





Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

Issue 10 (2015-11-20)

This issue is the tenth official release, and includes the following changes:
Password encryption algorithm supports SHA256.

Issue 09 (2015-01-12)

This issue is the ninth official release, and includes the following changes:
Add the parameter **oauth_token** for sendSms.

Issue 08 (2014-05-08)

This issue is the eighth official release, and includes the following changes:
Modify the **Response** chapter.

Issue 07 (2014-05-04)

This issue is the seventh official release, and includes the following changes:
Change the description of **serviceld** and **bundleID**.

Issue 06 (2014-04-17)

This issue is the sixth official release, and includes the following changes:
Modify the RequestSOAPHeader example.

Issue 05 (2013-12-23)

This issue is the fifth official release, and includes the following changes:

Add the fake ID function in Phase2.3 version.

To use the fake ID function, SPs must modify their systems to change all mobile numbers involved in old and new services to fake IDs so that numbers sent by the SP systems to the SDP are all fake IDs. The SDP converts the received fake IDs to mobile numbers for service processing.

SPs must obtain the mapping between mobile numbers involved in old services and fake IDs from the MTN carrier.

If SPs still use mobile numbers when the fake ID function is enabled, service processing will fail.

Issue 04 (2013-09-22)

This issue is the fourth official release, and includes the following changes:
Change the format from the Huawei style to MTN style.

Issue 03 (2013-08-09)

This issue is the third official release, and includes the following changes:
Added API Functions, Level of Requirement for Parameters, Request Format, Response Format, Namespaces in the Overview.

Issue 02 (2013-08-02)

This issue is the second official release, and includes the following changes:
Updated Chapter 3 APIs for Sending SMS Messages for adding **bundleID** field.



Change History

Issue 01 (2013-05-27)

This issue is the first official release.



Contents

1 Overview	1
1.1 API Functions	1
1.2 Level of Requirement for Parameters	3
1.3 Request Format	3
1.4 Response Format	4
1.5 Namespace	5
1.6 SOAPAction	6
2 APIs for Receiving SMS Messages	7
2.1 Process	7
2.2 startSmsNotification	9
2.2.1 Function	9
2.2.2 Request URI	9
2.2.3 Request	9
2.2.4 Response	14
2.2.5 Error Codes	14
2.3 notifySmsReception	16
2.3.1 Function	16
2.3.2 Request URI	16
2.3.3 Request	16
2.3.4 Response	23
2.3.5 Error Codes	23
2.4 stopSmsNotification	23
2.4.1 Function	23
2.4.2 Request URI	23
2.4.3 Request	24
2.4.4 Response	27
2.4.5 Error Codes	27
2.5 getReceivedSms	29
2.5.1 Function	29
2.5.2 Request URI	29
2.5.3 Request	29
2.5.4 Response	33
2.5.5 Error Codes	35
3 APIs for Sending SMS Messages	38
3.1 Process	38
3.2 sendSms	42
3.2.1 Function	42
3.2.2 Request URI	42
3.2.3 Request	42
3.2.4 Response	50
3.2.5 Error Codes	51
3.3 startDeliveryReceiptNotification	54



Contents

3.3.1 API Function	54
3.3.2 Request URI	54
3.3.3 Request	54
3.3.4 Response	58
3.3.5 Error Codes	59
3.4 notifySmsDeliveryReceipt	60
3.4.1 Function	60
3.4.2 Request URI	60
3.4.3 Request	60
3.4.4 Response	65
3.4.5 Error Codes	66
3.5 stopDeliveryReceiptNotification	66
3.5.1 Function	66
3.5.2 Request URI	66
3.5.3 Request	66
3.5.4 Response	69
3.5.5 Error Codes	70
3.6 getSmsDeliveryStatus	72
3.6.1 Function	72
3.6.2 Request URI	72
3.6.3 Request	72
3.6.4 Response	76
3.6.5 Error Codes	77
4 API Error Responses	80
4.1 Service Error Response	80
4.2 Policy Error Response	81

01

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 service Partners, Developers, and API Partners. In this document, partners are mainly the service Partners, Developers, and API Partners 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 the SDP (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).	<ul style="list-style-type: none">startSmsNotificationnotifySmsReceptionstopSmsNotification
	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).	getReceivedSms



1 Overview

Function	Subfunction	Description	API
Sending SMS messages	Sending SMS messages to users	The App (functioning as the client) invokes the sendSms API to send SMS messages through the SDP (functioning as the server).	sendSms
	Receiving status reports in Notify mode	<p>The App (functioning as the client) invokes the startDeliveryReceiptNotification API to subscribe to mobile originated (MO) SMS message status report notification on the SDP (functioning as the server).</p> <p>When receiving an MO SMS message status report from a user, the SDP (functioning as the client) invokes the notifySmsDeliveryReceipt API to send the MO SMS message status report to the App (functioning as the server).</p> <p>Before the App is brought offline, the App (functioning as the client) invokes the stopDeliveryReceiptNotification API to unsubscribe from MO SMS message status report notification on the SDP (functioning as the server).</p>	<ul style="list-style-type: none"> startDeliveryReceiptNotification notifySmsDeliveryReceipt stopDeliveryReceiptNotification
	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).	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



1 Overview

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 is 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 3.0 request in the following format:

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




1 Overview

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 3.0 protocol.

