SDP Solution

[API Reference (SMS, ParlayX 2.1)]

Ethio telecom

Huawei Technologies Co., Ltd

Contents

1 Overview

1.1 API Functions

The SDP provides SMS capability application programming interfaces (APIs) for third-party applications (App for short) to connect to it and use its SMS capability to send and receive SMS messages. The App is generally developed by various partners of the SDP.

NOTE

Partners are the enterprises and individuals who sign a contract and cooperate with carriers in utilizing the SDP. Partners include SPs, CPs, Developers, and Enterprises. In this document, partners are mainly the SPs, Developers, and Enterprises who use APIs for secondary development.

Table 1-1 describes functions of SMS capability APIs provided by the SDP.

Table 1-1 Functions of SMS capability APIs

Function	Subfunction	Description	API
Receiving SMS messages	Receiving SMS messages in Notify mode	The App (functioning as the client) invokes the startSmsNotification API to subscribe to mobile originated (MO) SMS message notification on theSDP (functioning as the server). When receiving an MO SMS message from a user, the SDP (functioning as the client) invokes the notifySmsReception API to send the MO SMS message to the App (functioning as the server). Before the App is brought offline, the App (functioning as the client) invokes the stopSmsNotification API to unsubscribe from MO SMS message notification on the SDP (functioning as the server).	• 2.2 startSmsNotification • 2.3 notifySmsReception • 2.4 stopSmsNotification
	Receiving SMS messages in Get mode	The App (functioning as the client) invokes the getReceivedSms API to obtain MO SMS messages from the SDP (functioning as the server).	2.5 getReceivedSms
Sendi	Sending SMS	The App (functioning as the client) invokes	3.2 sendSms

Function	Subfunction	Description	API
ng SMS messages	messages to users	the sendSms API to send SMS messages through the SDP (functioning as the server).	
	Receiving status reports in Notify mode	After the App sends an SMS message to a user, the SMSC sends a status report to the SDP. The SDP (functioning as the client) then invokes the notifySmsDeliveryReceipt API to send the status report to the App (functioning as the server).	3.3 notifySmsDeliveryReceipt
	Receiving status reports in Get mode	After the App sends an SMS message to a user, the SMSC sends a status report to the SDP. The App (functioning as the client) then invokes the getSmsDeliveryStatus API to obtain the status report from the SDP (functioning as the server).	3.4 getSmsDeliveryStatus

The App receives MO SMS messages and status reports from the SDP in either of the following modes:

- Notify

  The SDP notifies the App of MO SMS messages and status reports immediately when receiving them from users. To use this mode, the App must subscribe to MO SMS message notification on the SDP. This mode has the following features:

  - Real-time

    The App receives users' SMS messages and status reports in real time, which provides a pleasant user experience in interactive services.

  - High requirements for hardware

    If the App hardware performance does not match that of the SDP, the App may fail to process surging requests in a timely manner.

- Get

  The App periodically obtains MO SMS messages and status reports from the SDP. This mode has the following features:

  - Non-real-time

    The App cannot receive users' SMS messages and status reports in real time, which degrades user experience in interactive services.

  - Low requirements for hardware

Partners can select a mode based on the service or application requirements. The Notify mode is recommended. Partners must use the Notify mode to receive messages involved in an on-demand SMS service or application.

1.2 Level of Requirement for Parameters

The App must develop APIs based on the level of requirement for each parameter.

Table 1-2 Level of requirement for parameters

Type	Description
Mandatory	A parameter is always mandatory in a request. Parameters with the **Mandatory** requirement are used for access authentication or service processing. If a parameter with the **Mandatory** requirement is left empty in a request, access authentication or service processing fails and the request fails.
Conditional	A parameter is mandatory or optional in specified conditions. Parameters with the **Conditional** requirement are used for access authentication or service processing in specified conditions. If the specified conditions are met but a parameter with the **Conditional** requirement is left empty in a request, access authentication or service processing fails and the request fails.
Optional	A parameter is always optional. Parameters with the **Optional** requirement are not used for service processing.

1.3 Request Format

The SDP provides the Parlay X 2.1 request in the following format:

```
<soapenv:Envelope>
   <soapenv:Header>
        <parameter>…</parameter>
…
   </soapenv:Header>
   <soapenv:Body>
        <parameter>…</parameter>
…
   </soapenv:Body>
</soapenv:Envelope>
```

Table 1-3 Request format

Element	Description
<soapenv:Envelope>	Root element in a request, which specifies the namespace.
<soapenv:Header>	Request header. Parameters in this element are defined by the SDP and are mainly information to be processed by the SDP services, including access authentication parameters.
<soapenv:Body>	Request body. Parameters in this element comply with the Parlay X 2.1 protocol.

NOTE

When Apps invoke SDP APIs, requests cannot contain the following XML characters: & > < ' "

If the preceding characters are really required, you must add escape characters before such characters. Otherwise, SDP APIs will fail to be invoked.

1.4 Response Format

Success Response Format

The SDP provides the Parlay X 2.1 API success responses in the following format:

```
<soapenv:Envelope>
   <soapenv:Body>
        <parameter>…</parameter>
…
   </soapenv:Body>
</soapenv:Envelope>
```

Table 1-4 Success response format

Element	Description
<soapenv:Envelope>	Root element in a success response, which specifies the namespace.
<soapenv:Body>	Success response body. Parameters in this element comply with the Parlay X 2.1 protocol.

Error Response Format

The SDP provides the Parlay X 2.1 API error responses in the following format:

```
<soapenv:Envelope>
  <soapenv:Body>
    <soapenv:Fault>
        <parameter>…</parameter>
…

      <detail>
        <parameter>…</parameter>
…
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Table 1-5 Error response format

Element	Description
<soapenv:Envelope>	Root element in an error response, which specifies the namespace.
<soapenv:Body>	Error response body, contains the <soapenv:Fault> and <detail> elements. This element specifies the error code and error details.
<soapenv:Fault>	Error code and description. For details about error responses, see 4 API Error Responses.
<detail>	Error details, which are the same as the <soapenv:Fault> element information.

1.5 Namespaces

When coding API functions, partners must follow the namespaces of SMS capability APIs and data types in the APIs.

- Table 1-6 describes namespaces of SMS capability APIs.
- The namespace of data types in SMS capability APIs is **http://www.csapi.org/schema/parlayx/sms/v2_2**.

NOTE

The SMS capability APIs involve the following data types: SimpleReference, SmsMessage, and DeliveryInformation. For details about the structure of the data types, see the parameter description for the matching API requests or responses.

Table 1-6 Namespaces of SMS capability APIs

Namespace	API
http://www.csapi.org/wsdl/parlayx/sms/send/v2_2	• 3.2 sendSms • 3.4 getSmsDeliveryStatus
http://www.csapi.org/wsdl/parlayx/sms/receive/v2_2	3.2 sendSms
http://www.csapi.org/wsdl/parlayx/sms/notification/v2_2	• 2.3 notifySmsReception • 3.3 notifySmsDeliveryReceipt
http://www.csapi.org/wsdl/parlayx/sms/notification_manager/v2_3	• 2.2 startSmsNotification • 2.4 stopSmsNotification

1.6 SOAPAction

Leave the **SOAPAction** parameter empty.

The following is an example of the **SOAPAction** parameter setting in an HTTP header:

