Final draft ETSI ES 202 504-5 V1.1.1 (2008-02)

ETSI Standard

Open Service Access (OSA);
Parlay X Web Services;
Part 5: Multimedia Messaging
(Parlay X 3)

Reference
DES/TISPAN-01034-5-OSA

Keywords
API, OSA, service

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN), and is now submitted for the ETSI standards Membership Approval Procedure.

The present document is part 5 of a multi-part deliverable covering Open Service Access (OSA); Parlay X Web Services, as identified below:

Part 1: "Common";

Part 2: "Third Party Call";

Part 3: "Call Notification";

Part 4: "Short Messaging";

Part 5: "Multimedia Messaging";

Part 6: "Payment";

Part 7: "Account Management";

Part 8: "Terminal Status";

Part 9: "Terminal Location";

Part 10: "Call Handling";

Part 11: "Audio Call";

Part 12: "Multimedia Conference";

Part 13: "Address List Management";

Part 14: "Presence";

Part 15: "Message Broadcast";

Part 16: "Geocoding";

Part 17: "Application-driven Quality of Service (QoS)";

Part 18: "Device Capabilities and Configuration";

Part 19: "Multimedia Streaming Control";

Part 20: "Multimedia Multicast Session Management".

The present document has been defined jointly between ETSI, The Parlay Group (http://www.parlay.org) and the 3GPP.

The present document forms part of the Parlay X 3.0 set of specifications.

The present document is equivalent to 3GPP TS 29.199-05 V7.2.0 (Release 7).

1 Scope

The present document is part 5 of the Stage 3 Parlay X 3 Web Services specification for Open Service Access (OSA).

The OSA specifications define an architecture that enables application developers to make use of network functionality through an open standardized interface, i.e. the OSA APIs.

The present document specifies the Multimedia Messaging Web Service. The following are defined here:

- Name spaces.
- Sequence diagrams.
- Data definitions.
- Interface specification plus detailed method descriptions.
- Fault definitions.
- Service Policies.
- WSDL Description of the interfaces.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific.

- For a specific reference, subsequent revisions do not apply.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
  - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
  - for informative references.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

For online referenced documents, information sufficient to identify and locate the source shall be provided. Preferably, the primary source of the referenced document should be cited, in order to ensure traceability. Furthermore, the reference should, as far as possible, remain valid for the expected life of the document. The reference shall include the method of access to the referenced document and the full network address, with the same punctuation and use of upper case and lower case letters.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are indispensable for the application of the present document. For dated references, only the edition cited applies. For non-specific references, the latest edition of the referenced document (including any amendments) applies.

[1] W3C Recommendation (2 May 2001): "XML Schema Part 2: Datatypes".

NOTE: Available at: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

[2] ETSI ES 202 504-1: "Open Service Access (OSA); Parlay X Web Services; Part 1: Common (Parlay X 3)".

[3] W3C Note (11 December 2000): "SOAP Messages with Attachments".

NOTE: Available at: http://www.w3.org/TR/SOAP-attachments.

[4] IETF RFC 2822: "Internet Message Format".

NOTE: Available at: http://www.ietf.org/rfc/rfc2822.txt

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ES 202 504-1 [2] and the following apply:

Whitespace: See definition for CFWS as defined in RFC 2822 [4].

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ES 202 504-1 [2] and the following apply:

EMS	Enhanced Messaging Service
IM	Instant Messaging
MMS	Multimedia Messaging Service
MMS-C	Multimedia Messaging Service-Centre
OMA	Open Mobile Alliance
SMS	Short Message Service
WAP	Wireless Application Protocol

4 Detailed service description

Currently, in order to programmatically receive and send Multimedia Messages, it is necessary to write applications using specific protocols to access MMS functions provided by network elements (e.g. MMS-C). This approach requires application developers to have a high degree of network expertise.

This contribution defines a Multimedia Messaging Web Service that can map to SMS, EMS, MMS, IM, E-mail, etc.

The choice is between defining one set of interfaces per messaging network or a single set common to all networks; e.g. we could define sendMMS, sendEMS, sendSMS, etc., or just use sendMessage. Although the more specific the API the easier it is to use, there are advantages to a single set of network-neutral APIs. These advantages include:

- improved service portability;
- lower complexity, by providing support for generic user terminal capabilities only.

For this version of the Parlay X 3 specification, we provide sets of interfaces for two messaging Web Services: Short Messaging (part 4) and Multimedia Messaging (the present document), which provides generic messaging features (including SMS).

Multimedia Messaging provides operations (see clause 8.1, **SendMessage** API) for sending a Multimedia message to the network and a polling mechanism for monitoring the delivery status of a sent Multimedia message. It also provides an asynchronous notification mechanism for delivery status (see clause 8.3, **MessageNotification** API). In addition, a mechanism is provided to start and stop the notification of delivery receipts (see clause 8.4, **MessageNotificationManager** API).

Multimedia Messaging also allows an application to receive Multimedia messages. Both a polling (see clause 8.2, **ReceiveMessage** API) and an asynchronous notification mechanism (see clause 8.3, **MessageNotification** API and clause 8.4, **MessageNotificationManager** API) are available.