1.4 Response Format

Success Response Format

The SDP provides the Parlay X 3.0 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 3.0 protocol.

Error Response Format

The SDP provides the Parlay X 3.0 API error responses in the following format:

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

    <detail>
      <parameter>...</parameter>
    ...
  </detail>
```



1 Overview

```
</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 API Error Responses.
<detail>	Error details, which are the same as the <soapenv:Fault> element information.

1.5 Namespace

Partners must follow the specified namespaces of data types when developing SMS capability APIs.

- Table 1-6 describes the namespaces of SMS capability APIs.
- The namespace of data types used by the SMS capability APIs is **http://www.csapi.org/schema/parlayx/sms/v3_0**.

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/wsd/parlayx/sms/send/v3_1	<ul style="list-style-type: none"> • sendSms • getSmsDeliveryStatus
http://www.csapi.org/wsd/parlayx/sms/receive/v3_1	<ul style="list-style-type: none"> • sendSms
http://www.csapi.org/wsd/parlayx/sms/notification/v3_1	<ul style="list-style-type: none"> • notifySmsReception • notifySmsDeliveryReceipt
http://www.csapi.org/wsd/parlayx/sms/notification_manager/v3_2	<ul style="list-style-type: none"> • startSmsNotification • stopSmsNotification • startDeliveryReceiptNotification • stopDeliveryReceiptNotification



1.6 SOAPAction

Leave the **SOAPAction** parameter empty.

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

SOAPAction: ""

02

APIs for Receiving SMS Messages

2.1 Process

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.

2 APIs for Receiving SMS Messages

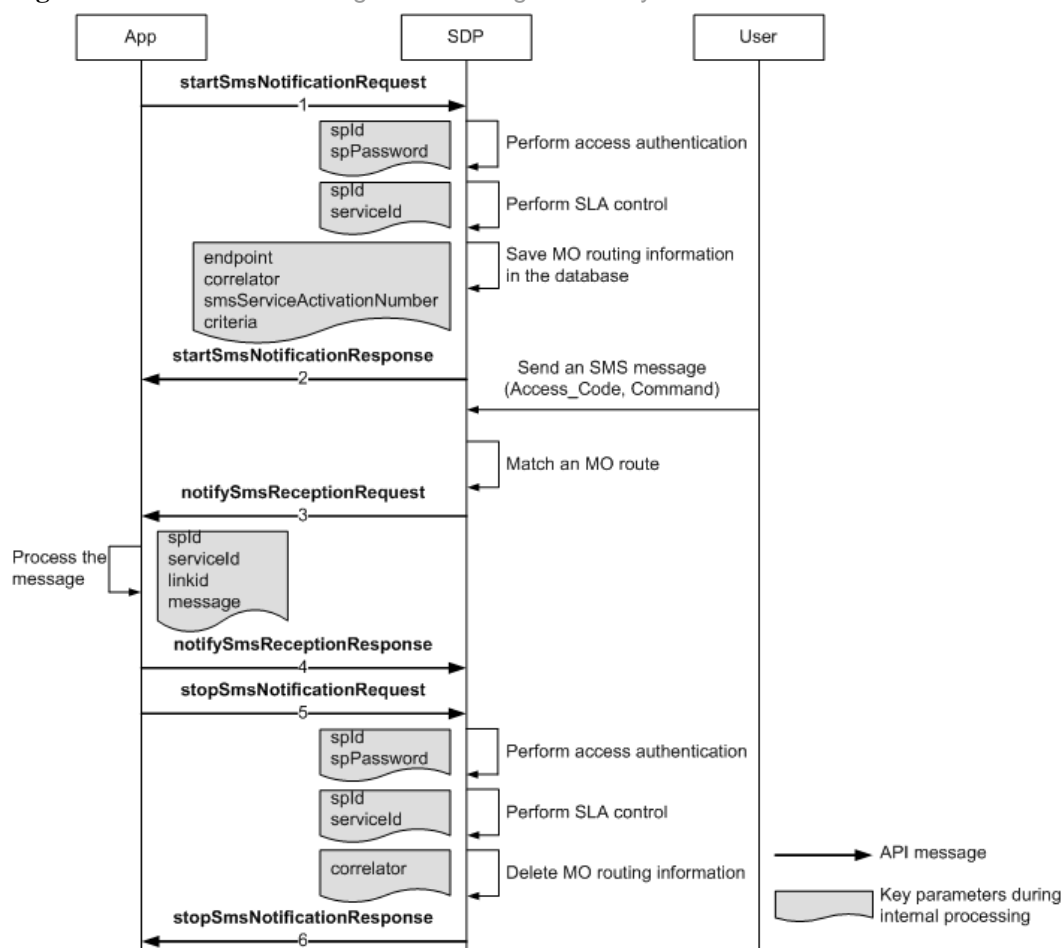
Figure 2-1 Process of receiving SMS messages in Notify mode