```
SOAPAction: ""
```

2 APIs for Receiving SMS Messages

2.1 Process

Notify Mode

The process of the App receiving SMS messages in Notify mode consists of the following main steps:

- Subscribing to MO SMS message notification: After the subscription, the SDP notifies the App of MO SMS messages immediately when receiving them from users.
- Receiving MO SMS messages: The App receives MO SMS messages from the SDP in real time.
- Unsubscribing from MO SMS message notification: After the unsubscription, the SDP no longer notifies the App of MO SMS messages.

Figure 2-1 shows the process of receiving SMS messages in Notify mode.

Figure 2-1 Process of receiving SMS messages in Notify mode

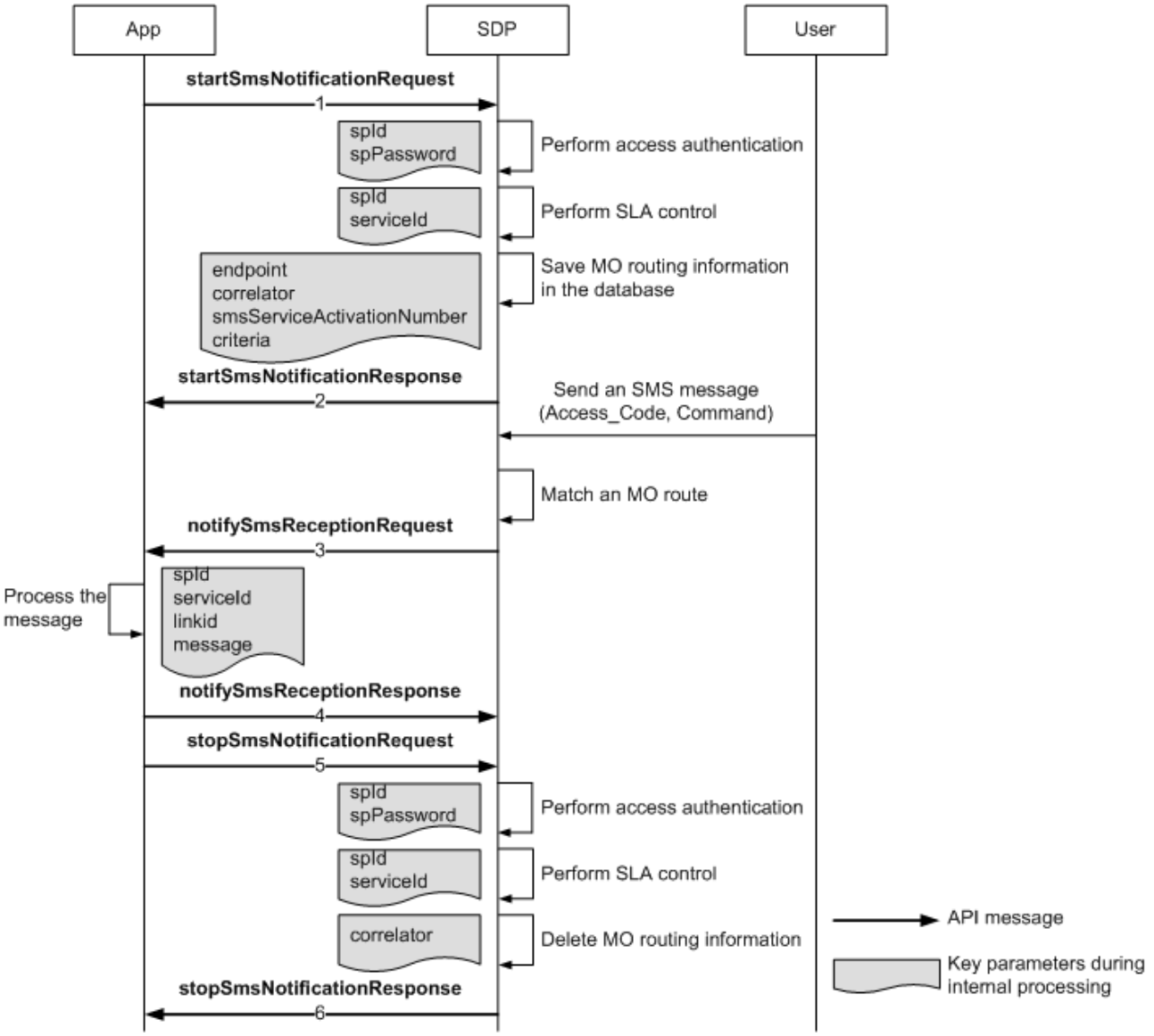

Table 2-1 describes the process.

Table 2-1 Description for the process of receiving SMS messages in Notify mode

Step	Description
1-2	• The App sends a request to the SDP to subscribe to MO SMS message notification. • The SDP performs authentication and service level agreement (SLA) control based on fields in the request, saves MO routing information in the database, and sends a response to the App.
3-4	• The SDP receives an MO SMS message from a user, matches an MO route based on the access code and command word, and sends a notification of the SMS message to the App.

Step	Description
	• The App parses the notification and sends a response to the SDP.
5-6	• The App sends a request to the SDP to unsubscribe from MO SMS message notification when the App is to be brought offline. • The SDP performs authentication and SLA control based on fields in the request, deletes MO routing information from the database, and sends a response to the App.

Get Mode

Figure 2-2 shows the process of receiving SMS messages in Get mode.

Figure 2-2 Process of receiving SMS messages in Get mode

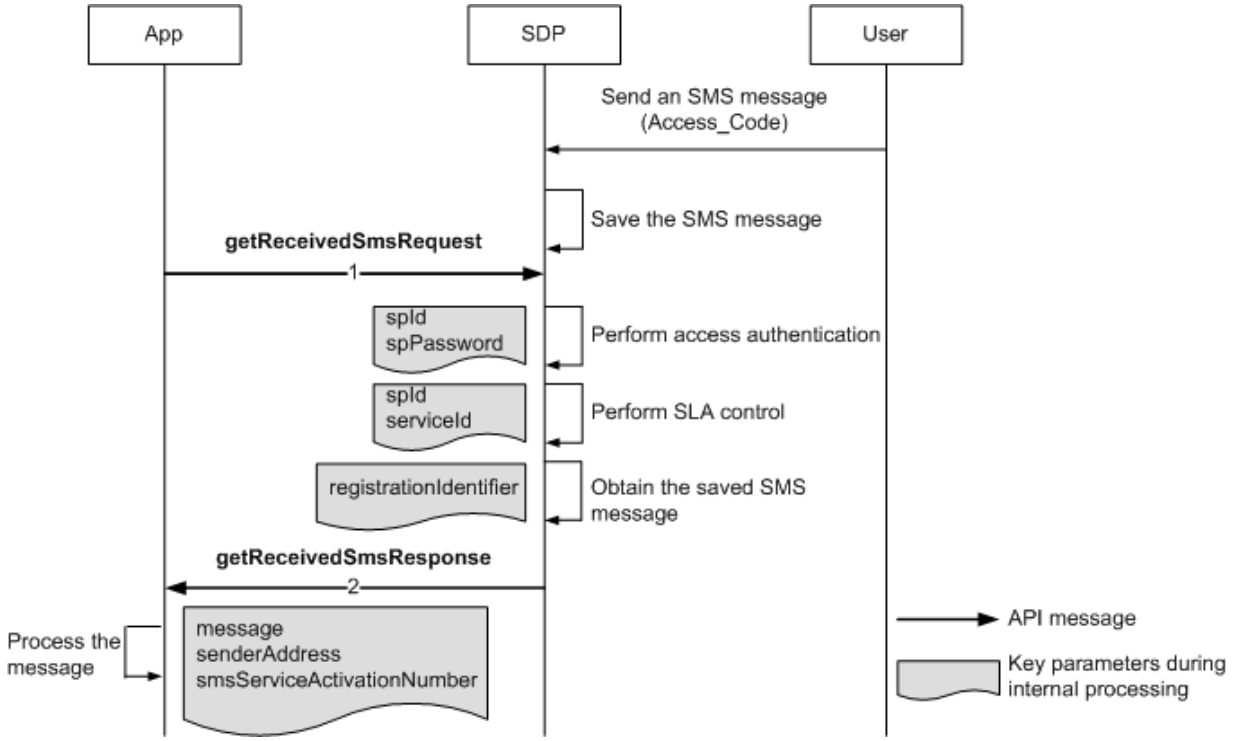

Table 2-2 describes the process.

Table 2-2 Description for the process of receiving SMS messages in Get mode

Step	Description
1–2	• After receiving an MO SMS message from a user, the SDP stores the MO SMS message for 48 hours by default. • The App sends a request to the SDP to obtain the MO SMS messages. • The SDP performs authentication and service level agreement (SLA) control based on fields in the request, and sends a response to the App.

Step	Description
	• The App processes the response message.

2.2 startSmsNotification

2.2.1 Function

The App (functioning as the client) invokes the startSmsNotification API to subscribe to MO SMS message notification on the SDP (functioning as the server).

The startSmsNotification API sends the routing information for the App to receive MO SMS message notifications to the SDP. When receiving the API request, the SDP saves the MO routing information of the App. After MO SMS message notification is subscribed, the SDP sends MO SMS messages received from users to the App based on the MO routing information.

Partners must code the App based on the API field requirements so that the App can send correct requests to the SDP. The SDP sends a response within 60 seconds by default.

2.2.2 Request URI

The request URI is the destination URI of startSmsNotificationRequest messages sent by the App to the SDP. The URI is provided by the SDP in the following format:

http://*IP*:*Port*/SmsNotificationManagerService/services/SmsNotificationManager

In the format, **IP** and **Port** indicate the service IP address and Parlay X 2.1 port number of the API provided by the SDP. Contact carriers to obtain the IP address and port number.

2.2.3 Request

The App functions as the client and sends a startSmsNotificationRequest message to the SDP to subscribe to MO SMS message notification.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v2="http://www.huawei.com.cn/schema/common/v2_1"
xmlns:loc="http://www.csapi.org/schema/parlayx/sms/notification_manager/v2_3/local
">
   <soapenv:Header>
      <RequestSOAPHeader xmlns="http://www.huawei.com.cn/schema/common/v2_1">
         <spId>000201</spId>
         <spPassword>e6434ef249df55c7a21a0b45758a39bb</spPassword>
         <serviceId>35000001000001</serviceId>
         <timeStamp>20100731064245</timeStamp>
      </RequestSOAPHeader>
   </soapenv:Header>
   <soapenv:Body>
      <loc:startSmsNotification>
         <loc:reference>
```

```
            <endpoint>http://10.138.38.139:9080/notify</endpoint>
            <interfaceName>notifySmsReception</interfaceName>
            <correlator>00001</correlator>
          </loc:reference>
          <loc:smsServiceActivationNumber>1234501</loc:smsServiceActivationNumber>
          <loc:criteria>demand</loc:criteria>
        </loc:startSmsNotification>
      </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 2-3 describes parameters in a startSmsNotificationRequest message header.

Table 2-3 Parameters in a startSmsNotificationRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • An SP can log in to the SDP management portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • A Developer can log in to the Developer Portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • An Enterprise must contact the carrier. [Example] 000201
spPassword	xsd: string	100	Conditional	Authentication key for the SDP to authenticate partners. The value is a character string encrypted from **spId** + **Password** + **timeStamp** by MD5. The encryption formula is: spPassword = MD5(spId + Password + timeStamp). In the formula: • **spId** and **timeStamp**: authentication ID and timestamp. • **Password**: password for partners to access the SDP. – An SP can obtain the password from the email notification received after successful registration.

Parameter	Type	Length	Level of Requirement	Description
				– A Developer can log in to the Developer Portal, choose **Member Center** > **Account** > **Registration Information** > **Invoke Password**, and set the password. – An Enterprise must contact the carrier to obtain the password. The SDP supports authentication by **SP ID + Password**, **SP ID + IP address + Password**, or **SP ID + IP address**. Partners select an authentication mode during registration. If a partner selects authentication by **SP ID + Password** or **SP ID + IP address + Password**, this parameter is mandatory in requests sent by this partner. [Example] e6434ef249df55c7a21a0b45758a39bb
serviceId	xsd: string	21	Conditional	Service ID. The ID is automatically allocated by the SDP to services after successful release. An SP can log in to the SDP management portal and query service information for the ID. This parameter is mandatory in a request sent by an SP and can be left empty in a request sent by a Developer or an Enterprise. [Example] 35000001000001
timeStamp	xsd: string	14	Conditional	Timestamp (UTC time). The value is used in MD5 encryption of **spPassword**. This parameter is mandatory when the **spPassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245

Message Body Parameters

Table 2-4 describes parameters in a startSmsNotificationRequest message body.

Table 2-4 Parameters in a startSmsNotificationRequest message body

Parameter	Type	Length	Level of Requirement	Description
reference	common:SimpleReference	-	Mandatory	Reference information. Reference information contains the App service address, API name, and correlator ID that a partner provides for the SDP. Based on the reference information, the SDP sends an MO SMS message notification to the matching App when receiving an SMS message from a user. The **reference** parameter is of the SimpleReference type and contains multiple sub-parameters. For details about the SimpleReference type, see Table 2-5.
smsServiceActivationNumber	xsd:anyURI	20	Mandatory	Access code. The value is planned and allocated by carriers. The SDP uses the access code and command word to match an MO route and routes user requests to the App. To obtain the access code: • An SP can log in to the SDP management portal and query service information. SPs can extend access codes allocated by carriers. In an extended access code, the prefix is allocated by carriers and the extension is defined by SPs. • A Developer or an Enterprise must contact the carrier. [Example] 1234501
criteria	xsd:string	50	Conditional	Service ordering or subscription command word. Users send SMS messages containing command words to order or subscribe to services. The SDP uses the access code and command word to match an MO route and routes user requests to the App. The command word is defined by SPs during service release. An SP can log in to the SDP management portal and query service information for the command word. This parameter is mandatory in a request sent by an SP who has configured the command word during service release. This parameter can be left empty in a request sent by a Developer, an Enterprise, or an SP who does not configure the command word during service release. [Example] demand

Table 2-5 describes the parameter structure of the SimpleReference type.

Table 2-5 Parameter structure of the SimpleReference type

Parameter	Type	Length	Level of Requirement	Description
endpoint	xsd:anyURI	512	Mandatory	Service address to which an SMS message is sent. [Example] http://10.138.38.139:9080/notify
interfaceName	xsd:string	20	Optional	Name of the API that receives SMS message notifications. The value can be customized. [Example] notifySmsReception
correlator	xsd:string	50	Mandatory	Correlator ID that associates a startSmsNotificationRequest message with a stopSmsNotificationRequest message. When the App sends a startSmsNotificationRequest message to the SDP, the SDP records the correlator ID. When the App sends a stopSmsNotificationRequest message to the SDP, the SDP unsubscribes from MO SMS message notification based on the correlator ID. The value is a random number defined by partners and must be unique. [Example] 00001

2.2.4 Response

The SDP functions as the server, processes startSmsNotificationRequest messages received from the App, and sends startSmsNotificationResponse messages to the App.

This topic provides a success response example. If a request fails, the SDP sends an error response that contains an error code. For details about error responses, see API Error Responses.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:startSmsNotificationResponse
xmlns:ns1="http://www.csapi.org/schema/parlayx/sms/notification_manager/v2_3/local
```

```
"/>
   </soapenv:Body>
</soapenv:Envelope>
```

