|  |  |  |  |
| --- | --- | --- | --- |
| logo-orange-prospect.gif |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | Reference : |  |
|  |  | Version : |  |
|  |  | Date : |  |
|  | Status : | Draft |
|  | Usage : |  |
|  | Author : |  |
|  | Type : | |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **document approval** | | | |
| **Company** | **Name (function)** | **Date** | **Visa** |
| Capgemini | N. Beitz (CP) |  |  |
| Orange | M. Ebrel (CP) |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Diffusion** | | | | |
| **Addresses** | **Company** | **Number** | **For action** | **For info** |
| Project Team | Capgemini AS France | 1 |  | X |
|  | Orange | 1 | X |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Updates** | | | |
| **Version** | **Date** | **Author** | **Reasons** |
| V1.0 | 21/06/2013 | Capgemini |  |
| V1.1 | 16/09/2013 | Capgemini | Update with Orange review |
| V1.2 | 18/09/2013 | Capgemini | Usage overview of File Transfer |
| V1.3 | 20/09/2013 | Capgemini | Update with Orange review & cURL examples |
| V1.4 | 24/09/2013 | Capgemini | Update with Orange review |
| V1.5 | 25/09/2013 | Capgemini | Update with Orange review |

|  |  |  |  |
| --- | --- | --- | --- |
| **Reference Documents** | | | |
| **Id** | **Name** | **Origin** | **Version** |
| 1. | RCS Profile of RESTful Network APIs | OMA | 2.0 |
| 2. | Common definitions for RESTful Network APIs | OMA | 1.0 |
| 3. | RESTful Network API for Chat | OMA | 1.0 |
| 4. | RESTful Network API for FileTransfer | OMA | 1.0 |
| 5. |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Deposit** | | |
| **URL** | **Anonymous access ?** | **Administrator** |
| https://coconet2.capgemini.com/sf/go/doc2064916 | N |  |

***CONTENTS***

1. Introduction 5

1.1 Scope 5

1.2 Version 1.0 6

2. Technical informations 7

2.1 Tools 7

2.2 Hackathon platform 7

3. Usage overview 8

3.1 Chat usage overview 8

3.1.1 Call-flow 8

3.1.2 Notification subscription 9

3.1.3 Chat session creation 11

3.1.4 Sending messages 12

3.1.5 Receiving messages 13

3.1.6 Closing the chat session 14

3.1.7 Unsubscribing to the notification 15

3.2 File transfer usage overview 16

3.2.1 Call-flow 16

3.2.2 Notification subscription 16

3.2.3 File transfer with an URL 17

3.2.4 File transfer with embedded file 18

3.2.5 Notifications during the file transfer 19

3.2.6 Unsubscribing the notification 21

4. cURL Examples 22

4.1 Chat usage examples 22

4.1.1 Notification subscription 22

4.1.1.1 XML 22

4.1.1.2 JSON 23

4.1.2 Chat session creation 24

4.1.2.1 XML 24

4.1.2.2 JSON 24

4.1.3 Sending messages 25

4.1.3.1 XML 25

4.1.3.2 JSON 26

4.1.4 Sending isComposing notification 27

4.1.4.1 XML 27

4.1.4.2 JSON 27

4.1.5 Updating status of received messages 28

4.1.5.1 XML 28

4.1.5.2 JSON 29

4.1.6 Closing the chat session 29

4.1.6.1 XML 29

4.1.7 Unsubscribing the notification 30

4.2 File Transfer usage examples 31

4.2.1 Notification subscription 31

4.2.1.1 XML 31

4.2.1.2 JSON 31

4.2.2 File transfer with an URL 32

4.2.2.1 XML 32

4.2.2.2 JSON 34

4.2.3 File transfer with embedded file 35

4.2.3.1 XML 35

4.2.3.2 JSON 36

4.2.4 Unsubscribing the notification 37

5. API Reference 39

5.1 Common API specifications 39

5.1.1 Common considerations 39

5.1.1.1 Content-type negotiation 39

5.1.1.2 Resource creation 39

5.1.1.3 Resource URL considerations 39

5.1.1.4 User identifiers 39

5.1.2 Data types 40

5.1.2.1 XML namespaces 40

5.1.2.2 Structures 40

5.1.2.3 Enumerations 41

5.2 Chat API definition 42

5.2.1 Resources Summary 42

5.2.2 Data types 46

5.2.2.1 XML namespaces 46

5.2.2.2 Structures 46

5.2.2.3 Enumerations 52

5.3 File Transfer API definition 53

5.3.1 Resources Summary 53

5.3.2 Data types 56

5.3.2.1 XML namespaces 56

5.3.2.2 Structures 56

5.3.2.3 Enumerations 60

5.3.2.4 MIME multipart representation 61

5.4 Fault définitions 63

5.4.1 Common faults 63

5.4.1.1 HTTP Response Codes 63

5.4.1.2 HTTP Response Codes in Response to Notifications 64

5.4.1.3 Service Exceptions 64

5.4.1.4 Policy Exceptions 65

5.4.1.5 Examples of exceptions 66

5.4.2 Chat API Exception Definitions 66

5.4.2.1 Service Exceptions 66

5.4.2.2 Policy Exceptions 66

5.4.3 File transfer API Exception Definitions 66

5.4.3.1 Service Exceptions 66

5.4.3.2 Policy Exceptions 66

# Introduction

[Joyn™](http://www.joynus.com/) is a certification trade mark of the GSMA (GSM Association) for RCS (Rich Communication Suite) which is an industry standard that is being adopted across the world by mobile operators. RCS is a set of new communication services to improve traditional communications with real times services.

This developer guide offers a description of the first set of the joyn Network API API which is under standardization with [GSMA](http://www.gsma.com/rcs)/OMA.

The NetAPI exposes HTTP NetAPI and ensures the protocol binding based on the requirements for Chat (also known as Instant Messaging) and File Transfer using the REST architectural style. It provides examples, 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.

## Scope

This document is the reference for developers of the NetAPI/RCS interface implemented for the Hackathon event. This API is a subpart of the NetAPI defined by OMA and implements, with limitations, the following specifications:

* [2] Common definitions for RESTful Network APIs

<http://technical.openmobilealliance.org/Technical/release_program/RESTNetAPICommon_v1_0.aspx>

* [3] RESTful Network API for Chat

<http://technical.openmobilealliance.org/Technical/release_program/RESTfulNetworkAPI_Chat_V1_0.aspx>

* [4] RESTful Network API for FileTransfer

<http://technical.openmobilealliance.org/Technical/release_program/RESTNetAPI_FileTransfer_v1_0.aspx>

The document contains:

* Usage overview of the NetAPI (part 2).
* Example of usage of the NetAPI, with cURL commands (part 4).
* The complete reference of the NetAPI, including resources URL, structures, and so on (part 5).

## Version 1.0

Version 1.0 of this specification supports the following operations:

* Managing 1-1 chat sessions:
  + Managing subscriptions to chat-related event notifications
  + Creating a 1-1 chat session
  + Sending and receiving 1-1 chat messages
  + Reporting the status of 1-1 chat messages
  + Receiving notifications about the status of 1-1 chat messages
  + Receiving notifications about chat session events
* Managing file transfer session:
  + Managing subscriptions to file transfer-related notifications
  + Managing file transfer sessions
  + Sending files
  + Receiving notifications about file transfer session events
  + Receiving notifications about Receiver acceptance

Version 1.0 of this specification DOES NOT support:

* Chat session invitation from RCS world.
* File Transfer invitation from RCS world.
* Long polling notifications.

Currently, the access to the NetAPI does not require authentication or authorization.

# Technical informations

## Tools

The NetAPI exposes a REST interface, thus you do not need any SDK. Furthermore, you can use any language to implement a NetAPI client.

However, you may want some third-party librairies and/or tools to use the NetAPI:

* A REST or HTTP client.
* XML or JSON parser/formatter library.
* Optionnaly, a library supporting the multipart MIME encoding for file transfers.

## Hackathon platform

The following part provides the technical data for using the NetAPI during the hackathon event.

URL of the NetAPI for chat: <http://172.20.65.93/netapi/chat/v1/>

URL of the NetAPI for file transfer: <http://172.20.65.93/netapi/filetransfer/v1/>

Due to firewalling limitations, the network flows from and to the NetAPI is limited:

* The HTTP interface used by the REST client to handle notifications must listen on tcp/80.
* Furthermore, the NetAPI has access to the Internet, but only outgoing requests to tcp/80 are allowed. This restriction applies in particular when a REST client provides an URL as subject to a file transfer.

Only the mobile phones provided by Orange for the event are usable as target through the NetAPI.

The access to the NetAPI does not need authorization or authentication.

# Usage overview

This sections includes examples of the use of the NetAPI for chat and file transfer. Each example shows the content of the requests sent or received by the REST client.

## Chat usage overview

The Chat API provides a way to chat with a mobile phone RCSe compliant. The REST client through the Chat API has access to a set of methods to create a chat session, and send messages and notifications through it.

The REST client must provide an HTTP interface which will be called by the NetAPI to deliver notifications related tothe chat session: messages sent by the mobile, status of the session, or status of the message sent by the REST client.The following example describes typical REST client interactions::

* Subscribe to the notification system.
* Create a chat session with a mobile phone.
* Exchange messages with the remote phone.
* Exchange notifications event (delivery and displayed) of the messages.
* Close the chat session
* Unsubscribe to the notification system.

### Call-flow

The following figure shows the studied use-case. The REST client initiating the chat session stands on the left side. The mobile phone is shown on the right side.

Please note that the description of the interactions between the NetAPI and the mobile phone follows the RCS specifications and is out of the scope of this document.



Each step of this chat session is detailed in the following parts.

### Notification subscription

First step is to subscribe the notification for a user. This action will allow the user to receive notifications from NetAPI Gateway.

The REST client must send a POST request to the URL:

http://*<REST\_ip>*/netapi/chat/v1/ tel%3A%2B33611223355/subscriptions

Where *REST\_ip* is the IP address of the NetAPI.

The content of the request is a “chatNotificationSubscription” structure (as describe on 5.2.2.2 ). The next figure shows an example of XML content:

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

<chat:chatNotificationSubscription xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<confirmedChatSupported>true</confirmedChatSupported>

<adhocChatSupported>false</adhocChatSupported>

<clientCorrelator>12345</clientCorrelator>

<duration>5000</duration>

<callbackReference>

<notifyURL>http://10.67.102.152/test/</notifyURL>

<callbackData>123</callbackData>

<notificationFormat>XML</notificationFormat>

</callbackReference>

</chat:chatNotificationSubscription>

The REST client must provide the address of his HTTP interface in the notifyURL field: this is the URL which will be called by the NetAPI when it needs to notify events to the REST client (in the example, http://10.67.102.152/test/).

The REST client could send the same request in JSON format:

{"chatNotificationSubscription":

{

"callbackReference":

{

"notifyURL":"http://10.67.102.152/test/",

"callbackData":"123",

"notificationFormat":"XML”

},

"confirmedChatSupported":true,

"adhocChatSupported":false,

"duration":5000,

"clientCorrelator":"12345"

}

}

If everything goes right, the interface responds with code “201 Created”. The response body contains the same structure, but a new field appeared. Called ‘resourceURL’, this is the identifier of the subscription (for instance: “http://<REST\_ip>/netapi/chat/v1/tel%3A%2B33611223355/subscriptions/814838ba-d773-4753-60d3684f6ca”). It will be needed when the REST client will ask to unsubscribe the notification.