Figure 1 shows an example scenario using **sendMessage** and **getMessageDeliveryStatus** to send data to subscribers and to determine if the data has been received by the subscriber. The application invokes a Web Service to retrieve a stock quote (1) and (2) and sends the current quote - **sendMessage** - using the Parlay X Interface (3) of the Multimedia Messaging Web Service. After invocation, the Multimedia Message Web Service sends the message to an MMS-C using the MM7 interface (4) for onward transmission (5) to the subscriber over the Mobile network.

Later, when the next quote is ready, the application checks to see - **getMessageDeliveryStatus** - if the previous quote has been successfully delivered to the subscriber. If not, it may for instance perform an action (not shown) to provide a credit for the previous message transmission. This way, the subscriber is only charged for a stock quote if it is delivered on time.

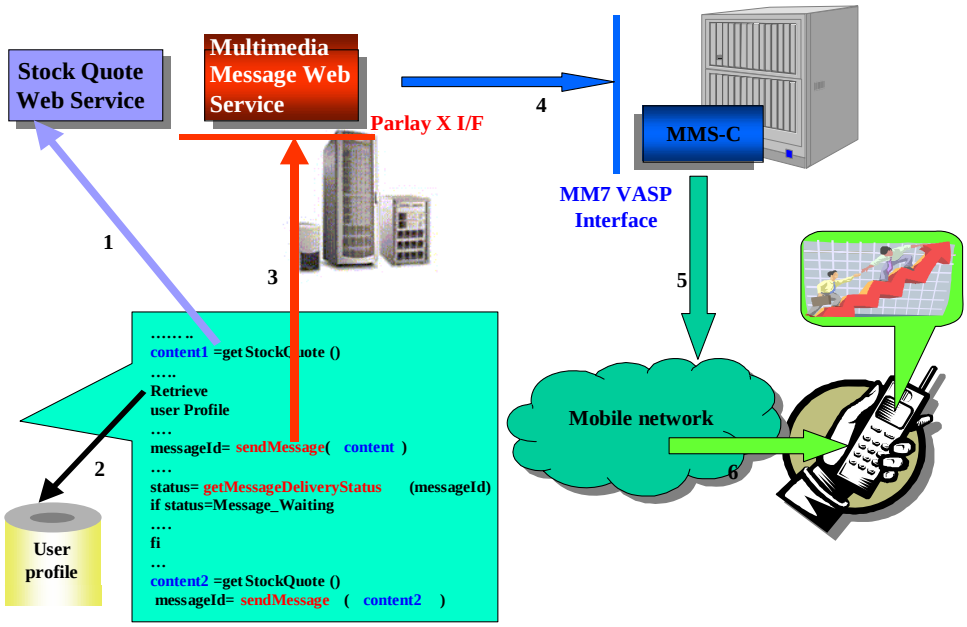

Figure 1: Multimedia Messaging Scenario

Alternatively this service could have been built using WAP Push features in the network.

Figure 2 shows an example scenario using **sendMessage** and **getMessageDeliveryStatus** to send a link to subscribers and to determine if the data has been received by the subscriber. The application invokes a Web Service to generate a stock quote graph (1) and (2) and sends the current quote as a WAP Push link - **sendMessage** - using the Parlay X Interface (3) of the Multimedia Messaging Web Service. After invocation, the Multimedia Message Web Service sends the message to an SMS-C (4) for onward transmission (5) to the subscriber over the Mobile network. The subscriber can then open the link and access his content.

