ETSI TR 102 397-4-2 V1.1.1 (2005-12)

Technical Report

**Open Service Access (OSA);
Mapping of Parlay X Web Services to Parlay/OSA APIs;
Part 4: Short Messaging Mapping;
Sub-part 2: Mapping to Multi-Media Messaging**

Reference

DTR/TISPAN-01021-04-02-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

Contents

Intellectual Property Rights

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN).

The present document is part 4, sub-part 2, of a multi-part deliverable providing an informative mapping of Parlay X Web Services to the Parlay Open Service Access (OSA) APIs and, where applicable, to IMS, as identified below:

Part 1: "Common Mapping";

Part 2: "Third Party Call Mapping";

Part 3: "Call Notification Mapping";

Part 4: "Short Messaging Mapping";

Sub-part 1: "Mapping to User Interaction";

Sub-part 2: "Mapping to Multi-Media Messaging";

Part 5: "Multimedia Messaging Mapping";

Part 6: "Payment Mapping";

Part 7: "Account Management Mapping";

Part 8: "Terminal Status Mapping";

Part 9: "Terminal Location Mapping";

Part 10: "Call Handling Mapping";

Part 11: "Audio Call Mapping";

Part 12: "Multimedia Conference Mapping";

Part 13: "Address list Management Mapping";

Part 14: "Presence Mapping".

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

1 Scope

The Parlay X Web Services provide powerful yet simple, highly abstracted, imaginative, telecommunications functions that application developers and the IT community can both quickly comprehend and use to generate new, innovative applications.

The Open Service Access (OSA) specifications define an architecture that enables application developers to make use of network functionality through an open standardized interface, i.e. the Parlay/OSA APIs.

The present document is part 4, sub-part 2, of an informative mapping of Parlay X Web Services to Parlay/OSA APIs.

The present document specifies the mapping of the Parlay X Short Messaging Web Service to the Parlay/OSA Multi-Media Messaging Service Capability Feature (SCF).

2 References

For the purposes of this Technical Report (TR) the following references apply:

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

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

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

[3] ETSI TR 102 397-1: "Open Service Access (OSA); Mapping of Parlay X Web Services to Parlay/OSA APIs; Part 1: Common Mapping".

[4] ETSI TS 123 040: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Technical realization of Short Message Service (SMS) (3GPP TS 23.040)".

[5] 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 TR 102 397-1 [3] and the following apply:

Shortcode: Short telephone number, usually 4 to 6 digits long. This is represented by the 'tel:' URI defined in TR 102 397-1 [3].

Whitespace: See definition for CFWS as defined in RFC 2822 [5].

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TR 102 397-1 [3] and the following apply:

MMM	Multi-Media Messaging
SMS	Short Message Service
URI	Uniform Resource Identifier

4 Mapping description

The Short Messaging capability can be implemented with the Parlay/OSA Multi-Media Messaging SCF.

It is applicable to ETSI OSA 3.x, Parlay/OSA 5.x and 3GPP Release 6.x.

5 Sequence Diagrams

5.1 Send Short Message to One or More Addresses (Messaging Paradigm)

This describes where an application sends a short message to one or more addresses. The use case is the same whether the message is text, ringtone or a logo, however a different operation on the Parlay X **SendSms** interface is used for each. For the diagram below replace **sendSms** with **sendSmsLogo** or **sendSmsRingtone** as appropriate.

1. The application requests the sending of a short message to multiple addresses using the **sendSms** operation. If the contents of the **sendSmsRequest** message are invalid for any reason, the appropriate service or policy exception is thrown. Otherwise processing continues as described below.

2. The web service creates a Multi-Media Messaging interface object for this application request (single-shot, page mode); no source or destination address information is provided in the method invocation. If the method invocation fails for any reason, the appropriate service or policy exception is thrown. Otherwise processing continues as described below.

3. A **sendSmsResponse** message is returned to the application containing a unique identifier for this SMS delivery request.

4. The web service invokes the `sendMessageReq` method on the Multi-Media Messaging interface object to send the message to each individual destination address.

5. The application can invoke the **getSmsDeliveryStatus** operation at any time after it receives the **sendSmsResponse** message and use the unique identifier it received in this message to obtain the current delivery status for each individual destination address. At this stage, the status returned for each address is either **MessageWaiting** or, in the event of an error, **DeliveryImpossible**.

6. The web service processes an invocation of the `sendMessageRes` method indicating that the message has been successfully sent to the destination address(es). However it does not indicate that the message was delivered or read.

7. The application can invoke the **getSmsDeliveryStatus** operation. At this stage, the status returned for each individual destination address is one of the following:

   - **DeliveryImpossible,** in the event an error occurred.

   - **DeliveredToNetwork**, otherwise.

8. The web service processes one or more invocations of the `messageStatusReport` method, one for each destination address associated with the message, which contains the `terminal delivery related` status.

9. If the **receiptRequest** part of the associated, original **sendSmsRequest** message was present, and this capability is supported by the web service, then the web service invokes the **notifySmsDeliveryReceipt** operation to notify the application of the final status of the SMS delivery to an individual destination address.

10. The application can invoke the **getSmsDeliveryStatus** operation. At this stage, the status returned for an individual destination address depends on whether a `messageStatusReport` method has been invoked for that address. If the method has not been invoked, the delivery status is as described in step 7. Otherwise this method has been invoked and the delivery status is one of the following:

    - **DeliveredToTerminal**, if `deliveryReportType` parameter value = `P_MESSAGE_REPORT_DELIVERED.`
    - **DeliveryImpossible**, if `deliveryReportType` parameter value = `P_MESSAGE_REPORT_ NOT_DELIVERABLE.`
    - **DeliveryUncertain**, if `deliveryReportType` parameter value = `P_MESSAGE_REPORT_ DELIVERY_UNDEFINED.`

11. If the web service has not yet received all the requested `terminal delivery related` status reports, it may optionally invoke the `queryStatusReq` method to poll the network for this information.

12. The web service processes an invocation of the `queryStatusRes` method containing `terminal delivery related` status for all destination addresses associated with the message.

13. If the **receiptRequest** part of the associated, original **sendSmsRequest** message was present, and this capability is supported by the web service, then the web service invokes the **notifySmsDeliveryReceipt** operation to notify the application of the final status of the SMS delivery to an individual destination address. (However if the delivery status is unchanged from the status previously reported to the application, then the web service does not need to invoke this operation.)

14. The application can invoke the **getSmsDeliveryStatus** operation. At this stage, the status returned for all associated destination addresses reflects the results provided by the `queryStatusRes` method (step 12), i.e.:

    - **DeliveredToTerminal**, if `deliveryReportType` parameter value = `P_MESSAGE_REPORT_DELIVERED.`
    - **DeliveryImpossible**, if `deliveryReportType` parameter value = `P_MESSAGE_REPORT_ NOT_DELIVERABLE.`
    - **DeliveryUncertain**, if `deliveryReportType` parameter value = `P_MESSAGE_REPORT_ DELIVERY_UNDEFINED.`

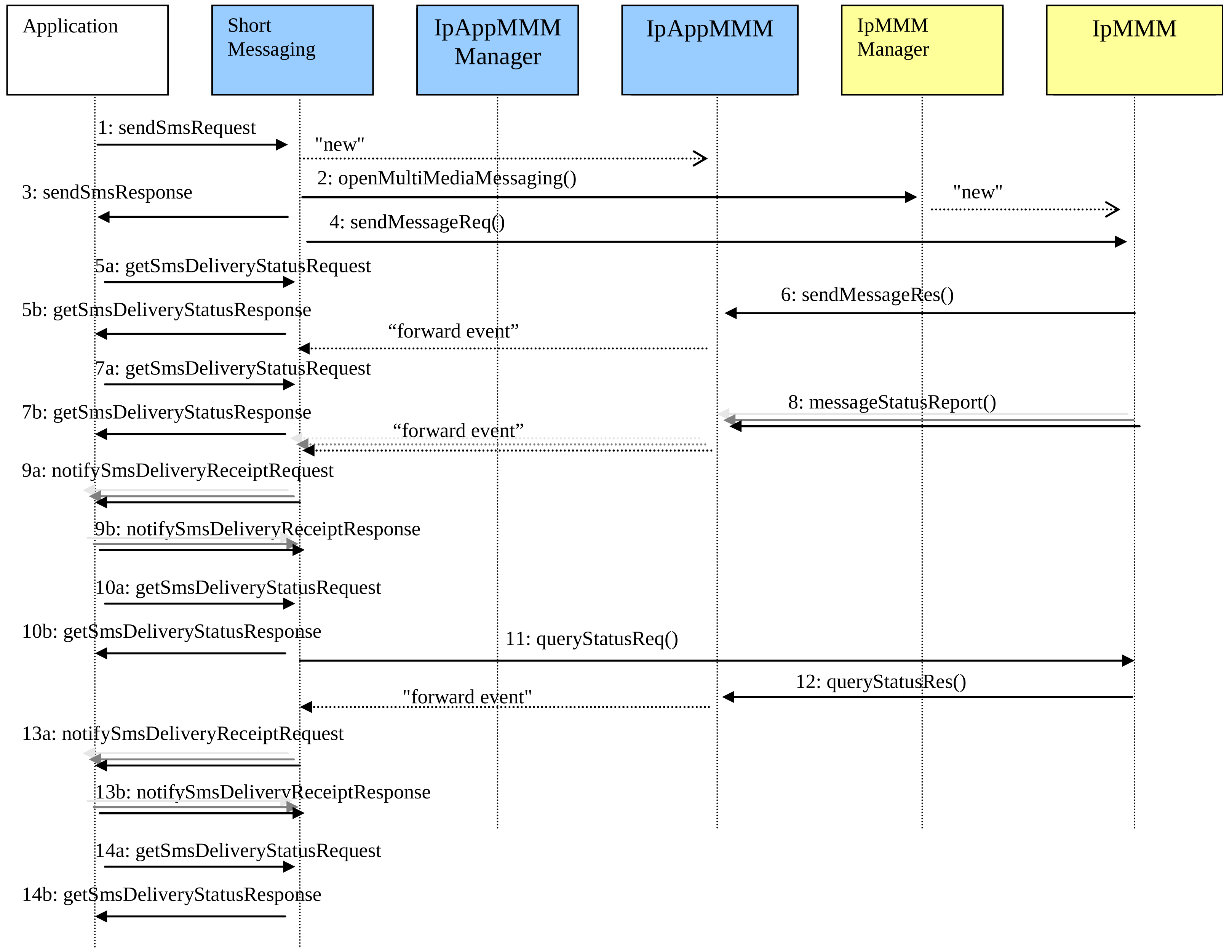
Application
Short Messaging
IpAppMMM Manager
IpAppMMM
IpMMM Manager
IpMMM
1: sendSmsRequest
"new"
2: openMultiMediaMessaging()
3: sendSmsResponse
"new"
4: sendMessageReq()
5a: getSmsDeliveryStatusRequest
6: sendMessageRes()
5b: getSmsDeliveryStatusResponse
“forward event”
7a: getSmsDeliveryStatusRequest
8: messageStatusReport()
7b: getSmsDeliveryStatusResponse
“forward event”
9a: notifySmsDeliveryReceiptRequest
9b: notifySmsDeliveryReceiptResponse
10a: getSmsDeliveryStatusRequest
10b: getSmsDeliveryStatusResponse
11: queryStatusReq()
12: queryStatusRes()
"forward event"
13a: notifySmsDeliveryReceiptRequest
13b: notifySmsDeliveryReceiptResponse
14a: getSmsDeliveryStatusRequest
14b: getSmsDeliveryStatusResponse

Figure 1

5.2 Notification of Short Message Reception and Retrieval (Messaging Paradigm)

1. The application registers for the reception of short messages by invoking **startSmsNotification**. The request includes event criteria consisting of a value for the short message destination address (the **smsServiceActivationNumber** part) and an optional text string for matching against the first word of the message body (the **criteria** part); also a URI for a Web Service implementing the **SmsNotification** interface on the client application side, and a correlation value for identifying this event registration request.