### Chat session creation

The next step is to create the chat session. This is done by a POST request to the URL:

http://*<REST\_ip>*/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223355

Where <tel:+33611223355> is the caller and tel:+33611223344is the called phone number in the chat session

The content of the request must be a “chatSessionInformation” structure (as describe on 5.2.2.2). For instance, in XML format:

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

<chat:chatSessionInformation xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<subject>Dinner tonight</subject>

<originatorAddress>tel:+33611223355</originatorAddress>

<originatorName>Alice</originatorName>

<tParticipantAddress>tel:+33611223344</tParticipantAddress>

<tParticipantName>Bob</tParticipantName>

<clientCorrelator>23456</clientCorrelator>

</chat:chatSessionInformation>

Be aware that the caller and the called parameters must be the same between the URL and in the request content, otherwise an error is thrown by the interface.

The same request could be sent in JSON format:

{"chatSessionInformation":

{

"subject": "Dinner tonight",

"clientCorrelator": 23456,

"originatorAddress": "tel:+33611223344",

"originatorName": "Alice",

"tParticipantAddress": "tel:+33611223355",

"tParticipantName": "Bob"

}

}If everything goes right, the interface responds with code “201 Created”. The response body contains the same structure, but a new field appeared, called ‘resourceURL’. This time, this is the identifier of the chat session. It must be provided in the next requests, and will be included in the events occurring in the chat session.

Shortly after the chat session creation, the REST client will be notified with an event of type ChatEventNotification with the eventType ‘Accepted’. An example of such event could be:

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

<chat:chatEventNotification xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<callBackData>123</callBackData>

<link rel="ChatSessionInformation" href=" http://<*REST\_ip*>/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/ tel%3A%2B33611223344/b9a06e30-c9b6-4792-8d8e-888f22411d61"/>

<link rel="ChatNotificationSubscription" href="http://<REST\_ip>/netapi/chat/v1/tel%3A%2B33611223355/subscriptions/814838ba-d773-4753-60d3684f6ca"/>

<eventType>Accepted</eventType>

<eventDescription>Accepted</eventDescription>

</chat:chatEventNotification>

Or in JSON:

{"chatEventNotification": {

"callBackData":"abcd",

"link":[

{"rel":"ChatSessionInformation","href":"http:\/\/<REST\_ip>\/netapi\/chat\/v1\/tel%3A%2B33611223355\/oneToOne\tel%3A%2B33611223344\/770a737f-09c0-474c-a1d1-3f52496fa6db"},

{"rel":"ChatNotificationSubscription","href":"http:\/\/<REST\_ip>\/netapi\/chat\/v1\/tel%3A%2B33611223355\/subscriptions\/7e88e653-2a9f-4925-a1b4-f265aa490454"}],

"eventType":"Accepted",

"eventDescription":"Accepted"

}}

This event shows that the interface is ready to handling messages in the chat session. Note that at this point, the mobile phone shows no chat session. It will display the invitation to a chat only when the REST client sends a first message in the chat.

### Sending messages

Now that the chat session is ready to be used, it is time to send our fist message. This is done by a POST request to the resourceURL returned by the interface when creating the chat session. For instance:

http://*<REST\_ip>*/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/ tel%3A%2B33611223344/b9a06e30-c9b6-4792-8d8e-888f22411d61/messages

The content of the request must be a “chatMessage” structure (as describe on 5.2.2.2). An example of content in XML format:

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

<chat:chatMessage xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<text>Hello World</text>

<reportRequest>Delivered</reportRequest>

<reportRequest>Displayed</reportRequest>

</chat:chatMessage>

Or in JSON:

{"chatMessage": {

"reportRequest": "Delivered",

"reportRequest": "Displayed",

"text": "Hello World"

}}

The field reportRequest requires the mobile phone to notify the REST client when the message is delivered and/or displayed to the user.

The interface responds with code “201 Created” with a resourceURL identifying the message, such as http://<*REST\_ip*>/netapi/chat/v1/ tel%3A%2B33611223355/oneToOne/ tel%3A%2B33611223344/b9a06e30-c9b6-4792-8d8e-888f22411d61/messages/a7b833249ec14dd4a21bbe23645eb85

If the REST client asked for reports of delivered and/or displayed message, it will receive shortly after the corresponding events, sent by the RCS mobile phone and relayed by the interface:

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

<chat:chatMessageStatusNotification xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<callBackData>123</callBackData>

<link rel="ChatSessionInformation" href="http://<REST\_ip>/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel:+336111223344/b9a06e30-c9b6-4792-8d8e-888f22411d61"/>

<link rel="ChatNotificationSubscription" href="http://<REST\_ip>/netapi/chat/v1/tel%3A%2B33611223355/subscriptions/814838ba-d773-4753-60d3684f6ca"/>

<link rel="ChatMessage" href="http://<REST\_ip>/netapi/chat/v1/ tel%3A%2B33611223355/oneToOne/tel:+336111223344/b9a06e30-c9b6-4792-8d8e-888f22411d61/messages/a7b833249ec14dd4a21bbe23645eb85"/>

<status>Displayed</status>

<errorCode></errorCode>

<description></description>

</chat:chatMessageStatusNotification>

Or in JSON:

{"chatMessageStatusNotification":{

"callBackData":"123",

"link":[

{"rel":"ChatSessionInformation","href":" http:\/\/<REST\_ip>\/netapi\/chat\/v1\/tel%3A%2B33611223355\/oneToOne\/tel%3A%2B33611223344\/b9a06e30-c9b6-4792-8d8e-888f22411d61"},

{"rel":"ChatMessage","href":"http:\/\/<REST\_ip>\/netapi\/chat\/v1\/tel%3A%2B33611223355\/oneToOne\/tel%3A%2B33611223344\/b9a06e30-c9b6-4792-8d8e-888f22411d61\/messages\/a7b833249ec14dd4a21bbe23645eb85"},

{"rel":"ChatNotificationSubscription","href":" http:\/\/<REST\_ip>\/netapi\/chat\/v1\/tel:+33612345678\/subscriptions\/814838ba-d773-4753-60d3684f6ca "}],

"status":" Displayed ",

"errorCode":"","description":""

}}

### Receiving messages

When the mobile phone sends a message, the interface pushes a notification to the REST client. The structure included in the notification is the following:

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

<chat:chatMessageNotification xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<callBackData>123</callBackData>

<link rel="ChatSessionInformation" href="http://<REST\_ip>/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/ tel%3A%2B33611223344/b9a06e30-c9b6-4792-8d8e-888f22411d61"/>

<link rel="ChatMessage" href="http://<REST\_ip>/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/ tel%3A%2B33611223344/b9a06e30-c9b6-4792-8d8e-888f22411d61/messages/QlA0DXiw"/>

<senderAddress>*tel:+* 33611223344</senderAddress>

<senderName>Bob</senderName>

<chatMessage>

<text>Comment ça va ?</text>

<resourceURL>http://<REST\_ip>/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/ tel%3A%2B33611223344/b9a06e30-c9b6-4792-8d8e-888f22411d61/messages/QlA0DXiw </resourceURL>

</chatMessage>

</chat:chatMessageNotification>

Or in JSON:

{"chatMessageNotification":{

"callbackData":"abcd","link":",

"link":[

{"rel":"ChatSessionInformation","href":" http:\/\/<REST\_ip>\/netapi\/chat\/v1\/tel%3A%2B33611223355\/oneToOne\/tel%3A%2B33611223344\/b9a06e30-c9b6-4792-8d8e-888f22411d61"},

{"rel":"ChatMessage","href":" http:\/\/<REST\_ip>\/netapi\/chat\/v1\/tel%3A%2B33611223355\/oneToOne\/tel%3A%2B33611223344\/b9a06e30-c9b6-4792-8d8e-888f22411d61\/messages\/a7b833249ec14dd4a21bbe23645eb85"},

{"rel":"ChatNotificationSubscription","href":" http:\/\/<REST\_ip>\/netapi\/chat\/v1\/tel%3A%2B33611223355\/subscriptions\/814838ba-d773-4753-60d3684f6ca "}],

"senderAddress":"tel :+33611223344",

"senderName":"Bob",

"chatMessage":{

"text":"Comment ça va ?",

"resourceURL":"http:\/\/<REST\_ip>\/netapi\/chat\/v1\/tel%3A%2B33611223355\/oneToOne\/tel%3A%2B33611223344\/b9a06e30-c9b6-4792-8d8e-888f22411d61\/messages\/a7b833249ec14dd4a21bbe23645eb85"

}}}

The REST client may now confirm to the mobile phone that it has received and displayed the message. This is done by a PUT request to the resourceURL of the message, for instance:

http://<*REST\_ip*>/netapi/chat/v1/ tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/b9a06e30-c9b6-4792-8d8e-888f22411d61/messages/a7b833249ec14dd4a21bbe23645eb85

The content of the request is a MessageStatusReport structure, which contains the status of the message:

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

<chat:messageStatusReport xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<status>Displayed</status>

</chat:messageStatusReport>

Or in JSON:

{"messageStatusReport": {"status": "Displayed"}}

The interface responds with code “204 No Content” and an empty body.

Note that the REST client can get the status of each message, at any time. It can send a GET request to the resourceURL of the message, completed with /status. For instance:

http://<*REST\_ip*>/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/b9a06e30-c9b6-4792-8d8e-888f22411d61/messages/a7b833249ec14dd4a21bbe23645eb85/status

The result is a MessageStatusReport, the same as above.

### Closing the chat session

Closing a chat session in simply done by sending a DELETE request to the resourceURL identifying the session, without body content.

When the close is complete, the REST client is notified by a ChatEventNotification, of type SessionEnded:

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

<chat:chatEventNotification xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<callBackData>123</callBackData>

<link rel="ChatSessionInformation" href="http://<*REST\_ip*>/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/2ff64c45-8621-4684-8a01-5aacc98370d0"/>

<link rel="ChatNotificationSubscription" href="http://<*REST\_ip*>/chat/v1/tel%3A%2B33611223355/subscriptions/b8756097-99df-431e-8b9d-c19f20555335"/>

<eventType>SessionEnded</eventType>

<eventDescription>Chat session closed by originator.</eventDescription>

</chat:chatEventNotification>

Or, in JSON:

{"chatEventNotification":{

"callBackData":"abcd",

"link":[

{"rel":"ChatSessionInformation","href":"http:\/\/<REST\_ip>\/netapi\/chat\/v1\/tel%3A%2B33611223355\/oneToOne\tel%3A%2B33611223344\/770a737f-09c0-474c-a1d1-3f52496fa6db"},

{"rel":"ChatNotificationSubscription","href":"http:\/\/<REST\_ip>\/netapi\/chat\/v1\/tel%3A%2B33611223355\/subscriptions\/7e88e653-2a9f-4925-a1b4-f265aa490454"}],

"eventType":"SessionEnded",

"eventDescription":" Chat session closed by originator."

}}

At this point, the resourceURL identifying the chat session is destroyed. These elements are no longer available in the NetAPI.

### Unsubscribing to the notification

If you don’t need to be notified of chat events any longer, you can unsubscribe to the notifications by deleting the associated resource. This is achieved by sending a DELETE request to the resourceURL identifying the subscription, with an empty body.

The URL is the identifier provided at subscription, such as:

http://<REST\_ip>/netapi/chat/v1/tel%3A%2B33611223355/subscriptions/814838ba-d773-4753-60d3684f6ca

## File transfer usage overview

The File Transfer API provides a way to send a file to a RCS user.

The REST client must provide an HTTP interface which will be called by the NetAPI to deliver notifications about the delivery of the transfer.