2.2.5 Error Codes

Table 2-6 describes startSmsNotification error codes that the SDP may return upon an exception. For details about the error codes, see the *SDP Solution Error Code Reference*.

Table 2-6 startSmsNotification error codes

Error Code	Description
SVC0001	Service error.
SVC0002	Invalid input value.
SVC0005	Duplicate correlator.
SVC0008	Overlapping criteria.
SVC0901	Access authentication or authorization error.
SVC0905	Parameter error.

2.3 notifySmsReception

2.3.1 Function

The SDP (functioning as the client) invokes the notifySmsReception API to send MO SMS messages to the App (functioning as the server).

After the App subscribes to MO SMS message notification through the 2.2 startSmsNotification API, the SDP invokes the notifySmsReception API to send MO SMS messages received from users to the App. If the MO SMS messages fail to be sent, the SDP resends the messages to the App when any of the cached message resending criteria is met. Cached SMS messages can be resent for a maximum of five times. SMS messages can be resent at least 1800 seconds after a sending failure.

Partners must code the App based on the API field requirements so that the App can correctly parse and respond to requests received from the SDP. The App must send a response to the SDP within 30 seconds.

2.3.2 Request URI

The request URI is the destination URI of notifySmsReceptionRequest messages sent by the SDP to the App. The URI is defined by the App.

2.3.3 Request

The SDP functions as the client and sends a notifySmsReceptionRequest message to the App to report an MO SMS message.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header>
      <ns1:NotifySOAPHeader xmlns:ns1="http://www.huawei.com.cn/schema/common/v2_1">
         <ns1:spRevId>sdp</ns1:spRevId>
         <ns1:spRevpassword>206D88BB7F3D154B130DD6E1E0B8828B</ns1:spRevpassword>
         <ns1:spId>000201</ns1:spId>
         <ns1:serviceId>35000001000001</ns1:serviceId>
         <ns1:timeStamp>111029084631570</ns1:timeStamp>
         <ns1:linkid>12345678901111</ns1:linkid>
         <ns1:traceUniqueID>10000120010111062302172100011</ns1:traceUniqueID>
      </ns1:NotifySOAPHeader>
   </soapenv:Header>
   <soapenv:Body>
      <ns2:notifySmsReception
xmlns:ns2="http://www.csapi.org/schema/parlayx/sms/notification/v2_2/local">
         <ns2:correlator>00001</ns2:correlator>
         <ns2:message>
            <message>Hello world</message>
            <senderAddress>tel:8612312345678</senderAddress>
            <smsServiceActivationNumber>tel:1234501</smsServiceActivationNumber>
            <dateTime>2010-08-09T00:00:00.000+08:00</dateTime>
         </ns2:message>
      </ns2:notifySmsReception>
   </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 2-7 describes parameters in a notifySmsReceptionRequest message header.

Table 2-7 Parameters in a notifySmsReceptionRequest message header

Parameter	Type	Length	Level of Requirement	Description
spRevId	xsd: string	20	Conditional	Reverse authentication ID for the App to authenticate the SDP. The ID is set by SPs during registration. An SP can log in to the SDP management portal and query account information for the ID. This parameter is mandatory in a request sent to an SP who has configured authentication information during registration. This parameter

Parameter	Type	Length	Level of Requirement	Description
				can be left empty in a request sent to a Developer, an Enterprise, or an SP who does not configure authentication information. [Example] sdp
spRevpassword	xsd:string	100	Conditional	Reverse authentication key for the App to authenticate the SDP. The key is a character string encrypted from **spRevId** + **Password** + **timeStamp** by message digest algorithm 5 (MD5). The encryption formula is: spRevPassword = MD5(spRevId + Password + timeStamp). In the formula: • **spRevId** and **timeStamp**: reverse authentication ID and timestamp. • **Password**: access password allocated by an SP to the SDP. An SP can obtain the password from the email notification received after successful registration. This parameter is mandatory in a request sent to an SP who has configured authentication information during registration. This parameter can be left empty in a request sent to a Developer, an Enterprise, or an SP who does not configure authentication information. [Example] 206D88BB7F3D154B130DD6E1E0B8828B
spId	xsd:string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • An SP can log in to the SDP management portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • A Developer can log in to the Developer Portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • An Enterprise must contact the carrier. [Example] 000201

Parameter	Type	Length	Level of Requirement	Description
serviceId	xsd: string	21	Conditional	Service ID. The ID is automatically allocated by the SDP to services after successful release. An SP can log in to the SDP management portal and query service information for the ID. This parameter is mandatory in a request sent to an SP, and can be left empty in a request sent to a Developer or an Enterprise. [Example] 35000001000001
timeStamp	xsd: string	14	Conditional	Timestamp (UTC time). The value is used in MD5 encryption of **spRevpassword**. This parameter is mandatory when the **spRevpassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245
linkid	xsd:string	20	Conditional	Service order ID. The ID is automatically generated by the SDP when a user orders a service in the SDP. This parameter is mandatory during service ordering by SMS message. [Example] 12345678901111
traceUniqueID	xsd:string	30	Mandatory	Transaction ID. The ID is automatically generated by the SDP and is used only to trace messages during the SDP commissioning. The App ignores this parameter. [Example] 10000120010111062302172100011

Message Body Parameters

Table 2-8 describes parameters in a notifySmsReceptionRequest message body.

Table 2-8 Parameters in a notifySmsReceptionRequest message body

Parameter	Type	Length	Level of Requirement	Description
correlator	xsd:string	50	Mandatory	Correlator ID that associates a startSmsNotificationRequest message with a notifySmsReceptionRequest message. The SDP directly obtains the value of **correlator** in the 2.2 startSmsNotification API request. [Example] 00001
message	SmsMessage	-	Mandatory	SMS message information. This parameter contains the SMS message content, sender's mobile number, access code, and time information. The **message** parameter is of the SmsMessage type and contains multiple sub-parameters. For details about the SmsMessage type, see Table 2-9.

Table 2-9 describes the parameter structure of the SmsMessage type.