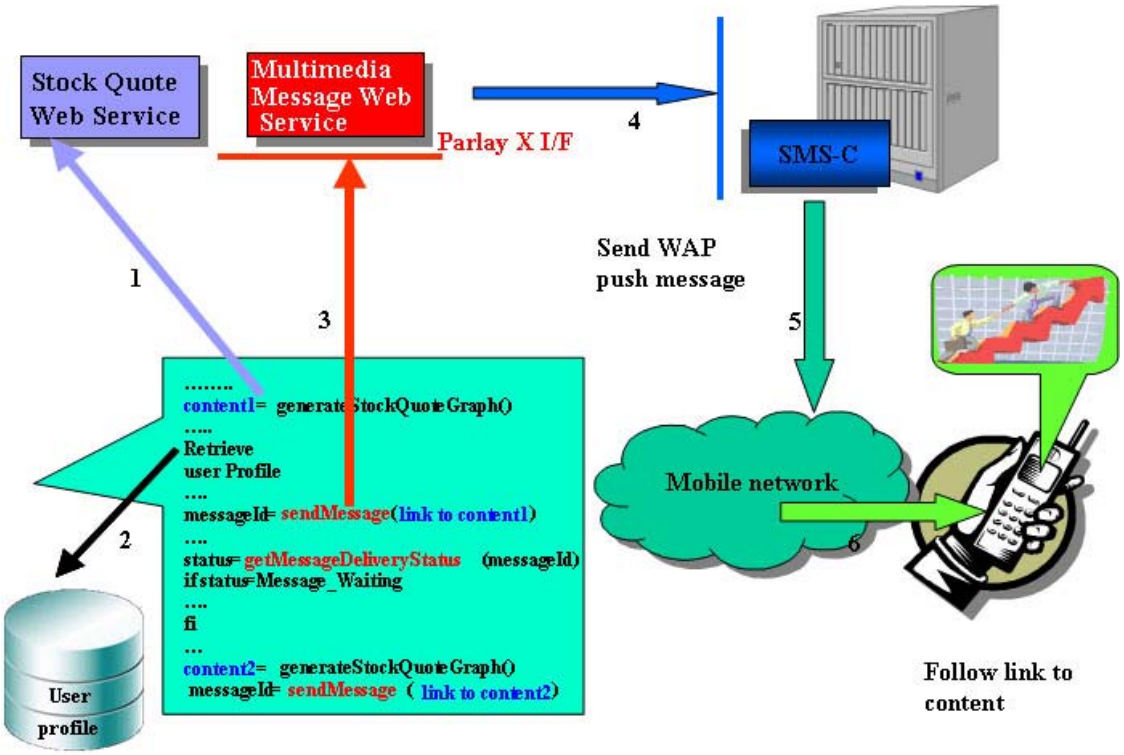

Figure 2: WAP Push scenario

5 Namespaces

The SendMessage interface uses the namespace:

http://www.csapi.org/wsdl/parlayx/multimedia_messaging/send/v3_1

The ReceiveMessage interface uses the namespace:

http://www.csapi.org/wsdl/parlayx/multimedia_messaging/receive/v3_1

The MessageNotification interface uses the namespace:

http://www.csapi.org/wsdl/parlayx/multimedia_messaging/notification/v3_1

The MessageNotificationManager interface uses the namespace:

http://www.csapi.org/wsdl/parlayx/multimedia_messaging/notification_manager/v3_2

The data types are defined in the namespace:

http://www.csapi.org/schema/parlayx/multimedia_messaging/v3_0

The "xsd" namespace is used in the present document to refer to the XML Schema data types defined in XML Schema [1]. The use of the name "xsd" is not semantically significant.

6 Sequence diagrams

6.1 Send picture

With the advent of picture capable phones, the exchange of photos to mobile phones is becoming more common place. This sequence diagram shows an application where a person can send a picture from an online photo album to a mobile phone.

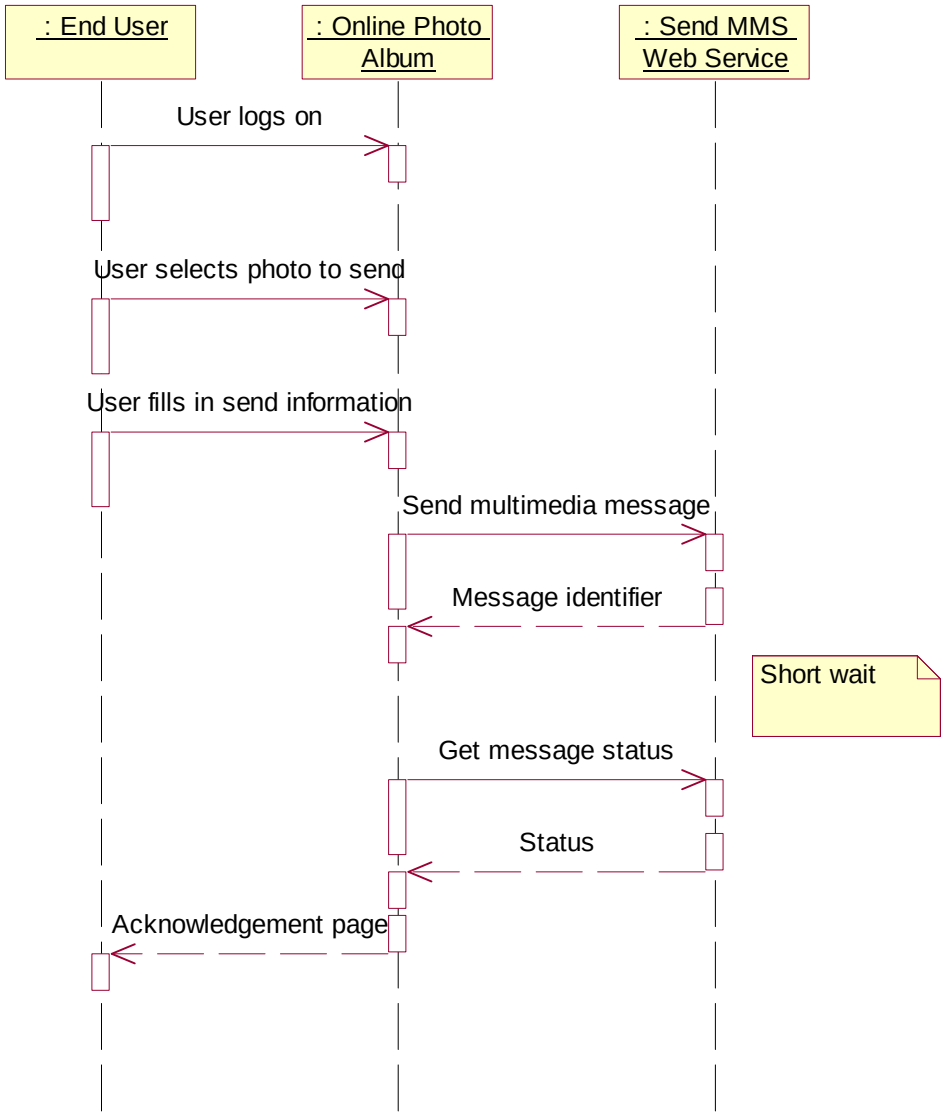

Figure 3

6.2 Send WAP Push message

For phones capable of receiving WAP Push messages, link to content can be sent using this example. The suggested MIME type for the attachment, as defined by the OMA, is **text/vnd.wap.sl** for sending HTTP links or WAP links to a mobile phone. This sequence diagram shows an application sending a link to a mobile phone, and the mobile phone using the link to retrieve the content.