The following example describes typical REST client interactions by showing how to:

* Subscribe to the notification system.
* Create a file transfer session with a mobile phone.
* Unsubscribe to the notification system.

### Call-flow

The following figure illustrates a typical file transfer use-case. The REST client initiating the file transfer session stands on the left side. The mobile phone is shown on the right side.

Please note that the description of the interactions between the NetAPI and the mobile phone follows the RCS specifications and is out of the scope of this document.



### Notification subscription

The subscription to the notification of file transfer follows the same rules as the chat sessions. The only difference stands in the URL called and the structure that is sent to the NetAPI.

The REST client must send a POST request to the URL:

http://*<REST\_ip>*/netapi/filetransfer/v1/*tel:+33612345678*/subscriptions

Where *REST\_ip* is the IP address of the NetAPI and *user* is the phone number of the originator of the chat session (ex: ‘tel:+33612345678’).

The content of the request is a “fileTransferSubscription” structure (as described in 5.3.2.2 ). The next figure shows an example of XML content:

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

<ft:fileTransferSubscription xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<clientCorrelator>12345</clientCorrelator>

<duration>5000</duration>

<callbackReference>

<notifyURL>http://<rest\_client\_ip>/test/</notifyURL>

<callbackData>123</callbackData>

<notificationFormat>XML<notificationFormat>

</callbackReference>

</ft:fileTransferSubscription>

Or, in JSON:

{“fileTransferSubscription”: {

“callbackReference”: {

“notifyURL”: “http://<rest\_client\_ip>/test/”,

“callbackData”: “123”;

“notificationFormat”: “XML”;

},

“duration”: 5000,

“clientCorrelator”: 12345,

}}

The REST client must provide the address of his HTTP interface in the notifyURL field: this is the URL which will be called by the NetAPI when it needs to notify events to the REST client.

The NetAPI responds with code “201 Created”. The response body contains the same structure, but a new field appeared. Called ‘resourceURL’, this is the identifier of the subscription (for instance: “http://<REST\_ip>/netapi/filetransfer/v1/tel:+33612345789/subscriptions/814838ba-d773-4753-60d3684f6ca”). The REST client needs this identifier to unsubscribe the notification at the end of the file transfer.

### File transfer with an URL

Next step is the creation of the file transfer session. This is done by a POST request to the URL:

http://*<REST\_ip>*/netapi/filetransfer/v1/ tel%3A%2B33611223355/

The content of the request must be a “FileTransferSessionInformation” structure (as describe on ). For instance, in XML format:

<ft:fileTransferSessionInformation xmlns:ft="urn:oma:xml:rest:netapi:filetransfer:1">

<originatorAddress>tel:+33611223355</originatorAddress>

<originatorName>Max Muster</originatorName>

<receiverAddress>*tel:+336111223344*</receiverAddress>

<receiverName>Peter E. Xample</receiverName>

<fileInformation>

<fileSelector>

<name>OP-RSC-illustration\_light\_background-01.png</name>

<type>image/png</type>

<size>60457</size>

<hash>

<algorithm>sha-1</algorithm>

<value>1f36609d7cebf3c4fe7f50c0d79259d5b55c6ec9</value>

</hash>

</fileSelector>

<fileDisposition>Attachment</fileDisposition>

<fileDescription>This is my latest picture</fileDescription>

<fileDate>

<cDate>2013-08-21T12:11:10-04:00</cDate>

</fileDate>

<fileURL>http://www.orangepartner.com/sites/default/files/pictures/OP-RSC-illustration\_light\_background-01.png</fileURL>

</fileInformation>

<clientCorrelator>123</clientCorrelator>

</ft:fileTransferSessionInformation>

Here, the fileURL parameter provides the address of the file. Be aware that the value of the parameter “originatorAddress” must be the same between the URL and in the request content, otherwise an error is thrown by the interface.

The same request could be sent in JSON format:

{"fileTransferSessionInformation": {

"clientCorrelator": "123",

"fileInformation": {

"fileDate": {

"cDate": "2013-08-21T12:11:10-04:00"},

"fileDescription": "This is my latest picture",

"fileDisposition": "Attachment",

"fileSelector": {

"hash": {

"algorithm": "sha-1",

"value": "1f36609d7cebf3c4fe7f50c0d79259d5b55c6ec9"

},

"name": "OP-RSC-illustration\_light\_background-01.png",

"size": "60457",

"type": "image/png"

},

"fileURL": "http://www.orangepartner.com/sites/default/files/pictures/OP-RSC-illustration\_light\_background-01.png "

},

"originatorAddress": "tel:+33611223355",

"originatorName": "Max Muster",

"receiverAddress": "tel:+33611223344",

"receiverName": "Peter E. Xample"

}}

If everything goes right, the interface responds with code “201 Created”. The response body contains the same structure, but a new field appeared, called ‘resourceURL’. This time, this is the identifier of the file transfer session. It will be included in the events occurring in the session.

Then, the NetAPI downloads the file and initiates the file transfer with the mobile phone.

### File transfer with embedded file

The NetAPI also supports an embedded file in the request instead of a file URL. In this case, the request content provided by the REST client is a bit more complex:

* The fileURL parameter must not appear in the structure.
* The body content must have the type multipart/form-data, including :
  + The structure describing the session (in XML or JSON format), with the name “root-fields”.
  + The file to be transferred, with a name identical to the file name specified in the structure.

The figure below shows such a request. More information is provided in .

------=\_Part\_0\_17368719.1377849863246

Content-Type: application/xml; name=root-fields

Content-Transfer-Encoding: binary

Content-Disposition: form-data; name="root-fields"; filename="root-fields"

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

<ft:fileTransferSessionInformation xmlns:ft="urn:oma:xml:rest:netapi:filetransfer:1">

<originatorAddress>tel:+33611223355</originatorAddress >

<originatorName>Max Muster</originatorName>

<receiverAddress>*tel:+336111223344* </receiverAddress>

<receiverName>Peter E. Xample</receiverName>

<fileInformation>

<fileSelector>

<name>sunset.jpg</name>

<type>image/jpeg</type>

<size>4096</size>

<hash>

<algorithm>sha-1</algorithm>

<value>58231FE8653BBCF371362F86D471913EE4B1DF2F</value>

</hash>

</fileSelector>

<fileDisposition>Attachment</fileDisposition>

<fileDescription>This is my latest picture</fileDescription>

<fileDate>

<cDate>2013-08-21T12:11:10-04:00</cDate>

</fileDate>

</fileInformation>

<clientCorrelator>104567</clientCorrelator>

</ft:fileTransferSessionInformation>

------=\_Part\_0\_17368719.1377849863246

Content-Type: image/jpeg; name=sunset.jpg

Content-Transfer-Encoding: binary

Content-Disposition: form-data; name="attachments"; filename="sunset.jpg"

*[.. image jpg]*

----=\_Part\_0\_17368719.1377849863246--

The NetAPI checks the parameters and the file submitted and initiates the file transfer to the RCS mobile phone.

### Notifications during the file transfer

The NetAPI sends several notifications during the file transfer. If everything goes right, it pushes the following notification:

* A first ReceiverAcceptanceNotification event with status “connected” when the mobile phone accepts the transfer.
* A first FileTransferEventNotification with type “Successful”, “Declined” or “Failed”, showing if the mobile phone has received the file, declined the transfer, or if the transfer failed for whatever reason.
* A second ReceiverAcceptanceNotification event with status “disconnected” when the mobile phone disconnects the session.
* A second FileTransferEventNotification with type “SessionEnded” when the transfer is complete (this event is sent regardless the type of the first FileTransferEventNotification).

An example of the ReceiverSessionStatusNotification in XML format:

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

<ft:receiverAcceptanceNotification xmlns:ft="urn:oma:xml:rest:netapi:filetransfer:1">

<callBackData>123</callBackData>

<link rel="FileTransferSessionInformation" href="http://<REST\_ip>/netapi/filetransfer/v1/tel%3A%2B33611223355/sessions/5bd08325-e752-4231-be2b-bddd9a7cea1c"/>

<link rel="FileTransferSubscription" href="http:// <REST\_ip>/netapi/filetransfer/v1/ el%3A%2B33611223355/subscriptions/7780215e-4d92-4501-b259-6b2a498d8b35"/>

<receiverAddress>*tel:+336111223344* </receiverAddress>

<receiverName>Peter E. Xample</receiverName>

<receiverSessionStatus>

<status>Connected</status>

</receiverSessionStatus>

</ft:receiverAcceptanceNotification>

Or in JSON:

{"ReceiverAcceptanceNotification":

{"callBackData":"abcd",

"link":[

{"rel":"FileTransferSessionInformation","href":"http:\/\/<REST\_ip>\/netapi\/filetransfer\/v1\/tel%3A%2B33611223355\/sessions\/191d10ba-bece-4057-82f5-e40ab5d121e6"},

{"rel":"FileTransferSubscription","href":"http:\/\/<REST\_ip>\/netapi\/filetransfer\/v1\/tel%3A%2B33611223355\/subscriptions\/ce05c18a-5edd-415c-886f-60809dcb4f18"}],

"receiverAddress":"*tel:+336111223344*",

"receiverName":"Peter E. Xample",

"receiverSessionStatus":{

"status":"Connected

}}}

An example of a FileTransferEventNotification in XML format:

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

<ft:fileTransferEventNotification xmlns:ft="urn:oma:xml:rest:netapi:filetransfer:1">

<callBackData>123</callBackData>

<link rel="FileTransferSessionInformation" href="http://<REST\_ip>/netapi/filetransfer/v1/tel%3A%2B33611223355/sessions/5bd08325-e752-4231-be2b-bddd9a7cea1c"/>

<link rel="FileTransferSubscription" href="http:// <REST\_ip>/netapi/filetransfer/v1/tel%3A%2B33611223355/subscriptions/7780215e-4d92-4501-b259-6b2a498d8b35"/>

<eventType>SessionEnded</eventType>

<eventDescription>Session has ended.</eventDescription>

</ft:fileTransferEventNotification>

Or in JSON:

{"fileTransferEventNotification":{

"callBackData":"abcd",

"link":[

{"rel":"FileTransferSessionInformation","href":"http:\/\/<REST\_ip>\/netapi\/filetransfer\/v1\/tel%3A%2B33611223355\/sessions\/1e986092-54c3-4513-b625-955a691d1128"},

{"rel":"FileTransferSubscription","href":"http:\/\/<REST\_ip>\/netapi\/filetransfer\/v1\/tel%3A%2B33611223355\/subscriptions\/ce05c18a-5edd-415c-886f-60809dcb4f18"}],

"eventType":"SessionEnded",

"eventDescription":"Session has ended."

}}

You do not need to close the file transfer session. The session is automatically destroyed by the NetAPI when the transfer is complete.

### Unsubscribing the notification

If you don’t need any more to be notified of chat events for this particular user, you can unsubscribe the notifications by deleting the associated resource. This is achieved by sending a DELETE request to the resourceURL identifying the subscription, with an empty body.

The URL is the identifier provided at subscription, such as:

http://<REST\_ip>/netapi/filetransfer/v1/tel%3A%2B33611223355/subscriptions/814838ba-d773-4753-60d3684f6ca

# cURL Examples

The following parts show example of calls to the NetAPI. It uses cURL as REST client in a command line.

Note: each section shows the result of the cURL command in verbose mode. In this mode, it dumps the HTTP headers of the request, the HTTP headers of the response and the body of the response. The body of the request, containing de JSON or XML structure, is not shown.

This document is delivered with a zip archive file, in which you will find the files needed for the cURL commands.