Table 2-9 Parameter structure of the SmsMessage type

Parameter	Type	Length	Level of Requirement	Description
message	xsd:string	700	Mandatory	SMS message content. [Format] *[Command word] [Message body]* In the format, *[Command word]* is optional. Its value can be a service ordering or subscription command word in a request sent to an SP, and is left empty in a request sent to a Developer or an Enterprise. [Example] order Hello world In the example, **order** is a service ordering command word.
senderAddress	xsd:anyURI	30	Mandatory	Mobile number of the sender. [Format] tel:*[Prefix][Country code][Mobile number]* In the format, *[Prefix]* and *[Country code]* are

Parameter	Type	Length	Level of Requirement	Description
				optional. The value of *[Prefix]*, if contained, can be **+**, **+0**, **+00**, **0**, or **00**. [Example] tel:8612312345678
smsServiceActivationNumber	xsd:anyURI	20	Mandatory	Access code. The value is planned and allocated by carriers. The SDP uses the access code and command word to match an MO route and routes user requests to the App. To obtain the access code: • An SP can log in to the SDP management portal and query service information. SPs can extend access codes allocated by carriers. In an extended access code, the prefix is allocated by carriers and the extension is defined by SPs. • A Developer or an Enterprise must contact the carrier. [Format] tel:*Access code* [Example] tel:1234501
dateTime	xsd:dateTime	30	Optional	Date and time (UTC time) when the SDP receives the SMS message. [Format] *yyyy-MM-dd*T*HH:mm:ss.SSSZ* [Example] 2010-08-09T00:00:00.000+08:00

2.3.4 Response

The App functions as the server, processes the request messages received from the SDP, and sends the response messages to the SDP.

The response is constructed based on the WSDL specification by the partner that provides the App.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/sms/notification/v2_2/local">
   <soapenv:Header/>
   <soapenv:Body>
      <loc:notifySmsReceptionResponse/>
```

```
   </soapenv:Body>
</soapenv:Envelope>
```

2.3.5 Error Codes

The App returns error codes to the SDP when an exception occurs in response to notifySmsReceptionRequest messages. The error codes are defined by partners.

2.4 stopSmsNotification

2.4.1 Function

The App (functioning as the client) invokes the stopSmsNotification API to unsubscribe from MO SMS message notification on the SDP (functioning as the server). This API is invoked by the App when it is to be brought offline.

After MO SMS message notification is unsubscribed, the SDP does not send MO SMS messages received from users to the App.

Partners must code the App based on the API field requirements so that the App can send correct requests to the SDP. The SDP sends a response within 60 seconds by default.

2.4.2 Request URI

The request URI is the destination URI of stopSmsNotificationRequest messages sent by the App to the SDP. The URI is provided by the SDP in the following format:

http://*IP*:*Port*/SmsNotificationManagerService/services/SmsNotificationManager

In the format, *IP* and *Port* indicate the service IP address and Parlay X 2.1 port number of the API provided by the SDP. Contact carriers to obtain the IP address and port number.

2.4.3 Request

The App functions as the client and sends a stopSmsNotificationRequest message to the SDP to unsubscribe from MO SMS message notification.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v2="http://www.huawei.com.cn/schema/common/v2_1"
xmlns:loc="http://www.csapi.org/schema/parlayx/sms/notification_manager/v2_3/local
">
   <soapenv:Header>
      <v2:RequestSOAPHeader>
         <v2:spId>000201</v2:spId>
         <v2:spPassword>e6434ef249df55c7a21a0b45758a39bb</v2:spPassword>
         <v2:serviceId>35000001000001</v2:serviceId>
         <v2:timeStamp>201007211126</v2:timeStamp>
      </v2:RequestSOAPHeader>
   </soapenv:Header>
   <soapenv:Body>
      <loc:stopSmsNotification>
        <loc:correlator>00001</loc:correlator>
```

```
      </loc:stopSmsNotification>
    </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 2-10 describes parameters in a stopSmsNotificationRequest message header.

Table 2-10 Parameters in a stopSmsNotificationRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • An SP can log in to the SDP management portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • A Developer can log in to the Developer Portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • An Enterprise must contact the carrier. [Example] 000201
spPassword	xsd: string	100	Conditional	Authentication key for the SDP to authenticate partners. The value is a character string encrypted from **spId** + **Password** + **timeStamp** by MD5. The encryption formula is: spPassword = MD5(spId + Password + timeStamp). In the formula: • **spId** and **timeStamp**: authentication ID and timestamp. • **Password**: password for partners to access the SDP. – An SP can obtain the password from the email notification received after successful registration. – A Developer can log in to the Developer Portal, choose **Member Center** > **Account** > **Registration Information** > **Invoke Password**, and set the password.

Parameter	Type	Length	Level of Requirement	Description
				– An Enterprise must contact the carrier to obtain the password. The SDP supports authentication by **SP ID + Password**, **SP ID + IP address + Password**, or **SP ID + IP address**. Partners select an authentication mode during registration. If a partner selects authentication by **SP ID + Password** or **SP ID + IP address + Password**, this parameter is mandatory in requests sent by this partner. [Example] e6434ef249df55c7a21a0b45758a39bb
serviceId	xsd: string	21	Conditional	Service ID. The ID is automatically allocated by the SDP to services after successful release. An SP can log in to the SDP management portal and query service information for the ID. This parameter is mandatory in a request sent by an SP and can be left empty in a request sent by a Developer or an Enterprise. [Example] 35000001000001
timeStamp	xsd: string	14	Conditional	Timestamp (UTC time). The value is used in MD5 encryption of **spPassword**. This parameter is mandatory when the **spPassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245

Message Body Parameters

Table 2-11 describes parameters in a stopSmsNotificationRequest message body.

Table 2-11 Parameters in a stopSmsNotificationRequest message body

Parameter	Type	Length	Level of Requirement	Description
correlator	xsd:string	50	Mandatory	Correlator ID that associates a startSmsNotificationRequest message with a stopSmsNotificationRequest message. When the App sends a startSmsNotificationRequest message to the SDP, the SDP records the correlator ID. When the App sends a stopSmsNotificationRequest message to the SDP, the SDP unsubscribes from MO SMS message notification based on the correlator ID. The value must be the same as the value of **correlator** in the startSmsNotificationRequest message body. [Example] 00001

2.4.4 Response

The SDP functions as the server, processes stopSmsNotificationRequest messages received from the App, and sends stopSmsNotificationResponse messages to the App.

This topic provides a success response example. If a request fails, the SDP sends an error response that contains an error code. For details about error responses, see API Error Responses.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:stopSmsNotificationResponse
xmlns:ns1="http://www.csapi.org/schema/parlayx/sms/notification_manager/v2_3/local
"/>
   </soapenv:Body>
</soapenv:Envelope>
```

2.4.5 Error Codes

Table 2-12 describes stopSmsNotification error codes that the SDP may return upon an exception. For details about the error codes, see the *SDP Solution Error Code Reference*.

Table 2-12 stopSmsNotification error codes

Error Code	Description
SVC0001	Service error.
SVC0002	Invalid input value.
SVC0901	Access authentication or authorization error.
SVC0905	Parameter error.

2.5 getReceivedSms

2.5.1 Function

The App (functioning as the client) invokes the getReceivedSms API to obtain MO SMS messages from the SDP (functioning as the server). The SDP saves MO SMS messages received from the SMSC for only 48 hours.

When the App invokes the getReceivedSms API, the SDP sends MO SMS messages to the App. The number of MO SMS messages sent by the SDP to the App is determined by the SLA of the App.

Partners must code the App based on the API field requirements so that the App can send correct requests to the SDP. The SDP sends a response within 60 seconds by default.

2.5.2 Request URI

The request URI is the destination URI of getReceivedSmsRequest messages sent by the App to the SDP. The URI is provided by the SDP in the following format:

http://*IP*:*Port*/ReceiveSmsService/services/ReceiveSms

In the format, **IP** and **Port** indicate the service IP address and Parlay X 2.1 port number of the API provided by the SDP. Contact carriers to obtain the IP address and port number.

2.5.3 Request

The App functions as the client and sends a getReceivedSmsRequest message to the SDP to receive MO SMS messages.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v2="http://www.huawei.com.cn/schema/common/v2_1"
xmlns:loc="http://www.csapi.org/schema/parlayx/sms/receive/v2_2/local">
  <soapenv:Header>
    <v2:RequestSOAPHeader>
      <v2:spId>000201</v2:spId>
      <v2:spPassword>e6434ef249df55c7a21a0b45758a39bb</v2:spPassword>
      <v2:serviceId>35000001000001</v2:serviceId>
```

```
        <v2:timeStamp>201007211126</v2:timeStamp>
      </v2:RequestSOAPHeader>
    </soapenv:Header>
    <soapenv:Body>
      <loc:getReceivedSms>
        <loc:registrationIdentifier>1234501</loc:registrationIdentifier>
      </loc:getReceivedSms>
    </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 2-13 describes parameters in a getReceivedSmsRequest message header.

Table 2-13 Parameters in a getReceivedSmsRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • An SP can log in to the SDP management portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • A Developer can log in to the Developer Portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • An Enterprise must contact the carrier. [Example] 000201
spPassword	xsd: string	100	Conditional	Authentication key for the SDP to authenticate partners. The value is a character string encrypted from **spId** + **Password** + **timeStamp** by MD5. The encryption formula is: spPassword = MD5(spId + Password + timeStamp). In the formula: • **spId** and **timeStamp**: authentication ID and timestamp. • **Password**: password for partners to access the SDP. – An SP can obtain the password from the

Parameter	Type	Length	Level of Requirement	Description
				email notification received after successful registration. – A Developer can log in to the Developer Portal, choose **Member Center** > **Account** > **Registration Information** > **Invoke Password**, and set the password. – An Enterprise must contact the carrier to obtain the password. The SDP supports authentication by **SP ID + Password**, **SP ID + IP address + Password**, or **SP ID + IP address**. Partners select an authentication mode during registration. If a partner selects authentication by **SP ID + Password** or **SP ID + IP address + Password**, this parameter is mandatory in requests sent by this partner. [Example] e6434ef249df55c7a21a0b45758a39bb
serviceId	xsd: string	21	Conditional	Service ID. The ID is automatically allocated by the SDP to services after successful release. An SP can log in to the SDP management portal and query service information for the ID. This parameter is mandatory in a request sent by an SP and can be left empty in a request sent by a Developer or an Enterprise. [Example] 35000001000001
timeStamp	xsd: string	14	Conditional	Timestamp (UTC time). The value is used in MD5 encryption of **spPassword**. This parameter is mandatory when the **spPassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245

Message Body Parameters

Table 2-14 describes parameters in a getReceivedSmsRequest message body.