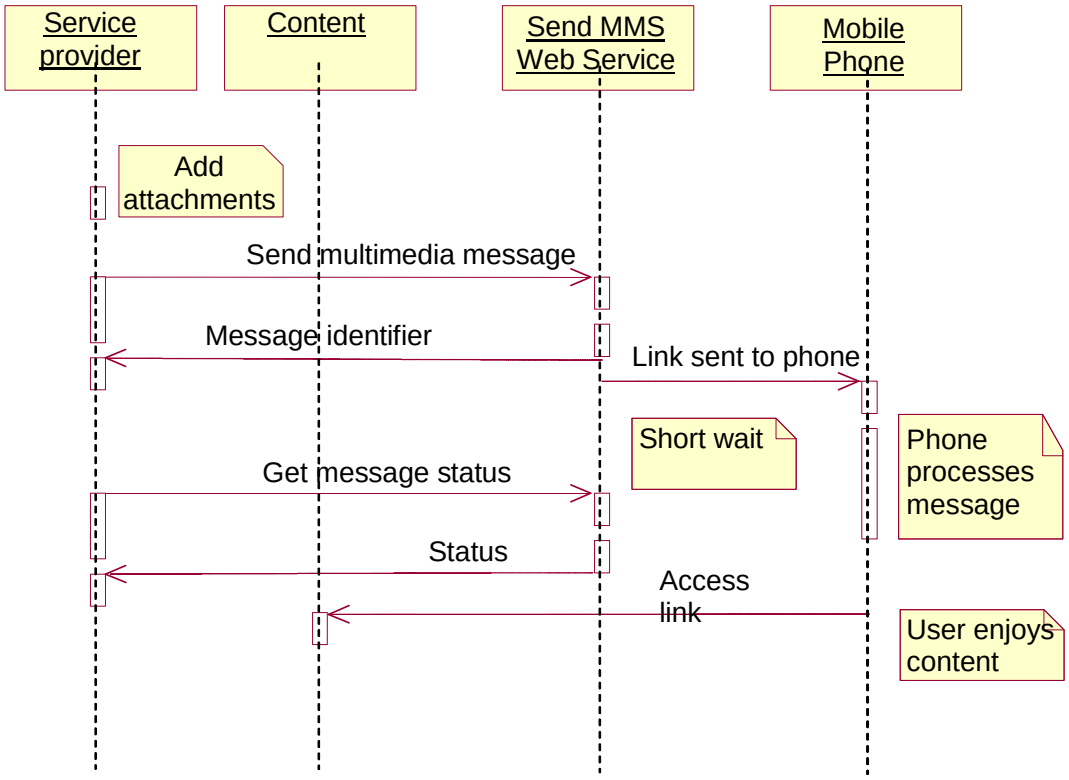

Figure 4

7 XML Schema data type definition

7.1 DeliveryStatus enumeration

List of delivery status values.

Enumeration value	Description
DeliveredToNetwork	Successful delivery to the network enabler responsible for distributing the multimedia message further in the network.
DeliveryUncertain	Delivery status unknown: e.g. because it was handed off to another network.
DeliveryImpossible	Unsuccessful delivery; the message could not be delivered before it expired.
MessageWaiting	The message is still queued for delivery. This is a temporary state, pending transition to one of the preceding states.
DeliveredToTerminal	Successful delivery to Terminal.
DeliveryNotificationNot Supported	Unable to provide delivery receipt notification. The **notifyMessageDeliveryReceipt** operation will provide "DeliveryNotificationNotSupported" to indicate that delivery receipt for the specified address in a **sendMessageRequest** message is not supported.

7.2 MessagePriority enumeration

List of delivery priority values.

Enumeration value	Description
Default	Default message priority
Low	Low message priority
Normal	Normal message priority
High	High message priority

7.3 DeliveryInformation structure

Delivery status information.

Element name	Element type	Optional	Description
address	xsd:anyURI	No	Address associated with the delivery status. The address field is coded as a URI.
deliveryStatus	DeliveryStatus	No	Indicates delivery status for the destination address.

7.4 MessageReference structure

Message information.

Element name	Element type	Optional	Description
messageIdentifier	xsd:string	Yes	If present, contains a reference to a message stored in the Parlay X gateway. If the message is pure text, this parameter is not present.
messageService ActivationNumber	xsd:string	No	Number associated with the invoked Message service, i.e. the destination address used by the terminal to send the message.
senderAddress	xsd:anyURI	No	Indicates message sender address.
subject	xsd:string	Yes	If present, indicates the subject of the received message. This parameter will not be used for SMS services.
priority	MessagePriority	No	The priority of the message: default is Normal.
message	xsd:string	Yes	If present, then the **messageIdentifier** is not present and this parameter contains the whole message. The type of the message is always pure ASCII text in this case. The message will not be stored in the Parlay X gateway.
dateTime	xsd:dateTime	Yes	Time when message was received by operator

7.5 MessageURI structure

Message location information.