## Chat usage examples

### Notification subscription

#### XML

Command (requires the file sub\_chat.xml):

curl -v -H "Accept: application/xml" -H "Content-type: application/xml" -d "@sub\_chat.xml" [http://172.20.65.93/netapi/chat/v1/tel:+33611223355/subscriptions](http://10.67.114.218/netapi/chat/v1/tel:+33611223355/subscriptions)

Result:

> POST /netapi/chat/v1/tel:+33611223355/subscriptions HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/xml

> Content-type: application/xml

> Content-Length: 537

>

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 08:12:13 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/subscriptions

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/xml;charset=UTF-8

< Transfer-Encoding: chunked

<

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

<chat:chatNotificationSubscription xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<callbackReference>

<notifyURL>http://10.67.102.152/test/</notifyURL>

<callbackData>123</callbackData>

<notificationFormat>XML</notificationFormat>

</callbackReference>

<confirmedChatSupported>true</confirmedChatSupported>

<adhocChatSupported>false</adhocChatSupported>

<duration>5000</duration>

<clientCorrelator>12345</clientCorrelator>

<resourceURL>http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/subscriptions/2ae59f0c-8425-4518-80cf-03e03ae1a177</resourceURL>

</chat:chatNotificationSubscription>

#### JSON

Command (requires the file sub\_chat.json):

curl -v -H "Accept: application/json" -H "Content-type: application/json" -d "@sub\_chat.json" http://172.20.65.93/netapi/chat/v1/tel:+33611223355/subscriptions

Result:

> POST /netapi/chat/v1/tel:+33611223355/subscriptions HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/json

> Content-type: application/json

> Content-Length: 271

>

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 08:16:04 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/subscriptions

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/json;charset=UTF-8

< Content-Length: 378

<

{"chatNotificationSubscription":{"callbackReference":{"notifyURL":"http:\/\/10.67.102.152\/test\/","callbackData":123,"notificationFormat":"JSON"},"confirmedChatSupported":false,"adhocChatSupported":true,"duration":1200,"clientCorrelator":12345,"resourceURL":"http:\/\/172.20.65.93\/netapi\/chat\/v1\/tel%3A%2B33611223355\/subscriptions\/f8990b80-d0ab-488f-8cd4-41e79e2c6fed"}}

### Chat session creation

#### XML

Command (requires the file chat\_session.xml):

curl -v -H "Accept: application/xml" -H "Content-type: application/xml" -d "@chat\_session.xml" “http://172.20.65.93/netapi/chat/v1/tel:+33611223355/oneToOne/tel%3A%2B33611223344"

Result:

> POST /netapi/chat/v1/tel:+33611223355/oneToOne/tel%3A%2B33611223344 HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/xml

> Content-type: application/xml

> Content-Length: 426

>

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 08:19:48 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/xml;charset=UTF-8

< Transfer-Encoding: chunked

<

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

<chat:chatSessionInformation xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<subject>Dinner tonight</subject>

<originatorAddress>tel:+33611223355</originatorAddress>

<originatorName>Alice</originatorName>

<tParticipantAddress>tel:+33611223344</tParticipantAddress>

<tParticipantName>Bob</tParticipantName>

<status>Connected</status>

<clientCorrelator>23456</clientCorrelator>

<resourceURL>http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/348e4638-04fb-409b-a9ee-14efa1f74de8</resourceURL>

</chat:chatSessionInformation>

#### JSON

Command (requires the file chat\_session.json):

curl -v -H "Accept: application/json" -H "Content-type: application/json" -d "@chat\_session.json" "http://172.20.65.93/netapi/chat/v1/tel:+33611223344/oneToOne/tel:+33611223355"

Result:

> POST /netapi/chat/v1/tel:+33611223344/oneToOne/tel:+33611223355 HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/json

> Content-type: application/json

> Content-Length: 231

>

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 08:24:12 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223344/oneToOne/tel%3A%2B33611223355

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/json;charset=UTF-8

< Content-Length: 380

<

{"chatSessionInformation":{"subject":"Dinner tonight","originatorAddress":"tel:+33611223344","originatorName":"Alice","tParticipantAddress":"tel:+33611223355","tParticipantName":"Bob","status":"Connected","clientCorrelator":23456,"resourceURL":"http:\/\/172.20.65.93\/netapi\/chat\/v1\/tel%3A%2B33611223344\/oneToOne\/tel%3A%2B33611223355\/18d32ba5-7892-4d29-aa50-38a9c3c21cd4"}}}

### Sending messages

#### XML

Command (requires the file chat\_msg.xml. The session identifier must be changed to a valid one):

curl -v -H "Accept: application/xml" -H "Content-type: application/xml" -d "@chat\_msg.xml" [http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/f2c786ae-9065-43eb-9804-7f820c887b72/messages](http://10.67.114.218/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/f2c786ae-9065-43eb-9804-7f820c887b72/messages)

Result:

> POST /netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/f2c786ae-9065-43eb-9804-7f820c887b72/messages HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/xml

> Content-type: application/xml

> Content-Length: 232

>

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 09:11:12 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/f2c786ae-9065-43eb-9804-7f820c887b72/messages

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/xml;charset=UTF-8

< Transfer-Encoding: chunked

<

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

<chat:chatMessage xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<resourceURL>http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/f2c786ae-9065-43eb-9804-7f820c887b72/messages/36a799370a30436b8c3989a1ed8f28</resourceURL>

<text>Hello World</text>

<reportRequest>Delivered</reportRequest>

<reportRequest>Displayed</reportRequest>

</chat:chatMessage>

#### JSON

Command (requires the file chat\_msg.json. The session identifier must be changed to a valid one):

curl -v -H "Accept: application/json" -H "Content-type: application/json" -d "@chat\_msg.json" "http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/586830e9-d178-4281-90b7-f67e360f1a4d/messages"

Result:

> POST /netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/586830e9-d178-4281-90b7-f67e360f1a4d/messages HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/json

> Content-type: application/json

> Content-Length: 103

>

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 09:14:43 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/586830e9-d178-4281-90b7-f67e360f1a4d/messages

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/json;charset=UTF-8

< Content-Length: 253

<

{"chatMessage":{"resourceURL":"http:\/\/172.20.65.93\/netapi\/chat\/v1\/tel%3A%2B33611223355\/oneToOne\/tel%3A%2B33611223344\/586830e9-d178-4281-90b7-f67e360f1a4d\/messages\/a9d7d494002740b39e29866a0adae6","text":"How are you?","reportRequest":"Sent"}}

### Sending isComposing notification

#### XML

Command (requires the file chat\_iscomposing.xml. The session identifier must be changed to a valid one):

curl -v -H 'Accept: application/xml' -H 'Content-type: application/xml' http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/4d68f9cb-5230-4762-9c89-52b8d2a14c16/messages -d @chat\_iscomposing.xml

Result:

> POST /netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/e80c0a4f-c5b8-4b61-ae01-ba0978c3a517/messages HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/xml

> Content-type: application/xml

> Content-Length: 213

>

} [data not shown]

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Tue, 24 Sep 2013 15:19:41 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/e80c0a4f-c5b8-4b61-ae01-ba0978c3a517/messages

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/xml;charset=UTF-8

< Transfer-Encoding: chunked

<

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

<chat:isComposing xmlns:chat="urn:oma:xml:rest:netapi:chat:1">

<resourceURL>http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/e80c0a4f-c5b8-4b61-ae01-ba0978c3a517/messages/21ca14ed44f54a229b94c9c79e2983</resourceURL>

<state>active</state>

<contenttype>text/plain</contenttype>

<refresh>90</refresh>

</chat:isComposing>

#### JSON

Command (requires the file chat\_iscomposing.json. The session identifier must be changed to a valid one):

curl -v -H 'Accept: application/json' -H 'Content-Type: application/json' http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/ee00728b-f92d-41ca-aa0a-bdcfe6de8559/messages -d @chat\_iscomposing.json

Result:

> POST /netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/ee00728b-f92d-41ca-aa0a-bdcfe6de8559/messages HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/json

> Content-Type: application/json

> Content-Length: 94

>

} [data not shown]

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Tue, 24 Sep 2013 15:24:41 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/oneToOne/tel%3A%2B33611223344/ee00728b-f92d-41ca-aa0a-bdcfe6de8559/messages

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/json;charset=UTF-8

< Content-Length: 271

<

{"isComposing":{"resourceURL":"http:\/\/172.20.65.93\/netapi\/chat\/v1\/tel%3A%2B33611223355\/oneToOne\/tel%3A%2B33611223344\/ee00728b-f92d-41ca-aa0a-bdcfe6de8559\/messages\/87eeacc1db7541db8c08d2326aa80e","state":"active","contenttype":"text\/plain","refresh":90}}'

### Updating status of received messages

#### XML

Command (requires the file chat\_status.xml. The session and message identifiers must be changed to valid ones):

curl -v -X PUT -H 'Accept: application/xml' -H 'Content-type: application/xml' -d @chat\_status.xml http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223344/oneToOne/tel%3A%2B33611223355/91ac8575-c241-47f2-8c01-18976f585d28/messages/d4cfbe00181649429fa036568460f3/status

Result:

> PUT /netapi/chat/v1/tel%3A%2B33611223344/oneToOne/tel%3A%2B33611223355/8bbbd661-f814-4c97-b418-13e2cea583e8/messages/12ec13048639467c87cfd910e0fa60/status HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/xml

> Content-type: application/xml

> Content-Length: 164

>

< HTTP/1.1 204 No Content

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 09:30:25 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223344/oneToOne/tel%3A%2B33611223355/8bbbd661-f814-4c97-b418-13e2cea583e8/messages/12ec13048639467c87cfd910e0fa60/status

< Server: Restlet-Framework/2.1.2

<

#### JSON

Command (requires the file chat\_status.json. The session and message identifiers must be changed to valid ones):

curl -v -X PUT -H 'Accept: application/json' -H 'Content-Type: application/json' -d @chat\_status.json [http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223344/oneToOne/tel%3A%2B33611223355/fa84054c-695d-460f-a3f1-e21eafc3b878/messages/e474fb76cfdd47b0854167f7b77adc/status](http://10.67.114.218/netapi/chat/v1/tel%3A%2B33611223344/oneToOne/tel%3A%2B33611223355/fa84054c-695d-460f-a3f1-e21eafc3b878/messages/e474fb76cfdd47b0854167f7b77adc/status)

Result:

> PUT /netapi/chat/v1/tel%3A%2B33611223344/oneToOne/tel%3A%2B33611223355/fa84054c-695d-460f-a3f1-e21eafc3b878/messages/e474fb76cfdd47b0854167f7b77adc/status HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/json

> Content-Type: application/json

> Content-Length: 48

>

< HTTP/1.1 204 No Content

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 09:31:55 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223344/oneToOne/tel%3A%2B33611223355/fa84054c-695d-460f-a3f1-e21eafc3b878/messages/e474fb76cfdd47b0854167f7b77adc/status

< Server: Restlet-Framework/2.1.2

<

### Closing the chat session

#### XML

Command (the session identifier must be changed to a valid one):

curl -v -X DELETE [http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223344/oneToOne/tel%3A%2B33611223355/4ae7337b-0739-4305-80fe-26a926484a03](http://10.67.114.218/netapi/chat/v1/tel%3A%2B33611223344/oneToOne/tel%3A%2B33611223355/4ae7337b-0739-4305-80fe-26a926484a03)

Result:

> DELETE /netapi/chat/v1/tel%3A%2B33611223344/oneToOne/tel%3A%2B33611223355/4ae7337b-0739-4305-80fe-26a926484a03 HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: \*/\*