Table 2-14 Parameters in a getReceivedSmsRequest message body

Parameter	Type	Length	Level of Requirement	Description
registrationIdentifier	xsd:string	20	Mandatory	Access code. The value is planned and allocated by carriers. The SDP uses the access code and command word to match an MO route and routes user requests to the App. To obtain the access code: • An SP can log in to the SDP management portal and query service information. SPs can extend access codes allocated by carriers. In an extended access code, the prefix is allocated by carriers and the extension is defined by SPs. • A Developer or an Enterprise must contact the carrier. [Example] 1234501

2.5.4 Response

The SDP functions as the server, processes getReceivedSmsRequest messages received from the App, and sends getReceivedSmsResponse messages to the App.

This topic provides a success response example and describes parameters in the response. If a request fails, the SDP sends an error response that contains an error code. For details about error responses, see API Error Responses.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:getReceivedSmsResponse
xmlns:ns1="http://www.csapi.org/schema/parlayx/sms/receive/v2_2/local">
         <ns1:result>
            <message>Hello world</message>
            <senderAddress>tel:8612312345678</senderAddress>
            <smsServiceActivationNumber>tel:1234501</smsServiceActivationNumber>
            <dateTime>2010-08-09T00:00:00.000+08:00</dateTime>
         </ns1:result>
      </ns1:getReceivedSmsResponse>
   </soapenv:Body>
```