   - Note that the application may also register offline for the reception of short messages: i.e. without using the Parlay X interface and the **startSmsNotification** operation. The registration request should at a minimum specify the message destination address. The request may also specify a URI for a Web Service implementing the **SmsNotification** interface on the client application side and/or the optional text string criteria. The registration request is assigned a unique registration identifier.

2. A check is made within the web service to see if a notification for the given short message destination address is active. If no notification is active, then the Short Messaging web service requests that a notification be created by the MMM SCS; note that the optional text string criteria (for matching against the first word in the SMS body) is not sent to the MMM SCS. Otherwise a notification is already active and the request is not made.

3. The MMM SCS sends a `reportNotification` containing a set of one (or more) short message(s) and related message information, where the destination address of each message is the same: i.e. equivalent to the value specified in the event criteria (steps 1 and 2).

4. For each short message, the web service verifies the first word of the message body matches the value of an optional text string criteria associated with this destination address. If a message is verified, then the web service stores the message and notifies the application by invoking the **notifySmsReception** operation on the corresponding, previously provisioned, application web service. Otherwise, if a message cannot be verified, the web service discards it.

5. The application may invoke the **getReceivedSms** operation to request a list of received short messages matching a registration identifier associated with off-line provisioned notification criteria. The web service returns the list of any such messages and deletes them.

6. to 8.Repeat of steps 3 through 5. In step 8, only messages received by the web service during step 6, which match the registration identifier associated with off-line provisioned notification criteria, can be "bulk" retrieved by this **getReceivedSms** operation.

9. The application terminates an existing registration for the reception of short messages by invoking the **stopSMSNotification** operation. The request includes the same correlation value previously specified in an earlier **startSMSNotification** operation (step 1).

   - Note that the application may also deregister offline for the reception of short messages: i.e. without using the Parlay X interface and the **stopSmsNotification** operation. The deregistration request would specify the registration identifier associated with the original, offline registration operation (step 1).

10. A check is made within the web service to see if the registration identifier (correlation value) represents the last active notification for the corresponding destination address. If it is the last, then the web service requests that the notification be destroyed by the MMM SCS. Otherwise at least one other notification (i.e. associated with a different text string criteria value) remains active for this destination address and the request is not made.

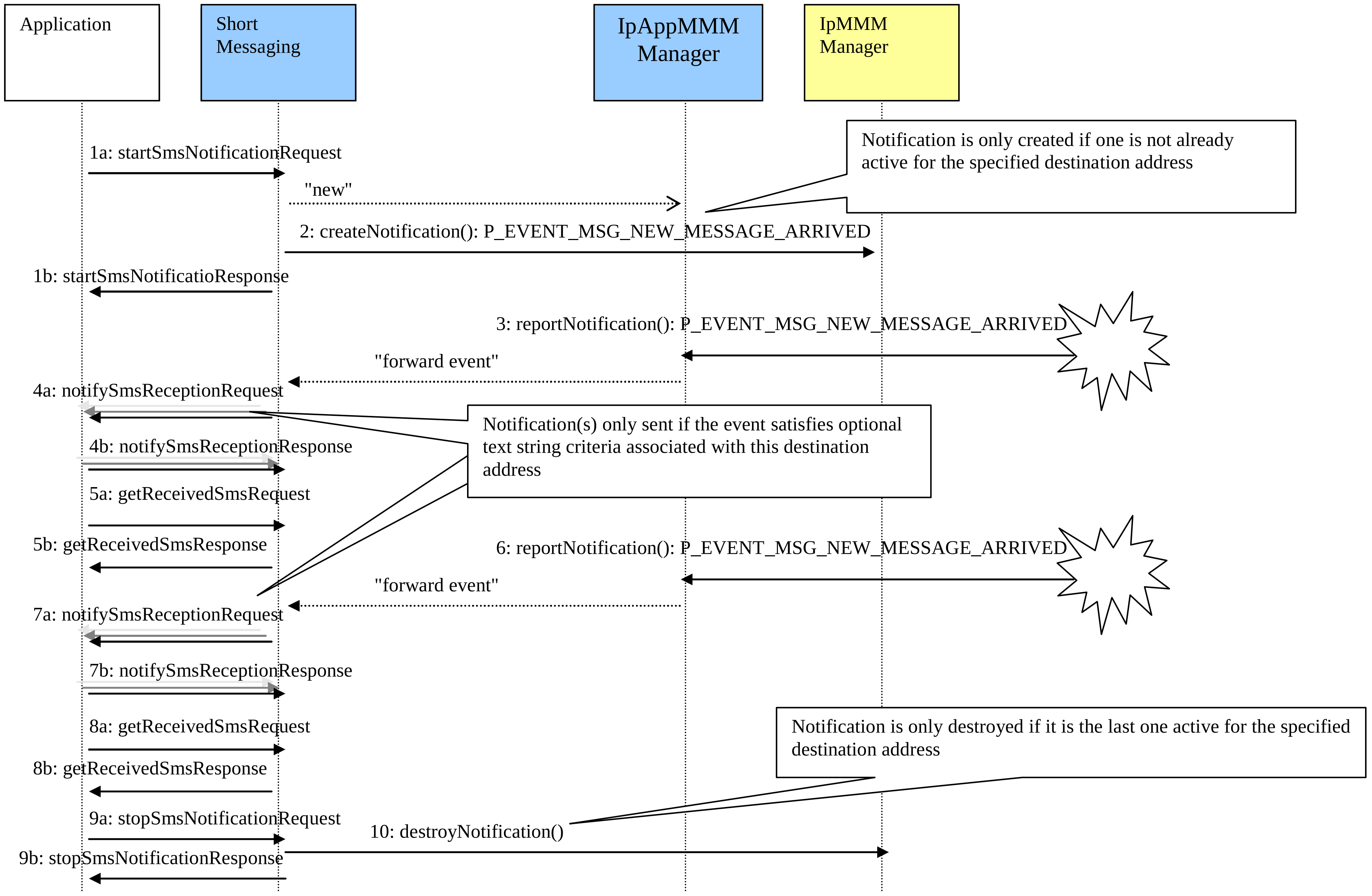
Application
Short Messaging
IpAppMMM Manager
IpMMM Manager
1a: startSmsNotificationRequest
"new"
Notification is only created if one is not already active for the specified destination address
2: createNotification(): P_EVENT_MSG_NEW_MESSAGE_ARRIVED
1b: startSmsNotificatioResponse
3: reportNotification(): P_EVENT_MSG_NEW_MESSAGE_ARRIVED
"forward event"
4a: notifySmsReceptionRequest
Notification(s) only sent if the event satisfies optional text string criteria associated with this destination address
4b: notifySmsReceptionResponse
5a: getReceivedSmsRequest
5b: getReceivedSmsResponse
6: reportNotification(): P_EVENT_MSG_NEW_MESSAGE_ARRIVED
"forward event"
7a: notifySmsReceptionRequest
7b: notifySmsReceptionResponse
8a: getReceivedSmsRequest
Notification is only destroyed if it is the last one active for the specified destination address
8b: getReceivedSmsResponse
9a: stopSmsNotificationRequest
10: destroyNotification()
9b: stopSmsNotificationResponse

Figure 2

5.3 Send Short Message to One or More Addresses (Mailbox Paradigm)

This describes where an application sends a short message to one or more addresses. The use case is the same whether the message is text, ringtone or a logo, however a different operation on the Parlay X **SendSms** interface is used for each. For the diagram below replace **sendSms** with **sendSmsLogo** or **sendSmsRingtone** as appropriate.

1. The application requests the sending of a short message to multiple addresses using the **sendSms** operation. If the contents of the **sendSmsRequest** message are invalid for any reason, the appropriate service or policy exception is thrown. Otherwise processing continues as described below.

2. If a mailbox for the requesting application is not already open, then the web service opens a Mailbox interface object. If the method invocation fails for any reason, the appropriate service or policy exception is thrown. Otherwise processing continues as described below.

3. A **sendSmsResponse** message is returned to the application containing a unique identifier for this SMS delivery request.

4. The web service invokes the `putMessageReq` method one or more times on the Mailbox interface object to place the message in an 'outbox' to be sent to each individual destination address. Note that, by invoking the method separately for each individual destination address, the web service receives a `messageId` for each destination that can be subsequently used to poll for delivery status on a per destination basis, e.g. in step 8.

5. The application can invoke the **getSmsDeliveryStatus** operation at any time after it receives the **sendSmsResponse** message and use the unique identifier it received in this message to obtain the current delivery status for each individual destination address. At this stage, the status returned for each address is either **MessageWaiting** or, in the event of an error, **DeliveryImpossible**.

6. The web service processes invocations of the `putMessageRes` method indicating that the message has been successfully sent to the destination address(es). However it does not indicate that the message was delivered or read.

7. The application can invoke the **getSmsDeliveryStatus** operation. At this stage, the status returned for each individual destination address is one of the following:

   - **DeliveryImpossible,** in the event an error occurred.

   - **DeliveredToNetwork**, otherwise.

8. The web service invokes the `getMessageInfoPropertiesReq` method one or more times on the Mailbox interface object, one for each destination address associated with the message, to poll for message delivery status.