>

< HTTP/1.1 204 No Content

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 09:34:34 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223344/oneToOne/tel%3A%2B33611223355/4ae7337b-0739-4305-80fe-26a926484a03

< Server: Restlet-Framework/2.1.2

<

### Unsubscribing the notification

Command (the subscription identifier must be changed to a valid one):

curl -v -X DELETE "http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/subscriptions/3f55eacc-6a61-4959-af6e-cabfd333755c"

Result :

> DELETE /netapi/chat/v1/tel%3A%2B33611223355/subscriptions/3f55eacc-6a61-4959-af6e-cabfd333755c HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: \*/\*

>

< HTTP/1.1 204 No Content

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 09:37:06 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/chat/v1/tel%3A%2B33611223355/subscriptions/3f55eacc-6a61-4959-af6e-cabfd333755c

< Server: Restlet-Framework/2.1.2

<

## File Transfer usage examples

### Notification subscription

#### XML

Command line (requires the file sub\_ft.xml):

curl -v -H "Accept: application/xml" -H "Content-Type: application/xml" -d "@sub\_ft.xml" "http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223355/subscriptions"

Result:

> POST /netapi/filetransfer/v1/tel%3A%2b33611223355/subscriptions HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93:80

> Accept: application/xml

> Content-Type: application/xml

> Content-Length: 351

>

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Fri, 20 Sep 2013 14:44:07 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223355/subscriptions

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/xml;charset=UTF-8

< Transfer-Encoding: chunked

<

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

<ft:fileTransferSubscription xmlns:ft="urn:oma:xml:rest:netapi:filetransfer:1">

<callbackReference>

<notifyURL>http://10.67.102.152/test/</notifyURL>

<callbackData>abcd</callbackData>

<notificationFormat>XML</notificationFormat>

</callbackReference>

<duration>5000</duration>

<clientCorrelator>12345</clientCorrelator>

<resourceURL>http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223355/subscriptions/7d7d7f58-29ef-4f54-b2d0-315f0fc8de9f</resourceURL>

</ft:fileTransferSubscription>

#### JSON

Command line (requires the file sub\_ft.json):

curl -v -H "Accept: application/json" -H "Content-Type: application/json" -d "@sub\_ft.json" "http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223355/subscriptions"

Result:

> POST /netapi/filetransfer/v1/tel%3A%2b33611223355/subscriptions HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/json

> Content-Type: application/json

> Content-Length: 195

>

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 06:54:36 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223355/subscriptions

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/json;charset=UTF-8

< Content-Length: 325

<

{"fileTransferSubscription":{"callbackReference":{"notifyURL":"http:\/\/10.67.102.152\/test","callbackData":"abcd","notificationFormat":"XML"},"duration":5000,"clientCorrelator":12345,"resourceURL":"http:\/\/172.20.65.93\/netapi\/filetransfer\/v1\/tel%3A%2b33611223355\/subscriptions\/557136e4-7a76-43b2-b131-9064270cefff"}}

### File transfer with an URL

We assume that you want to send the file <http://images.wikia.com/pixar/images/4/43/Sunset.jpg> to the mobile phone +33611223344.

#### XML

Command (requires the file ft\_url.xml):

curl -v -H "Accept: application/xml" -H "Content-Type: application/xml" -d "@ft\_url.xml" "http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223355/sessions"

Résult:

> POST /netapi/filetransfer/v1/tel%3A%2b33611223355/sessions HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/xml

> Content-Type: application/xml

> Content-Length: 933

>

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 07:04:56 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223355/sessions

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/xml;charset=UTF-8

< Transfer-Encoding: chunked

<

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

<ft:fileTransferSessionInformation xmlns:ft="urn:oma:xml:rest:netapi:filetransfer:1">

<originatorAddress>tel:+33611223355</originatorAddress>

<originatorName>Max Muster</originatorName>

<receiverAddress>tel:+33611223344</receiverAddress>

<receiverName>Peter E. Xample</receiverName>

<status>Invited</status>

<fileInformation>

<fileSelector>

<name>Sunset.jpg</name>

<type>text/plain</type>

<size>10268</size>

<hash>

<algorithm>sha-1</algorithm>

<value>5040cf93baddc7ae5b6189a4cda1e571142dee0f</value>

</hash>

</fileSelector>

<fileDisposition>Attachment</fileDisposition>

<fileDescription>This is my latest picture</fileDescription>

<fileDate>

<cDate>Sun Aug 21 00:00:00 CEST 2011</cDate>

</fileDate>

<fileURL>http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223355/sessions/fd1cc535-7dfe-4f41-9bbe-7e2f6a0989e0/files/339aa4fe50494c5bbe5e9cdb194e32</fileURL>

</fileInformation>

<clientCorrelator>123</clientCorrelator>

<resourceURL>http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223355/sessions/fd1cc535-7dfe-4f41-9bbe-7e2f6a0989e0</resourceURL>

#### JSON

Command (requires the file ft\_url.json):

curl -v -H "Accept: application/json" -H "Content-Type: application/json" -d "@ft\_url.json" [http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223355/sessions](http://10.67.114.218/netapi/filetransfer/v1/tel%3A%2b33611223355/sessions)

Result:

> POST /netapi/filetransfer/v1/tel%3A%2b33611223355/sessions HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/json

> Content-Type: application/json

> Content-Length: 718

>

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 07:09:28 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223355/sessions

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/json;charset=UTF-8

< Content-Length: 827

<

{"fileTransferSessionInformation":{"originatorAddress":"tel:+33611223355","originatorName":"Max Muster","receiverAddress":"tel:+33611223344","receiverName":"Peter E. Xample","status":"Invited","fileInformation":{"fileSelector":{"name":"Sunset.jpg","type":"image\/jpeg","size":4096,"hash":{"algorithm":"sha-1","value":["58231FE8653BBCF371362F86D471913EE4B1DF2F"]}},"fileDisposition":"Attachment","fileDescription":"This is my latest picture","fileDate":{"cDate":"Sun Aug 21 00:00:00 CEST 2011"},"fileURL":"http:\/\/172.20.65.93\/netapi\/filetransfer\/v1\/tel%3A%2b33611223355\/sessions\/03fd54e2-91ae-401c-8aa9-19c6bc32ebb2\/files\/acd928afad2143bb80633ffb3f0ec0"},"clientCorrelator":104567,"resourceURL":"http:\/\/172.20.65.93\/netapi\/filetransfer\/v1\/tel%3A%2b33611223355\/sessions\/03fd54e2-91ae-401c-8aa9-19c6bc32ebb2"}}

### File transfer with embedded file

We assume that you want to send an image you have on your hard drive (sunset.jpg) to the mobile phone +33611223344.

#### XML

Command (requires the file ft\_embed.xml):

curl -v -H "Accept: application/xml" -F "root-fields=@ft\_embed.xml;type=application/xml" -F "attachments=@sunset.jpg;type=image/jpeg" "http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2B33611223355/sessions"

Result:

> POST /netapi/filetransfer/v1/tel%3A%2B33611223355/sessions HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/xml

> Content-Length: 33092

> Expect: 100-continue

> Content-Type: multipart/form-data; boundary=----------------------------75aae5b19149

>

< HTTP/1.1 100 Continue

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Fri, 20 Sep 2013 15:07:57 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2B33611223355/sessions

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/xml;charset=UTF-8

< Transfer-Encoding: chunked

<

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

<ft:fileTransferSessionInformation xmlns:ft="urn:oma:xml:rest:netapi:filetransfer:1">

<originatorAddress>tel:+33611223355</originatorAddress>

<originatorName>Max Muster</originatorName>

<receiverAddress>tel:+33611223344</receiverAddress>

<receiverName>Peter E. Xample</receiverName>

<status>Invited</status>

<fileInformation>

<fileSelector>

<name>sunset.jpg</name>

<type>image/jpeg</type>

<size>10312</size>

<hash>

<algorithm>sha-1</algorithm>

<value>0d13ce1141e4a204251a0b4fbd440da78d2b8dca</value>

</hash>

</fileSelector>

<fileDisposition>Attachment</fileDisposition>

<fileDescription>This is my latest picture</fileDescription>

<fileDate>

<cDate>Sun Aug 21 00:00:00 CEST 2011</cDate>

</fileDate>

<fileURL>http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2B33611223355/sessions/b1d0df91-414a-4dd3-840a-9af8dfbf8111/files/324ef9869d3449428a5b99a83d1231</fileURL>

</fileInformation>

<clientCorrelator>123</clientCorrelator>

<resourceURL>http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2B33611223355/sessions/b1d0df91-414a-4dd3-840a-9af8dfbf8111</resourceURL>

</ft:fileTransferSessionInformation>

#### JSON

Command (requires the file ft\_embed.json):

curl -v -H "Accept: application/json" -F "root-fields=@ft\_embed.json;type=application/json" -F "attachments=@sunset.jpg;type=image/jpeg" [http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2B33611223355/sessions](http://10.67.114.218/netapi/filetransfer/v1/tel%3A%2B33611223355/sessions)

Result:

> POST /netapi/filetransfer/v1/tel%3A%2B33611223355/sessions HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: application/json

> Content-Length: 32857

> Expect: 100-continue

> Content-Type: multipart/form-data; boundary=----------------------------5da08e515574

>

< HTTP/1.1 100 Continue

< HTTP/1.1 201 Created

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Mon, 23 Sep 2013 07:13:23 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2B33611223355/sessions

< Server: Restlet-Framework/2.1.2

< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept

< Content-Type: application/json;charset=UTF-8

< Content-Length: 827

<

{"fileTransferSessionInformation":{"originatorAddress":"tel:+33611223355","originatorName":"Max Muster","receiverAddress":"tel:+33611223344","receiverName":"Peter E. Xample","status":"Invited","fileInformation":{"fileSelector":{"name":"sunset.jpg","type":"image\/jpeg","size":4096,"hash":{"algorithm":"sha-1","value":["58231FE8653BBCF371362F86D471913EE4B1DF2F"]}},"fileDisposition":"Attachment","fileDescription":"This is my latest picture","fileDate":{"cDate":"Sun Aug 21 00:00:00 CEST 2011"},"fileURL":"http:\/\/172.20.65.93\/netapi\/filetransfer\/v1\/tel%3A%2B33611223355\/sessions\/d1d8823d-9837-465c-9d83-628b3ada60a6\/files\/108350f0296e4e649bb414b5d22c4f"},"clientCorrelator":104567,"resourceURL":"http:\/\/172.20.65.93\/netapi\/filetransfer\/v1\/tel%3A%2B33611223355\/sessions\/d1d8823d-9837-465c-9d83-628b3ada60a6"}}

### Unsubscribing the notification

Command (the subscription identifier must be changed to a valid one):

curl –v -X DELETE http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223344/subscriptions/a2dfda3b-93ea-4905-8cf0-e98022d9c4f2

Result:

> DELETE /netapi/filetransfer/v1/tel%3A%2b33611223344/subscriptions/a0af62d6-9651-4c85-a0dd-dac6176a9323 HTTP/1.1

> User-Agent: curl/7.29.0

> Host: 172.20.65.93

> Accept: \*/\*

>

< HTTP/1.1 204 No Content

< Server: Apache-Coyote/1.1

< X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1

< Date: Fri, 20 Sep 2013 15:13:35 GMT

< Accept-Ranges: bytes

< Location: http://172.20.65.93/netapi/filetransfer/v1/tel%3A%2b33611223344/subscriptions/a0af62d6-9651-4c85-a0dd-dac6176a9323

< Server: Restlet-Framework/2.1.2

<

# API Reference

## Common API specifications

### Common considerations

#### Content-type negotiation

The interface supports XML and JSON formats for the bodies of REST request and response. According to the OMA specifications ([REST\_Common] §5.4 "Content type negotiation"), the format used is a choice of the REST client using either (in order of decrease priority):

* the resFormat query parameter in the URL, if present,
* the Accept header in the REST request, if present,
* the Content-type header in the REST request
* the notificationFormat parameter (for notification subscriptions only).