```
</soapenv:Envelope>
```

Message Body Parameters

Table 2-15 describes parameters in a getReceivedSmsResponse message body.

Table 2-15 Parameters in a getReceivedSmsResponse message body

Parameter	Type	Length	Level of Requirement	Description
result	SmsMessage[0..unbounded]	-	Optional	SMS message content. This parameter contains the SMS message content, sender's mobile number, access code, and time information. The **result** parameter is of the SmsMessage type and contains multiple sub-parameters. For details about the SmsMessage type, see Table 2-16.

Table 2-16 describes the parameter structure of the SmsMessage type.

Table 2-16 Parameter structure of the SmsMessage type

Parameter	Type	Length	Level of Requirement	Description
message	xsd:string	700	Mandatory	SMS message content. [Example] Hello world
senderAddress	xsd:anyURI	30	Mandatory	Mobile number of the sender. [Format] tel:*[Prefix][Country code][Mobile number]* In the format, *[Prefix]* and *[Country code]* are optional. The value of *[Prefix]* can be **+**, **+0**, **+00**, **0**, or **00**. [Example] tel:8612312345678
smsServiceActivationNumber	xsd:anyURI	20	Mandatory	Access code. The value is planned and allocated by carriers. The SDP uses the access code and command word to match an MO route and routes user requests to the App. To obtain the access code: • An SP can log in to the SDP

Parameter	Type	Length	Level of Requirement	Description
				management portal and query service information. SPs can extend access codes allocated by carriers. In an extended access code, the prefix is allocated by carriers and the extension is defined by SPs. • A Developer or an Enterprise must contact the carrier. [Format] tel:*Access code* [Example] tel:1234501
dateTime	xsd:dateTime	30	Optional	Date and time when the SDP receives the SMS message. [Format] *yyyy-MM-dd*T*HH:mm:ss.SSSZ* [Example] 2010-08-09T00:00:00.000+08:00

2.5.5 Error Codes

Table 2-17 describes getReceivedSms error codes that the SDP may return upon an exception. For details about the error codes, see the *SDP Solution Error Code Reference*.

Table 2-17 getReceivedSms error codes

Error Code	Description
SVC0001	Service error.
SVC0002	Invalid input value.
SVC0901	Access authentication or authorization error.
SVC0905	Parameter error.

3 APIs for Sending SMS Messages

3.1 Process

The App sends SMS messages to users and receives status reports in Notify or Get mode to determine whether the SMS messages are successfully sent.

If the App does not require status reports, the process of sending SMS messages is from steps 1 to 2 in Figure 3-2.

Notify Mode

Figure 3-1 shows the process of the App sending SMS messages and receiving status reports in Notify mode.

Figure 3-1 Process of sending SMS messages and receiving status reports in Notify mode

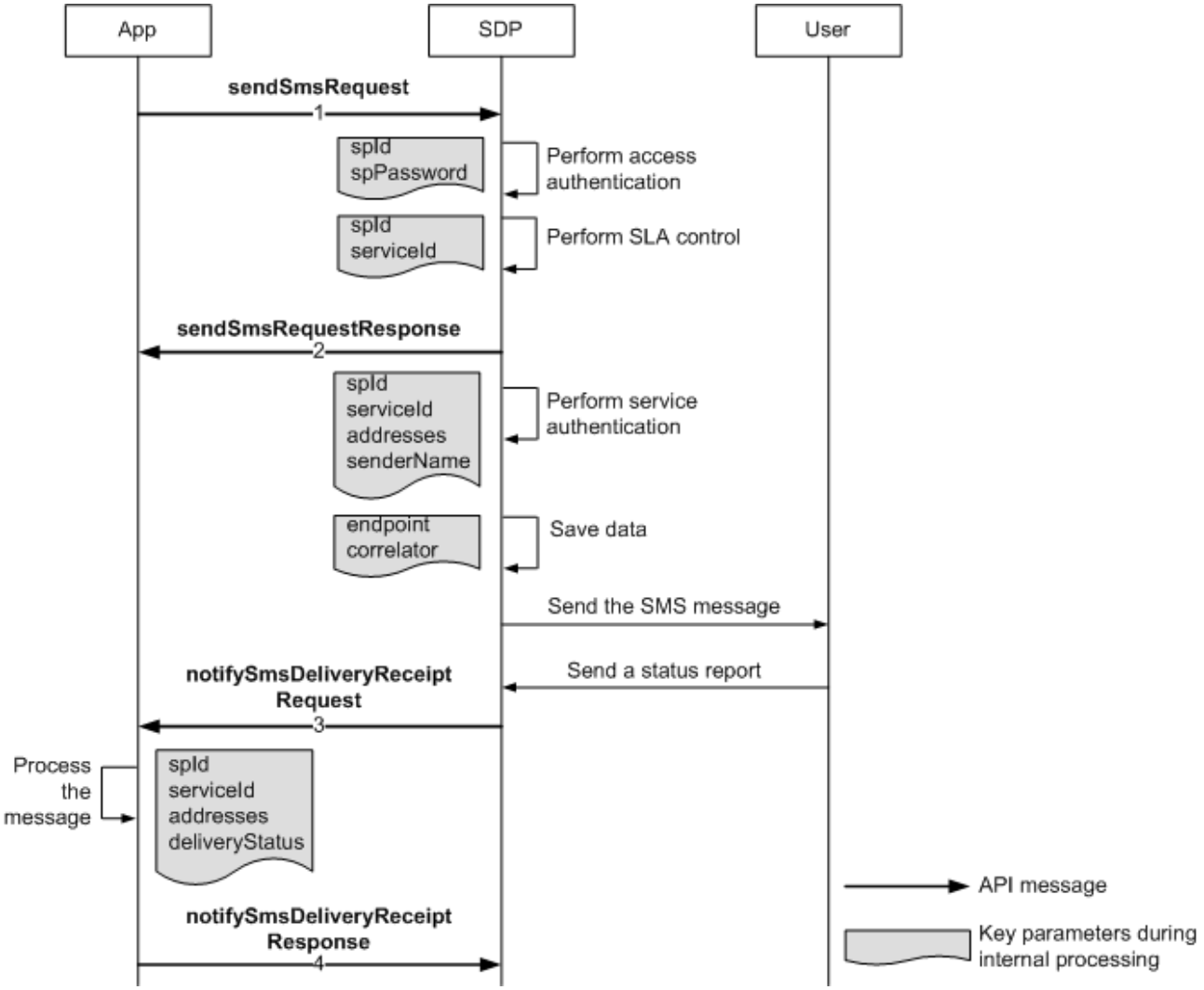

Table 3-1 describes the process.

Table 3-1 Description for the process of sending SMS messages and receiving status reports in Notify mode

Step	Description
1-2	• The App sends a request to the SDP to send an SMS message. • The SDP performs authentication and SLA control based on fields in the request and sends a response to the App. Then the SDP performs service authentication, saves the values of **endpoint** and **correlator** in the request, and sends the SMS message to the user.
3-4	• The SDP receives a status report and sends a notification of the status report to the App based on **endpoint**. • The App parses the notification and sends a response to the SDP.

Get Mode

Figure 3-2 shows the process of the App sending SMS messages and receiving status reports in Get mode.

Figure 3-2 Process of sending SMS messages and receiving status reports in Get mode

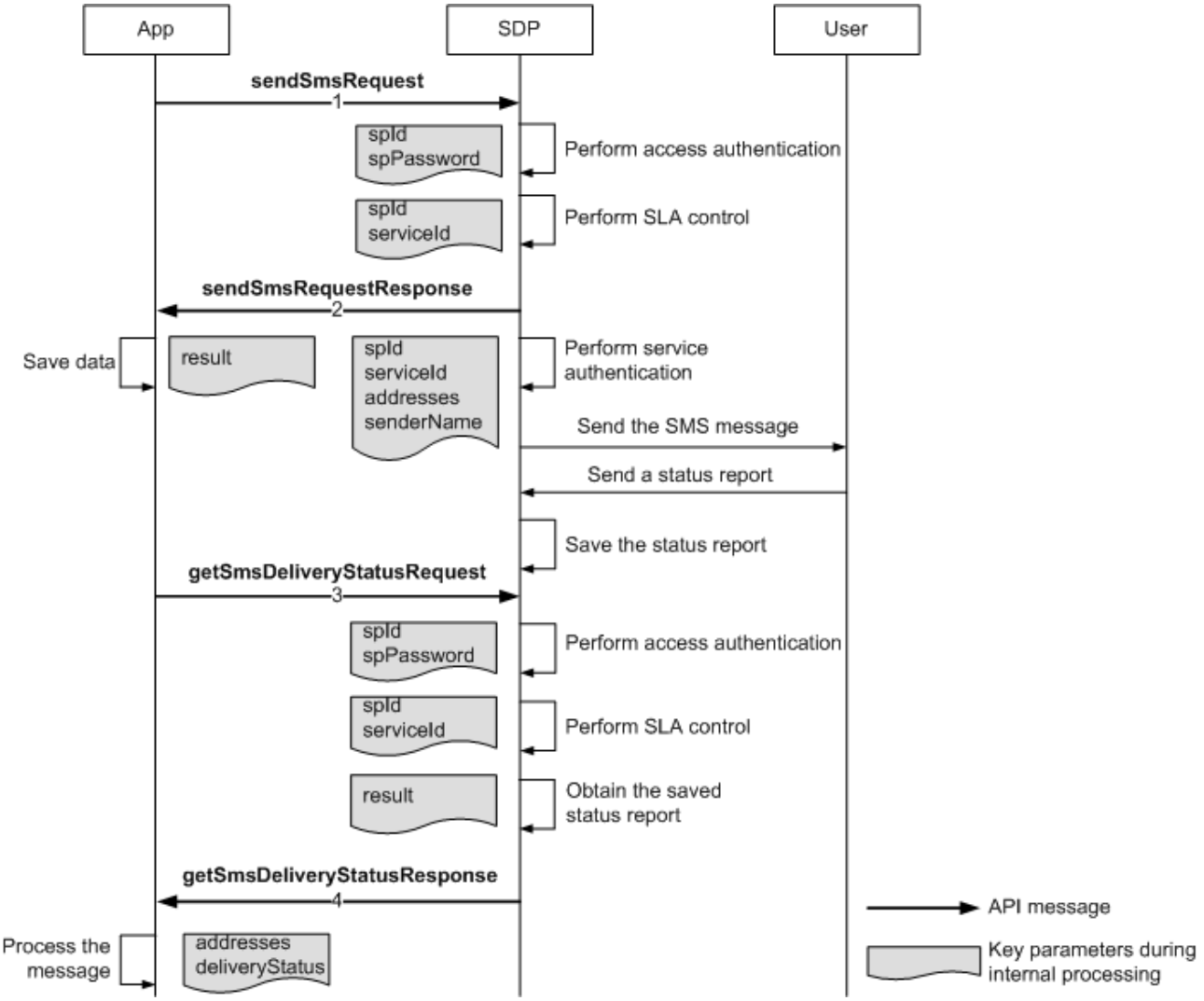

Table 3-2 describes the process.

Table 3-2 Description for the process of sending SMS messages and receiving status reports in Get mode

Step	Description
1-2	• The App sends a request to the SDP to send an SMS message. • The SDP performs authentication and SLA control based on fields in the request and sends a response to the App. Then the SDP performs service authentication and sends the SMS message to the user.
3-4	• The SDP receives a status report from the user and saves the report for a

Step	Description
	specific period (48 hours by default). • The App sends a request to the SDP at scheduled time to obtain the status report. • The SDP performs authentication and SLA control based on fields in the request, obtains the saved status report, and sends a response to the App. • The App processes the response.

3.2 sendSms

3.2.1 Function

The App (functioning as the client) invokes the sendSms API to send SMS messages to the SDP (functioning as the server).

Partners must code the App based on the API field requirements so that the App can send correct requests to the SDP. The SDP sends a response within 60 seconds by default.

3.2.2 Request URI

The request URI is the destination URI of sendSmsRequest messages sent by the App to the SDP. The URI is provided by the SDP in the following format:

http://*IP*:*Port*/SendSmsService/services/SendSms

In the format, **IP** and **Port** indicate the service IP address and Parlay X 2.1 port number of the API provided by the SDP. Contact carriers to obtain the IP address and port number.

3.2.3 Request

The App functions as the client and sends sendSmsRequest messages to the SDP.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v2="http://www.huawei.com.cn/schema/common/v2_1"
xmlns:loc="http://www.csapi.org/schema/parlayx/sms/send/v2_2/local">
   <soapenv:Header>
      <v2:RequestSOAPHeader>
         <v2:spId>000201</v2:spId>
         <v2:spPassword>e6434ef249df55c7a21a0b45758a39bb</v2:spPassword>
         <v2:serviceId>35000001000001</v2:serviceId>
         <v2:timeStamp>20100731064245</v2:timeStamp>
         <v2:OA>8612312345678</v2:OA>
         <v2:FA>8612312345678</v2:FA>
         <v2:linkid>12345678901111</v2:linkid>
         <v2:presentid>22345678901113</v2:presentid>
      </v2:RequestSOAPHeader>
   </soapenv:Header>
   <soapenv:Body>
```

```
      <loc:sendSms>
         <loc:addresses>tel:8612312345678</loc:addresses>
         <loc:senderName>321123</loc:senderName>
         <loc:message>Hello World.</loc:message>
         <loc:receiptRequest>
            <endpoint>http://10.138.38.139:9080/notify</endpoint>
            <interfaceName>SmsNotification</interfaceName>
            <correlator>00001</correlator>
         </loc:receiptRequest>
      </loc:sendSms>
   </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 3-3 describes parameters in a sendSmsRequest message header.

Table 3-3 Parameters in a sendSmsRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • An SP can log in to the SDP management portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • A Developer can log in to the Developer Portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • An Enterprise must contact the carrier. [Example] 000201
spPassword	xsd: string	100	Conditional	Authentication key for the SDP to authenticate partners. The value is a character string encrypted from **spId** + **Password** + **timeStamp** by MD5. The encryption formula is: spPassword = MD5(spId + Password + timeStamp). In the formula: • **spId** and **timeStamp**: authentication ID and timestamp.

Parameter	Type	Length	Level of Requirement	Description
				• **Password**: password for partners to access the SDP. – An SP can obtain the password from the email notification received after successful registration. – A Developer can log in to the Developer Portal, choose **Member Center** > **Account** > **Registration Information** > **Invoke Password**, and set the password. – An Enterprise must contact the carrier to obtain the password. The SDP supports authentication by **SP ID + Password**, **SP ID + IP address + Password**, or **SP ID + IP address**. Partners select an authentication mode during registration. If a partner selects authentication by **SP ID + Password** or **SP ID + IP address + Password**, this parameter is mandatory in requests sent by this partner. [Example] e6434ef249df55c7a21a0b45758a39bb
serviceId	xsd: string	21	Conditional	Service ID. The ID is automatically allocated by the SDP to services after successful release. An SP can log in to the SDP management portal and query service information for the ID. This parameter is mandatory in an SP's request, and can be left empty in a Developer's or an Enterprise's request. [Example] 35000001000001
timeStamp	xsd: string	14	Conditional	Timestamp (UTC time). The value is used in MD5 encryption of **spPassword**. This parameter is mandatory when the **spPassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245
OA	xsd:strin	30	Conditional	Mobile number of the service originator.

Parameter	Type	Length	Level of Requirement	Description
	g			This parameter is mandatory in a request for sending a single SMS message, and can be left empty in a request for sending bulk SMS messages. • In an SP's request for sending an SMS message to a user who subscribes to or orders a service, the value is the mobile number of the user. In an SP's request for sending an SMS message to a gift recipient, the value is the mobile number of the gift sender. • In an Enterprise's or a Developer's request, the value is the mobile number of the message recipient. [Example] 8612312345678
FA	xsd:string	30	Conditional	Mobile number of the charged party. When sending a single message, set **FA** to the value of **OA**. When sending a group message, you can leave the **FA** field empty. Charging is executed based on the **address** field.
linkid	xsd:string	20	Conditional	Service order ID. The ID is automatically generated by the SDP when a user orders a service in the SDP. This parameter is mandatory during on-demand service delivery by SMS message. The SDP sends the value to SPs as follows in different scenarios: • Invokes the ServiceOnDemand API to send the value when a user orders a service on the SDP portals. • Invokes the notifySmsReception API to send the value when a user orders a service by sending an SMS message. [Example] 12345678901111
presentid	xsd:string	15	Conditional	Service gift ID. The ID is automatically generated by the SDP when a user sends a service to another user as a gift on the SDP. This parameter is mandatory in an SP's

Parameter	Type	Length	Level of Requirement	Description
				request for sending an SMS message to a gift recipient, and can be left empty in an Enterprise's or a Developer's request. The SDP invokes the assignPresentToUser API to send the value to SPs. [Example] 22345678901113

Message Body Parameters

Table 3-4 describes parameters in a sendSmsRequest message body.

Table 3-4 Parameters in a sendSmsRequest message body

Parameter	Type	Length	Level of Requirement	Description
addresses	xsd:anyURI[1..unbounded]	30	Mandatory	Mobile number of the message recipient. [Format] tel:*[Prefix][Country code][Mobile number]* In the format, *[Prefix]* and *[Country code]* are optional. The value of *[Prefix]*, if contained, can be **+**, **+0**, **+00**, **0**, or **00**. [Example] tel:8612312345678