9. The web service processes invocations of the `getMessageInfoPropertiesRes` method containing message delivery status.

10. If the **receiptRequest** part of the associated, original **sendSmsRequest** message was present, and this capability is supported by the web service, then the web service invokes the **notifySmsDeliveryReceipt** operation to notify the application of the final status of the SMS delivery to an individual destination address.

11. The application can invoke the **getSmsDeliveryStatus** operation. At this stage, the status returned for an individual destination address depends on whether a `getMessageInfoPropertiesRes` method has been invoked for that address. If the method has not been invoked, the delivery status is as described in step 7. Otherwise this method has been invoked and the delivery status is one of the following:

   - **DeliveredToTerminal**, if MessageStatus parameter value = `P_MMM_SENT_MSG_STATUS_DELIVERED, P_MMM_SENT_MSG_STATUS_READ or P_MMM_SENT_MSG_STATUS_DELETED_UNREAD.`

   - **DeliveryImpossible**, if MessageStatus parameter value = `P_MMM_SENT_MSG_STATUS_NOT_DELIVERABLE or P_MMM_SENT_MSG_STATUS_EXPIRED.`

- **DeliveryUncertain**, if `MessageStatus` parameter value = `P_MMM_SENT_MSG_STATUS_SENT.`

12. If the web service has not yet received a final message delivery status for all the destination addresses, it may optionally (re-)invoke the `getMessageInfoPropertiesReq` method one or more times on the Mailbox interface object to poll for message delivery status.

13. The web service processes invocations of the `getMessageInfoPropertiesRes` method containing message delivery status.

14. If the **receiptRequest** part of the associated, original **sendSmsRequest** message was present, and this capability is supported by the web service, then the web service invokes the **notifySmsDeliveryReceipt** operation to notify the application of the final status of the SMS delivery to an individual destination address. (However if the delivery status is unchanged from the status previously reported to the application in step 10, then the web service does not need to invoke this operation.)

15. The application can invoke the **getSmsDeliveryStatus** operation. At this stage, the status returned for all associated destination addresses reflects the results provided by the `getMessageInfoPropertiesRes` methods (steps 9 and 13), i.e.:

    - **DeliveredToTerminal**, if MessageStatus parameter value = `P_MMM_SENT_MSG_STATUS_DELIVERED, P_MMM_SENT_MSG_STATUS_READ or P_MMM_SENT_MSG_STATUS_DELETED_UNREAD.`

    - **DeliveryImpossible**, if MessageStatus parameter value = `P_MMM_SENT_MSG_STATUS_NOT_DELIVERABLE or P_MMM_SENT_MSG_STATUS_EXPIRED.`

    - **DeliveryUncertain**, if MessageStatus parameter value = `P_MMM_SENT_MSG_STATUS_SENT.`

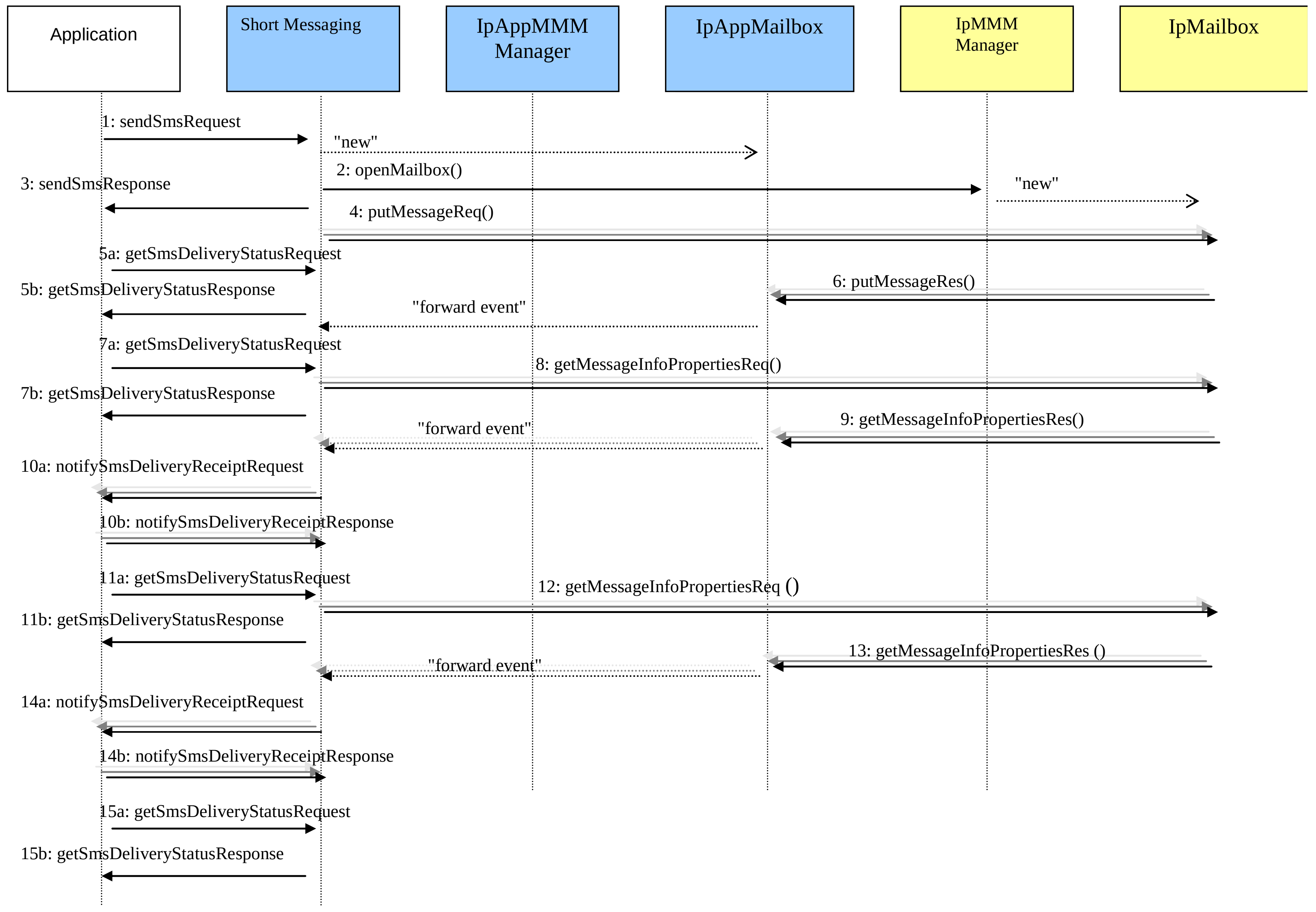
Application
Short Messaging
IpAppMMM Manager
IpAppMailbox
IpMMM Manager
IpMailbox
1: sendSmsRequest
"new"
2: openMailbox()
3: sendSmsResponse
"new"
4: putMessageReq()
5a: getSmsDeliveryStatusRequest
5b: getSmsDeliveryStatusResponse
6: putMessageRes()
"forward event"
7a: getSmsDeliveryStatusRequest
8: getMessageInfoPropertiesReq()
7b: getSmsDeliveryStatusResponse
9: getMessageInfoPropertiesRes()
"forward event"
10a: notifySmsDeliveryReceiptRequest
10b: notifySmsDeliveryReceiptResponse
11a: getSmsDeliveryStatusRequest
12: getMessageInfoPropertiesReq ()
11b: getSmsDeliveryStatusResponse
13: getMessageInfoPropertiesRes ()
"forward event"
14a: notifySmsDeliveryReceiptRequest
14b: notifySmsDeliveryReceiptResponse
15a: getSmsDeliveryStatusRequest
15b: getSmsDeliveryStatusResponse

Figure 3

5.4 Notification of Short Message Reception and Retrieval (Mailbox Paradigm)

1. The application registers for the reception of short messages by invoking **startSmsNotification**. The request includes event criteria consisting of a value for the short message destination address (the **smsServiceActivationNumber** part) and an optional text string for matching against the first word of the message body (the **criteria** part); also a URI for a Web Service implementing the **SmsNotification** interface on the client application side, and a correlation value for identifying this event registration request. The web service maps the short message destination address to a unique mailbox identifier supported by the messaging system.

   - Note that the application may also register offline for the reception of short messages: i.e. without using the Parlay X interface and the **startSmsNotification** operation. The registration request should at a minimum specify the message destination address, which maps to a unique mailbox identifier supported by the messaging system. The request may also specify a URI for a Web Service implementing the **SmsNotification** interface on the client application side and/or the optional text string criteria. The registration request is assigned a unique registration identifier.

2. A check is made within the web service to see if a notification for the given short message destination address is active. If no notification is active, then the Short Messaging web service requests that a notification be created by the MMM SCS; note that the optional text string criteria (for matching against the first word in the SMS body) is not sent to the MMM SCS. Otherwise a notification is already active and the request is not made.

3. The MMM SCS sends a `reportNotification` containing a set of one (or more) received message notification(s) and related message information, where the mailbox identifier of each message is the same: i.e. equivalent to the value specified in the event criteria (steps 1 and 2).

4. The web service opens a Mailbox interface object associated with the mailbox identifier reported in the event notification (step 3).

5. The web service invokes the `getMessageContentReq` method one or more times on the Mailbox interface object to request the retrieval of the entire body of each message reported in the event notification (step 3).

6. The web service processes invocations of the `getMessageContentRes` method containing the entire body of each message reported in the event notification (step 3).

7. For each retrieved message, the web service verifies the first word of the message body matches the value of an optional text string criteria associated with this destination address. If a message is verified, then the web service stores the message and notifies the application by invoking the **notifySmsReception** operation on the corresponding, previously provisioned, application web service. Otherwise, if a message cannot be verified, the web service discards it.

8. If the web service discards a retrieved message, it may also invoke either the `deleteMessageReq` or `moveMessageReq` method on the Mailbox interface object to clean-up the mailbox and folder. If the web service stores a retrieved message, it may also invoke the `setMessageInfoPropertiesReq` method on the Mailbox interface object to change the value of the `MessageStatus` element from `P_MMM_RECEIVED_MSG_STATUS_UNREAD` to `P_MMM_RECEIVED_MSG_STATUS_READ`.

9. The application may invoke the **getReceivedSms** operation to request a list of received short messages matching a registration identifier associated with off-line provisioned notification criteria. The web service returns the list of any such messages and deletes them.

10. Repeat of step 3 for the same message destination address and mailbox identifier.

11 to 14. Repeat of steps 5 through 8. Note that step 4 is not repeated as the mailbox interface object is already open.