If the REST client asks for a format different of XML or JSON, the server returns a 406 response code.

#### Resource creation

Typically, a resource is created either following a POST request (to create a child of an existing resource that is addressed by the request), or following a PUT request (to create a new resource as addressed by the request).

If a resource has been created on the server, the server returns an HTTP response with a "201 Created" header and the Location header containing the location of the created resource, and includes in the response body a representation of the created resource.

[Further note that REST resource representations are designed in such a way that they can include a self-reference. (i.e. resourceURL element.). A self-reference is always present in any data structure that is a representation of a resource created by POST, and can be included as necessary in other cases. Since a self-reference can be defined as a mandatory or optional element to accommodate different situations, the normative aspects on the REST client and on the server in each optional usage instance in the specification are clarified as follows: the resourceURL SHALL NOT be included in POST requests by the REST client, but is included in POST requests representing notifications by the server to the REST client, when a complete representation of the resource is embedded in the notification. The resourceURL is also included in responses to any HTTP method that returns an entity body, and in PUT requests.

#### Resource URL considerations

Each resource URL consists of fixed and variable parts. For fixed parts, the exact string value is defined by this specification. For variable parts, rules how to build the string value are defined by this specification.

Resource URL variables are denoted by a name in curly brackets, such as {apiVersion}.

Special characters (such as ‘+’, ‘:’, …) must be url-encoded. For instance, for the user identifiers specified in resource URL : ‘tel%3A%2B63311223344’.

#### User identifiers

Currently, the NetAPI interface is only allowing user’s identity based on a MSISDN phone number in a Tel URI format (i.e. tel:+33601020304). Any other user identifier is denied by the interface with the ServiceException SVC0004.

### Data types

#### XML namespaces

This section defines data types which are shared among two or more RESTful Network APIs.

The namespace for the common data types is:

**urn:oma:xml:rest:netapi:common:1**

#### Structures

**Type: CallbackReference**

An application can use the CallbackReference data structure to subscribe to notifications.

If a parameter callbackData has been passed in a particular subscription, the server copies it into each notification which is related to that particular subscription.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| notifyURL | xsd:anyURI | No | Notify Callback URL |
| 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. These data can be used by the application in the processing of the notification, e.g. for correlation purposes. |
| notificationFormat | NotificationFormat | Yes | Default: XML  Application can specify format of the resource representation in notifications that are related to this subscription. The choice is between {XML, JSON} |

Note: In case the application requires correlating notifications to the related subscription, it can either submit a different notifyURL in each subscription, or use the optional callbackData parameter as a correlator.

**Type: ResourceReference**

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| resourceURL | xsd:anyURI | No | The URL that addresses the resource. The resourceURL SHALL NOT be included in POST requests by the client, but is 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 is also included in responses to any HTTP method that returns an entity body, and in PUT requests. |

**Type: Link**

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| rel | xsd:string | No | Describes the relationship between the URI and the resource |
| href | xsd:anyURI | No | URI |

An element of type Link is provided by the server to point to other resources that are in relationship with the resource.

The rel attribute is a string. The possible values for the string are defined in each RESTful Network API. Rel and href are realized as attributes in the XSD.

**Type: RequestError**

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| link | Link[0..unbounded] | Yes | Link to elements external to the resource |
| serviceException | ServiceException | Choice | Exception Details |
| policyException | PolicyException | Choice | Exception Details |

XSD modelling uses a “choice” to select either a serviceException or a policyException.

**Type: ServiceException**

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| messageId | xsd:string | No | Message identifier, with prefix SVC |
| text | xsd:string | No | Message text, with replacement variables marked with %n, where n is an index into the list of <variables> elements, starting at 1 |
| variables | xsd:string [0..unbounded] | Yes | Variables to substitute into Text string |

**Type: PolicyException**

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| messageId | xsd:string | No | Message identifier, with prefix POL |
| text | xsd:string | No | Message text, with replacement variables marked with %n, where n is an index into the list of <variables> elements, starting at 1 |
| variables | xsd:string [0..unbounded] | Yes | Variables to substitute into Text string |

#### Enumerations

**Enumeration: NotificationFormat**

This enumeration models the possible dispositions of a file transmitted in file transfer.

| **Enumeration** | **Description** |
| --- | --- |
| XML | Notification about new inbound message would use XML format in the POST request |
| JSON | Notification about new inbound message would use JSON format in the POST request |

## Chat API definition

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

### Resources Summary

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

The "apiVersion" URL variable have the value "v1" to indicate that the API corresponds to this version of the specification.

The figure below visualizes the resource structure defined by this specification.

|  |  |
| --- | --- |
| **Resource URI** | **Lifecycle** |
| {*serverRoot*}/chat/{*version*}/{*userid*} | Implicit creation when a REST client subscribes notification for this user. The resource is destroyed if no sub-branch still in uses (no active subscription or oneToOne chat). |
| /subscriptions |
| /{*subscriptionId*} | Available from the subscription of the notification to the cancel of this subscription. |
| /oneToOne | Implicit creation when a REST client creates a chat session between {*userid*} and {*otherUserId*}. The resource is destroyed if no sub-branch still in uses (no active oneToOne chat). |
| /{*otherUserId*} |
| /{*sessionId*} | Resources available during the chat session life time. I.e., when the chat session is closed, all sub-resources are not available in the NetAPI anymore. |
| /status |
| /messages |
| /{*messageId*} |
| /status |

**Figure 1 - Resources 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: Allow the client to manage subscriptions for chat notifications**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Resource URL** | **Base URL: http://{serverRoot} /chat/{apiVersion}/{userId}** | **Data Structures** | **HTTP verbs** | | | |
| **GET** | **PUT** | **POST** | **DELETE** |
| All subscriptions to chat event notifications | /subscriptions | ChatSubscriptionList (used for GET)  ChatNotificationSubscription (used for POST) | Read the list of active chat notification subscriptions | No | Create new subscription to chat notifications | No |
| Individual subscription to chat event notifications | /subscriptions/{subscriptionId} | ChatNotificationSubscription | Read an individual chat notification subscription | No | No | Cancel subscription and stop corresponding notifications |

**Purpose: Allow the client to handle 1-1 chats**

| **Resource URL** | **Base URL: http://{serverRoot} /chat/{apiVersion}/{userId}/ oneToOne** | **Data Structures** | **HTTP verbs** | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **GET** | **PUT** | **POST** | **DELETE** |
| All 1-1 chat sessions between two users | /{otherUserId} | ChatSessionInformation (used for POST) | No | No | Create a 1-1 chat session | No |
| Individual 1-1 chat session | /{otherUserId}/{sessionId} | ChatSessionInformation | Read 1-1 chat session information | No | No | Cancel invitation (Originator)  Decline invitation (Terminating Participant)  Terminate session |
| 1-1 chat session status | /{otherUserId}/{sessionId}/status | ParticipantSessionStatus | No | Accept a 1-1 chat session invitation | No | No |

**Purpose: Allow the client to handle 1-1 chat messages**

| **Resource URL** | **Base URL: http://{serverRoot} /chat/{apiVersion}/{userId}/ oneToOne/{otherUserId}/ {sessionId}** | **Data Structures** | **HTTP verbs** | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **GET** | **PUT** | **POST** | **DELETE** |
| Chat messages in a 1-1 chat | /messages | Choice : ChatMessage or isComposing (used for POST) | No | No | Create (send) a chat message or an isComposing message | No |
| Individual message status in a 1-1 chat | /messages/{messageId}/status | MessageStatusReport | Read the status of a chat message | Report the status of a chat message | No | No |

**Purpose: Allow the client to receive chat notifications**

| **Resource URL** | **Base URL:**  **<Specified by the client>** | **Data Structures** | **HTTP verbs** | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Client notification containing incoming message | Specified by client when subscription is created or provisioned | ChatMessageNotification | No | No | Notify client about incoming chat message | No |
| Client notification about message status | Specified by client when subscription is created or provisioned | ChatMessageStatusNotification | No | No | Notify client about the status of a chat message it has sent | No |
| Client notification about chat session events | Specified by client when subscription is created or provisioned | ChatEventNotification | No | No | Notify client about chat events | No |
| Client notification about subscription cancellation | Specified by client when subscription is created or provisioned | ChatSubscriptionCancellationNotification | No | No | Notify client that a subscription has been cancelled (e.g. expired) | No |

### Data types

#### XML namespaces

The XML namespace for the Chat API data types is:

**urn:oma:xml:rest:netapi:chat:1**

The XML schema for the data structures is defined in the section below.

#### Structures

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

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

**Type: ChatSubscriptionList**

This type represents a list of chat notification subscriptions.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| chatNotificationSubscription | ChatNotificationSubscription  [0..unbounded] | Yes | Array of chat notification subscriptions |
| resourceURL | xsd:anyURI | No | Self referring URL |

**Type: ChatNotificationSubscription**

This type represents a subscription to chat-related event notifications, i.e. all notifications of type ChatEventNotification, ChatMessageNotification, ChatSubscriptionCancellationNotification and ChatMessageStatusNotification targeted at a particular user.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| callbackReference | common:CallbackReference | No | Client's Notification URL and OPTIONAL callbackData |
| confirmedChatSupported | xsd:boolean | Yes | In resource-creating requests, this flag signals to the server that this client supports Confirmed 1-1 Chats. In case this is present and set to true, the client supports Confirmed 1-1 Chats.  In the created resource, the server sets this flag to true as it supports Confirmed 1-1 Chats.  Default: true |
| adhocChatSupported | xsd:boolean | Yes | In resource-creating requests, this flag signals to the server that this client supports Ad-hoc 1-1 Chats. In case this is absent or set to true, the client supports for Ad-hoc 1-1 Chats.  In the created resource, the server sets this flag to false as it does not support Ad-hoc 1-1 Chats.  Default: false |
| 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. |
| 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.  In case the element is present, the server does not alter its value, and provides it as part of the representation of this resource. |
| resourceURL | xsd:anyURI | Yes | Self referring URL.  The resourceURL SHALL NOT be included in POST requests by the client, but is 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 is also included in responses to any HTTP method that returns an entity body, and in PUT requests. |

**Type: ChatEventNotification**

This type represents a notification about chat events that only need to convey the type of event without additional typespecific parameters.

More specific notification types are defined below.

| **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 notifications about chat events |
| link | common:Link [0..unbounded] | Yes | Links to other resources that are in relationship to the notification (e.g. related chat session) |
| eventType | EventType | No | Type of event |
| eventDescription | xsd:string | Yes | Textual description of the event |

**Type: ChatMessageNotification**

This type represents a notification delivering an incoming chat message.

| **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 notifications about chat events |
| link | common:Link [0..unbounded] | Yes | Links to other resources that are in relationship to the notification (e.g. related chat session) |
| senderAdress | xsd:anyURI | No | Identifier of the Participant that sent the message (e.g. 'tel' URI) |
| senderName | xsd:string | Yes | Name of the Sender |
| chatMessage | ChatMessage | Choice | The actual message |
| isComposing | IsComposing | Choice | “isComposing” message |
| dateTime | xsd:dateTime | Yes | The time when the message was sent |

XSD modelling uses a “choice” to select either chatMessage or isComposing.