Element name	Element type	Optional	Description
bodyText	xsd:string	Yes	Contains the message body if it is encoded as ASCII text.
fileReferences	xsd:anyURI [0..unbounded]	Yes	This is an array of URI references to all the attachments in the Multimedia message. These are URIs to different files, e.g. GIF pictures or pure text files.

7.5 ScheduledDeliveryStatus enumeration

List of scheduled multimedia mesage delivery status values.

Enumeration value	Description
Scheduled	The Message has been scheduled, the scheduled time has not started.
NotSent	Message could not be sent before end of scheduled time.
Sent	The Message has been sent within the scheduled time.
Cancelled	Message has been cancelled. Some messages may have been sent.
PartiallySent	Message is sent to some, but not to all the recipients.
StatusUnavailable	Unable to provide delivery information.

7.6 ScheduledDeliveryInformation structure

Scheduled delivery information.

Element name	Element type	Optional	Description
deliveryStatus	ScheduledDeliveryStatus	No	Indicates the delivery result for the destination address.
numberOf MessagesSent	xsd:int	Yes	If applicable, the number of messages already sent.

8 Web Service interface definition

8.1 Interface: SendMessage

Operations to send messages and check status on sent messages.

8.1.1 Operation: sendMessage

Request to send a Message to a set of destination addresses, returning a **requestIdentifier** to identify the message. The **requestIdentifier** can subsequently be used by the application to poll for the message status, i.e. using **getMessageDeliveryStatus** to see if the message has been delivered or not. The content is sent as one or more attachments as specified in SOAP Messages with Attachments [3].

addresses may include group URIs as defined in the Address List Management specification. If groups are not supported, a fault (POL0006) will be returned to the application.

Optionally the application can also indicate the sender address (**senderAddress**), i.e. the string that is displayed on the user's terminal as the originator of the message, the message **priority**, the message **subject**, the **charging** information and a **receiptRequest**. The **receiptRequest** which is a **SimpleReference** structure indicates the application endpoint, interface used for notification of delivery receipt and a correlator that uniquely identifies the sending request. By invoking this operation with the optional **receiptRequest** part the application requires to receive the notification of the status of the message delivery.

The optional **receiptRequest** message part is not used (or will be overridden) in case the **startDeliveryReceiptNotification** operation is used when the application requires to receive delivery receipt notifications. This is to avoid overlapping criteria.

If the notification mechanism is not supported by a network a fault (SVC0283) will be returned to the application and the message will not be sent to the addresses specified. Notification to the application is done by invoking the **notifyMessageDeliveryReceipt** operation at the endpoint specified in **receiptRequest**.

The correlator provided in the **receiptRequest** must be unique for this Web Service and application at the time the notification is initiated, otherwise a **ServiceException** (SVC0005) will be returned to the application.

8.1.1.1 Input message: sendMessageRequest

Part name	Part type	Optional	Description
addresses	xsd:anyURI [1..unbounded]	No	Destination addresses for the Message.
senderAddress	xsd:string	Yes	Message sender address. This parameter is not allowed for all 3rd party providers. Parlay X server needs to handle this according to a SLA for the specific application and its use can therefore result in a PolicyException.
subject	xsd:string	Yes	Message subject. If mapped to SMS this parameter will be used as the **senderAddress**, even if a separate **senderAddress** is provided.
priority	MessagePriority	Yes	Priority of the message. If not present, the network will assign a priority based on an operator policy.
charging	common:ChargingInformation	Yes	Charging to apply to this message.
receiptRequest	common:SimpleReference	Yes	It defines the application endpoint, interfaceName and correlator that will be used to notify the application when the message has been delivered to a terminal or if delivery is impossible. It is not used (or will be overridden) in case the **startDeliveryReceiptNotification** operation is used.

NOTE: The input message contains one or more attachments, with appropriate content as defined by SOAP Messages with Attachments [3].

8.1.1.2 Output message: sendMessageResponse

Part name	Part type	Optional	Description
result	xsd:string	No	It is a correlation identifier that is used in a **getMessageDeliveryStatus** operation invocation, i.e. to poll for the delivery status of all of the sent Messages.

8.1.1.3 Referenced faults

ServiceException from ES 202 504-1 [2]:

- SVC0001 - Service error.
- SVC0002 - Invalid input value.
- SVC0004 - No valid addresses.
- SVC0006 - Invalid group.
- SVC0283 - Delivery Receipt Notification not supported.

PolicyException from ES 202 504-1 [2]:

- POL0001 - Policy error.
- POL0006 - Groups not allowed.
- POL0007 - Nested groups not allowed.
- POL0008 - Charging not supported.

8.1.2 Operation: getMessageDeliveryStatus

This is a poll method used by the application to retrieve delivery status for each message sent as a result of a previous **sendMessage** message invocation. The **requestIdentifier** parameter identifies this previous message invocation.

This operation can be invoked multiple times by the application even if the status has reached a final value. However, after the status has reached a final value, status information will be available only for a limited period of time as defined by a service policy.

8.1.2.1 Input message: getMessageDeliveryStatusRequest

Part name	Part type	Optional	Description
requestIdentifier	xsd:string	No	Identifier related to the delivery status request.

8.1.2.2 Output message: getMessageDeliveryStatusResponse

Part name	Part type	Optional	Description
result	DeliveryInformation [0..unbounded]	Yes	It is an array of status of the messages that were previously sent. Each array element represents a sent message: i.e. its destination address and its delivery status.