15. The application may invoke the **getReceivedSms** operation to request a list of received short messages matching a registration identifier associated with off-line provisioned notification criteria. The web service returns the list of any such messages and deletes them. Note that only messages received by the web service since the previous invocation (step 9), can be "bulk" retrieved by this **getReceivedSms** operation.

16. The application terminates an existing registration for the reception of short messages by invoking the **stopSMSNotification** operation. The request includes the same correlation value previously specified in the earlier **startSMSNotification** operation (step 1).

    - Note that the application may also deregister offline for the reception of short messages: i.e. without using the Parlay X interface and the **stopSmsNotification** operation. The deregistration request would specify the registration identifier associated with the original, offline registration operation (step 1).

17. A check is made within the web service to see if the registration identifier (correlation value) represents the last active notification for the corresponding destination address. If it is the last, then the web service requests that the notification be destroyed by the MMM SCS. Otherwise at least one other notification (i.e. associated with a different text string criteria value) remains active for this destination address and the request is not made.

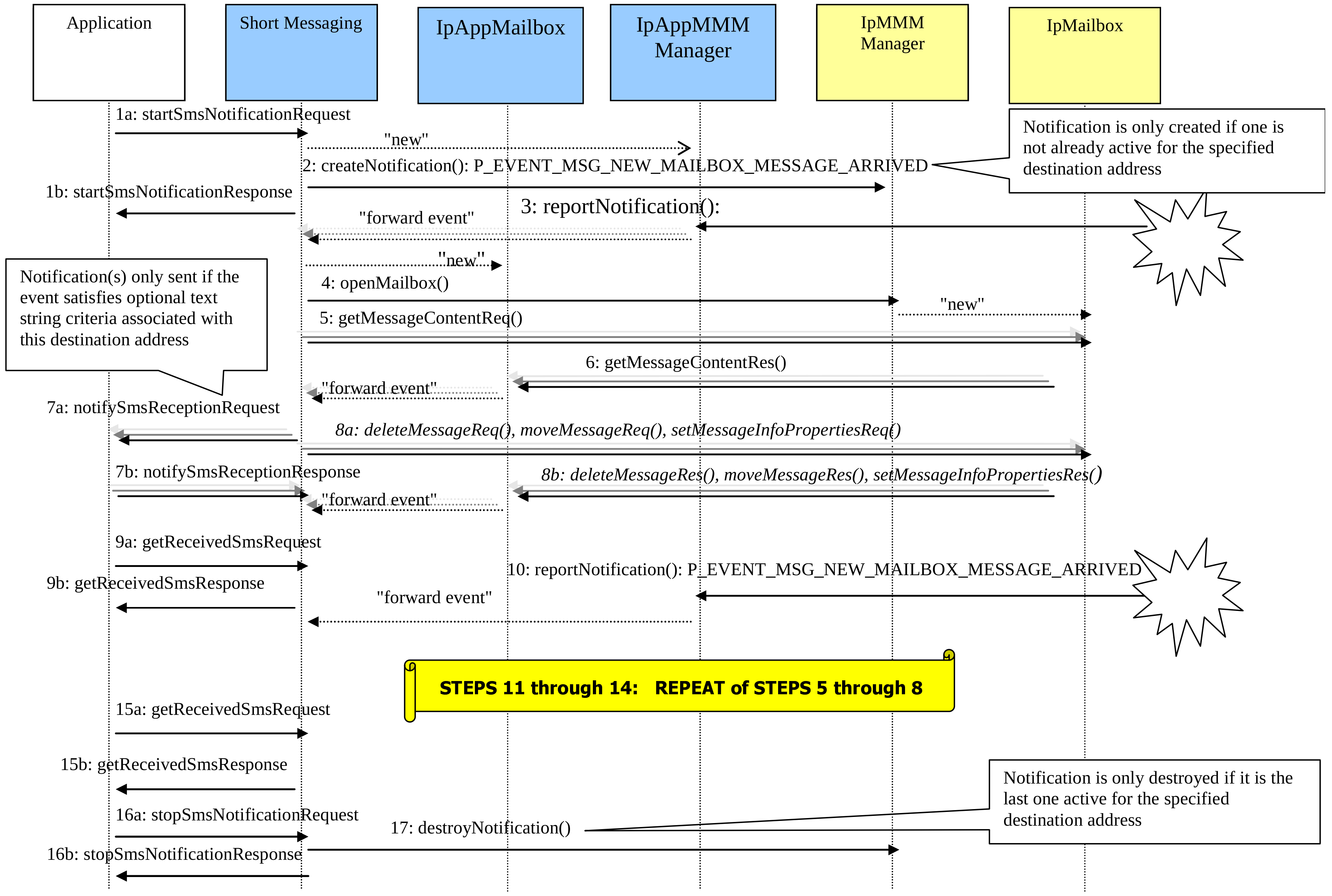
Application
Short Messaging
IpAppMailbox
IpAppMMM Manager
IpMMM Manager
IpMailbox
1a: startSmsNotificationRequest
"new"
2: createNotification(): P_EVENT_MSG_NEW_MAILBOX_MESSAGE_ARRIVED
Notification is only created if one is not already active for the specified destination address
1b: startSmsNotificationResponse
"forward event"
3: reportNotification():
"new"
4: openMailbox()
"new"
5: getMessageContentReq()
Notification(s) only sent if the event satisfies optional text string criteria associated with this destination address
6: getMessageContentRes()
"forward event"
7a: notifySmsReceptionRequest
8a: deleteMessageReq(), moveMessageReq(), setMessageInfoPropertiesReq()
7b: notifySmsReceptionResponse
8b: deleteMessageRes(), moveMessageRes(), setMessageInfoPropertiesRes()
"forward event"
9a: getReceivedSmsRequest
10: reportNotification(): P_EVENT_MSG_NEW_MAILBOX_MESSAGE_ARRIVED
9b: getReceivedSmsResponse
"forward event"
STEPS 11 through 14: REPEAT of STEPS 5 through 8
15a: getReceivedSmsRequest
15b: getReceivedSmsResponse
Notification is only destroyed if it is the last one active for the specified destination address
16a: stopSmsNotificationRequest
17: destroyNotification()
16b: stopSmsNotificationResponse

Figure 4

6 Detailed Mapping Information

6.1 Operations (Messaging Paradigm)

6.1.1 sendSms

The sequence diagram in clause 5.1 Send Short Message to One or More Addresses (Messaging Paradigm) (figure 1) illustrates the flow for the **sendSms** operation.

The **sendSms** operation is synchronous from the Parlay X client's point of view. It is mapped to the following Parlay/OSA methods:

- `IpMMMManager.openMMM.`
- `IpMMM.sendMessageReq.`

6.1.1.1 Mapping to `IpMMMManager.openMMM`

The `IpMMMManager.openMMM` method is invoked with the following parameters:

Name	Type	Comment
defaultDestination AddressList	TpTerminating AddressList	Not mapped. [Optional parameter]
defaultSource Address	TpAddress	Not mapped. [Optional parameter]
appMMM	IpAppMMMRef	Reference to callback (internal)

The result from `IpMMMManager.openMMM` is of type `TpMMMIdentifier` and identifies the MMM interface object upon which future methods are invoked: e.g. `IpMMM.sendMessageReq`. It is also correlated with the value of the **requestIdentifier** part returned to the application in the **sendSmsResponse** message.

Parlay exceptions thrown by `IpMMMManager.openMMM` are mapped to Parlay X exceptions as defined in clause 6.3 Exceptions.

6.1.1.2 Mapping to `IpMMM.sendMessageReq`

The `IpMMM.sendMessageReq` method is invoked with the following parameters:

Name	Type	Comment
sessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMMM`].
sourceAddress	TpAddress	The address used to represent the sender of the message. For alphanumeric SMS addresses - i.e. the optional **senderName** part of **sendSmsRequest** - the address plan `P_ADDRESS_PLAN_UNDEFINED` is used.
destination AddressList	TpTerminating AddressList	Specifies the addresses to which the SMS should be sent. It is constructed based on the URIs provided in the **addresses** part of **sendSmsRequest**, mapped as described in TR 102 397-1 [3]. Only the `ToAddressList` element of `TpTerminatingAddressList` is populated.
deliveryType	TpMessage DeliveryType	Set to the `P_MMM_SMS` value (GSM 7-bit character set only) or to the `P_MMM_SMS_BINARY` value (for Unicode SMS).
message Treatment	TpMessage TreatmentSet	Consists of the following elements: • a `DeliveryReport` element with value set to a value of "9", which represents a logical "OR" (and request for notification) of ONLY the following delivery states: `P_MESSAGE_REPORT_DELIVERY_UNDEFINED`; `P_MESSAGE_REPORT_DELIVERED`; and `P_MESSAGE_REPORT_NOT_DELIVERABLE.` • a `BillingID` element constructed from the **code** element of the optional **charging** part (if present); • a `DeliveryTime` element set to a value of `P_MMM_SEND_IMMEDIATE`; • a `ValidityTime` element set to a vendor-specific value.
message	TpOctetSet	The actual message that needs to be sent: i.e. the **message** part.
additionalHeaders	TpMessage HeaderFieldSet	Not mapped.

The result from `IpMMM.sendMessageReq` is of type `TpAssignmentID` and is used internally to correlate the callbacks. Specifically it is used to correlate with future invocations of the `IpMMM.queryStatusReq` method and of `IpAppMMM callback interface methods`.

Parlay exceptions thrown by `IpMMM.sendMessageReq` are not mapped to Parlay X exceptions. Instead they are reported to the application in a set of one or more **notifySmsDeliveryReceiptRequest** messages and/or in a **getSmsDeliveryStatusResponse** message, with the following part values:

- [**notifySmsDeliveryReceiptRequest** message only] **correlator** has the value of the **correlator** element of the **receiptRequest** part of the **sendSmsRequest** message;
- the **deliveryStatus.address** element has an address value contained in the `ToAddressList` element of the `terminatingAddressList` parameter of the `IpMMM.sendMessageReq` method, mapped as described in TR 102 397-1 [3];
- the **deliveryStatus.deliveryStatus** element has the value: **DeliveryImpossible**.

6.1.2 sendSmsLogo

The sequence diagram in clause 5.1 Send Short Message to One or More Addresses (Messaging Paradigm) (figure 1) illustrates the flow for the **sendSms** operation. The flow for the **sendSmsLogo** operation is identical.

