example shows also an alternative way to indicate desired content type in response from the server, by using URL query parameter "?resFormat" which is described in [REST_NetAPI_Common].

6.2.3.1.1 Request

```
GET /exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001?resFormat=XML HTTP/1.1

Host: example.com
```

6.2.3.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Mon, 28 Jun 2010 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareNotificationSubscription xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
    <callbackReference>
        <notifyURL>http://application.example.com/imageshare/notifications/77777</notifyURL>
        <callbackData>abcd</callbackData>
    </callbackReference>
    <duration>7200</duration>
    <clientCorrelator>12345</clientCorrelator>
    <resourceURL>http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001</resourceURL>
</is:imageShareNotificationSubscription>
```

6.2.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET, DELETE' field in the response as per section 14.7 of [RFC2616].

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, DELETE' field in the response as per section 14.7 of [RFC2616].

6.2.6 DELETE

This operation is used to cancel a subscription and to stop corresponding notifications.

6.2.6.1 Example: Cancelling a subscription (Informative)

6.2.6.1.1 Request

```
DELETE /exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001 HTTP/1.1
Accept: application/xml
Host: example.com
```

6.2.6.1.2 Response

```
HTTP/1.1 204 No Content
Date: Mon, 28 Jun 2010 17:51:59 GMT
```

6.3 Resource: All image share sessions

The resource used is:

http://{serverRoot}/imageshare/{apiVersion}/{userId}/sessions

This resource represents the active image share sessions for a particular user.

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 clients want 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.

6.3.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful image share API, see section 7 .

6.3.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response as per section 14.7 of [RFC2616].

6.3.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response as per section 14.7 of [RFC2616].

6.3.5 POST

This operation is used to create a new image share session.

For the source of the image file, this operation can either include the actual file content or an file repository URL link to the image file.

This operation supports the image share session with or without CS call.

6.3.5.1 Example 1: Creating a new image share session with file content (no CS call related) (Informative)

6.3.5.1.1 Request

```
POST /exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions HTTP/1.1
Content-Type: multipart/form-data; boundary="===============123456==";
Content-Length: nnnn
Accept: application/xml
Host: example.com
MIME-Version: 1.0

--===============123456==
Content-Disposition: form-data; name="root-fields"
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareSessionInformation xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
  <originatorAddress>tel:+19585550100</originatorAddress >
  <originatorName>Alice</originatorName>
  <receiverAddress>tel:+19585550101</receiverAddress>
  <receiverName>Bob</receiverName>
  <fileInformation>
    <fileSelector>
      <type>image/jpeg</type>
      <name>sunset.jpg</name>
      <size>4096</size>
      <hash>
        <algorithm>sha-1</algorithm>
        <value>58231FE8653BBCF371362F86D471913EE4B1DF2F</value>
      </hash>
    </fileSelector>
    <fileDescription>This is my latest picture</fileDescription>
   </fileInformation>
  <clientCorrelator>12345</clientCorrelator>
</is:imageShareSessionInformation>

--===============123456==

Content-Disposition: form-data; name=”attachment”; filename=”sunset.jpg”
Content-Type: image/jpeg
Content-Length: nnnn

JPEG ...binary image data...

--===============123456==--
```

6.3.5.1.2 Response

```
HTTP/1.1 201 Created
Content-Type: application/xml
Content-Length: nnnn
Location: http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001
Date: Mon, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareSessionInformation xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
   <originatorAddress>tel:+19585550100</originatorAddress>
  <originatorName>Alice</originatorName>
  <receiverAddress>tel:+19585550101</receiverAddress>
  <receiverName>Bob</receiverName>
 <status>Initial</status>
  <fileInformation>
    <fileSelector>
      <type>image/jpeg</type>
      <name>sunset.jpg</name>
      <size>4096</size>
      <hash>
        <algorithm>sha-1</algorithm>
        <value>58231FE8653BBCF371362F86D471913EE4B1DF2F</value>
      </hash>
    </fileSelector>
    <fileDescription>This is my latest picture</fileDescription>
   </fileInformation>
<clientCorrelator>12345</clientCorrelator>
<resourceURL>http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001</resourceURL>
</is:imageShareSessionInformation>
```

Note that alternatively to returning a copy of the created resource, the location of created resource could be returned using the common:resourceReference root element (see section 6.1.5.2.2).

6.3.5.2 Example 2: Creating a new image share session with file repository URL (CS call related) (Informative)

6.3.5.2.1 Request

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

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareSessionInformation xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
   <originatorAddress>tel:+19585550100</originatorAddress>
   <originatorName>Alice</originatorName>
  <callObjectRef>http://example.com/exampleAPI/call/tel%3A%2B19585550101/sessions/callSess001</callObjectRef>
   <fileInformation>
     <fileSelector>
       <type>image/jpeg</type>
       <name>sunset.jpg</name>
       <size>4096</size>
       <hash>
         <algorithm>sha-1</algorithm>
         <value>58231FE8653BBCF371362F86D471913EE4B1DF2F</value>
       </hash>
     </fileSelector>
     <fileDescription>This is my latest picture</fileDescription>
   </fileInformation>
   <fileURL>http://mypictures.com/album/sunset.jpg</fileURL>
   <clientCorrelator>12345</clientCorrelator>
</is:imageShareSessionInformation>
```

6.3.5.2.2 Response

```
HTTP/1.1 201 Created
Content-Type: application/xml
Content-Length: nnnn
Location: http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001
Date: Mon, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareSessionInformation xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
   <originatorAddress>tel:+19585550100</originatorAddress>
   <originatorName>Alice</originatorName>
   <receiverAddress>tel:+19585550101</receiverAddress>
  <callObjectRef>http://example.com/exampleAPI/call/tel%3A%2B19585550101/sessions/callSess001</callObjectRef>
  <receiverName>Bob</receiverName>
  <status>Initial</status>
  <fileInformation>
     <fileSelector>
       <type>image/jpeg</type>
       <name>sunset.jpg</name>
       <size>4096</size>
       <hash>
         <algorithm>sha-1</algorithm>
```

```
        <value>58231FE8653BBCF371362F86D471913EE4B1DF2F</value>
      </hash>
    </fileSelector>
    <fileDescription>This is my latest picture</fileDescription>
  </fileInformation>
   <fileURL>http://mypictures.com/album/sunset.jpg</fileURL>
  <clientCorrelator>12345</clientCorrelator>
  <resourceURL>http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001</resourceURL>
</is:imageShareSessionInformation>
```

Note that alternatively to returning a copy of the created resource, the location of created resource could be returned using the common:resourceReference root element (see section 6.1.5.2.2).

6.3.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response as per section 14.7 of [RFC2616].

6.4 Resource: Individual image share session

The resource used is:

http://{serverRoot}/imageshare/{apiVersion}/{userId}/sessions/{sessionId}

This resource represents an image share session.

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 clients want 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
sessionId	Identifier of the session

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 the RESTful image share API, see section 7.

6.4.3 GET

This operation is used to retrieve image share session information.

6.4.3.1 Example 1: Retrieving an image share session information (no CS call related) (Informative)

6.4.3.1.1 Request

```
GET /exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001 HTTP/1.1
Accept: application/xml
Host: example.com
```

6.4.3.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Mon, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareSessionInformation xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
   <originatorAddress>tel:+19585550100</originatorAddress>
   <originatorName>Alice</originatorName>
   <receiverAddress>tel:+19585550101</receiverAddress>
  <receiverName>Bob</receiverName>
  <status>Connected</status>
  <fileInformation>
     <fileSelector>
       <type>image/jpeg</type>
       <name>sunset.jpg</name>
       <size>4096</size>
       <hash>
         <algorithm>sha-1</algorithm>
         <value>58231FE8653BBCF371362F86D471913EE4B1DF2F</value>
       </hash>
     </fileSelector>
     <fileDescription>This is my latest picture</fileDescription>
   </fileInformation>
   <clientCorrelator>12345</clientCorrelator>
   <resourceURL>http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001</resourceURL>
</is:imageShareSessionInformation>
```

6.4.3.2 Example 2: Retrieving an image share session information(CS call related) (Informative)

6.4.3.2.1 Request

```
GET /exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001 HTTP/1.1
Accept: application/xml
Host: example.com
```

6.4.3.2.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Mon, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareSessionInformation xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
   <originatorAddress>tel:+19585550100</originatorAddress>
   <originatorName>Alice</originatorName>
   <receiverAddress>tel:+19585550101</receiverAddress>
  <callObjectRef>http://example.com/exampleAPI/call/tel%3A%2B19585550101/sessions/callSess001</callObjectRef>
  <receiverName>Bob</receiverName>
  <status>Connected</status>
  <fileInformation>
     <fileSelector>
       <type>image/jpeg</type>
       <name>sunset.jpg</name>
       <size>4096</size>
       <hash>
         <algorithm>sha-1</algorithm>
         <value>58231FE8653BBCF371362F86D471913EE4B1DF2F</value>
       </hash>
     </fileSelector>
     <fileDescription>This is my latest picture</fileDescription>
   </fileInformation>
    <fileURL>http://mypictures.com/album/sunset.jpg</fileURL>
   <clientCorrelator>12345</clientCorrelator>
   <resourceURL>http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001</resourceURL>
</is:imageShareSessionInformation>
```

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, DELETE' field in the response as per section 14.7 of [RFC2616].