8.1.2.3 Referenced faults

ServiceException from ES 202 504-1 [2]:

- SVC0001 - Service error.
- SVC0002 - Invalid input value.

PolicyException from ES 202 504-1 [2]:

- POL0001 - Policy error.
- POL0010 - Retention time interval expired.

8.1.3 Operation: scheduleMessage

Request to schedule sending a message to a set of destination addresses, returning a **requestIdentifier** to identify the message. The **requestIdentifier** can subsequently be used by the application to poll for the message status or cancel the scheduled message.

8.1.3.1 Input message: scheduleMessageRequest

Part name	Part type	Optional	Description
addresses	xsd:anyURI [1..unbounded]	No	Destination addresses for the message.
senderAddress	xsd:string	Yes	Message sender address. This parameter is not allowed for all 3rd party providers. Parlay X server needs to handle this according to a SLA for the specific application and its use can therefore result in a PolicyException.
subject	xsd:string	Yes	Message subject. If mapped to SMS this parameter will be used as the senderAddress, even if a separate senderAddress is provided.
priority	MessagePriority	Yes	Priority of the message. If not present, the network will assign a priority based on an operator policy.
charging	common:Charging Information	Yes	Charging to apply to this message.
startTime	xsd:dateTime	No	Specifies the time to start sending out the scheduled message.
stopTime	xsd:dateTime	No	Specifies the time to stop sending out the message. Any message not sent before **stopTime** will not be sent.

NOTE: The input message contains one or more attachments, with appropriate content as defined by SOAP Messages with Attachment**s [**3**].**

8.1.3.2 Output message: scheduleMessageResponse

Part name	Part type	Optional	Description
result	xsd:string	No	It is a correlation identifier

8.1.3.3 Referenced faults

ServiceException from ES 202 504-1 [2]:

- SVC0001 - Service error.
- SVC0002 - Invalid input value.
- SVC0004 - No valid addresses.
- SVC0006 - Invalid group.

PolicyException from ES 202 504-1 [2]:

- POL0001 - Policy error.
- POL0006 - Groups not allowed.
- POL0007 - Nested groups not allowed.
- POL0008 - Charging not supported.

8.1.4 Operation: cancelScheduledMessage

The application invokes the **cancelScheduledMessage** operation to cancel the previously scheduled message request identified by **requestIdentifier**. If the period scheduled for sending the message has started, some of the messages may have been sent.

8.1.4.1 Input message: cancelScheduledMessageRequest

Part name	Part type	Optional	Description
requestIdentifier	xsd:string	No	It identifies a specific message schedule request

8.1.4.2 Output message : cancelScheduledMessageResponse

Part name	Part type	Optional	Description
None			

8.1.4.3 Referenced faults

ServiceException from ES 202 504-1 [2]:

- SVC0001 - Service error.
- SVC0002 - Invalid input value.

PolicyException from ES 202 504-1 [2]:

- POL0001 - Policy error.

8.1.5 Operation: getScheduledMessageStatus

Gets the schedule and status of a scheduled message.

8.1.5.1 Input message: getScheduledMessageStatusRequest

Part name	Part type	Optional	Description
requestIdentifier	xsd:string	No	It identifies a specific message schedule request

8.1.5.2 Output message: getScheduledMessageStatusResponse

Part name	Part type	Optional	Description
result	ScheduledDeliveryInformation	No	Indicates the delivery result for the destination addresses and, if applicable, the number of messages already sent.

8.1.5.3 Referenced faults

ServiceException from ES 202 504-1 [2]:

- SVC0001 - Service error.
- SVC0002 - Invalid input value.

PolicyException from ES 202 504-1 [2]:

- POL0001 - Policy error.
- POL0010 - Retention time interval expired.

8.2 Interface: ReceiveMessage

Operations to retrieve messages that have been received.

8.2.1 Operation: getReceivedMessages

This method enables the application to poll for new messages received that fulfil the criteria identified by **registrationIdentifier**. The **priority** parameter may be used by the application to retrieve references to higher priority messages, e.g. if **Normal** is chosen, only references to high priority and normal priority messages are returned. If the priority parameter is omitted all message references are returned.

The operation returns a new list of received messages: i.e. only the received messages that the application has not retrieved by previous invocations of this operation. Moreover, each received message will be automatically removed from the server after an agreed time interval, as defined by a service policy.

8.2.1.1 Input message: getReceivedMessagesRequest

Part name	Part type	Optional	Description
registrationIdentifier	xsd:string	No	Identifies the provisioning step that enables the application to receive notification of Message reception according to specified criteria.
priority	MessagePriority	Yes	The priority of the messages to poll from the Parlay X gateway. All messages of the specified priority and higher will be retrieved. If not specified, all messages shall be returned, i.e. the same as specifying Low.

8.2.1.2 Output message: getReceivedMessagesResponse

Part name	Part type	Optional	Description
result	MessageReference [0..unbounded]	Yes	It contains an array of messages received according to the specified filter of **registrationIdentifier** and **priority**.

8.2.1.3 Referenced faults

ServiceException from ES 202 504-1 [2]:

- SVC0001 - Service error.
- SVC0002 - Invalid input value.

PolicyException from ES 202 504-1 [2]:

- POL0001 - Policy error.
- POL0010 - Retention time interval expired.