The **sendSmsLogo** operation is synchronous from the Parlay X client's point of view. It is mapped to the same Parlay/OSA methods as the **sendSms** operation (clause 6.1.1 sendSms). The only difference is the mapping to the `deliveryType`, `message` and `additionalHeaders` parameters of the `IpMMM.sendMessageReq` method, as follows:

- The `deliveryType` parameter is set to the `P_MMM_SMS_BINARY` value (for Unicode SMS).
- The `message` parameter contains the actual logo that needs to be sent. It is constructed from the **image** part of the **sendSmsLogoRequest** message.

- There are two alternatives for the mapping of the **smsFormat** part of the **sendSmsLogoRequest** message:
  - the `Subject` element of the `additionalHeaders` parameter; or
  - the `ExtensionField` element of the `additionalHeaders` parameter, in an RFC 2822 [5] compliant format and a value of either '**SmsFormat:EMS**' or '**SmsFormat:SmartMessaging**'.

6.1.3 sendSmsRingtone

The sequence diagram in clause 5.1 Send Short Message to One or More Addresses (Messaging Paradigm) (figure 1) illustrates the flow for the **sendSms** operation. The flow for the **sendSmsRingtone** operation is identical.

The **sendSmsRingtone** operation is synchronous from the Parlay X client's point of view. It is mapped to the same Parlay/OSA methods as the **sendSms** operation (clause 6.1.1 sendSms). The only difference is the mapping to the `message` and `additionalHeaders` parameters of the `IpMMM.sendMessageReq` method, as follows:

- The `message` parameter contains the actual ringtone (in RTX text format) that needs to be sent. It is constructed from the **ringtone** part of the **sendSmsRingtoneRequest** message.
- There are two alternatives for the mapping of the **smsFormat** part of the **sendSmsRingtoneRequest** message:
  - the `Subject` element of the `additionalHeaders` parameter; or
  - the `ExtensionField` element of the `additionalHeaders` parameter, in an RFC 2822 [5] compliant format and a value of either '**SmsFormat:EMS**' or '**SmsFormat:SmartMessaging**'.

6.1.4 getSmsDeliveryStatus

The sequence diagram in clause 5.1 Send Short Message to One or More Addresses (Messaging Paradigm) (figure 1) illustrates the flow for the **getSmsDeliveryStatus** operation.

The **getSmsDeliveryStatus** operation is synchronous from the Parlay X client's point of view. It is mapped to/from the following Parlay/OSA methods:

- `IpAppMMM.sendMessageRes.`
- `IpAppMMM.sendMessageErr.`
- `IpAppMMM.messageStatusReport.`
- `IpMMM.queryStatusReq.`
- `IpAppMMM.queryStatusRes.`
- `IpAppMMM.queryStatusErr.`

`The delivery status provided to the Parlay X client will depend on the timing of the` **getSmsDeliveryStatus** operation invocation. If a message status report is received from the network as a result of an earlier **sendSmsXxx**-related operation, then the delivery status information provided in the `IpAppMMM.messageStatusReport` callback is mapped. If such a report has not been received, then the `IpMMM.queryStatusReq` method is invoked.

6.1.4.1 Mapping from `IpAppMMM.sendMessageRes`

The `IpAppMMM.sendMessageRes` method is invoked with the following parameters:

Name	Type	Comment
sessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMMM`].
assignmentID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMMM.sendMessageReq`].

In the absence of more recent delivery status information (i.e. as provided in an `IpAppMMM.messageStatusReport` or an `IpAppMMM.queryStatusRes` method), this method results in the assignment of the **DeliveredToNetwork** value to the **deliveryStatus** element of each **DeliveryInformation** parameter of the **deliveryStatus** part of a **getSmsDeliveryStatusResponse** message.

6.1.4.2 Mapping from `IpAppMMM.sendMessageErr`

The `IpAppMMM.sendMessageErr` method is invoked with the following parameters:

Name	Type	Comment
sessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMMM`]
assignmentID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMMM.sendMessageReq`]
error	TpMessaging Error	Maps to the **DeliveryImpossible** value of the **deliveryStatus** element of each **DeliveryInformation** parameter of the **deliveryStatus** part of a **getSmsDeliveryStatusResponse** message
errorDetails	TpString	Not mapped

6.1.4.3 Mapping from `IpAppMMM.messageStatusReport`

The `IpAppMMM.messageStatusReport` method is invoked with the following parameters:

Name	Type	Comment
sessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMMM`]
assignmentID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMMM.sendMessageReq`]
destinationAddress	TpAddress	Maps to the **address** element of one **DeliveryInformation** parameter of the **deliveryStatus** part of **getSmsDeliveryStatusResponse**
deliveryReportType	TpMessageDeliveryReportType	Maps to the **deliveryStatus** element of one **DeliveryInformation** parameter of the **deliveryStatus** part of **getSmsDeliveryStatusResponse**, as follows: • `P_MESSAGE_REPORT_ DELIVERY_UNDEFINED` maps to **DeliveryUncertain** • `P_MESSAGE_REPORT_DELIVERED` maps to **DeliveredToTerminal** • `P_MESSAGE_REPORT_NOT_DELIVERABLE` maps to **DeliveryImpossible**
deliveryReportInfo	TpString	Not mapped

6.1.4.4 Mapping to `IpMMM.queryStatusReq`

The `IpMMM.queryStatusReq` method is invoked with the following parameters:

Name	Type	Comment
sessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMMM`]
assignmentID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMMM.sendMessageReq`]

Parlay exceptions thrown by `IpMMM.queryStatusReq` are not mapped to Parlay X exceptions.

6.1.4.5 Mapping from `IpAppMMM.queryStatusRes`

The `IpAppMMM.queryStatusRes` method is invoked with the following parameters:

Name	Type	Comment
sessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMMM`]
assignmentID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMMM.sendMessageReq`]
result	TpQueryStatus ReportSet	This is a set of tuples where each tuple contains a `DestinationAddress` of the message, together with the `ReportedStatus` for that address. Each tuple maps to the **address** and **deliveryStatus** elements of one **DeliveryInformation** parameter of the **deliveryStatus** part of the **getSmsDeliveryStatusResponse** message. The mapping to the **deliveryStatus** element is as follows: • `P_MESSAGE_REPORT_ DELIVERY_UNDEFINED` maps to **DeliveryUncertain** • `P_MESSAGE_REPORT_DELIVERED` maps to **DeliveredToTerminal** • `P_MESSAGE_REPORT_NOT_DELIVERABLE` maps to **DeliveryImpossible** In the event that the messaging system provides additional delivery states to those requested in the `messageTreatment` parameter (clause 6.1.1.2 Mapping to `IpMMM.sendMessageReq`), the mapping to the **deliveryStatus** element is as follows: • `P_MESSAGE_REPORT_READ` and `P_MESSAGE_REPORT_DELETED_UNREAD` map to **DeliveredToTerminal** • `P_MESSAGE_REPORT_EXPIRED` maps to **DeliveryImpossible**

6.1.4.6 Mapping from `IpAppMMM.queryStatusErr`

The `IpAppMMM.queryStatusErr` method is invoked with the following parameters:

Name	Type	Comment
sessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMMM`].
assignmentID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMMM.sendMessageReq`].
error	TpMessaging Error	For each destination address with a current **deliveryStatus** value of **DeliveredToNetwork**, the **deliveryStatus** is updated to the **DeliveryUncertain** value. This updated value is reported to the application in a **DeliveryInformation** parameter of the **deliveryStatus** part of a **getSmsDeliveryStatusResponse** message.
errorDetails	TpString	Not mapped.

6.1.5 notifySmsDeliveryReceipt

The sequence diagram in clause 5.1 Send Short Message to One or More Addresses (Messaging Paradigm) (figure 1)illustrates the flow for the **notifySmsDeliveryReceipt** operation, which is mapped from the following Parlay/OSA methods:

- `Parlay exceptions thrown by IpMMM.sendMessageReq, as described in clause` 6.1.1.2 Mapping to `IpMMM.sendMessageReq.`
- `IpAppMMM.sendMessageErr.`
- `IpAppMMM.messageStatusReport.`
- `IpAppMMM.queryStatusRes.`

6.1.5.1 Mapping from `IpAppMMM.sendMessageErr`

The `IpAppMMM.sendMessageErr` method is invoked with the following parameters:

Name	Type	Comment
sessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMMM`].
assignmentID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMMM.sendMessageReq`].
error	TpMessaging Error	Results in the assignment of the following values to the **DeliveryInformation** parameter of the **deliveryStatus** part of a **notifySmsDeliveryReceiptRequest** message: • the **address** element contains the associated message destination address; • the **deliveryStatus** element has the value: **DeliveryImpossible.**
errorDetails	TpString	Not mapped.

In addition, the **correlator** part of the **notifySmsDeliveryReceiptRequest** message is assigned the value of the **correlator** element of the **receiptRequest** part of the **sendSmsXxxRequest** message to which it relates.

6.1.5.2 Mapping from `IpAppMMM.messageStatusReport`

The `IpAppMMM.messageStatusReport` method is invoked with the following parameters:

Name	Type	Comment
sessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMMM`].
assignmentID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMMM.sendMessageReq`].
destinationAddress	TpAddress	Maps to the **address** element of the **DeliveryInformation** parameter of the **deliveryStatus** part of **getSmsDeliveryStatusResponse**
deliveryReportType	TpMessageDelive ryReportType	Maps to the **deliveryStatus** element of the **DeliveryInformation** parameter of the **deliveryStatus** part of **notifySmsDeliveryReceiptRequest**, as follows: • `P_MESSAGE_REPORT_DELIVERED` maps to **DeliveredToTerminal** • `P_MESSAGE_REPORT_NOT_DELIVERABLE` maps to **DeliveryImpossible** Note that the `P_MESSAGE_REPORT_ DELIVERY_UNDEFINED` value does not represent a final delivery status, and does not result in the generation of a **notifySmsDeliveryReceiptRequest** message.
deliveryReportInfo	TpString	Not mapped.

In addition, the **correlator** part of the **notifySmsDeliveryReceiptRequest** message is assigned the value of the **correlator** element of the **receiptRequest** part of the **sendSmsXxxRequest** message to which it relates.