**Type: ChatMessageStatusNotification**

This type represents a notification about the status of a chat message.

| **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 notifications about chat events |
| link | common:Link [0..unbounded] | Yes | Links to other resources that are in relationship to the notification (e.g. related chat session) |
| status | MessageStatus | No | Indicates the status of the message |
| errorCode | xsd:string | Yes | Code of the error, if any |
| description | xsd:string | Yes | Description of the error, if any |

**Type: ChatMessage**

This type represents a chat message.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| text | xsd:string | No | content of a chat message |
| reportRequest | MessageStatus [0..unbounded] | Yes | List of status events to report |
| resourceURL | xsd:anyURI | Yes | Self referring URL.  The resourceURL SHALL NOT be included in POST requests by the client, but is 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 is also included in responses to any HTTP method that returns an entity body, and in PUT requests. |

**Type: MessageStatusReport**

This type represents a response to a chat message notification.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| status | MessageStatus | No | Indicates the status of the message |

**Type: ParticipantSessionStatus**

This type represents the status of a Participant in the chat session.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| status | ParticipantStatus | No | Status of the Participant  The client is not allowed to use in requests the remaining values of the ParticipantStatus enumeration.  If the client uses one of these in a request, the server responds with an HTTP status code “400 Bad request” and returns a SVC0003 exception with the list of valid values set to “Connected”. |

**Type: ChatSessionInformation**

This type represents information about a 1-1 chat session.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| subject | xsd:string | Yes | Topic of the chat session, which MAY be set by the Originator and is passed to the invited Participant. |
| originatorAddress | xsd:anyURI | No | The address (e.g. 'tel' URI) of the Originator.  If originatorAddress is also part of the request URL, the two MUST have the same value. |
| originatorName | xsd:string | Yes | Human readable name of the Originator |
| tParticipantAddress | xsd:anyURI | No | The address (e.g. 'tel' URI) of the Terminating Participant  If tParticipantAddress is also part of the request URL, the two MUST have the same value. |
| tParticipantName | xsd:string | Yes | Human readable name of the Terminating Participant |
| status | ParticipantStatus | Yes | Connection status of the Terminating Participant  Set by the server, SHALL NOT be present in request bodies during resource creation |
| 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.  In case the element is present, the server does not alter its value, and provides it as part of the representation of this resource. |
| resourceURL | xsd:anyURI | Yes | Self referring URL.  The resourceURL SHALL NOT be included in POST requests by the client, but is 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 is also included in responses to any HTTP method that returns an entity body, and in PUT requests. |

**Type: IsComposing**

This type represents a message indicates to the recipient that the Sender is editing (composing) a message.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| state | xsd:string | No | Sender state, as defined in [RFC3994]. One of “idle”, “active” |
| lastactive | xsd:dateTime | Yes | Time of last activity, as defined in [RFC3994] |
| contenttype | xsd:string | Yes | Type of message being created, as defined in [RFC3994]  This element contains either a MIME media type, or a combination of media type and subtype. |
| refresh | xsd:positiveInteger | Yes | Time interval in seconds after which the Receiver can expect an update from the Sender, as defined in [RFC3994] |
| (any) | any[0..unbounded] | Yes | Any element from another namespace, as defined in [RFC3994] |

**Type: ChatSubscriptionCancellationNotification**

A type containing the 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. |
| reason | common:ServiceError | Yes | Reason notification is being discontinued. Is present if the reason is different from a regular expiry of the subscription. |
| link | common:Link[1..unbounded] | No | Link to other resources that are in relationship with the resource. |

#### Enumerations

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

**Enumeration: ParticipantStatus**

This enumeration defines the possible values for chat Participant status.

| **Enumeration** | **Description** |
| --- | --- |
| Invited | Participant was invited to the session. |
| Connected | Participant is connected to the session. |
| Disconnected | Participant is disconnected from the session. |

**Enumeration: EventType**

This enumeration is defines the types of events. It is used in notifications.

| **Enumeration** | **Description** |
| --- | --- |
| SessionCancelled | The Originator has cancelled the chat session during the invite phase |
| SessionEnded | The session has ended |
| Declined | The Participant has declined the chat session invite |
| Accepted | The Participant has accepted the chat invite |
| Timeout | The session invitation to the Participant has timed out |
| Unreachable | The Participant could not be reached or is unknown |

**Enumeration: MessageStatus**

This enumeration defines the possible values for the message status.

| **Enumeration** | **Description** |
| --- | --- |
| Sent | Message was sent and has not yet reached the recipient. Initial status of a message, not used in PUT requests from the client. |
| Delivered | Message was delivered to the client. |
| Displayed | Message was displayed by the client. |
| Failed | Message was not delivered to the client. Only used in notifications from the server, but not in PUT requests from the client. |

## File Transfer API definition

### Resources Summary

This section summarizes all the resources used by the RESTful Network API for File Transfer.

The "apiVersion" URL variable SHALL have the value "v1" to indicate that the API corresponds to this version of the specification.

The figure below visualizes the resource structure defined by this specification.

|  |  |
| --- | --- |
| **Resource URI** | **Lifecycle** |
| {serverRoot}/filetransfer/{version}/{userid} | Implicit creation when a REST client subscribes notification for this user. The resource is destroyed when no more sub-branches resources is active for the considered user. |
| /subscriptions |
| /{subscriptionId} | Available from the subscription of the notification to the cancel of this subscription. |
| /sessions | Implicit creation when a REST client starts a File Transfer session. This sub-branch is deleted when no more FT session still alive. |
| /{sessionId} | Resources available during the file transfer session life time. I.e., when the session is closed, all sub-resources are not available in the NetAPI anymore. |
| /status |

**Figure 2 - 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: Allow client to manage file transfer notifications subscriptions**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Resource URL** | **Base URL: http://{serverRoot}/filetran**  **sfer/{apiVersion}/{userId}** | **Data Structures** | **HTTP verbs** | | | |
| **GET** | **PUT** | **POST** | **DELETE** |
| All subscriptions to file transfer notifications | /subscriptions | FileTransferSubscriptionList (used for GET)  FileTransferSubscription (used for POST) | Read all active file transfer subscriptions related to a user | No | Create new subscription for file transfer notification of an user | No |
| Individual subscription to file transfer notifications | /subscriptions/{subscriptionId} | FileTransferSubscription | Retrieves an active file transfer subscription related to a user | No | No | Cancel subscription and stop corresponding notifications |

**Purpose: To allow client to manage 1-1 file transfer sessions**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Resource URL** | **Base URL: http://{serverRoot}/filetran**  **sfer/{apiVersion}/{userId}** | **Data Structures** | **HTTP verbs** | | | |
| **GET** | **PUT** | **POST** | **DELETE** |
| All 1-1 file transfer sessions | /sessions | FileTransferSessionInformation (used for POST) | No | No | Create a new 1-1 file transfer session | No |
| Individual 1-1 file transfer session | /sessions/{sessionId} | FileTransferSessionInformation | Retrieve file transfer session information | No | No | Cancel or abort invitation (Originator) |

**Purpose: To allow server to notify client about file transfer session status**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Resource URL** | **Base URL: <Specified by the client>** | **Data Structures** | **HTTP verbs** | | | |
| **GET** | **PUT** | **POST** | **DELETE** |
| Client notification about file transfer events | Specified by client when subscription is created or provisioned | FileTransferEventNotification | No | No | Notify the client about file transfer events | No |
| Client notification about Receiver acceptance | Specified by client when subscription is created or provisioned | ReceiverAcceptanceNotification | No | No | Notify the client about Receiver accepted the invitation | 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 |

### Data types

#### XML namespaces

The XML namespace for the Chat API data types is:

**urn:oma:xml:rest:netapi:filetransfer:1**

The XML schema for the data structures is defined in the section below.

#### Structures

The subsections of this section define the data structures used in the File Transfer API.

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

**Type: FileTransferSessionInformation**

This type represents information about a File Transfer Session.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| originatorAddress | xsd:anyURI | No | Address of the Originator of this file transfer session (e.g. 'tel' URI) |
| originatorName | xsd:string | Yes | Name of the Originator of this file transfer session |
| receiverAddress | xsd:anyURI | No | Address of the Receiver of this file transfer session( e.g. 'tel' URI) |
| receiverName | xsd:string | Yes | Name of the Receiver of this file transfer session |
| status | status | Yes | Connection status of the Receiver. Set by the server. SHALL NOT be present in request bodies during resource creation. |
| fileInformation | FileInformation | No | The file’s attributes. |
| 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.  In case the element is present, the server does not alter its value, and provides it as part of the representation of this resource. |
| resourceURL | xsd:anyURI | Yes | Self referring URL.  The resourceURL SHALL NOT be included in POST requests by the client, but is 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 is also included in responses to any HTTP method that returns an entity body, and in PUT requests. |

**Type: FileInformation**

This type represents a set of attributes of a file.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| fileSelector | FileSelector | No | A tuple of file : file name, size, type and hash as specified in [RFC5547] |
| fileDisposition | FileDisposition | Yes | It is used by the file sender to indicate a preferred disposition of the file. To indicate that a file should be automatically rendered, the value is “Render”. To indicate that a file should not be automatically rendered, the value is “Attachment”.  See [RFC5547].  Default value is “Attachment”. |
| fileDescription | xsd:string | Yes | Human-readable short description of the file which could be set by the Originator. |
| fileDate | FileDate | Yes | The dates on which the file was created, modified, or last read as specified in [RFC5547]. |
| fileIcon | xsd:anyURI | Yes | This field is not used by the interface. It stands for compatibility reasons. |
| fileURL | xsd:anyURI | Yes | The URL link to actual file content.  In POST operation during resource creation of Create a new 1-1 file transfer session, it is the external file repository URL set by Originator.  If it is present, it indicates that there is no file content included in the request operation.  If it is not present, it indicates that the actual file content is included in the HTTP request body during resource creation of Create a new 1-1 file transfer session or Send file. File content can be represented as multipart/form-data entity bodies, where the first entry of the form are the root fields and the second entry of the form are the file content.  This field is not used in session invitation notifications. |
| resourceURL | xsd:anyURI | Yes | Self referring URL.  The resourceURL SHALL NOT be included in POST requests by the client, but is 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 is also included in responses to any HTTP method that returns an entity body, and in PUT requests. |

**Type: FileSelector**

This type represents the basic information of a file such as name.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| name | xsd:string | No | The name of the file.  See [RFC5547].  Note: in a file transfer session, the file name is unique. |
| type | xsd:string | No | The MIME type of the file. It is concatenated by type, “/” and subtype.  See [RFC5547]. |
| size | xsd: unsignedLong | Yes | The size of the file in octets.  See [RFC5547]. |
| hash | HashInformation | Yes | The file hash information including hash algorithm and hash value.  See [RFC5547]. |

**Type: FileDate**

This type represents the dates on which the file was created, modified, or last read, it MAY contain any combination of “cDate”, “mDate” and “rDate”.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| cDate | xsd:dateTime | Yes | The date on which the file was last created. See [RFC5547]. |
| mDate | xsd:dateTime | Yes | The date on which the file was last modified.  See [RFC5547]. |
| rDate | xsd:dateTime | Yes | The date on which the file was last read.  See [RFC5547]. |

**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).  See [RFC5547]. |
| value | xsd:hexBinary | No | The hash value of the file.  See [RFC5547]. |