8.2.2 Operation: getMessageURIs

This method will read the different parts of the message, create local files in the Parlay Gateway and return URI references to them. The application can then simply read each file or just have them presented as links to the end-user. The URIs to the files will be active for as long as the message remains on the server: i.e. an agreed time interval as defined by a service policy.

8.2.2.1 Input message: getMessageURIsRequest

Part name	Part type	Optional	Description
messageRefIdentifier	xsd:string	No	The identity of the message to retrieve.

8.2.2.2 Output message: getMessageURIsResponse

Part name	Part type	Optional	Description
result	MessageURI	No	It contains the complete message, i.e. the textual part of the message, if such exists, and a list of file references for the message attachments, if any.

8.2.2.3 Referenced faults

ServiceException from ES 202 504-1 [2]:

- SVC0001 - Service error.
- SVC0002 - Invalid input value.

PolicyException from ES 202 504-1 [2]:

- POL0001 - Policy error.
- POL0010 - Retention time interval expired.

8.2.3 Operation: getMessage

This method will read the whole message. The data is returned as an attachment, as defined in SOAP Messages with Attachments [3], in the return message. Note that the received message is only available on the server for an agreed time interval following receipt, as defined by a service policy.

8.2.3.1 Input message: getMessageRequest

Part name	Part type	Optional	Description
messageRefIdentifier	xsd:string	No	The identity of the message

8.2.3.2 Output message: getMessageResponse

Part name	Part type	Optional	Description
None			

8.2.3.3 Referenced faults

ServiceException from ES 202 504-1 [2]:

- SVC0001 - Service error.
- SVC0002 - Invalid input value.

PolicyException from ES 202 504-1 [2]:

- POL0001 - Policy error.
- POL0010 - Retention time interval expired.

8.3 Interface: MessageNotification

MessageNotification is the application side notification interface to which notifications about multimedia messages are delivered.

8.3.1 Operation: notifyMessageReception

The notification to the application will occur only if the multimedia message fulfils the criteria specified when starting the multimedia message notification.

8.3.1.1 Input message: notifyMessageReceptionRequest

Part name	Part type	Optional	Description
correlator	xsd:string	No	Correlator provided in request to set up this notification.
message	MessageReference	No	This parameter contains all the information associated with the received message.

8.3.1.2 Output message: notifyMessageReceptionResponse

Part name	Part type	Optional	Description
None			

8.3.1.3 Referenced faults

None.

8.3.2 Operation: notifyMessageDeliveryReceipt

The **notifyMessageDeliveryReceipt** method must be implemented by a Web Service at the ***application side*** if it requires notification of message delivery receipt. It will be invoked by the Parlay X 3 server to notify the application when a message sent by an application has been delivered to the terminal of the recipient or if delivery is impossible. The notification will occur if and only if the status of the sent message is **DeliveredToTerminal** or **DeliveryImpossible** and the application has specified interest in notification using one of the following mutually exclusive mechanisms:

- when sending a message by specifying the optional **receiptRequest** part. The correlator returned corresponds to the identifier specified by the application in the **receiptRequest** part of the original **sendMessage** request.
- by invoking the **startDeliveryReceiptNotification** operation requesting to receive delivery receipt notifications. The correlator returned corresponds to the identifier specified by the application in the **reference** of the original **startDeliveryReceiptNotification** request

When a message is sent to multiple addresses, the server will send a notification for each terminal as and when a message is delivered to the terminal.

The following three different message delivery status will be returned in the **notifyMessageDeliveryReceiptRequest** message:

- **DeliveryImpossible**: unsuccessful delivery; the message could not be delivered before it expired.
- **DeliveredToTerminal**: when the message has been successfully delivered to the terminal.
- **DeliveredNotificationNotSupported**: if notification is supported by the network but it does not support delivery receipt for one or more addresses specified in the **sendMessageRequest** message. The service will send this status for those addresses.

8.3.2.1 Input message: notifyMessageDeliveryReceiptRequest

Part name	Part type	Optional	Description
correlator	xsd:string	No	This correlator was provided by the application in either the **startDeliveryReceiptNotificationRequest** message or the **sendMessageRequest** message
deliveryStatus	DeliveryInformation	No	It lists the variations on the delivery status of the message to a terminal. Possible values are: • DeliveryImpossible • DeliveredToTerminal • DeliveryNotificationNotSupported

8.3.2.2 Output message: notifyMessageDeliveryReceiptResponse

Part name	Part type	Optional	Description
None			

8.3.2.3 Referenced faults

None.

8.4 Interface: MessageNotificationManager

The multimedia message notification manager enables applications to set up and tear down notifications for multimedia messages, online.

8.4.1 Operation: startMessageNotification

Start notifications to the application for a given Message Service activation number and criteria.

The **messageServiceActivationNumber** is an Address Data item, e.g. a Shortcode, as defined in ES 202 504-1 [2].

The **correlator** provided in the **reference** must be unique for the application Web Service at the time the notification is initiated, otherwise a fault (SVC0005) will be returned to the application.

If specified, criteria will be used to filter messages that are to be delivered to an application. The use of criteria will allow different notification endpoints to receive notifications for the same **messageServiceActivationNumber**. If criteria are not provided, or is an empty string, then all messages for the **messageServiceActivationNumber** will be delivered to the application. If **criteria** values overlap then SVC0008 will be returned to the application and the notification will not be set up. The combination of **messageServiceActivationNumber** and **criteria** must be unique, so that a notification will be delivered to only one notification endpoint. If no match is found, the message will not be delivered to the application.