6.1.5.3 Mapping from `IpAppMMM.queryStatusRes`

The `IpAppMMM.queryStatusRes` method is invoked with the following parameters:

Name	Type	Comment
sessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMMM`]
assignmentID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMMM.sendMessageReq`]
result	TpQueryStatus ReportSet	This is a list of each `DestinationAddress` of the message, together with the `ReportedStatus` for that address. These elements map to the **address** and **deliveryStatus** elements of the **DeliveryInformation** parameter of the **deliveryStatus** part of **notifySmsDeliveryReceiptRequest**. The mapping to the **deliveryStatus** element is as follows: • `P_MESSAGE_REPORT_DELIVERED` maps to **DeliveredToTerminal** • `P_MESSAGE_REPORT_NOT_DELIVERABLE` maps to **DeliveryImpossible** In the event that the messaging system provides additional delivery states to those requested in the `messageTreatment` parameter (clause 6.1.1.2 Mapping to `IpMMM.sendMessageReq`), the mapping to the **deliveryStatus** element is as follows: • `P_MESSAGE_REPORT_READ` and `P_MESSAGE_REPORT_DELETED_UNREAD` map to **DeliveredToTerminal** • `P_MESSAGE_REPORT_EXPIRED` maps to **DeliveryImpossible** Note that the `P_MESSAGE_REPORT_ DELIVERY_UNDEFINED` value does not represent a final delivery status, and does not result in the generation of a **notifySmsDeliveryReceiptRequest** message

In addition, the **correlator** part of the **notifySmsDeliveryReceiptRequest** message is assigned the value of the **correlator** element of the **receiptRequest** part of the **sendSmsXxxRequest** message to which it relates.

6.1.6 startSmsNotification

The sequence diagram in clause 5.2 Notification of Short Message Reception and Retrieval (Messaging Paradigm) (figure 2) illustrates the flow for the **startSmsNotification** operation, which is mapped to the Parlay/OSA method: `IpMMMManager.createNotification`, provided there is no existing notification already established for the destination address contained in the **smsServiceActivationNumber** part.

6.1.6.1 Mapping to IpMMMManager.createNotification

The `IpMMMManager.createNotification` is invoked with the following parameters:

Name	Type	Comment
appMMM Manager	IpAppMMM ManagerRef	Not mapped. Reference to callback (internal).
eventCriteria	TpMessaging EventCriteriaSet	Contains a single element specifying the event notification criteria, for the messaging event: `P_EVENT_MSG_NEW_MESSAGE_ARRIVED`. The criteria consist of 3 fields: • The `SourceAddress` is not mapped. It is set to be valid for all senders. • The `DestinationAddress` is constructed based on the URI provided in the **smsServiceActivationNumber** part of the **startSmsNotificationRequest** message, mapped as described in TR 102 397-1 [3]. • The `CreateMultiMediaMessagingSession` element is not mapped. It is set to a value of 'FALSE': i.e. the SCF will not create a MMM session object when a new message arrives.

The result from `IpMMMManager.createNotification` is of type `TpAssignmentID` and is used internally to correlate the callbacks. Specifically it is correlated with the value of the **reference** part received from the application in the **startSmsNotificationRequest** message and the **correlator** part returned to the application in the **notifySmsReceptionRequest** message.

Note that the **reference** part and the optional **criteria** part of a **startSmsNotificationRequest** message are not mapped to `IpMMMManager.createNotification`. Instead the web service uses all the text string criteria values associated with a specific destination address to parse any event reported for that address by the `IpAppMMMManager.reportNotification method. The web service determines whether the event is valid - i.e. there is a match with a` text string criteria value. If valid, the web service stores the message and `selects the previously provisioned application callback web service to receive the` **notifySmsReceptionRequest** `message. If invalid, the web service discards the event notification.`

Parlay exceptions thrown by `IpMMMManager.createNotification` are mapped to Parlay X exceptions as defined in clause 6.3 Exceptions.

6.1.7 notifySmsReception

The **notifySmsReception** operation is mapped from the following Parlay/OSA methods:

- `IpAppMMMManager.reportNotification`, as illustrated in the sequence diagram in clause 5.2 Notification of Short Message Reception and Retrieval (Messaging Paradigm) (figure 2).
- `IpAppMMM.messageReceived`, which contains a message received for a remote party within the context of the conversation or session currently active. The message may be, but is not necessarily in reply to a message sent by the application using the `IpMMM.sendMessageReq` method (clause 6.1.1.2 Mapping to `IpMMM.sendMessageReq`). Note that the reference information for the application web service, upon which the **notifySmsReception** operation is invoked, must be provisioned offline, since online provisioning using the **SmsNotificationManager** interface is only applicable for messages which are received outside the context of the conversation or session.

6.1.7.1 Mapping from IpAppMMMManager.reportNotification

The `IpAppMMMManager.reportNotification` method is invoked with the following parameters:

Name	Type	Comment
assignmentID	TpAssignmentID	Not mapped. [The value provide in the result from `IpMMMManager.createNotification`].
eventInfo	TpMessaging EventInfoSet	Contains a set of SMS messages with the same destination address and an event type = `EventNewMessageArrived`. The mapping of each message (type `TpNewMessageArrivedInfo`) to the **message** part of a **notifySmsReceptionRequest** message is described in clause 6.1.7.2 Mapping from `TpNewMessageArrivedInfo` .

The result from `IpAppMMMManager.reportNotification` is of type `IpAppMultiMediaMessagingRef`. It is set to null.

6.1.7.2 Mapping from `TpNewMessageArrivedInfo`

The mapping from `TpNewMessageArrivedInfo` to the **message** part of a **notifySmsReceptionRequest** message is as follows:

Name	Type	Comment
SourceAddress	TpAddress	Maps to the **senderAddress** element of the **message** part. The data type mapping from `TpAddress` to **xsd:anyURI** is described in TR 102 397-1 [3].
DestinationAddress Set	TpAddressSet	Consists of a single destination address element, which maps to the **smsServiceActivationNumber** element of the **message** part. The data type mapping from `TpAddress` to **xsd:anyURI** is described in TR 102 397-1 [3].
Message	TpOctetSet	Maps to the **message** element of the **message** part.
Headers	TpMessage HeaderFieldSet	Not mapped. [Contains header information which could be duplicated in the `Message` element, depending on its format].
MultiMedia MessagingIdentifier	TpMultiMedia MessagingIdentifier	Not applicable. This parameter is null, reflecting the criteria value included in the `IpMMMManager.createNotification` invocation.

Note that this mapping occurs if there is at least one active notification established for the value of the `eventInfo.DestinationAddress(Set)` element, an associated `application callback web service,` and one of the following conditions is satisfied:

- There is only one active notification that was defined without the optional text string criteria value.
- There is one active notification that was defined with the optional text string criteria value and that value matches the first word in the value of the `eventInfo.Message` element.
    - Note that the 'first word' in the message is defined as the initial characters after discarding any leading Whitespace and ending with a Whitespace or end of message. The matching shall be case-insensitive.

6.1.7.3 Mapping from IpAppMMM.messageReceived

The `IpAppMMM.messageReceived` method is invoked with the following parameters:

Name	Type	Comment
sessionID	TpSessionID	Not mapped. [The value provide in the result from `IpMMMManager.openMMM – clause 6.1.1.1` Mapping to `IpMMMManager.openMMM`]
message	TpOctetSet	Maps to the **message** element of the **message** part.
headers	TpMessage HeaderFieldSet	The `Sender` set element maps to the **senderAddress** element of the **message** part of the **notifySmsReceptionRequest** message. The data type mapping from `TpAddress` to **xsd:anyURI** is described in TR 102 397-1 [3].

The **senderName** part of the original **sendSmsXxxRequest** message associated with this multimedia session, which was established as described in clause 6.1.1.1 Mapping to `IpMMMManager.openMMM`, is mapped to the **smsServiceActivationNumber** element of the **message** part of the **notifySmsReceptionRequest** message.

As previously noted, the endpoint definition of the application web service to which the **notifySmsReceptionRequest** message is sent, including the value of the **correlator** part, is provisioned offline.

6.1.8 getReceivedSms

The sequence diagram in clause 5.2 Notification of Short Message Reception and Retrieval (Messaging Paradigm) (figure 2) illustrates the flow for the **getReceivedSms** operation. It is not explicitly mapped to any Parlay/OSA method. Instead, the **getReceivedSms** operation is a bulk retrieval capability for previously received short messages matching criteria defined in an off-line provisioning step. This retrieval operation includes matching messages previously and individually delivered to the application via the **notifySmsReception** operation.

6.1.9 stopSMSNotification

The sequence diagram in clause 5.2 Notification of Short Message Reception and Retrieval (Messaging Paradigm) (figure 2) illustrates the flow for the **stopSmsNotification** operation, which is mapped to the Parlay/OSA method: `IpMMMManager.destroyNotification`, provided that the referenced notification is the last active notification for the associated destination address. Otherwise at least one other notification (i.e. associated with a different text string criteria value) remains active for this destination address and the mapping is not performed.

6.1.9.1 Mapping to IpMMMManager.destroyNotification

The `IpMMMManager.destroyNotification` is invoked with the following parameters:

Name	Type	Comment
assignmentID	TpAssignmentID	Not mapped. (The value provide in the result from `IpMMMManager.createNotification and correlated with the` value of the **reference** part received from the application in the original **startSmsNotificationRequest** message and the value of the **correlator** part received from the application in the **stopSmsNotificationRequest** message).

Parlay exceptions thrown by `IpMMMManager.destroyNotification` are mapped to Parlay X exceptions as defined in clause 6.3 Exceptions.

6.2 Operations (Mailbox Paradigm)

6.2.1 sendSms

The sequence diagram in clause 5.3 Send Short Message to One or More Addresses (Mailbox Paradigm) (figure 3) illustrates the flow for the **sendSms** operation.

The **sendSms** operation is synchronous from the Parlay X client's point of view. It is mapped to the following Parlay/OSA methods:

- `IpMMMManager.openMailbox, if not already opened for the application.`
- `IpMMM.putMessageReq.`

6.2.1.1 Mapping to `IpMMMManager.openMailbox`

The `IpMMMManager.openMailbox` method is invoked with the following parameters:

Name	Type	Comment
mailboxID	TpString	Not mapped. [Specifies the identity of the application's mailbox in the messaging system].
authenticationInfo	TpString	Not mapped. [Authentication information needed to open the application's mailbox, such as a key or password].
appMailbox	IpAppMailboxRef	Reference to callback (internal).

The result from `IpMMMManager.openMailbox` is of type `TpMailboxIdentifier` and identifies the Mailbox interface object upon which future methods are invoked: e.g. `IpMailbox.putMessageReq`. It is also correlated with the value of the **requestIdentifier** part returned to the application in the **sendSmsResponse** message

Parlay exceptions thrown by `IpMMMManager.openMailbox` are mapped to Parlay X exceptions as defined in clause 6.3 Exceptions.