senderName	xsd:string	20	Mandatory	Name of the message sender, which is displayed on users' terminals. • In an SP's request, the value is a service access code obtained from carriers before service release. • In an Enterprise's or a Developer's request, the value is an access code allocated by carriers during capability product purchase. [Example] 321123
message	xsd:string	700	Mandatory	Content of the SMS message to be sent. When the length of an SMS message exceeds the maximum length (160 GSM 7-bit characters or 70 Unicode characters) supported by the short message service center (SMSC), the message is split into multiple sub messages.

Parameter	Type	Length	Level of Requirement	Description
				[Example] Hello World.
receiptRequest	common:SimpleReference	-	Conditional	Information required for status report notification. The App sends the App URL, API name, and correlator ID information to the SDP, which then uses the information for status report notification. The **receiptRequest** parameter is of the SimpleReference type and contains multiple sub-parameters. For details about the SimpleReference type, see Table 3-5.

Table 3-5 describes the parameter structure of the SimpleReference type.

Table 3-5 Parameter structure of the SimpleReference type

Parameter	Type	Length	Level of Requirement	Description
endpoint	xsd:anyURI	512	Mandatory	URL of the App for receiving status reports. [Example] http://10.138.38.139:9080/notify
interfaceName	xsd:string	20	Optional	Name of the API for receiving status reports. The value can be customized. [Example] SmsNotification
correlator	xsd:string	50	Mandatory	Correlator ID that associates an SMS message with a status report. When receiving a sendSmsRequest message from the App, the SDP saves the correlator ID. When receiving a status report from the SMSC, the SDP sends the status report and the corresponding correlator ID to the App. The value is a random number defined by partners and must be unique. [Example] 00001

3.2.4 Response

The SDP functions as the server, processes sendSmsRequest messages received from the App, and sends sendSmsResponse messages to the App.

This topic provides a success response example and describes parameters in the response. If a request fails, the SDP sends an error response that contains an error code. For details about error responses, see API Error Responses.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
       <ns1:sendSmsResponse
xmlns:ns1="http://www.csapi.org/schema/parlayx/sms/send/v2_2/local">
        <ns1:result>10000120030111102906571400014l</ns1:result>
      </ns1:sendSmsResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Message Body Parameters

Table 3-6 describes parameters in a sendSmsResponse message body.

Table 3-6 Parameters in a sendSmsResponse message body

Parameter	Type	Length	Level of Requirement	Description
result	xsd:string	30	Mandatory	A string of 30 digits. When the App invokes the getSmsDeliveryStatus API to obtain status reports, the request must contain this parameter value. [Example] 10000120030111102906571400014l

3.2.5 Error Codes

Table 3-7 describes sendSms error codes that the SDP may return upon an exception.

Table 3-7 sendSms error codes

Error Code	Description
SVC0001	Service error. The error code is %1.

Error Code	Description
SVC0002	Invalid input value.
SVC0280	The length exceeds the threshold.
SVC0901	Access authentication fails.
SVC0905	Parameter error.
POL0003	There are too many addresses.
POL0006	The function of sending a message to a group is not supported.
POL0900	The bulk messaging function is not supported.
POL0904	The message sending rate exceeds the threshold.

For details about the error codes, see the *SDP Solution Error Code Reference*.

3.3 notifySmsDeliveryReceipt

3.3.1 Function

The SDP (functioning as the client) invokes the notifySmsDeliveryReceipt API to send status reports to the App (functioning as the server).

The App invokes the 3.2 sendSms API to send SMS messages to users. Requests of the 3.2 sendSms API contain the status report receiving address. After the SDP sends a request to the SMSC, the SMSC sends a status report. The SDP receives the status report and invokes the notifySmsDeliveryReceipt API to send the status report to the App. If the status report fails to be sent, the SDP does not resend it.

Partners must code the App based on the API field requirements so that the App can correctly parse and respond to requests received from the SDP. The App must send a response to the SDP within 30 seconds.

3.3.2 Request URI

The request URI is the destination URI of notifySmsDeliveryReceiptRequest messages sent by the SDP to the App. The URI is defined by the App.

3.3.3 Request

The SDP functions as the client and sends a notifySmsDeliveryReceiptRequest message to the App to send status reports.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header>
```

```
      <ns1:NotifySOAPHeader xmlns:ns1="http://www.huawei.com.cn/schema/common/v2_1">
        <ns1:spRevId>sdp</ns1:spRevId>
        <ns1:spRevpassword>206D88BB7F3D154B130DD6E1E0B8828B</ns1:spRevpassword>
        <ns1:spId>000201</ns1:spId>
        <ns1:serviceId>35000001000001</ns1:serviceId>
        <ns1:timeStamp>111029084631570</ns1:timeStamp>
        <ns1:traceUniqueID>10000120010111062302172100011</ns1:traceUniqueID>
      </ns1:NotifySOAPHeader>
    </soapenv:Header>
    <soapenv:Body>
      <ns2:notifySmsDeliveryReceipt
xmlns:ns2="http://www.csapi.org/schema/parlayx/sms/notification/v2_2/local">
        <ns2:correlator>00001</ns2:correlator>
        <ns2:deliveryStatus>
          <address>tel:8612312345678</address>
          <deliveryStatus>DeliveredToTerminal</deliveryStatus>
        </ns2:deliveryStatus>
      </ns2:notifySmsDeliveryReceipt>
    </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 3-8 describes parameters in a notifySmsDeliveryReceiptRequest message header.

Table 3-8 Parameters in a notifySmsDeliveryReceiptRequest message header

Parameter	Type	Length	Level of Requirement	Description
spRevId	xsd: string	20	Conditional	Reverse authentication ID for the App to authenticate the SDP. The ID is set by SPs during registration. An SP can log in to the SDP management portal and query account information for the ID. • This parameter is mandatory in a request sent to an SP who has configured authentication information during registration. • This parameter can be left empty in a request sent to a Developer, an Enterprise, or an SP who does not configure authentication information. [Example] sdp
spRevpassword	xsd: string	100	Conditional	Reverse authentication key for the App to authenticate the SDP. The key is a character string encrypted from **spRevId** + **Password** + **timeStamp** by MD5. The encryption formula is: spRevPassword = MD5(spRevId + Password + timeStamp). In the

Parameter	Type	Length	Level of Requirement	Description
				formula: • **spRevId** and **timeStamp**: reverse authentication ID and timestamp. • **Password**: access password allocated by an SP to the SDP. An SP can obtain the password from the email notification received after successful registration. – This parameter is mandatory in a request sent to an SP who has configured authentication information during registration. – This parameter can be left empty in a request sent to a Developer, an Enterprise, or an SP who does not configure authentication information. [Example] 206D88BB7F3D154B130DD6E1E0B8828B
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • An SP can log in to the SDP management portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • A Developer can log in to the Developer Portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • An Enterprise must contact the carrier. [Example] 000201
serviceId	xsd: string	21	Conditional	Service ID. The ID is automatically allocated by the SDP to services after successful release. An SP can log in to the SDP management portal and query service information for the ID. This parameter is mandatory in a request sent to an SP and can be left empty in a request sent to a Developer or an Enterprise. [Example]

Parameter	Type	Length	Level of Requirement	Description
				35000001000001
timeStamp	xsd:string	14	Conditional	Timestamp (UTC time). The value is used in MD5 encryption of **spRevpassword**. This parameter is mandatory when the **spRevpassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245
traceUniqueID	xsd:string	30	Mandatory	Transaction ID. This parameter is automatically generated by the SDP and is used only to trace messages during the SDP commissioning. The App ignores this parameter. [Example] 10000120010111062302172100011

Message Body Parameters

Table 3-9 describes parameters in a notifySmsDeliveryReceiptRequest message body.

Table 3-9 Parameters in a notifySmsDeliveryReceiptRequest message body

Parameter	Type	Length	Level of Requirement	Description
correlator	xsd:string	50	Mandatory	Correlator ID that associates an SMS message with a status report. When invoking the sendSms API to send an SMS message, the App defines a correlator ID by the **correlator** parameter and sends it to the SDP in the sendSmsRequest message. When receiving a status report of the SMS message, the SDP sets the correlator ID to the value of the **correlator** parameter in a notifySmsDeliveryReceiptRequest message to send the status report. [Example] 00001

Parameter	Type	Length	Level of Requirement	Description
deliveryStatus	DeliveryInformation	-	Mandatory	Status description. The **deliveryStatus** parameter is of the DeliveryInformation type and contains multiple sub-parameters. For details about the DeliveryInformation type, see Table 3-10.

Table 3-10 describes the parameter structure of the DeliveryInformation type.

Table 3-10 Parameter structure of the DeliveryInformation type

Parameter	Type	Length	Level of Requirement	Description
Address	xsd:anyURI	30	Mandatory	Mobile number of the sender. [Format] tel:*[Prefix][Country code][Mobile number]* In the format, *[Prefix]* and *[Country code]* are optional. The value of *[Prefix]* can be +, **+0**, **+00**, **0**, or **00**. [Example] tel:8612312345678
deliveryStatus	DeliveryStatus	40	Mandatory	Status description. [Enumerated values of **DeliveryStatus**] • **DeliveredToTerminal**: The message has been successfully delivered to the terminal. • **DeliveryImpossible**: The message fails to be delivered because of a network error. • **DeliveryNotificationNotSupported**: The SMSC does not provide the function of sending status reports. The SDP constructs status reports. [Example] DeliveredToTerminal

3.3.4 Response

The App functions as the server, processes the request messages received from the SDP, and sends the response messages to the SDP.

The response is constructed based on the WSDL specification by the partner that provides the App.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/sms/notification/v2_2/local">
   <soapenv:Header/>
   <soapenv:Body>
      <loc:notifySmsDeliveryReceiptResponse/>
   </soapenv:Body>
</soapenv:Envelope>
```