8.4.1.1 Input message: startMessageNotificationRequest

Part name	Part type	Optional	Description
reference	common: SimpleReference	No	Notification endpoint definition
messageServiceActivationNumber	xsd:anyURI	No	The destination address of the multimedia message
criteria	xsd:string	Yes	The text to match against to determine the application to receive the notification. This text is matched against the first word, as defined as the initial characters after discarding any leading Whitespace and ending with a Whitespace or end of the string. The matching shall be case-insensitive. If the subject of the multimedia message is present it shall be used as the string. If not, the string is defined as the first plain/text part of the content.

8.4.1.2 Output message: startMessageNotificationResponse

Part Name	Part Type	Optional	Description
None			

8.4.1.3 Referenced faults

ServiceException from ES 202 504-1 [2]:

- SVC0001 - Service error.
- SVC0002 - Invalid input value.
- SVC0005 - Duplicate correlator.
- SVC0008 - Overlapping Criteria.

PolicyException from ES 202 504-1 [2]:

- POL0001 - Policy error.

8.4.2 Operation: stopMessageNotification

The application may end a multimedia message notification using this operation.

8.4.2.1 Input message: stopMessageNotificationRequest

Part name	Part type	Optional	Description
correlator	xsd:string	No	Correlator of request to end

8.4.2.2 Output message: stopMessageNotificationResponse

Part Name	Part Type	Optional	Description
None			

8.4.2.3 Referenced faults

ServiceException from ES 202 504-1 [2]:

- SVC0001 - Service error.
- SVC0002 - Invalid input value.

PolicyException from ES 202 504-1 [2]:

- POL0001 - Policy error.

8.4.3 Operation: startDeliveryReceiptNotification

This operation starts notifications to the application for delivery receipts. The **reference** part will identify where to send the delivery receipts. The **notifyMessageDeliveryReceipt** operation (see clause 8.3.2) must be implemented by a Web Service at the application side if it requires notification of multimedia message delivery receipt. When the Service Provider supports the **startDeliveryReceiptNotification** operation, its use overrides the delivery receipting mechanism supported in the **SendMessage** interface (see clause 8.1.1: **sendMessage** operation).

8.4.3.1 Input message: startDeliveryReceiptNotificationRequest

Part name	Part type	Optional	Description
reference	common:SimpleReference	No	Notification endpoint definition
filterCriteria	xsd:string	No	Allows the service to filter flexibly. One example would be for the Service Provider to filter based on the first 4 digits in MSISDN. This however is implementation specific and will be left to the Service Provider.

8.4.3.2 Output message: startDeliveryReceiptNotificationResponse

Part Name	Part Type	Optional	Description
None			

8.4.3.3 Referenced faults

ServiceException from ES 202 504-1 [2]:

- SVC0001 - Service error.
- SVC0002 - Invalid input value.
- SVC0005 - Duplicate correlator.
- SVC0008 - Overlapping Criteria.
- SVC0283 - Delivery Receipt Notification not supported.

PolicyException from ES 202 504-1 [2]:

- POL0001 - Policy error.

8.4.4 Operation: stopDeliveryReceiptNotification

The application may end delivery receipt notification using this operation.

8.4.4.1 Input message: stopDeliveryReceiptNotificationRequest

Part name	Part type	Optional	Description
correlator	xsd:string	No	Correlator of request to end

8.4.4.2 Output message: stopDeliveryReceiptNotificationResponse

Part Name	Part Type	Optional	Description
None			

8.4.4.3 Referenced faults

ServiceException from ES 202 504-1 [2]:

- SVC0001 - Service error.
- SVC0002 - Invalid input value.

PolicyException from ES 202 504-1 [2]:

- POL0001 - Policy error.

9 Fault definitions

9.1 ServiceException

9.1.1 Void

The fault code SVC0230 is reserved and shall not be used.

9.1.2 SVC0283: Delivery Receipt Notification not supported

Name	Description
message Id	SVC0283
text	Delivery Receipt Notification not supported
variables	

10 Service policies

Service policies for this service.

Name	Type	Description
GroupSupport	xsd:boolean	Groups may be included with addresses
NestedGroupSupport	xsd:boolean	Nested groups are supported in group definitions
ChargingSupported	xsd:boolean	Charging supported for send message operation
StatusRetentionTime	common:TimeMetric	A time interval that begins after the status of a message delivery request has reached a final value. During this interval, the delivery status information remains available for retrieval by the application.
MessageRetentionTime	common:TimeMetric	A time interval that begins after the the receipt of a message. During this interval, the message remains available for retrieval by the application.

Annex A (normative): WSDL for Multimedia Messaging

The document/literal WSDL representation of this interface specification is compliant to ES 202 504-1 [2] and is contained in text files (contained in archive es_20250405v010101m0.zip) which accompany the present document.

Annex B (informative): Bibliography

ETSI TR 121 905: " Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Vocabulary for 3GPP Specifications (3GPP TR 21.905)".

History

Document history			
V1.1.1	February 2008	Membership Approval Procedure	MV 20080425: 2008-02-26 to 2008-04-25