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, DELETE' field in the response as per section 14.7 of [RFC2616].

6.4.6 DELETE

This operation ends an image share session.

It is used by an application of the Originator to cancel an image share session before the session invitation has been accepted.

It is used by an application of the Receiver to decline an image share session after the session invitation has been accepted.

It is used by an application of the Originator or an application of the Receiver to terminate an image share session.

6.4.6.1 Example: Terminating an image share session (Informative)

6.4.6.1.1 Request

```
DELETE /exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001 HTTP/1.1
Accept: application/xml
Host: example.com
```

6.4.6.1.2 Response

```
HTTP/1.1 204 No Content
Date: Mon, 28 Jul 2011 17:51:59 GMT
```

6.5 Resource: Individual image share session status

The resource used is:

http://{serverRoot}/imageshare/{apiVersion}/{userId}/sessions/{sessionId}/status

This resource represents the status of the image session and is used for accepting an image share invitation, by means of updating the status.

6.5.1 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 clients want 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
sessionId	Identifier of the session

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

6.5.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful image share API, see section 7.

6.5.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 14.7 of [RFC2616].

6.5.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response as per section 14.7 of [RFC2616].

6.5.5 POST

This operation is used for accepting an image share invitation, by means of updating the status.

6.5.5.1 Example1: Accepting an image share invitation with acceptance of the image file (Informative)

6.5.5.1.1 Request

```
POST /exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001/status HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:receiverSessionStatus xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
    <status>Connected</status>
    <fileAcceptance>true</fileAcceptance>
</is:receiverSessionStatus>
```

6.5.5.1.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.5.5.2 Example2: Accepting an image share invitation without accepting image file (Informative)

6.5.5.2.1 Request

```
POST /exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001/status HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:receiverSessionStatus xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
  <status>Connected</status>
  <fileAcceptance>false</fileAcceptance>
 </is:receiverSessionStatus>
```

6.5.5.2.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.5.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response as per section 14.7 of [RFC2616].

6.6 Resource: Client notification about image share session invitation

This resource is a callback URL provided by the client for notification about image share session invitations. The RESTful Image Share API does not make any assumption about the structure of this URL. If this URL is a Client-side Notification URL, the server will POST notifications directly to it. If this URL is a Server-side Notification URL, the server uses it to determine the address of the Notification Server to which the notifications will subsequently be POSTed. The way the server determines the address of the Notification Server is out of scope of this specification.

Note: In the case when the client has set up a Notification Channel in order to use Long Polling to obtain the notifications, in order to retrieve the notifications, the client needs to use the Long Polling mechanism described in [REST_NetAPI_NotificationChannel], instead of the mechanism described below in section 6.6.5.

The following table gives detailed information about image share session invitation notification. It is also outlined which image share participant receives notifications of a particular type, whether a response is needed, and which links to other resources are contained in that notification.

In the "Notification sent to" column, the following values can occur:

- Originator: the Originator of the image share session
- Receiver: one individual Receiver in the image share session at a time
- All: Receiver and Originator of the image share session

Notification Root Element Type	Notification sent to	Response to Notification	Link rel	Link href Base URL: http://{serverRoot}/imageshare/{apiVersion}/{userId}/sessions
SessionInvitationNotification	Receiver	decline accept	ImageShareSessionInformation ReceiverSessionStatus	/{sessionId} /{sessionId}/status

Table 2: Image share session invitation notification

The application of Receiver can accept the request by updating the status, which is addressed by the URL passed in the "href" attribute of the "link" element with rel=" ReceiverSessionStatus".

Typically, this is **http://{serverRoot}/imageshare/{apiVersion}/{userId}/sessions/{sessionId}/status.**

The application of Receiver can decline the request by sending a DELETE request to one the URL passed in the "href" attribute of the "link" element with rel="ImagaeShareSessionInformation".

Typically, this is **http://{serverRoot}/imageshare/{apiVersion}/{userId}/sessions/{sessionId}**

If the application of Receiver fails to react within a time interval defined by service policies, the session invitation will time out and the session will terminate.

6.6.1 Request URL variables

Client provided if any.

6.6.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

6.6.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response as per section 14.7 of [RFC2616].

6.6.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response as per section 14.7 of [RFC2616].

6.6.5 POST

This operation is used to notify the client about image share session invitations.

6.6.5.1 Example1: Notify a client about image share session invitations (no CS call related) (Informative)

6.6.5.1.1 Request

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: application.example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:sessionInvitationNotification xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
  <callbackData>abcd</callbackData>
  <link rel="ImageShareSessionInformation"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001"/>
  <link rel="ReceiverSessionstatus"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001/status"/>
  <link rel="ImageShareNotificationSubscription"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001"/>
  <originatorAddress>tel:+19585550100</originatorAddress>
  <originatorName>Alice</originatorName>
  <receiverAddress>tel:+19585550101</receiverAddress>
  <receiverName>Bob</receiverName>
  <fileInformation>
    <fileSelector>
      <type>image/jpeg</type>
      <name>sunset.jpg</name>
      <size>4096</size>
      <hash>
        <algorithm>sha-1</algorithm>
        <value>58231FE8653BBCF371362F86D471913EE4B1DF2F</value>
      </hash>
    </fileSelector>
    <fileDescription>This is my latest picture</fileDescription>
  </fileInformation>
</is:sessionInvitationNotification>
```

6.6.5.1.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.6.5.2 Example1: Notify a client about image share session invitations (CS call related) (Informative)

6.6.5.2.1 Request

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: application.example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:sessionInvitationNotification xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
  <callbackData>abcd</callbackData>
  <link rel="ImageShareSessionInformation"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001"/>
  <link rel="ReceiverSessionstatus"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001/status"/>
  <link rel="ImageShareNotificationSubscription"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001"/>
  <originatorAddress>tel:+19585550100</originatorAddress>
  <originatorName>Alice</originatorName>
  <receiverAddress>tel:+19585550101</receiverAddress>
  <callObjectRef>http://example.com/exampleAPI/call/tel%3A%2B19585550101/sessions/callSess001</callObjectRef>
  <receiverName>Bob</receiverName>
  <fileInformation>
    <fileSelector>
      <type>image/jpeg</type>
      <name>sunset.jpg</name>
      <size>4096</size>
      <hash>
        <algorithm>sha-1</algorithm>
        <value>58231FE8653BBCF371362F86D471913EE4B1DF2F</value>
      </hash>
    </fileSelector>
    <fileDescription>This is my latest picture</fileDescription>
  </fileInformation>
</is:sessionInvitationNotification>
```

6.6.5.2.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.6.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response as per section 14.7 of [RFC2616].

6.7 Resource: Client notification about image share session acceptance

This resource is a callback URL provided by the client for notification about image share session acceptance. The RESTful Image Share API does not make any assumption about the structure of this URL. If this URL is a Client-side Notification URL, the server will POST notifications directly to it. If this URL is a Server-side Notification URL, the server uses it to determine the address of the Notification Server to which the notifications will subsequently be POSTed. The way the server determines the address of the Notification Server is out of scope of this specification.

Note: In the case when the client has set up a Notification Channel in order to use Long Polling to obtain the notifications, in order to retrieve the notifications, the client needs to use the Long Polling mechanism described in [REST_NetAPI_NotificationChannel], instead of the mechanism described below in section 6.7.5.

The following table gives detailed information about image share session acceptance notification. It is also outlined which image share participant receives notifications of a particular type, whether a response is needed, and which links to other resources are contained in that notification.