Table 2-1 describes the process.

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

Step	Description
1-2	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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. The App parses the notification and sends a response to the SDP.



2 APIs for Receiving SMS Messages

Step	Description
5-6	<ul style="list-style-type: none"> 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.

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 startSmsNotification messages sent by the App to the SDP to enable MO SMS message notification. The URI is provided by the SDP in the following format:

http://IP:Port/SmsNotificationManagerService/services/SmsNotificationManager/v3

In the format, *IP* and *Port* indicate the service IP address and Parlay X 3.0 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 enable MO SMS message notification.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:loc="http://www.csapi.org/schema/parlayx/sms/notification_manager/v3_2/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>
```



2 APIs for Receiving SMS Messages

```
<endpoint>http://10.138.38.139:9080/notify</endpoint>
<interfaceName>notifyMessageReception</interfaceName>
<correlator>12345</correlator>
</loc:reference>
<loc:smsServiceActivationNumber>1111</loc:smsServiceActivationNumber>
<!--Optional:-->
  <loc:criteria>demand</loc:criteria>
</loc:startSmsNotification>
</soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

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

Table 2-2 Parameters in a startSmsNotificationRequest message header

Parameter	Type	Length	Level of Requirement	Description
spld	xsd: string	21	Mandatory	<p>Partner ID.</p> <p>The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID:</p> <ul style="list-style-type: none"> • A service Partner and an API Partner 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. <p>[Example] 000201</p>



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
spPassword	xsd: string	100	Conditional	<p>Authentication key for the SDP to authenticate partners. 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. The value is a character string encrypted. The encryption formula is as follows:</p> <ul style="list-style-type: none"> • SHA-256: spPassword = Base64(SHA-256(spId + Password + timeStamp)) • MD5: spPassword = MD5(spId + Password + timeStamp) <p>In the formula:</p> <ul style="list-style-type: none"> • spId and timeStamp: authentication ID and timestamp. • Password: access password allocated by the SDP to a partner. <ul style="list-style-type: none"> – A service Partner and API Partner 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. <p> NOTE To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks.</p> <p>[Example] e6434ef249df55c7a21a0b45758a39bb</p>



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
serviceld	xsd: string	21	Conditional	<p>Service ID.</p> <p>The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID.</p> <p>The serviceld must be contained during invocation of a service interface developed by service partners and other partners, and must not be contained during invocation of a capability interface developed by API partners, other partners, and developers.</p> <p>[Example] 35000001000001</p>
timeStamp	xsd: string	14	Conditional	<p>Time stamp (UTC time).</p> <p>The value is used in MD5 encryption of spPassword. This parameter is mandatory when the spPassword parameter is required.</p> <p>[Format] yyyyMMddHHmmss</p> <p>[Example] 20100731064245</p>

Message Body Parameters

Table 2-3 describes parameters in a **startSmsNotificationRequest** message body.

Table 2-3 Parameters in a startSmsNotificationRequest message body

Parameter	Type	Length	Level of Requirement	Description
reference	common:SimpleReference	-	Mandatory	<p>Reference.</p> <p>The App sends the App URL, API name, and correlator ID information to the SDP, which then uses the information for the MO SMS message notification.</p> <p>The reference parameter is of the SimpleReference type and contains multiple sub-parameters. For details about the SimpleReference type, see Table 2-4.</p>



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
smsServiceActivationNumber	xsd:anyURI	20	Mandatory	<p>Access code.</p> <p>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:</p> <ul style="list-style-type: none"> • A service Partner can log in to the SDP management portal and query service information. Service Partners can extend access codes allocated by carriers. In an extended access code, the prefix is allocated by carriers and the extension is defined by service Partners. • A Developer or an API Partner must contact the carrier. <p>[Example] 1111</p>
criteria	xsd:string	50	Conditional	<p>Service ordering or subscription command word.</p> <p>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 service Partners during service release. An service Partner can log in to the SDP management portal and query service information for the command word.</p> <p>This parameter is mandatory in a request sent by a service Partner who has configured the command word during service release. This parameter can be left empty in a request sent by a Developer, an API Partner, or a service Partner who does not configure the command word during service release.</p> <p>[Example] demand</p>

Table 2-4 describes the parameter structure of the SimpleReference type.