6.2.1.2 Mapping to `IpMailbox.putMessageReq`

The `IpMailbox.putMessageReq` method is invoked with the following parameters:

Name	Type	Comment
mailboxSessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMailbox`].
folderID	TpString	In order to send a message from the mailbox, the web service places the message in a designated folder, from which it will be sent. The folder to use is indicated by the service property P_PUT_MESSAGE_FOLDER_TO_SEND.
message	TpOctetSet	The actual message that needs to be sent. The message and the headers are transferred to the Messaging service. The message will be taken as is. No checking is done on the message. The web service constructs the content of this parameter from the parts of the **sendSmsRequest** message by including the following information: • the 'To:' header field contains a single destination address, derived from the **addresses** part; • the 'From:' header field contains an individual destination address, derived from the **senderName**; • the message 'body' field contains the message text, derived from the **message** part. Note that the optional **charging** part is not mapped.

The result from `IpMailbox.putMessageReq` is of type `TpAssignmentID` and is used internally to correlate the callback invocation of the `IpAppMailbox.getMessageRes/Err` method.

Parlay exceptions thrown by `IpMailbox.putMessageReq` are not mapped to Parlay X exceptions. Instead they are reported to the application in a **notifySmsDeliveryReceiptRequest** message and/or in a **getSmsDeliveryStatusResponse** message, with the following part values:

- [**notifySmsDeliveryReceiptRequest** message only] **correlator** has the value of the **correlator** element of the **receiptRequest** part of the **sendSmsRequest** message;
- the **deliveryStatus.address** element contains the associated message destination address, originally derived from the **addresses** part;
- the **deliveryStatus.deliveryStatus** element has the value: **DeliveryImpossible**.

6.2.2 sendSmsLogo

The sequence diagram in clause 5.3 Send Short Message to One or More Addresses (Mailbox Paradigm) (figure 3) illustrates the flow for the **sendSms** operation. The flow for the **sendSmsLogo** operation is identical.

The **sendSmsLogo** operation is synchronous from the Parlay X client's point of view. It is mapped to the same Parlay/OSA methods as the **sendSms** operation (clause 6.2.1 sendSms). The only difference is in the mapping to the `message` parameter of the `IpMailbox.putMessageReq` method, as follows:

- The **image** part of the **sendSmsLogoRequest** message, which contains the actual logo that needs to be sent, should be mapped to the 'body' field of the `message` parameter.
- The **smsFormat** part of the **sendSmsLogoRequest** message can be mapped to a 'subject' or 'extension' header field of the `message` parameter, e.g. containing a value of either '**SmsFormat:EMS**' or '**SmsFormat:SmartMessaging**'.

6.2.3 sendSmsRingtone

The sequence diagram in clause 5.3 Send Short Message to One or More Addresses (Mailbox Paradigm) (figure 3) illustrates the flow for the **sendSms** operation. The flow for the **sendSmsRingtone** operation is identical.

The **sendSmsRingtone** operation is synchronous from the Parlay X client's point of view. It is mapped to the same Parlay/OSA methods as the **sendSms** operation (clause 6.2.1 sendSms). The only difference is the mapping to the `message` parameter of the `IpMailbox.putMessageReq` method, as follows:

- The **ringtone** part of the **sendSmsRingtoneRequest** message, which contains the actual ringtone (in RTX text format) that needs to be sent, should be mapped to the 'body' field of the `message` parameter.
- The **smsFormat** part of the **sendSmsRingtoneRequest** message can be mapped to a 'subject' or 'extension' header field of the `message` parameter, e.g. containing a value of either '**SmsFormat:EMS**' or '**SmsFormat:SmartMessaging**'.

6.2.4 getSmsDeliveryStatus

The sequence diagram in clause 5.3 Send Short Message to One or More Addresses (Mailbox Paradigm) (figure 3) illustrates the flow for the **getSmsDeliveryStatus** operation.

The **getSmsDeliveryStatus** operation is synchronous from the Parlay X client's point of view. It is mapped to/from the following Parlay/OSA methods:

- `IpAppMailbox.putMessageRes.`
- `IpAppMailbox.putMessageErr.`
- `IpMailbox.`getMessageInfoPropertiesReq.
- `IpAppMailbox.`getMessageInfoPropertiesRes.
- `IpAppMailbox.`getMessageInfoPropertiesErr.

The delivery status provided to the Parlay X client will depend on the timing of the **getSmsDeliveryStatus** operation invocation. If the delivery status for some destination addresses is known, as a result of earlier invocations of the `IpMailbox.getMessageInfoPropertiesReq` method, then the delivery status information provided in the `IpAppMailbox.getMessageInfoPropertiesRes` callback methods is mapped. If such a report has not been received for some destination addresses, then the `IpMailbox.getMessageInfoPropertiesReq` method is invoked for each of those destination addresses.

6.2.4.1 Mapping from `IpAppMailbox.putMessageRes`

The `IpAppMailbox.putMessageRes` method is invoked with the following parameters:

Name	Type	Comment
mailboxSessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMailbox`].
requestID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMailbox.putMessageReq`]
messageID	TpString	Not mapped. [The new ID of the message which has been placed in the folder, from which it will be sent, as requested].

In the absence of more recent delivery status information (i.e. as provided in an `IpAppMailbox.getMessageInfoPropertiesRes` method), this method results in the assignment of the following values to one **DeliveryInformation** parameter of the **deliveryStatus** part of a **getSmsDeliveryStatusResponse** message:

- the **address** element contains the associated message destination address;
- the **deliveryStatus** element has the value: **DeliveredToNetwork.**

6.2.4.2 Mapping from `IpAppMailbox.putMessageErr`

The `IpAppMailbox.putMessageErr` method is invoked with the following parameters:

Name	Type	Comment
mailboxSessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMailbox`].
requestID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMailbox.putMessageReq`].
error	TpMessaging Error	Results in the assignment of the following values to one **DeliveryInformation** parameter of the **deliveryStatus** part of a **getSmsDeliveryStatusResponse** message: • the **address** element contains the associated message destination address; • the **deliveryStatus** element has the value: **DeliveryImpossible**.
errorDetails	TpString	Not mapped.

6.2.4.3 Mapping to `IpMailbox`.getMessageInfoPropertiesReq

The `IpMailbox.getMessageInfoPropertiesReq` method is invoked with the following parameters:

Name	Type	Comment
mailboxSessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMailbox`].
messageID	TpString	Not mapped. [The value provided in the result from `IpAppMailbox.putMessageRes`].

The result from `IpMailbox.getMessageInfoPropertiesReq` is of type `TpAssignmentID` and is used internally to correlate the callback invocation of the `IpAppMailbox.getMessageInfoPropertiesRes/Err` method.

Parlay exceptions thrown by `IpMailbox.getMessageInfoPropertiesReq` are not mapped to Parlay X exceptions.

6.2.4.4 Mapping from `IpAppMailbox.`getMessageInfoPropertiesRes

The `IpAppMailbox.getMessageInfoPropertiesRes` method is invoked with the following parameters:

Name	Type	Comment
mailboxSessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMailbox`].
requestID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMailbox.getMessageInfoPropertiesReq`].
messageID	TpString	Not mapped. [The value provided in the invocation of `IpMailbox.getMessageInfoPropertiesReq`].
returnedProperties	TpMessageInfo PropertySet	Provides various message properties (names and values). Of these, the value of a single element, `MessageStatus`, is mapped to the **deliveryStatus** element of one **DeliveryInformation** parameter of the **deliveryStatus** part of a **getSmsDeliveryStatusResponse** message, as follows: • **DeliveredToTerminal**, if `MessageStatus` parameter value = `P_MMM_SENT_MSG_STATUS_DELIVERED`, `P_MMM_SENT_MSG_STATUS_READ` or `P_MMM_SENT_MSG_STATUS_DELETED_UNREAD` • **DeliveryImpossible**, if `MessageStatus` parameter value = `P_MMM_SENT_MSG_STATUS_NOT_DELIVERABLE` or `P_MMM_SENT_MSG_STATUS_EXPIRED` • **DeliveryUncertain**, if `MessageStatus` parameter value = `P_MMM_SENT_MSG_STATUS_SENT` Note that the **address** element of the **DeliveryInformation** parameter contains the associated message destination address

6.2.4.5 Mapping from `IpAppMailbox.`getMessageInfoPropertiesErr

The `IpAppMailbox.getMessageInfoPropertiesErr` method is invoked with the following parameters:

Name	Type	Comment
mailboxSessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMailbox`]
requestID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMailbox.getMessageInfoPropertiesReq`]
error	TpMessaging Error	If this message destination address has a current **deliveryStatus** value of **DeliveredToNetwork**, then it is updated to the **DeliveryUncertain** value. This updated value is reported to the application in a **DeliveryInformation** parameter of the **deliveryStatus** part of a **getSmsDeliveryStatusResponse** message. Note that the **address** element of the **DeliveryInformation** parameter contains the associated message destination address
errorDetails	TpString	Not mapped

6.2.5 notifySmsDeliveryReceipt

The sequence diagram in clause 5.3 Send Short Message to One or More Addresses (Mailbox Paradigm) (figure 3) illustrates the flow for the **notifySmsDeliveryReceipt** operation, which is mapped from the following Parlay/OSA methods:

- `Parlay exceptions thrown by IpMailbox.putMessageReq, as described in clause` 6.2.1.2 Mapping to `IpMailbox.putMessageReq.`

- `IpAppMailbox.putMessageErr.`

- `IpAppMailbox.getMessageInfoPropertiesRes.`

6.2.5.1 Mapping from `IpAppMailbox.putMessageErr`

The `IpAppMailbox.putMessageErr` method is invoked with the following parameters:

Name	Type	Comment
mailboxSessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMailbox`].
requestID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMailbox.putMessageReq`].
error	TpMessaging Error	Results in the assignment of the following values to the **DeliveryInformation** parameter of the **deliveryStatus** part of a **notifySmsDeliveryReceiptRequest** message: • the **address** element contains the associated message destination address; • the **deliveryStatus** element has the value: **DeliveryImpossible**.
errorDetails	TpString	Not mapped.

In addition, the **correlator** part of the **notifySmsDeliveryReceiptRequest** message is assigned the value of the **correlator** element of the **receiptRequest** part of the **sendSmsXxxRequest** message to which it relates.