In the "Notification sent to" column, the following values can occur:

- Originator: the Originator of the image share session
- Receiver: one individual Receiver in the image share session at a time
- All: Receiver and Originator of the image share session

Notification Root Element Type	Notification sent to	Response to Notification	Link rel	Link href Base URL: http://{serverRoot}/ imageshare/{apiVersion}/ {userId}/sessions
SessionAcceptanceNotification	Originator	n/a	ImageShareSessionInformation	/{sessionId}/{sessionId}

Table 3: Image share session acceptance notification

6.7.1 Request URL variables

Client provided if any.

6.7.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

6.7.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 14.7 of [RFC2616].

6.7.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 14.7 of [RFC2616].

6.7.5 POST

This operation is used to notify the client about image share session acceptance.

6.7.5.1 Example 1: Notify a client about the acceptance of an image share session (no CS call related) (Informative)

6.7.5.1.1 Request

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: application.example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:sessionAcceptanceNotification xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
  <callbackData>abcd</callbackData>
   <link rel="ImageShareSessionInformation"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001"/>
<link rel="ImageShareNotificationSubscription"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001"/>
 <receiverAddress>tel:+19585550101</receiverAddress>
  <receiverName>Bob</receiverName>
  <receiverSessionStatus>
      <status>Connected</status>
     <fileAcceptance>true</fileAcceptance>
  </receiverSessionStatus>
</is:sessionAcceptanceNotification>
```

6.7.5.1.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.7.5.2 Example 2: Notify a client about the acceptance of an image share session (CS call related) (Informative)

6.7.5.2.1 Request

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: application.example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:sessionAcceptanceNotification xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
   <callbackData>abcd</callbackData>
   <link rel="ImageShareSessionInformation"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101sessions/sess001"/>
   <link rel="ImageShareNotificationSubscription"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001"/>
  <receiverAddress>tel:+19585550101</receiverAddress>
  <callObjectRef>http://example.com/exampleAPI/call/tel%3A%2B19585550101/sessions/callSess001</callObjectRef>
  <receiverName>Bob</receiverName>
  <receiverSessionStatus>
      <status>Connected</status>
      <fileAcceptance>false</fileAcceptance>
  </receiverSessionStatus>
</is:sessionAcceptanceNotification>
```

6.7.5.2.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.7.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 14.7 of [RFC2616].

6.8 Resource: Client notification about the image file link

This resource is a callback URL provided by the client for notification about image file link. The RESTful Image Share API does not make any assumption about the structure of this URL. If this URL is a Client-side Notification URL, the server will POST notifications directly to it. If this URL is a Server-side Notification URL, the server uses it to determine the address of the Notification Server to which the notifications will subsequently be POSTed. The way the server determines the address of the Notification Server is out of scope of this specification.

Note: In the case when the client has set up a Notification Channel in order to use Long Polling to obtain the notifications, in order to retrieve the notifications, the client needs to use the Long Polling mechanism described in [REST_NetAPI_NotificationChannel], instead of the mechanism described below in section 6.8.5.

The following table gives detailed information about image share session acceptance notification. It is also outlined which image share participant receives notifications of a particular type, whether a response is needed, and which links to other resources are contained in that notification.

In the "Notification sent to" column, the following values can occur:

- Originator: the Originator of the image share session
- Receiver: one individual Receiver in the image share session at a time
- All: Receiver and Originator of the image share session

Notification Root Element Type	Notification sent to	Response to Notification	Link rel	Link href Base URL: http://{serverRoot}/ imageshare/{apiVersion}/ {userId}/sessions
ImageFileNotification	Receiver	n/a	ImageShareSessionInformation	/{sessionId}/{sessionId}

Table 4: Image share image file link notification

After the Receiver has accepted the session invitation, the server will send an ImageFileNotification to the application of Receiver which includes the link to the actual file on content repository.

6.8.1 Request URL variables

Client provided if any.

6.8.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

6.8.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response as per section 14.7 of [RFC2616].

6.8.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response as per section 14.7 of [RFC2616].

6.8.5 POST

This operation is used to notify the client about image file availability.

6.8.5.1 Example 1: Notify a client about the image file link (Informative)

6.8.5.1.1 Request

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: application.example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:imageFileNotification xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
```

```
  <link rel="ImageShareSessionInformation"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001"/>
   <link rel="ImageShareNotificationSubscription"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001"/>
  <fileURL>http://example.com/exampleAPI/imageshare/tel%3A%2B19585550100/album/sunset.jpg</fileURL>
</is:imageFileNotification>
```

6.8.5.1.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.8.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 14.7 of [RFC2616].

6.9 Resource: Client notification about image share events

This resource is a callback URL provided by the client for notification about image share events. The RESTful Image Share API does not make any assumption about the structure of this URL. If this URL is a Client-side Notification URL, the server will POST notifications directly to it. If this URL is a Server-side Notification URL, the server uses it to determine the address of the Notification Server to which the notifications will subsequently be POSTed. The way the server determines the address of the Notification Server is out of scope of this specification.

Note: In the case when the client has set up a Notification Channel in order to use Long Polling to obtain the notifications, in order to retrieve the notifications, the client needs to use the Long Polling mechanism described in [REST_NetAPI_NotificationChannel], instead of the mechanism described below in section 6.9.5.

The following table gives an overview of the image share events notifications. It is also outlined which image share participant receives notifications of a particular type, whether a response is needed, and which links to other resources are contained in that notification.

In the “Notification sent to” column, the following values can occur:

- Originator: the Originator of the image share session
- Receiver: one individual Receiver in the image share session at a time
- All: Receiver and Originator of the image share session

EventType	Notification Root Element Type	Notification sent to	Response to Notification	Link rel	Link href Base URL: http://{serverRoot}/ imageshare/{apiVersion}/ {userId}/sessions
Declined	ImageShareEventNotification	Originator	n/a	ImageShareSessionInformation	/{sessionId}

SessionCancelled	ImageShareEventNotification	Receiver	n/a	ImageShareSessionInformation	/{sessionId}
SessionEnded	ImageShareEventNotification	All	n/a	ImageShareSessionInformation	/{sessionId}
Successful	ImageShareEventNotification	All	n/a	ImageShareSessionInformation	/{sessionId}
Failed	ImageShareEventNotification	All	n/a	ImageShareSessionInformation	/{sessionId}
Aborted	ImageShareEventNotification	Receiver	n/a	ImageShareSessionInformation	/{sessionId}

Table 5: Image share events notification

6.9.1 Request URL variables

Client provided if any.

6.9.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

6.9.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 14.7 of [RFC2616].

6.9.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 14.7 of [RFC2616].

6.9.5 POST

This operation is used to notify the client about image share events.

6.9.5.1 Example 1: Notify a client about image share event (ended) (Informative)

6.9.5.1.1 Request

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: application.example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareEventNotification xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
   <callbackData>abcd</callbackData>
   <link rel="ImageShareSessionInformation"
```

```
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001"/>
   <link rel="ImageShareNotificationSubscription"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001"/>
  <eventType>SessionEnded</eventType>
</is:imageShareEventNotification>
```

6.9.5.1.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.9.5.2 Example 2: Notify a client about image share event (declined) (Informative)

6.9.5.2.1 Request

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: application.example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareEventNotification xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
   <callbackData>abcd</callbackData>
  <link rel="ImageShareSessionInformation"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001"/>
   <link rel="ImageShareNotificationSubscription"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001"/>
 <eventType>Declined</eventType>
</is:imageShareEventNotification>
```

6.9.5.2.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.9.5.3 Example 3: Notify a client about image share event (cancelled) (Informative)

6.9.5.3.1 Request

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: application.example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareEventNotification xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
   <callbackData>abcd</callbackData>
  <link rel="ImageShareSessionInformation"
```

```
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001"/>
   <link rel="ImageShareNotificationSubscription"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001"/>
  <eventType>SessionCancelled</eventType>
</is:imageShareEventNotification>
```

6.9.5.3.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.9.5.4 Example 4: Notify a client about image share event (aborted) (Informative)

6.9.5.4.1 Request

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: application.example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareEventNotification xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
  <callbackData>abcd</callbackData>
  <link rel="ImageShareSessionInformation"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001"/>
   <link rel="ImageShareNotificationSubscription"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001"/>
  <eventType>Aborted</eventType>
</is:imageShareEventNotification>
```

6.9.5.4.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.9.5.5 Example 5: Notify a client about image share event (failed) (Informative)

6.9.5.5.1 Request

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: application.example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareEventNotification xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
   <callbackData>abcd</callbackData>
   <link rel="ImageShareSessionInformation"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101sessions/sess001"/>
   <link rel="ImageShareNotificationSubscription"
href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001"/>
   <eventType>Failed</eventType>
  </is:imageShareEventNotification>
```

6.9.5.5.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.9.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 14.7 of [RFC2616].