3.3.5 Error Codes

The App returns error codes to the SDP when an exception occurs in response to notifySmsDeliveryReceiptRequest messages. The error codes are defined by partners.

3.4 getSmsDeliveryStatus

3.4.1 Function

The App (functioning as the client) invokes the getSmsDeliveryStatus API to obtain status reports from the SDP (functioning as the server).

The App invokes the 3.2 sendSms API to send SMS messages to users. After the SDP sends a request to the SMSC, the SMSC sends a status report. The SDP receives the status report and saves it for 48 hours. When the App invokes the getSmsDeliveryStatus API, the SDP sends status reports within 48 hours to the App.

Partners must code the App based on the API field requirements so that the App can send correct requests to the SDP. The SDP sends a response within 60 seconds by default.

3.4.2 Request URI

The request URI is the destination URI of getSmsDeliveryStatusRequest messages sent by the App to the SDP. The URI is provided by the SDP in the following format:

http://*IP*:*Port*/SendSmsService/services/SendSms

In the format, *IP* and *Port* indicate the service IP address and Parlay X 2.1 port number of the API provided by the SDP. Contact carriers to obtain the IP address and port number.

3.4.3 Request

The App functions as the client and sends a getSmsDeliveryStatusRequest message to the SDP to obtain status reports.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v2="http://www.huawei.com.cn/schema/common/v2_1"
xmlns:loc="http://www.csapi.org/schema/parlayx/sms/send/v2_2/local">
   <soapenv:Header>
```

```
      <v2:RequestSOAPHeader>
         <v2:spId>000201</v2:spId>
         <v2:spPassword>e6434ef249df55c7a21a0b45758a39bb</v2:spPassword>
         <v2:serviceId>35000001000001</v2:serviceId>
         <v2:timeStamp>201007211126</v2:timeStamp>
      </v2:RequestSOAPHeader>
   </soapenv:Header>
   <soapenv:Body>
      <loc:getSmsDeliveryStatus>

<loc:requestIdentifier>100001200301111029065714000141</loc:requestIdentifier>
      </loc:getSmsDeliveryStatus>
   </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 3-11 describes parameters in a getSmsDeliveryStatusRequest header.

Table 3-11 Parameters in a getSmsDeliveryStatusRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • An SP can log in to the SDP management portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • A Developer can log in to the Developer Portal and query account information, or log in to the mailbox used for registration and view the email notification received after successful registration. • An Enterprise must contact the carrier. [Example] 000201
spPassword	xsd: string	100	Conditional	Authentication key for the SDP to authenticate partners. The value is a character string encrypted from **spId** + **Password** + **timeStamp** by MD5. The encryption formula is: spPassword = MD5(spId + Password + timeStamp). In the formula: • **spId** and **timeStamp**: authentication ID and timestamp.

Parameter	Type	Length	Level of Requirement	Description
				• **Password**: password for partners to access the SDP. – An SP can obtain the password from the email notification received after successful registration. – A Developer can log in to the Developer Portal, choose **Member Center** > **Account** > **Registration Information** > **Invoke Password**, and set the password. – An Enterprise must contact the carrier to obtain the password. The SDP supports authentication by **SP ID + Password**, **SP ID + IP address + Password**, or **SP ID + IP address**. Partners select an authentication mode during registration. If a partner selects authentication by **SP ID + Password** or **SP ID + IP address + Password**, this parameter is mandatory in requests sent by this partner. [Example] e6434ef249df55c7a21a0b45758a39bb
serviceId	xsd: string	21	Conditional	Service ID. The ID is automatically allocated by the SDP to services after successful release. An SP can log in to the SDP management portal and query service information for the ID. This parameter is mandatory in a request sent by an SP and can be left empty in a request sent by a Developer or an Enterprise. [Example] 35000001000001
timeStamp	xsd: string	14	Conditional	Timestamp (UTC time). The value is used in MD5 encryption of **spPassword**. This parameter is mandatory when the **spPassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245

Message Body Parameters

Table 3-12 describes parameters in a getSmsDeliveryStatusRequest message body.

Table 3-12 Parameters in a getSmsDeliveryStatusRequest message body

Parameter	Type	Length	Level of Requirement	Description
registrationIdentifier	xsd:string	30	Mandatory	ID of an SMS message request. The value is a string of 30 digits. The SDP sends the ID to the App by the **result** parameter in the sendSms API response. [Example] 100001200301111029065714000141

3.4.4 Response

The SDP functions as the server, processes getSmsDeliveryStatusRequest messages received from the App, and sends getSmsDeliveryStatusResponse messages to the App.

This topic provides a success response example and describes parameters in the response. If a request fails, the SDP sends an error response that contains an error code. For details about error responses, see API Error Responses.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:getSmsDeliveryStatusResponse
xmlns:ns1="http://www.csapi.org/schema/parlayx/sms/send/v2_2/local">
         <ns1:result>
            <address>tel:8612312345678</address>
            <deliveryStatus>DeliveredToTerminal</deliveryStatus>
         </ns1:result>
      </ns1:getSmsDeliveryStatusResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Message Body Parameters

Table 3-13 describes parameters in a getSmsDeliveryStatusResponse message body.

Table 3-13 Parameters in a getSmsDeliveryStatusResponse message body

Parameter	Type	Length	Level of Requirement	Description
result	DeliveryInformation [0..unbounded]	-	Optional	Status report information. The **deliveryStatus** parameter is of the DeliveryInformation type and contains multiple sub-parameters. For details about the DeliveryInformation type, see Table 3-14.

Table 3-14 describes the parameter structure of the DeliveryInformation type.

Table 3-14 Parameter structure of the DeliveryInformation type

Parameter	Type	Length	Level of Requirement	Description
Address	xsd:anyURI	30	Mandatory	Mobile number of the sender. [Format] tel:*[Prefix][Country code][Mobile number]* In the format, *[Prefix]* and *[Country code]* are optional. The value of *[Prefix]* can be **+**, **+0**, **+00**, **0**, or **00**. [Example] tel:8612312345678
deliveryStatus	DeliveryStatus	40	Mandatory	Status description. [Enumerated values of **DeliveryStatus**] • **DeliveredToTerminal**: The message has been successfully delivered to the terminal. • **MessageWaiting**: The message is waiting to be delivered. • **DeliveryUncertain**: The message is being delivered. • **DeliveredToNetwork**: The message has been delivered, but the terminal has not sent any status reports. • **DeliveryImpossible**: The message fails to be delivered because of a network error. • **DeliveryNotificationNotSupported**: The SMSC does not provide the function of sending status reports. The SDP constructs status reports. [Example]

Parameter	Type	Length	Level of Requirement	Description
				DeliveredToTerminal

3.4.5 Error Codes

Table 3-15 describes getSmsDeliveryStatus error codes that the SDP may return upon an exception. For details about the error codes, see the *SDP Solution Error Code Reference*.

Table 3-15 getSmsDeliveryStatus error codes

Error Code	Description
SVC0001	Service error.
SVC0002	Invalid input value.
SVC0901	Access authentication or authorization error.
SVC0905	Parameter error.
POL0903	The storage duration expires.
POL0904	The message sending rate exceeds the threshold.

4 API Error Responses

4.1 Service Error Response

A service error is caused by service operation exceptions irrelevant to policies. When a service error occurs, the server sends a service error response to the client. This topic provides a service error response example and describes parameters in the response.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>SVC0001</faultcode>
      <faultstring>Waiting for response timed out, message type is
OutwardGetLocReq</faultstring>
      <detail>
        <ns1:ServiceException
xmlns:ns1="http://www.csapi.org/schema/parlayx/common/v2_1">
          <messageId>SVC0001</messageId>
          <text>Waiting for response timed out, message type is OutwardGetLocReq.</text>
          <variables>OutwardGetLocReq</variables>
        </ns1:ServiceException>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Parameter Description

Table 4-1 describes parameters in a service error response.

Table 4-1 Parameters in a service error response

Parameter	Type	Level of Requirement	Description
faultcode	xsd:string	Mandatory	Result code. [Format]

Parameter	Type	Level of Requirement	Description
			SVC*ABCD* In the format, **SVC** identifies a service error response, and *ABCD* is a number ranging from 0001 to 9999. [Example] SVC0001
faultstring	xsd:string	Mandatory	Error description. The value can contain the variable *%#* in definition. When sending a response, the server replaces the variable *%#* with the value of **variables**. [Example] Waiting for response timed out, message type is OutwardGetLocReq.
messageId	xsd:string	Mandatory	The value is the same as that of **faultcode**.
text	xsd:string	Mandatory	The value is the same as that of **faultstring**.
variables	xsd:string [0..unbounded]	Optional	Value of the variable defined in the value of **faultstring**. [Example] OutwardGetLocReq

4.2 Policy Error Response

A policy error is caused by service level agreement (SLA) violation. When a policy error occurs, the server sends a policy error response to the client. This topic provides a policy error response example and describes parameters in the response.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>POL0006</faultcode>
      <faultstring>GroupAddr is not supported</faultstring>
      <detail>
        <ns1:PolicyException
xmlns:ns1="http://www.csapi.org/schema/parlayx/common/v2_1">
          <messageId>POL0006</messageId>
          <text>GroupAddr is not supported</text>
          <variables>GroupAddr</variables>
        </ns1:PolicyException>
```

```
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Parameter Description

Table 4-2 describes parameters in a policy error response.

Table 4-2 Parameters in a policy error response

Parameter	Type	Level of Requirement	Description
faultcode	xsd:string	Mandatory	Result code. [Format] POL*ABCD* In the format, **POL** identifies a policy error response, and *ABCD* is a number ranging from 0001 to 9999. [Example] POL0006
faultstring	xsd:string	Mandatory	Error description. The value can contain the variable *%#* in definition. When sending a response, the server replaces the variable *%#* with the value of **variables**. [Example] GroupAddr is not supported
messageId	xsd:string	Mandatory	The value is the same as that of **faultcode**.
text	xsd:string	Mandatory	The value is the same as that of **faultstring**.
variables	xsd:string [0..unbounded]	Conditional	Value of the variable defined in the value of **faultstring**. [Example] GroupAddr