6.2.5.2 Mapping from `IpAppMailbox.`getMessageInfoPropertiesRes

The `IpAppMailbox.getMessageInfoPropertiesRes` method is invoked with the following parameters:

Name	Type	Comment
mailboxSessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMailbox`].
requestID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMailbox.getMessageInfoPropertiesReq`].
messageID	TpString	Not mapped. [The value provided in the invocation of `IpMailbox.getMessageInfoPropertiesReq`].
returnedProperties	TpMessageInfo PropertySet	Provides various message properties (names and values). Of these, the value of a single element, `MessageStatus`, is mapped to the **deliveryStatus** element of the **DeliveryInformation** parameter of the **deliveryStatus** part of a **notifySmsDeliveryReceiptRequest** message, as follows: • **DeliveredToTerminal**, if `MessageStatus` parameter value = `P_MMM_SENT_MSG_STATUS_DELIVERED`, `P_MMM_SENT_MSG_STATUS_READ` or `P_MMM_SENT_MSG_STATUS_DELETED_UNREAD` • **DeliveryImpossible**, if `MessageStatus` parameter value = `P_MMM_SENT_MSG_STATUS_NOT_DELIVERABLE` or `P_MMM_SENT_MSG_STATUS_EXPIRED` Notes: • Other values of `MessageStatus`, e.g. `P_MMM_SENT_MSG_STATUS_SENT`, do not represent a final delivery status, and do not result in the generation of a **notifySmsDeliveryReceiptRequest** message. • The **address** element of the **DeliveryInformation** parameter contains the associated message destination address.

In addition, the **correlator** part of the **notifySmsDeliveryReceiptRequest** message is assigned the value of the **correlator** element of the **receiptRequest** part of the **sendSmsXxxRequest** message to which it relates.

6.2.6 startSmsNotification

The sequence diagram in clause 5.4 Notification of Short Message Reception and Retrieval (Mailbox Paradigm) (figure 4) illustrates the flow for the **startSmsNotification** operation, which is mapped to the Parlay/OSA method: `IpMMMManager.createNotification`, provided there is no existing notification already established for the destination address contained in the **smsServiceActivationNumber** part.

6.2.6.1 Mapping to IpMMMManager.createNotification

The `IpMMMManager.createNotification` is invoked with the following parameters:

Name	Type	Comment
appMMM Manager	IpAppMMM ManagerRef	Not mapped. Reference to callback (internal).
eventCriteria	TpMessaging EventCriteriaSet	Contains a single element specifying the event notification criteria, for the messaging event: `P_EVENT_MSG_NEW_MAILBOX_MESSAGE_ARRIVED`. The criteria consist of 2 fields: • `MailboxID`, which identifies a mailbox in the messaging system that is correlated with the short message destination address contained in the **smsServiceActivationNumber** part. • `AuthenticationInfo`, which provides the authentication information needed to open the mailbox, such as a key or password.

The result from `IpMMMManager.createNotification` is of type `TpAssignmentID` and is used internally to correlate the callbacks. Specifically it is correlated with the value of the **reference** part received from the application in the **startSmsNotificationRequest** message and the **correlator** part returned to the application in the **notifySmsReceptionRequest** message.

Note that the **reference** part and the optional **criteria** part of a **startSmsNotificationRequest** message are not mapped to `IpMMMManager.createNotification`. Instead the web service uses all the text string criteria values associated with a specific destination address to parse any received message event reported for that address by the `IpAppMMMManager.reportNotification method. The web service determines whether the event is valid - i.e. there is a match with a` text string criteria value. If valid, the web service stores the message and `selects the previously provisioned application callback web service to receive the` **notifySmsReceptionRequest** `message. If invalid, the web service discards the event notification.`

Parlay exceptions thrown by `IpMMMManager.createNotification` are mapped to Parlay X exceptions as defined in clause 6.3 Exceptions.

6.2.7 notifySmsReception

The sequence diagram in clause 5.4 Notification of Short Message Reception and Retrieval (Mailbox Paradigm) (figure 4) illustrates the flow for the **notifySmsReception** operation, which is mapped to/from the following Parlay/OSA methods:

- IpAppMMMManager.reportNotification.
- `IpMMMManager.openMailbox.`
- `IpMailbox.`getMessageContentReq.
- IpAppMailbox.getMessageContentRes.

6.2.7.1 Mapping from IpAppMMMManager.reportNotification

The `IpAppMMMManager.reportNotification` method is invoked with the following parameters:

Name	Type	Comment
assignmentID	TpAssignmentID	Not mapped. [The value provide in the result from `IpMMMManager.createNotification`].
eventInfo	TpMessaging EventInfoSet	Contains a set of one (or more) received message notification(s) and related message information. For each notification, the fields of the `EventNewMailboxMessageArrived` element are mapped as follows: • `MailboxID`: the mailbox identifier in each message notification is the same; i.e. it is equivalent to the value specified in the event criteria (clause 6.2.6.1 Mapping to IpMMMManager.createNotification). This field correlates with the short message destination address returned in the **smsServiceActivationNumber** part of a **notifySmsReceptionRequest** message. • `FolderID`: the folder identifier in each message notification specifies the identity of the folder in which the received message is stored • `MessageDescription` contains sub-fields, of which two are applicable for the mapping: • `MessageID`: the message identifier for the received message. • `From`: the sender of the received message, which maps to the **senderAddress** part of a **notifySmsReceptionRequest** message. The data type mapping from `TpAddress` to **xsd:anyURI** is described in TR 102 397-1 [3]. • `ExtendedHeaderInformation`: not applicable.

The result from `IpAppMMMManager.reportNotification` is of type `IpAppMultiMediaMessagingRef`. It is set to null.

6.2.7.2 Mapping to `IpMMMManager.openMailbox`

The `IpMMMManager.openMailbox` method is invoked with the following parameters:

Name	Type	Comment
mailboxID	TpString	Specifies the identity of the application's mailbox in the messaging system: i.e. as specified in the `eventInfo` parameter of the `reportNotification` method (clause 6.2.7.1 Mapping from IpAppMMMManager.reportNotification).
authenticationInfo	TpString	Specifies authentication information needed to open the application's mailbox, such as a key or password: i.e. as specified in the `AuthenticationInfo` field of the `eventCriteria` parameter of the `createNotification` method (clause 6.2.6.1 Mapping to IpMMMManager.createNotification).
appMailbox	IpAppMailboxRef	Reference to callback (internal)

The result from `IpMMMManager.openMailbox` is of type `TpMailboxIdentifier` and identifies the Mailbox interface object upon which future methods are invoked: e.g. `IpMailbox.getMessageContentReq`.

Parlay exceptions thrown by `IpMMMManager.openMailbox` are not mapped to Parlay X exceptions.

6.2.7.3 Mapping to `IpMailbox`.getMessageContentReq

The `IpMailbox.getMessageContentReq` method is invoked with the following parameters:

Name	Type	Comment
mailboxSessionID	TpSessionID	Not mapped. [The value provided in the result from `IpMMMManager.openMailbox`]
folderID	TpString	Not mapped. [The value provided in the `eventInfo` parameter of the `reportNotification` method (clause 6.2.7.1 Mapping from IpAppMMMManager.reportNotification)]
messageID	TpString	Not mapped. [The value provided in the `MessageDescription.MessageID` field of the `eventInfo` parameter of the `reportNotification` method (clause 6.2.7.1 Mapping from IpAppMMMManager.reportNotification)]

The result from `IpMailbox.getMessageContentReq` is of type `TpAssignmentID` and is used internally to correlate the callback invocation of the `IpAppMailbox.getMessageContentRes/Err` method.

Parlay exceptions thrown by `IpMailbox.getMessageContentReq` are not mapped to Parlay X exceptions.

6.2.7.4 Mapping from IpAppMailbox.getMessageContentRes

The `IpAppMailbox.getMessageContentRes` method is invoked with the following parameters:

Name	Type	Comment
mailboxSessionID	TpSessionID	Not mapped. [The value provide in the result from `IpMMMManager.openMailbox`].
requestID	TpAssignmentID	Not mapped. [The value provided in the result from `IpMailbox.getMessageContentReq`].
contentType	TpString	Not mapped.
contentTransfer Encoding	TpString	Not mapped.
content	TpOctetSet	Contains the body of the message. Maps to the **message** element of the **message** part of a **notifySmsReceptionRequest** message.

Note that this mapping occurs if there is at least one active notification established for the mailbox (i.e. as identified in the `MailboxID` element of the `eventInfo` parameter of `IpAppMMMManager.reportNotification`), an associated `application callback web service,` and one of the following conditions is satisfied:

- There is only one active notification that was defined without the optional text string criteria value
- There is one active notification that was defined with the optional text string criteria value and that value matches the first word in the value of the `content` parameter (of `IpAppMailbox.getMessageContentRes`).

NOTE: The 'first word' in the message is defined as the initial characters after discarding any leading Whitespace and ending with a Whitespace or end of message. The matching shall be case-insensitive.

6.2.8 getReceivedSms

The sequence diagram in clause 5.4 Notification of Short Message Reception and Retrieval (Mailbox Paradigm) (figure 4) illustrates the flow for the **getReceivedSms** operation. It is not explicitly mapped to any Parlay/OSA method. Instead, the **getReceivedSms** operation is a bulk retrieval capability for previously received short messages matching criteria defined in an off-line provisioning step. This retrieval operation includes matching messages previously and individually delivered to the application via the **notifySmsReception** operation.

6.2.9 `stopSMSNotification`

The sequence diagram in clause 5.4 Notification of Short Message Reception and Retrieval (Mailbox Paradigm) (figure 4) illustrates the flow for the **stopSmsNotification** operation, which is mapped to the Parlay/OSA method: `IpMMMManager.destroyNotification`, provided that the referenced notification is the last active notification for the associated destination address. Otherwise at least one other notification (i.e. associated with a different text string criteria value) remains active for this destination address and the mapping is not performed.

6.2.9.1 Mapping to IpMMMManager.destroyNotification

The `IpMMMManager.destroyNotification` is invoked with the following parameters:

Name	Type	Comment
assignmentID	TpAssignmentID	Not mapped. [The value provide in the result from `IpMMMManager.createNotification and correlated with the` value of the **reference** part received from the application in the original **startSmsNotificationRequest** message and the value of the **correlator** part received from the application in the **stopSmsNotificationRequest** message]

Parlay exceptions thrown by `IpMMMManager.destroyNotification` are mapped to Parlay X exceptions as defined in clause 6.3 Exceptions.

6.3 Exceptions

In addition to the common mapping of Parlay/OSA API method exceptions to Parlay X Web Service exceptions, which is defined in TR 102 397-1 [3], there are the following service-specific exception mappings:

Parlay/OSA Exception	Service Exception	Notes
P_MMM_INVALID_MAILBOX	SVC0001	With error number
P_MMM_INVALID_AUTHENTICATION_ INFORMATION	SVC0001	With error number

7 Additional Notes

No additional notes are provided.

History

Document history		
V1.1.1	December 2005	Publication