6.10 Resource: Client notification about subscription cancellations

This resource is a callback URL provided by the client for notification about subscription cancellations, which are usually due to the subscription expiring. The RESTful Image Share API does not make any assumption about the structure of this URL. If this URL is a Client-side Notification URL, the server will POST notifications directly to it. If this URL is a Server-side Notification URL, the server uses it to determine the address of the Notification Server to which the notifications will subsequently be POSTed. The way the server determines the address of the Notification Server is out of scope of this specification.

Note: In the case when the client has set up a Notification Channel in order to use Long Polling to obtain the notifications, in order to retrieve the notifications, the client needs to use the Long Polling mechanism described in [REST_NetAPI_NotificationChannel], instead of the mechanism described below in section 6.10.5.

The notification is sent by the server to the image share participant to whom the cancelled subscription belongs.

EventType	Notification Root Element Type	Notification sent to	Response to Notification	Link rel	Link href Base URL: http://{serverRoot}/imageshare/{apiVersion}/{userId}/sessions
n/a	SubscriptionCancellationNotification	Originator or Receiver	n/a	ImageShareNotificationSubscription	/subscriptions/{subscriptionId}

6.10.1 Request URL variables

Client provided if any.

6.10.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful Image Share API, see section 7.

6.10.3 GET

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

6.10.4 PUT

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

6.10.5 POST

This operation is used to notify the client about subscription cancellation.

6.10.5.1 Example: Notify a client about subscription cancellation (Informative)

6.10.5.1.1 Request

```
POST /imageshare/notifications/77777 HTTP/1.1
Accept: application/xml
Content-Type: application/xml
Host: application.example.com

<?xml version="1.0" encoding="UTF-8"?>
<is:subscriptionCancellationNotification xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
   <callbackData>abcd</callbackData>
   <link rel="ImageShareNotificationSubscription"
      href="http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001"/>
</is:subscriptionCancellationNotification>
```

6.10.5.1.2 Response

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

6.10.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' 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 Image Share API.

7.1.1 SVC1003: CS call not existing for Image Share

Name	Description
MessageID	SVC1003
Text	CS call object reference not existing for Image Share
Variables	None
HTTP status code(s)	400 Bad request

7.2 Policy Exceptions

For common Policy Exceptions refer to [REST_NetAPI_Common].

No policy exceptions are defined for the RESTful Image Share API in this release.

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 OMA-TS-REST_NetAPI _ImageShare-V1_0	10 May 2011	All	Baseline
	17 Jun 2011	5, 5.1, Appendix E	CRs implemented - OMA-ARC-REST-NetAPI-2011-0026R01-CR_ImageShare_Resources
	1 Aug 2011	2.1,3.2,5.1,5.2	CRs implemented - OMA-ARC-REST-NetAPI-2011-0131R02-CR_ImageShare_Resources_alignment_with_new_resource_model - OMA-ARC-REST-NetAPI-2011-0132R02-CR_ImageShare_Datatypes_alignment_with_new_resource_model
	09 Sep 2011	Many	CRs implemented - OMA-ARC-REST-NetAPI-2011-0181R03-CR_Imageshare_DataTypes_Enhancement - OMA-ARC-REST-NetAPI-2011-0182-CR_Imageshare_Resource_modification - OMA-ARC-REST-NetAPI-2011-0218-CR_Imageshare_telURI_change_and_others
	19 Sep 2011	Many	CRs implemented - OMA-ARC-REST-NetAPI-2011-0239R01-CR_subscription_flow_specificaiton_ACR
	09 Oct 2011	Many	CRs implemented - OMA-ARC-REST-NetAPI-2011-0262R02-CR_ImageShare_additional_data_types - OMA-ARC-REST-NetAPI-2011-0280R01-CR_imageshare_sequence_diagram
	31 Oct 2011	Many	CRs implemented - OMA-ARC-REST-NetAPI-2011-0288R02-CR_ImageShare_more_SeqenceDiagram_and_other_updates
	03 Nov 2011	Many	CRs implemented - OMA-ARC-REST-NetAPI-2011-0338R01-CR_ImageShare_Sequence_diagram_update - Correct some typos in figure 3 and figure 4
	10 Nov 2011	5.2.2	CRs implemented - OMA-ARC-REST-NetAPI-2011-0350-CR_ImageShare_DataType_improvement
	01 Dec2011	Many	CRs implemented - OMA-ARC-REST-NetAPI-2011-0374-CR_delete_fileRange_in_IS_TS - OMA-ARC-REST-NetAPI-2011-0418-CR_ImageShare_Appendix_G - OMA-ARC-REST-NetAPI-2011-0422R01-CR_ImageShare_SCR_ - OMA-ARC-REST-NetAPI-2011-0419R03-CR_ImageShare_detailed_specification_of_resources - Change "acr:authorization" to "acr:Authorization"

Document Identifier	Date	Sections	Description
	02 Feb 2012	Many	CRs implemented - OMA-ARC-REST-NetAPI-2012-0039-CR_ImageShare_CONR_section2_3 - OMA-ARC-REST-NetAPI-2012-0041-CR_ImageShare_CONR_editorial_changes - OMA-ARC-REST-NetAPI-2012-0055R01-CR_ImageShare_CONR_more_changes
	08 Feb 2012	Many	CRs implemented - OMA-ARC-REST-NetAPI-2012-0055R01-CR_ImageShare_CONR_more_changes
	20 Feb 2012	Many	CRs implemented - OMA-ARC-REST-NetAPI-2012-0060R03-CR_ImageShare_CONR_A032 - OMA-ARC-REST-NetAPI-2012-0070-CR_ImageShare_CONR_link_examples
	23 Feb 2012	Many	CRs implemented - OMA-ARC-REST-NetAPI-2012-0079R03-CR_ImageShare_CONR_resolution_more
	02 Mar 2012	Many	CRs implemented - OMA-ARC-REST-NetAPI-2012-0087-CR_ImageShare_CONR_TS_resolutions
	15 Mar 2012	Various	Editorial: changed front cover, header, updated copyright year
	19 Mar 2012	Many	CRs implemented - OMA-ARC-REST-NetAPI-2012-0098-CR_ImageShare_TS_more_fix
	27 Mar 2012	Many	CRs implemented OMA-ARC-REST-NetAPI-2012-0116R01-CR_ImageShare_TS_Exceptions_Notifications_with_TTL
	28 Mar 2012	Tables 7 & 8, G.1.2	Corrected copyright year to 2012. Editorial changes
Candidate Version OMA-TS-REST_NetAPI_ImageShare-V1_0	10 Apr 2012	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2012-0161-INP_REST_NetAPI_ImageShare_1_0_ERP_and_ETR_for_Candidate_Approval

Appendix B. Static Conformance Requirements (Normative)

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

B.1 SCR for REST.ImageShare Server

Item	Function	Reference	Requirement
REST-IMAGESHARE-SUPPORT-S-001-M	Support for the RESTful Image Share API	5,6	
REST-IMAGESHARE-SUPPORT-S-002-M	Support for the XML request & response format	6	
REST-IMAGESHARE-SUPPORT-S-003-M	Support for the JSON request & response format	6	
REST-IMAGESHARE-SUPPORT-S-004-O	Support for the application/x-www-form-urlencoded format	Appendix C	

B.1.1 SCR for REST.ImageShare.Subscriptions Server

Item	Function	Reference	Requirement
REST-IMAGESHARE-SUBSCR-S-001-M	Support for subscriptions to image share notifications	6.1	
REST-IMAGESHARE-SUBSCR-S-002-O	Read the list of active image share notification subscriptions – GET	6.1.3	
REST-IMAGESHARE-SUBSCR-S-003-M	Create new subscription to image share notifications – POST (XML or JSON)	6.1.5	
REST-IMAGESHARE-SUBSCR-S-004-O	Create new subscription to image share notifications – POST (application/x-www-form-urlencoded)	C.1	

B.1.2 SCR for REST.ImageShare.Individual.Subscription Server

Item	Function	Reference	Requirement
REST-IMAGESHARE-IND-SUBSCR-S-001-M	Support for access to an individual subscription to image share notifications	6.2	
REST-IMAGESHARE-IND-SUBSCR-S-002-O	Read an individual image share notification subscription – GET	6.2.3	

Item	Function	Reference	Requirement
REST-IMAGESHARE-IND-SUBSCR-S-003-M	Cancel subscription and stop corresponding notifications – DELETE	6.2.6	