**Type: FileTransferEventNotification**

This type represents a notification about file transfer events that only need to convey the type of event without additional

| **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 file transfer notifications. |
| link | common:Link [0..unbounded] | Yes | Links to other resources that are in relationship to the notification (e.g. related File Transfer session). |
| eventType | EventType | No | Type of event |
| eventDescription | xsd:string | Yes | Textual description of the event |

**Type: FileTransferSubscriptionList**

This type represents a list of FileTransfer notification subscriptions.

| **Element** | **Type** | **Optional** | **Description** |
| --- | --- | --- | --- |
| fileTransferSubscription | FileTransferSubscription [0..unbounded] | Yes | Array of File Transfer notification subscriptions |
| resourceURL | xsd:anyURI | No | Self referring URL |

**Type: FileTransferSubscription**

This type represents a subscription to file transfer related notifications, i.e. FileTransferEventNotification, and ReceiverAcceptanceNotification 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. |
| 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.  In case the element is present, the server does not alter its value, and provides it as part of the representation of this resource. |
| resourceURL | xsd:anyURI | Yes | Self referring URL.  The resourceURL SHALL NOT be included in POST requests by the client, but is 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 is also included in responses to any HTTP method that returns an entity body, and in PUT requests. |

**Type: ReceiverAcceptanceNotification**

This type represents the Receiver 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 file transfer notifications. |
| link | common:Link [0..unbounded] | Yes | Links to other resources that are in relationship to the notification (e.g. related ReceiverSessionStatus). |
| receiverAddress | xsd:anyURI | No | Address (e.g. 'sip' URI, 'tel' URI, 'acr' URI) of the Receiver of this file transfer session |
| receiverName | xsd:string | Yes | Name of the Receiver of this file transfer session |
| receiverSessionStatus | ReceiverSessionStatus | No | Status of a Receiver in the file transfer session |

**Type: SubscriptionCancellationNotification**

A type containing the subscription cancellation 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 file transfer notifications. |
| Reason | common:ServiceError | Yes | Reason notification is being discontinued. Is be present if the reason is different from a regular expiry of the subscription. |
| link | common:Link[1..unbounded] | No | Link to other resources that are in relationship with the resource. |

#### Enumerations

The subsections of this section define the enumerations used in the File Transfer API.

**Enumeration: FileDisposition**

This enumeration models the possible dispositions of a file transmitted in file transfer.

| **Enumeration** | **Description** |
| --- | --- |
| Render | Indicates that the file should be automatically rendered. |
| Attachment | Indicates that the file should not be automatically rendered. |

**Enumeration: EventType**

This enumeration defines the types of events. It is used in notifications.

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

**Enumeration: ReceiverStatus**

This enumeration defines the possible values for a Receiver in a file transfer session.

| **Enumeration** | **Description** |
| --- | --- |
| Invited | User was invited to the session. |
| Connected | User is connected to the session. |
| Disconnected | User is disconnected from the session. |

#### MIME multipart representation

The sending file operation can use MIME multipart/form-data representation. The POST request should contains at least two contents items.

First item must be named root-fields and it contains XML/JSON file transfer session and file details. Its content-type header must be as “application/<format>; name=root-fields”. Its content-Disposition header must be “form-data; name=”root-fields”; filename=” root-fields”;

The second item must be the file to be transferred. Its Content-Disposition header must be “form-data; name=”attachments”; filename=”<file name>””.

In case of POST request contains more than two items, structure of second and next items will change. They will represent as MIME subparts of a general request second part with the next headers:

Content-Disposition: form-data; name=“attachments”

Content-Type: multipart/mixed

Then, the possible file content SHALL be included as subparts, with:

Content-Disposition: attachments; filename=“<Name of the file>”

Content-Type: <Corresponding Content-Type>

Then, the possible file icon SHALL be included as subparts, with:

Content-Disposition: attachments; filename=“icon”

Content-Type: <Corresponding Content-Type>

The next figure shows an example of the body content of such a request:

------=\_Part\_0\_17368719.1377849863246

Content-Type: application/xml; name=root-fields

Content-Transfer-Encoding: binary

Content-Disposition: form-data; name="root-fields"; filename="root-fields"

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

<ft:fileTransferSessionInformation xmlns:ft="urn:oma:xml:rest:netapi:filetransfer:1">

<originatorAddress>tel:+33612345789</originatorAddress >

<originatorName>Max Muster</originatorName>

<receiverAddress>tel:+33609875432</receiverAddress>

<receiverName>Peter E. Xample</receiverName>

<fileInformation>

<fileSelector>

<name>sunset.jpg</name>

<type>image/jpeg</type>

<size>4096</size>

<hash>

<algorithm>sha-1</algorithm>

<value>58231FE8653BBCF371362F86D471913EE4B1DF2F</value>

</hash>

</fileSelector>

<fileDisposition>Attachment</fileDisposition>

<fileDescription>This is my latest picture</fileDescription>

<fileDate>

<cDate>2011-08-21T00:00:00-04:00</cDate>

</fileDate>

<fileIcon>cid:id3@alicepc.example.com</fileIcon>

</fileInformation>

<clientCorrelator>104567</clientCorrelator>

</ft:fileTransferSessionInformation>

------=\_Part\_0\_17368719.1377849863246

Content-Type: image/gif; name=icon

Content-Transfer-Encoding: binary

Content-Disposition: form-data; name="attachments"; filename="icon"

[..small preview icon...]

------=\_Part\_0\_17368719.1377849863246

Content-Type: image/jpeg; name=sunset.jpg

Content-Transfer-Encoding: binary

Content-Disposition: form-data; name="attachments"; filename="sunset.jpg"

[.. image jpg]

----=\_Part\_0\_17368719.1377849863246--

## Fault définitions

### Common faults

#### HTTP Response Codes

Following is a list of often used HTTP response codes for RESTful Network APIs. The full set of HTTP response codes can be found in [RFC2616].

|  |  |  |
| --- | --- | --- |
| **Response code** | **Reason** | **Description** |
| 200 | Ok | The operation was successful. |
| 201 | Created | The operation was successful, and a new resource has been created by the request. |
| 202 | Accepted | The request has been accepted for processing, but the processing has not been completed (yet). |
| 204 | No content | The operation was successful, and the response intentionally contains no data. |
| 300 | Multiple choices | The requested resource corresponds to any one of a set of representations, each with its own specific location. In the OMA RESTful Network APIs, this code is for instance used to signal the supported API versions in case an unsupported version was requested for a particular resource. |
| 303 | See others | The response to the request can be found under a different URI and can be retrieved using a GET method on that resource. |
| 304 | Not Modified | The condition specified in the conditional header(s) was not met for a read operation. |
| 400 | Bad Request | In the original HTTP meaning, this error signals invalid parameters in the request. In OMA RESTful Network APIs, this code is also used as the “catch-all” code for error situations triggered by a client request, for which no matching HTTP error code exists. |
| 401 | Unauthorized | Authentication has failed, but the application can retry the request using authorization. |
| 403 | Forbidden | The server understood the request, but is refusing to fulfil it (e.g. because application doesn't have permissions to access resource due to the policy constraints) |
| 404 | Not Found | The specified resource does not exist. |
| 405 | Method Not Allowed | The actual HTTP method (such as GET, PUT, POST, DELETE) is not supported by the resource |
| 406 | Not Acceptable | The content type requested is not acceptable for the resource. |
| 408 | Request Timeout | The client did not produce a response in the time the server was prepared to wait. |
| 409 | Conflict | Occurs in situations when two instances of an application are trying to modify a resource in parallel, in a nonsynchronized way. |
| 410 | Gone | The requested resource is no longer available at the server. |
| 411 | Length Required | The Content-Length header was not specified. |
| 412 | Precondition Failed | The condition specified in the conditional request header(s) was not met for an operation. |
| 413 | Request Entity too Large | The size of the request body exceeds the maximum size permitted by the server implementation. |
| 414 | Request-URI too Long | The length of the request URI exceeds the maximum size permitted by the server implementation. |
| 415 | Unsupported Media Type | The content type of the request body is unsupported by the server. |
| 500 | Internal Server Error | General, catch-all server-side error |
| 503 | Service unavailable | The server is currently unable to receive requests, but the request can be retried at a later time. |

#### HTTP Response Codes in Response to Notifications

The server assumes the notification has been sent successfully, regardless the HTTP response code sent by the client application.

#### Service Exceptions

Faults related to the operation of the service, not including policy related faults, result in the return of a ServiceException message.

**SVC0001: Service error**

| **Name** | **Description** |
| --- | --- |
| MessageId | SVC0001 |
| Text | A service error occurred. Error code is %1 |
| Variables | %1 Error code from service |
| HTTP status code(s) | 400 Bad request |

**SVC0002: Invalid input value**

| **Name** | **Description** |
| --- | --- |
| MessageId | SVC0002 |
| Text | Invalid input value for message part %1 |
| Variables | %1 - message part |
| HTTP status code(s) | 400 Bad request |

**SVC0003: Invalid input value with list of valid values**

| **Name** | **Description** |
| --- | --- |
| MessageId | SVC0003 |
| Text | Invalid input value for message part %1, valid values are %2 |
| Variables | %1 - message part  %2 - list of valid values |
| HTTP status code(s) | 400 Bad request |

**SVC0004: No valid address(es)**

| **Name** | **Description** |
| --- | --- |
| MessageId | SVC0004 |
| Text | No valid addresses provided in message part %1 |
| Variables | %1 - message part |
| HTTP status code(s) | 1. t found, 400 Bad request |

#### Policy Exceptions

Faults related to policies associated with the service result in the return of a PolicyException message.

**POL0001: Policy error**

| **Name** | **Description** |
| --- | --- |
| MessageId | POL0001 |
| Text | A policy error occurred. Error code is %1 |
| Variables | %1 Error code from service - meaningful to support, and may be documented in product documentation |
| HTTP status code(s) | 403 Forbidden |

This exception represents a general, catch-all policy error. It can be used if no more information regarding the error is available, or if it is not intended that the network shares more detailed information with the application.

**POL0011: Media Type not supported**

| **Name** | **Description** |
| --- | --- |
| MessageId | POL0011 |
| Text | Media type not supported |
| Variables | None |
| HTTP status code(s) | 406 Not acceptable |

**POL0013: Addresses duplication**

| **Name** | **Description** |
| --- | --- |
| MessageId | POL0013 |
| Text | Duplicated addresses |
| Variables | %1 – duplicated addresses |
| HTTP status code(s) | 400 Bad request |

**POL1016: File size limit exceeded**

| **Name** | **Description** |
| --- | --- |
| MessageId | POL1016 |
| Text | File size exceeds the limit %1 |
| Variables | %1 – file size limit |
| HTTP status code(s) | 1. rbidden |

#### Examples of exceptions

The interface checks originating userId format and sends a “400 Bad Request” error response with the service exception generated (see 5.4.1.3). The next figure shows the XML content of the response in this case:

<ServiceException>

<messageId>SVC0002</messageId>

<text>Invalid input value for message part %1</text>

<variable>Originating UserId tel:+ format is not valid : valid format is ‘tel:+digits’</variable>

</ServiceException>

### Chat API Exception Definitions

#### Service Exceptions

There are no additional specific Service Exception codes defined for this release of the Chat API.

#### Policy Exceptions

There are no additional specific Policy Exception codes defined for this release of the Chat API.

### File transfer API Exception Definitions

#### Service Exceptions

There are no additional specific Service Exception codes defined for this release of the File Transfer API.

#### Policy Exceptions

There are no additional specific Policy Exception codes defined for this release of the File Transfer API.