B.1.3 SCR for REST.ImageShare.Sessions Server

Item	Function	Reference	Requirement
REST-IMAGESHARE-SESS-S-001-M	Support for image share sessions	6.3	
REST-IMAGESHARE-SESS-S-002-M	Create a new image share session – POST(XML or JSON)	6.3.5	
REST-IMAGESHARE-SESS-S-003-O	Create a new image share session – POST(application/x-www-form-urlencoded)	C.2	

B.1.4 SCR for REST.ImageShare.Individual.Session Server

Item	Function	Reference	Requirement
REST-IMAGESHARE-IND-SESS-S-001-M	Support for individual image share sessions	6.4	
REST-IMAGESHARE-IND-SESS-S-002-O	Retrieve an image share session information – GET	6.4.3	
REST-IMAGESHARE-IND-SESS-S-003-M	Cancel invitation/ Decline Invitation/ Terminate an image share session – DELETE	6.4.6	

B.1.5 SCR for REST.ImageShare.Individual.Session.Status Server

Item	Function	Reference	Requirement
REST-IMAGESHARE-IND-SESS-STAT-S-001-M	Support for acceptance of the session	6.5	
REST-IMAGESHARE-IND-SESS-STAT-S-002-M	Accept an image share session invitation – POST (XML or JSON)	6.5.5	
REST-IMAGESHARE-IND-SESS-STAT-S-003-O	Accept an image share session invitation – POST(application/x-www-form-urlencoded)	C.3	

B.1.6 SCR for REST.ImageShare.Session.Invitation.Notifications Server

Item	Function	Reference	Requirement
REST-IMAGESHARE-INVITE-NOTIF-S-001-M	Support for notifications about image share session invitations	6.6	
REST-IMAGESHARE-INVITE-NOTIF-S-002-M	Image share session invitation notifications – POST (XML or JSON)	6.6.5	

B.1.7 SCR for REST.ImageShare.Session.Acceptance.Notifications Server

Item	Function	Reference	Requirement
REST-IMAGESHARE-ACCEPT-NOTIF-S-001-M	Support for notifications about image share session acceptance	6.7	
REST-IMAGESHARE-ACCEPT-NOTIF-S-002-M	Image share session acceptance notifications – POST (XML or JSON)	6.7.5	

B.1.8 SCR for REST.ImageShare.Link.Notifications Server

Item	Function	Reference	Requirement
REST-IMAGESHARE-LINK-NOTIF-S-001-M	Support for notifications about image file link	6.8	
REST-IMAGESHARE-LINK-NOTIF-S-002-M	Image file link notifications – POST (XML or JSON)	6.8.5	

B.1.9 SCR for REST.ImageShare.Events.Notifications Server

Item	Function	Reference	Requirement
REST-IMAGESHARE-EVENT-NOTIF-S-001-M	Support for notifications about image share events	6.9	
REST-IMAGESHARE-EVENT-NOTIF-S-002-M	Image share event notifications – POST (XML or JSON)	6.9.5	

B.1.10 SCR for REST.ImageShare.SubscriptionCancellation Server

Item	**Function**	**Reference**	**Requirement**
REST-IMAGESHARE-NOTIF-SUBCXL-S-001-M	Support for notifications about subscription	6.10	

Item	Function	Reference	Requirement
	cancellation		
REST-IMAGESHARE-NOTIF-SUBCXL-S-002-M	Subscription cancellation notifications – POST (XML or JSON)	6.10.5	

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

This section defines a format for the RESTful Image Share API requests where the body of the request is encoded using the application/x-www-form-urlencoded MIME type.

Note: only the request body is encoded as application/x-www-form-urlencoded, the response is still encoded as XML or JSON depending on the preference of the client and the capabilities of the server. Names and values MUST follow the application/x-www-form-urlencoded character escaping rules from [W3C_URLENC].

The encoding is defined below for the following Image Share REST operations which are based on POST requests:

- Creating a new subscription to image share notifications
- Creating a new image share session
- Accepting an image share session invitation

C.1 Creating a new subscription to image share notifications

This operation is used to create a new subscription to image share notifications. See section 6.1.5.

The notifyURL either contains the Client-side Notification URL (as defined by the client) or the Server-side Notification URL (as obtained during the creation of the Notification Channel [REST_NetAPI_NotificationChannel]).

The request parameters are as follows:

Name	Type/Values	Optional	Description
notifyURL	xsd:anyURI	No	Notification endpoint definition. For the use of Client-side Notification URLs and Server-side Notification URLs in this parameter, see sections 6.1 and 6.1.5.
callbackData	xsd:string	Yes	Data the application can register with the server when subscribing to notifications, and that are passed back unchanged in each of the related notifications.
notificationFormat	common:NotificationFormat	Yes	Application can specify format of the resource representation in notifications that are related to this subscription. The choice is between {XML, JSON}. Default: XML
duration	xsd:int	Yes	Period of time (in seconds) notifications are provided for. If set to “0” (zero), a default duration time, which is specified by the service policy, will be used. If the parameter is omitted, the notifications will continue until the maximum duration time, which is specified by the service policy, unless the notifications are stopped by deletion of subscription for notifications. This element MAY be given by the client during resource creation in order to signal the desired lifetime of the subscription. The server SHOULD return in this element the period of time for which the subscription

			will still be valid.
clientCorrelator	xsd:string	Yes	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 re-sending the message 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.

If the operation was successful, it returns an HTTP Status of "201 Created".

C.1.1 Example 1: Creating a new subscription to image share notifications using tel URI (Informative)

C.1.1.1 Request

```
POST /exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn
Accept: application/xml
Host: example.com

notifyURL=http%3A%2F%2Fapplication.example.com%2Fimageshare%2Fnotifications%2F77777&
callbackData=abcd&
duration=7200&
clientCorrelator=12345
```

C.1.1.2 Response

```
HTTP/1.1 201 Created
Content-Type: application/xml
Location: http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001
Content-Length: nnnn
Date: Thu, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareNotificationSubscription xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
    <callbackReference>
        <notifyURL>http://application.example.com/imageshare/notifications/77777</notifyURL>
        <callbackData>abcd</callbackData>
    </callbackReference>
    <duration>7200</duration>
    <clientCorrelator>12345</clientCorrelator>
    <resourceURL>http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001</resourceURL>
</is:imageShareNotificationSubscription>
```

Note that alternatively to returning a copy of the created resource, the location of created resource could be returned using the common:resourceReference root element (see section C.1.2.2).

C.1.2 Example 2: Creating a new subscription to image notifications using ACR (Informative)

C.1.2.1 Request

```
POST /exampleAPI/imageshare/v1/acr%3Apseudonym123/subscriptions/ HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn
Accept: application/xml
Host: example.com

notifyURL=http%3A%2F%2Fapplication.example.com%2Fimageshare%2Fnotifications%2F77777&
callbackData=abcd&
duration=7200&
clientCorrelator=12345
```

C.1.2.2 Response

```
HTTP/1.1 201 Created
Content-Type: application/xml
Location: http://example.com/exampleAPI/imageshare/v1/acr%3Apseudonym123/subscriptions/sub001
Content-Length: nnnn
Date: Thu, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<common:resourceReference xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <resourceURL>http://example.com/exampleAPI/imageshare/v1/acr%3Apseudonym123/subscriptions/sub001</resourceURL>
</common:resourceReference>
```

C.2 Creating a new image share session

This operation is used to create a new image share session. See section 6.3.5.

The request parameters are as follows:

Name	Type/Values	Optional	Description
originatorAddress	xsd:anyURI	No	Address (e.g. 'sip' URI, 'tel' URI, 'acr' URI) of the Originator. When the application acts behalf of the Originator, the originatorAddress MUST have the same value as the {userId} fragment in the resource URL path if the {userId} is also part of the request URL.
originatorName	xsd:string	Yes	Name of the Originator
receiverAddress	xsd:anyURI	Yes	Address (e.g. 'sip' URI, 'tel' URI, 'acr' URI) of the

			Receiver. It SHALL be present in request bodies during resource creation in case of image sharing without CS voice call. It SHALL not be present in request bodies during resource creation in case of image sharing with CS voice call. The server can get the receiverAddress using the callObjectRef received in the request bodies during resource creation in case of image sharing with CS voice call. When the application acts behalf of the Receiver, the receiverAddress MUST have the same value as the {userId} fragment in the resource URL path if the {userId} is also part of the request URL.
callObjectRef	xsd:anyURI	Yes	Reference to the CS voice object (to which the Receiver is linked) It SHALL be present in request bodies during resource creation in case of image sharing with CS voice call. It SHALL not be present in request bodies during resource creation in case of image sharing without CS voice call.
receiverName	xsd:string	Yes	Name of the Receiver. SHALL NOT be present when callObjectRef is used, and MAY be present when receiverAddress is used.
type	xsd:string	No	The MIME type of the file, as concatenated of type, "/" and subtype (e.g. image/jpeg). See RFC 2045.
name	xsd:string	Yes	The name of the file
size	xsd: unsignedLong	Yes	The size of the file in octets
algorithm	xsd:string	Yes	The hash algorithm used (only "sha-1" currently supported)
value	xsd: hexBinary	Yes	The hash value of the file
fileDescription	xsd:string	Yes	Human-readable short description of the image file (corresponding to 'i=' line in SDP) which could be set by the Originator. See [IR.79] & [RFC2327].
fileURL	xsd:anyURI	Yes	The file repository URL from where the file can be retrieved. If it is present in the POST operation during resource creation, the application of the Originator side needs

			to fetch the image file using this URL. Otherwise, if it is not present in the POST request during resource creation, the image file content is included in the HTTP body. The HTTP body can be represented as multipart/form-data entity bodies, as specified in 5.2.5.
clientCorrelator	xsd:string	Yes	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 re-sending the message 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.

If the operation was successful, it returns an HTTP Status of “201 Created”.

C.2.1 Example 1:Creating a new image share session with file content (no CS call related) (Informative)

C.2.1.1 Request

```
POST /exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions HTTP/1.1
Content-Type: multipart/form-data; boundary="===============123456==";
Content-Length: nnnn
Accept: application/xml
Host: example.com
MIME-Version: 1.0

--===============123456==
Content-Disposition: form-data; name=”root-fields”
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn

originatorAddress=tel%3A%2B19585550100&
originatorName=Alice&
receiverAddress=tel%3A%2B19585550101&
receiverName=Bob&
type=image%2Fjpeg&
name=sunset.jpg&
size=4096&
algorithm=sha-1&
value=58231FE8653BBCF371362F86D471913EE4B1DF2F&
fileDescription=This%20is%20my%20latest%20picture&
clientCorrelator=12345

--===============123456==
Content-Disposition: form-data; name=”attachment”; filename=”sunset.jpg”
Content-Type: image/jpeg
Content-Length: nnnn

JPEG ...binary image data...

--===============123456==--
```

C.2.1.2 Response

```
HTTP/1.1 201 Created
Content-Type: application/xml
Content-Length: nnnn
Location: http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001
Date: Mon, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareSessionInformation xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
   <originatorAddress>tel:+19585550100</originatorAddress>
  <originatorName>Alice</originatorName>
  <receiverAddress>tel:+19585550101</receiverAddress>
  <receiverName>Bob</receiverName>
 <status>Initial</status>
  <fileInformation>
    <fileSelector>
      <type>image/jpeg</type>
      <name>sunset.jpg</name>
      <size>4096</size>
      <hash>
        <algorithm>sha-1</algorithm>
        <value>58231FE8653BBCF371362F86D471913EE4B1DF2F</value>
      </hash>
    </fileSelector>
    <fileDescription>This is my latest picture</fileDescription>
   </fileInformation>
<clientCorrelator>12345</clientCorrelator>
<resourceURL>http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001</resourceURL>
</is:imageShareSessionInformation>
```

Note that alternatively to returning a copy of the created resource, the location of created resource could be returned using the common:resourceReference root element (see section C.1.2.2).

C.2.2 Example 2:Creating a new image share session with file repository URL (CS call related) (Informative)

C.2.2.1 Request

```
POST /exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn
Accept: application/xml
Host: example.com

originatorAddress=tel%3A%2B19585550100&
```

```
originatorName= Alice&
callObjectRef=http%3A%2F%2Fexample.com%2FexampleAPI%2Fcall%2Ftel%3A%2B19585550101%2Fsessions%2FcallSess001&
type=image%2Fjpeg&
name=sunset.jpg&
size=4096&
algorithm=sha-1&
value=58231FE8653BBCF371362F86D471913EE4B1DF2F&
fileDescription=This%20is%20my%20latest%20picture&
fileURL=http%3A%2F%2Fmypictures.com%2Falbum%2Fsunset.jpg&
clientCorrelator=12345
```

C.2.2.2 Response

```
HTTP/1.1 201 Created
Content-Type: application/xml
Content-Length: nnnn
Location: http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001
Date: Mon, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<is:imageShareSessionInformation xmlns:is="urn:oma:xml:rest:netapi:imageshare:1">
   <originatorAddress>tel:+19585550100</originatorAddress>
   <originatorName>Alice</originatorName>
   <receiverAddress>tel:+19585550101</receiverAddress>
  <callObjectRef>http://example.com/exampleAPI/call/tel%3A%2B19585550101/sessions/callSess001</callObjectRef>
  <receiverName>Bob</receiverName>
  <status>Initial</status>
  <fileInformation>
     <fileSelector>
       <type>image/jpeg</type>
       <name>sunset.jpg</name>
       <size>4096</size>
       <hash>
         <algorithm>sha-1</algorithm>
         <value>58231FE8653BBCF371362F86D471913EE4B1DF2F</value>
       </hash>
     </fileSelector>
     <fileDescription>This is my latest picture</fileDescription>
   </fileInformation>
    <fileURL>http://mypictures.com/album/sunset.jpg</fileURL>
   <clientCorrelator>12345</clientCorrelator>
   <resourceURL>http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001</resourceURL>
</is:imageShareSessionInformation>
```

Note that alternatively to returning a copy of the created resource, the location of created resource could be returned using the common:resourceReference root element (see section C.1.2.2).

C.3 Accepting an image share session invitation

This operation is used to accept an image share session invitation, see section 6.5.5.

The request parameters are as follows:

Name	Type/Values	Optional	Description
status	SessionStatus	No	Status of the Receiver. To indicate that the Receiver accepts the session invitation, this element MUST be set to "Connected"
fileAcceptance	xsd:Boolean	Yes	Accept (true) or reject (false) the image file by the Receiver. Default is true.

If the operation was successful, it returns an HTTP Status of "200 OK".

C.3.1 Example 1: Accepting an image share invitation with acceptance of the image file (Informative)

C.3.1.1 Request

```
POST /exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001/status HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn
Accept: application/xml
Host: example.com

status=Connected&
fileAcceptance=true
```

C.3.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Mon, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<is:receiverSessionStatus xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
    <status>Connected</status>
    <fileAcceptance>true</fileAcceptance>
</is:receiverSessionStatus>
```

C.3.2 Example2: Accepting an image share invitation without accepting image file (Informative)

C.3.2.1 Request

```
POST /exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001/status HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn
Accept: application/xml
```

```
Host: example.com

status=Connected&
fileAcceptance=false
```

C.3.2.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Mon, 28 Jul 2011 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<is:receiverSessionStatus xmlns:is ="urn:oma:xml:rest:netapi:imageshare:1">
    <status>Connected</status>
    <fileAcceptance>false</fileAcceptance>
</is:receiverSessionStatus>
```

Appendix D. JSON examples (Informative)

JSON (JavaScript Object Notation) is a lightweight, 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 [RFC4627].

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].

For full details on the operations themselves please refer to the section number indicated.

D.1 Reading all active image share notification subscriptions (section 6.1.3.1)

Request:

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

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Thu, 28 Jul 2011 17:51:59 GMT

{"imageShareSubscriptionList": {
    "imageShareNotificationSubscription": {
        "callbackReference": {
            "callbackData": "abcd",
            "notifyURL": "http://application.example.com/imageshare/notifications/77777"
        },
        "clientCorrelator": "12345",
        "duration": "7200",
        "resourceURL": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001"
    },
    "resourceURL": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions"
}}
```

D.2 Creating a new subscription to image share notifications using tel URI (section 6.1.5.1)

Request:

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

```
{"imageShareNotificationSubscription": {
    "callbackReference": {
        "callbackData": "abcd",
        "notifyURL": "http://application.example.com/imageshare/notifications/77777"
    },
    "clientCorrelator": "12345",
    "duration": "7200"
}}
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Location: http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001
Date: Thu, 28 Jul 2011 17:51:59 GMT

{"imageShareNotificationSubscription": {
    "callbackReference": {
        "callbackData": "abcd",
        "notifyURL": "http://application.example.com/imageshare/notifications/77777"
    },
    "clientCorrelator": "12345",
    "duration": "7200",
    "resourceURL": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001"
}}
```

D.3 Creating a new subscription to image share notifications using ACR (section 6.1.5.2)

Request:

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

{"imageShareNotificationSubscription": {
    "callbackReference": {
        "callbackData": "abcd",
        "notifyURL": "http://application.example.com/imageshare/notifications/77777"
    },
    "clientCorrelator": "12345",
    "duration": "7200"
}}
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://example.com/exampleAPI/imageshare/v1/acr%3Apseudonym123/subscriptions/sub001
Content-Length: nnnn
```

```
Date: Thu, 28 Jul 2011 17:51:59 GMT

{"resourceReference": {"resourceURL": "http://example.com/exampleAPI/imageshare/v1/acr%3Apseudonym123/subscriptions/sub001"}}
```

D.4 Reading an individual subscription (section 6.2.3.1)

Request:

```
GET /exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001?resFormat=JSON HTTP/1.1

Host: example.com
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Mon, 28 Jun 2010 17:51:59 GMT

{"imageShareNotificationSubscription": {
    "callbackReference": {
        "callbackData": "abcd",
        "notifyURL": "http://application.example.com/imageshare/notifications/77777"
    },
    "clientCorrelator": "12345",
    "duration": "7200",
    "resourceURL": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001"
}}
```

D.5 Cancelling a subscription (section 6.2.6.1)

Request:

```
DELETE /exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001 HTTP/1.1
Accept: application/json
Host: example.com
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 28 Jun 2010 17:51:59 GMT
```

D.6 Creating a new image share session with file content (no CS call related) (section 6.3.5.1)

Request:

```
POST /exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions HTTP/1.1
Content-Type: multipart/form-data; boundary="==============123456==";
Content-Length: nnnn
```

```
Accept: application/json
Host: example.com
MIME-Version: 1.0

--===============123456==
Content-Disposition: form-data; name=”root-fields”
Content-Type: application/xml
Content-Length: nnnn

{"imageShareSessionInformation": {
    "clientCorrelator": "12345",
    "fileInformation": {
        "fileDescription": "This is my latest picture",
        "fileSelector": {
            "hash": {
                "algorithm": "sha-1",
                "value": "58231FE8653BBCF371362F86D471913EE4B1DF2F"
            },
            "name": "sunset.jpg",
            "size": "4096",
            "type": "image/jpeg"
        }
    },
    "originatorAddress": "tel:+19585550100",
    "originatorName": "Alice",
    "receiverAddress": "tel:+19585550101",
    "receiverName": "Bob"
}}

--===============123456==

Content-Disposition: form-data; name=”attachment”; filename=”sunset.jpg”
Content-Type: image/jpeg
Content-Length: nnnn

JPEG ...binary image data...

--===============123456==--
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001
Content-Length: nnnn
Date: Mon, 28 Jul 2011 17:51:59 GMT

{"imageShareSessionInformation": {
    "clientCorrelator": "12345",
    "fileInformation": {
        "fileDescription": "This is my latest picture",
        "fileSelector": {
            "hash": {
                "algorithm": "sha-1",
                "value": "58231FE8653BBCF371362F86D471913EE4B1DF2F"
            },
            "name": "sunset.jpg",
            "size": "4096",
            "type": "image/jpeg"
        }
    },
    "originatorAddress": "tel:+19585550100",
    "originatorName": "Alice",
    "receiverAddress": "tel:+19585550101",
    "receiverName": "Bob",
    "resourceURL": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001",
    "status": "Initial"
}}
```

D.7 Creating a new image share session with file repository URL (CS call related) (section 6.3.5.2)

Request:

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

{"imageShareSessionInformation": {
    "callObjectRef": "http://example.com/exampleAPI/call/tel%3A%2B19585550101/sessions/callSess001",
    "clientCorrelator": "12345",
    "fileInformation": {
        "fileDescription": "This is my latest picture",
        "fileSelector": {
            "hash": {
                "algorithm": "sha-1",
                "value": "58231FE8653BBCF371362F86D471913EE4B1DF2F"
            },
            "name": "sunset.jpg",
            "size": "4096",
            "type": "image/jpeg"
        }
    },
    "fileURL": "http://mypictures.com/album/sunset.jpg",
    "originatorAddress": "tel:+19585550100",
    "originatorName": "Alice"
}}
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001
Content-Length: nnnn
```

```
Date: Mon, 28 Jul 2011 17:51:59 GMT

{"imageShareSessionInformation": {
    "callObjectRef": "http://example.com/exampleAPI/call/tel%3A%2B19585550101/sessions/callSess001",
    "clientCorrelator": "12345",
    "fileInformation": {
        "fileDescription": "This is my latest picture",
        "fileSelector": {
            "hash": {
                "algorithm": "sha-1",
                "value": "58231FE8653BBCF371362F86D471913EE4B1DF2F"
            },
            "name": "sunset.jpg",
            "size": "4096",
            "type": "image/jpeg"
        }
    },
    "fileURL": "http://mypictures.com/album/sunset.jpg",
    "originatorAddress": "tel:+19585550100",
    "originatorName": "Alice",
    "receiverAddress": "tel:+19585550101",
    "receiverName": "Bob",
    "resourceURL": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001",
    "status": "Initial"
}}
```

D.8 Retrieving an image share session information (no CS call related) (section 6.4.3.1)

Request:

```
GET /exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001 HTTP/1.1
Accept: application/json
Host: example.com
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Mon, 28 Jul 2011 17:51:59 GMT

{"imageShareSessionInformation": {
    "clientCorrelator": "12345",
    "fileInformation": {
        "fileDescription": "This is my latest picture",
        "fileSelector": {
            "hash": {
                "algorithm": "sha-1",
                "value": "58231FE8653BBCF371362F86D471913EE4B1DF2F"
            },
            "name": "sunset.jpg",
            "size": "4096",
            "type": "image/jpeg"
        }
    },
    "originatorAddress": "tel:+19585550100",
    "originatorName": "Alice",
    "receiverAddress": "tel:+19585550101",
    "receiverName": "Bob",
    "resourceURL": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001",
    "status": "Connected"
}}
```

D.9 Retrieving an image share session information (CS call related) (section 6.4.3.2)

Request:

```
GET /exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001 HTTP/1.1
Accept: application/json
Host: example.com
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Mon, 28 Jul 2011 17:51:59 GMT

{"imageShareSessionInformation": {
    "callObjectRef": "http://example.com/exampleAPI/call/tel%3A%2B19585550101/sessions/callSess001",
    "clientCorrelator": "12345",
    "fileInformation": {
        "fileDescription": "This is my latest picture",
        "fileSelector": {
            "hash": {
                "algorithm": "sha-1",
                "value": "58231FE8653BBCF371362F86D471913EE4B1DF2F"
            },
            "name": "sunset.jpg",
            "size": "4096",
            "type": "image/jpeg"
        }
    },
    "fileURL": "http://mypictures.com/album/sunset.jpg",
    "originatorAddress": "tel:+19585550100",
    "originatorName": "Alice",
    "receiverAddress": "tel:+19585550101",
    "receiverName": "Bob",
    "resourceURL": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001",
    "status": "Connected"
}}
```

D.10 Terminating an image share session (section 6.4.6.1)

Request:

```
DELETE /exampleAPI/imageshare/v1/tel%3A%2B19585550100/sessions/sess001 HTTP/1.1
Accept: application/json
Host: example.com
```

Response:

```
HTTP/1.1 204 No Content
Date: Mon, 28 Jul 2011 17:51:59 GMT
```

D.11 Accepting an image share invitation with acceptance of the image file (section 6.5.5.1)

Request:

```
POST /exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001/status HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"receiverSessionStatus": {
    "fileAcceptance": "true",
    "status": "Connected"
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

D.12 Accepting an image share invitation without accepting image file (section 6.5.5.2)

Request:

```
POST /exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001/status HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"receiverSessionStatus": {
    "fileAcceptance": "false",
    "status": "Connected"
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2011 17:51:59 GMT
```

D.13 Notify a client about image share session invitations (no CS call related) (section 6.6.5.1)

Request:

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: application.example.com

{"sessionInvitationNotification": {
    "callbackData": "abcd",
    "fileInformation": {
        "fileDescription": "This is my latest picture",
        "fileSelector": {
            "hash": {
                "algorithm": "sha-1",
                "value": "58231FE8653BBCF371362F86D471913EE4B1DF2F"
            },
            "name": "sunset.jpg",
            "size": "4096",
            "type": "image/jpeg"
        }
    },
    "link": [
        {
            "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001",
            "rel": "ImageShareSessionInformation"
        },
        {
            "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001/status",
            "rel": "ReceiverSessionstatus"
        },
        {
            "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001",
            "rel": "ImageShareNotificationSubscription "
        }
   ],
    "originatorAddress": "tel:+19585550100",
    "originatorName": "Alice",
    "receiverAddress": "tel:+19585550101",
    "receiverName": "Bob"
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

D.14 Notify a client about image share session invitations (CS call related) (section 6.6.5.2)

Request:

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: application.example.com

{"sessionInvitationNotification": {
    "callObjectRef": "http://example.com/exampleAPI/call/tel%3A%2B19585550101/sessions/callSess001",
    "callbackData": "abcd",
    "fileInformation": {
        "fileDescription": "This is my latest picture",
        "fileSelector": {
            "hash": {
                "algorithm": "sha-1",
                "value": "58231FE8653BBCF371362F86D471913EE4B1DF2F"
            },
            "name": "sunset.jpg",
            "size": "4096",
            "type": "image/jpeg"
        }
    },
    "link": [
        {
            "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001",
            "rel": "ImageShareSessionInformation"
        },
        {
            "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001/status",
            "rel": "ReceiverSessionstatus"
        },
        {
            "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001",
            "rel": "ImageShareNotificationSubscription "
        }
    ],
    "originatorAddress": "tel:+19585550100",
    "originatorName": "Alice",
    "receiverAddress": "tel:+19585550101",
    "receiverName": "Bob"
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

D.15 Notify a client about the acceptance of an image share session (no CS call related) (section 6.7.5.1)

Request:

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: application.example.com

{"sessionAcceptanceNotification": {
   "callbackData": "abcd",
   "link": [
   {
      "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001",
      "rel": "ImageShareSessionInformation"
   },
      {
         "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001",
         "rel": "ImageShareNotificationSubscription "
      }
   ],
   "receiverAddress": "tel:+19585550101",
   "receiverName": "Bob",
   "receiverSessionStatus": {
      "fileAcceptance": "true",
      "status": "Connected"
   }
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

D.16 Notify a client about the acceptance of an image share session (CS call related) (section 6.7.5.2)

Request:

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: application.example.com


{"sessionAcceptanceNotification": {
   "callObjectRef": "http://example.com/exampleAPI/call/tel%3A%2B19585550101/sessions/callSess001",
   "callbackData": "abcd",
   "link": [
   {
       "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001",
       "rel": "ImageShareSessionInformation"
   },
       {
           "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001",
           "rel": "ImageShareNotificationSubscription "
       }
   ],
   "receiverAddress": "tel%3A%2B19585550101",
   "receiverName": "Bob",
   "receiverSessionStatus": {
       "fileAcceptance": "false",
       "status": "Connected"
   }
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

D.17 Notify a client about the image file link (section 6.8.5.1)

Request:

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: application.example.com

{"imageFileNotification": {
    "fileURL": "http://example.com/exampleAPI/imageshare/tel%3A%2B19585550100/album/sunset.jpg",
    "link": [
    {
        "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001",
        "rel": "ImageShareSessionInformation"
    },
        {
            "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001",
            "rel": "ImageShareNotificationSubscription "
        }
    ]
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

D.18 Notify a client about image share event (ended) (section 6.9.5.1)

Request:

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: application.example.com

{"imageShareEventNotification": {
   "callbackData": "abcd",
   "eventType": "SessionEnded",
   "link": [
   {
      "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001",
      "rel": "ImageShareSessionInformation"
   },
      {
         "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001",
         "rel": "ImageShareNotificationSubscription "
      }
   ]
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

D.19 Notify a client about image share event (declined) (section 6.9.5.2)

Request:

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: application.example.com

{"imageShareEventNotification": {
   "callbackData": "abcd",
   "eventType": "Declined",
   "link": [
   {
      "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001",
      "rel": "ImageShareSessionInformation"
   },
      {
         "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001",
         "rel": "ImageShareNotificationSubscription "
```

```
        }
    ]
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

D.20 Notify a client about image share event (cancelled) (section 6.9.5.3)

Request:

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: application.example.com

{"imageShareEventNotification": {
    "callbackData": "abcd",
    "eventType": "SessionCancelled",
    "link": [
    {
        "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001",
        "rel": "ImageShareSessionInformation"
    },
        {
            "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001",
            "rel": "ImageShareNotificationSubscription "
        }
    ]
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

D.21 Notify a client about image share event (aborted) (section 6.9.5.4)

Request:

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
```

```
Host: application.example.com

{"imageShareEventNotification": {
   "callbackData": "abcd",
   "eventType": "Aborted",
   "link": [
   {
      "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001",
      "rel": "ImageShareSessionInformation"
   },
      {
         "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001",
         "rel": "ImageShareNotificationSubscription "
      }
   ]
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

D.22 Notify a client about image share event (failed) (section 6.9.5.5)

Request:

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: application.example.com

{"imageShareEventNotification": {
   "callbackData": "abcd",
   "eventType": "Failed",
   "link": [
   {
      "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/sessions/sess001",
      "rel": "ImageShareSessionInformation"
   },
      {
         "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550101/subscriptions/sub001",
         "rel": "ImageShareNotificationSubscription "
      }
   ]
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

D.23 Notify a client about subscription cancellation (section 6.10.5.1)

Request:

```
POST /imageshare/notifications/77777 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: application.example.com

{"subscriptionCancellationNotification": {
    "callbackData": "abcd",
    "link": {
        "href": "http://example.com/exampleAPI/imageshare/v1/tel%3A%2B19585550100/subscriptions/sub001",
        "rel": "ImageShareNotificationSubscription"
    }
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 28 Jul 2010 02:51:59 GMT
```

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)

As this version of the specification does not define any light-weight resources, this Appendix is empty.

Appendix G. Authorization aspects (Normative)

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

G.1 Use of Autho4API

The RESTful Image Share API MAY support the Autho4API authorization framework defined in [Autho4API_10].

A RESTful Image Share 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 Image Share API:

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

Scope value	**Description**	**For one-time access token**
oma_rest_imageshare.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_imageshare.sessions	Provide access to all defined operations on image share sessions	No
oma_rest_imageshare.subscr	Provide access to all defined operations on image share subscriptions	No

Table 6: Autho4API scope values for RESTful Image Share API

G.1.1.2 Downscoping

In the case where the Autho4API client requests authorization for " oma_rest_imageshare.all_{apiVersion}" scope, the Autho4API Authorization Server and/or resource owner MAY restrict the granted scope to some of the following scope values:

- "oma_rest_imageshare.sessions"
- "oma_rest_imageshare.subscr"

G.1.1.3 Mapping with resources and methods

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

Resource	URL Base URL: http://{serverRoot}/imageshare/{apiVersion}/{userId}	Section reference	HTTP verbs			
			GET	PUT	POST	DELETE
All image share sessions	/sessions	6.3	n/a	n/a	**all_{apiVersion} or sessions**	n/a
Individual image share session	/sessions/{sessionId}	6.4	**all_{apiVersion} or sessions**	n/a	n/a	**all_{apiVersion} or sessions**
Individual image share session status	/sessions/{sessionId}/status	6.5	n/a	n/a	**all_{apiVersion} or sessions**	n/a

Table 7: Required scope values for: image share sessions

Resource	URL Base URL: http://{serverRoot}/imageshare/{apiVersion}/{userId}	Section reference	HTTP verbs			
			GET	PUT	POST	DELETE
All subscriptions to image share notifications	/subscriptions	6.1	**all_{apiVersion} or subscr**	n/a	**all_{apiVersion} or subscr**	n/a
Individual subscription to image share notifications	/subscriptions/{subscriptionId}	6.2	**all_{apiVersion} or subscr**	n/a	n/a	**all_{apiVersion} or subscr**

Table 8: Required scope values for: image share subscriptions

G.1.2 Use of ‘acr:Authorization’

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

An ‘acr’ URI of the form ‘acr:Authorization’, where ‘Authorization’ 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:Authorization’ in a resource URL in place of the {userId} resource URL variable in the resource URL path, when the RESTful Image Share API is used in combination with [Autho4API_10].

In the case the RESTful Image Share API supports [Autho4API_10], the server:

- SHALL accept ‘acr:Authorization’ as a valid value for the resource URL variable {endUserId}.
- SHALL conform to [REST_Common_TS] section 5.8.1.1 regarding the processing of ‘acr:Authorization’.