Table 2-4 Parameter structure of the SimpleReference type

Parameter	Type	Length	Level of Requirement	Description
endpoint	xsd:anyURI	512	Mandatory	<p>Service address to which an SMS message is sent.</p> <p>[Example] http://10.138.38.139:9080/notify</p>



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
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] 12345

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 4 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/receive/v3_1/local"/>
  </soapenv:Body>
</soapenv:Envelope>
```

2.2.5 Error Codes

Table 2-5 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 2-5 startSmsNotification error codes

Error Code	Description	Cause
SVC0002	Criteria %1 is invalid.	The criteria value in the request body is invalid.



2 APIs for Receiving SMS Messages

Error Code	Description	Cause
	Criteria %1 is too long.	The length of the criteria value in the request body exceeds the specified range.
	ServiceActivationNumber is null.	The ServiceActivationNumber value in the request body is blank.
SVC0901	SPID is null!	The SPID value in the request header is blank or does not exist in the SDP.
	SPID %1 is invalid!	The SPID value in the request header is in an incorrect format.
	SPID %1 is not exist!	The SP specified by spid in the request header does not exist in the SDP.
	SP ip is null!	The IP address in the request header is blank.
	Sp ip %1 is not accepted!	The IP address in the request header is incorrect.
	Sp password is null!	The password value in the request header is blank.
	Timestamp is empty in soapheader.	The timeStamp value in the request header is blank.
	local SP password is null!	An internal error occurs in the SDP.
	Sp password is not accepted!	The password value in the request header is incorrect.
	The Sp has not logged in.	The Token value in the request header does not exist in the SDP and the SP needs to log in first.
	SP %1 is in blacklist!	The SP account is in the blacklist.
	The sp's Status is pre-deregistered.	The SP is in the pre-deregistered state.
	The sp's Status is deregistered.	The SP is in the deregistered state.
	SP status is locked.	The SP is in the locked state.
	The sp's Status is deregistered.	The SP is in the forbidden state.
	The sp 's status is pause.	The SP is in the paused state.
	The sp's Status is unknown.	An internal SDP service is abnormal.
	Service ID %1 is invalid!	The serviceID value in the request header is incorrect.



2 APIs for Receiving SMS Messages

Error Code	Description	Cause
	Service ID %1 is not existed!	The servicelD value in the request header does not exist in the SDP.
	Service %1 is in blacklist!	The servicelD value in the request header in the blacklist.
	The service status is configuring.	The service specified by servicelD in the request header is in the configuring state.
	The service status is suspended.	The service specified by servicelD in the request header is in the paused state.
	The service status is pre-deregistered.	The service specified by servicelD in the request header is in the pre-deregistered state.
	The service status is deregistered.	The service specified by servicelD in the request header is in the deregistered state.
	The service status is unknown.	An internal SDP service is abnormal.
	The API %1 is not existed.	This SP does not have the permission for using the API.
	The API status is disabled.	This SP does not have the permission for using the API.

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 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.



2 APIs for Receiving SMS Messages

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>35000001</ns1:spRevId>
      <ns1:spRevpassword>206D88BB7F3D154B130DD6E1E0B8828B</ns1:spRevpassword>
      <ns1:spId>000201</ns1:spId>
      <ns1:serviceId>35000001000001</ns1:serviceId>
      <ns1:timeStamp>20100731064245</ns1:timeStamp>
      <ns1:linkid>12345678901111</ns1:linkid>
      <ns1:traceUniqueID>100001200101110623021721000011</ns1:traceUniqueID>
    </ns1:NotifySOAPHeader>
  </soapenv:Header>
  <soapenv:Body>
    <ns2:notifySmsReception
xmlns:ns2="http://www.csapi.org/schema/parlayx/sms/notification/v3_1/local">
      <ns2:correlator>12345</ns2:correlator>
      <ns2:message>
        <message>Hello world</message>
        <senderAddress>tel:8612312345678</senderAddress>
        <smsServiceActivationNumber>tel:12321</smsServiceActivationNumber>
        <dateTime>2010-08-09T00:00:00.000+08:00</dateTime>
      </ns2:message>
    </ns2:notifySmsReception>
  </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 2-6 describes parameters in a **notifySmsReceptionRequest** message header.



2 APIs for Receiving SMS Messages

Table 2-6 Parameters in a notifySmsReceptionRequest message header

Parameter	Type	Length	Level of Requirement	Description
spRevId	xsd: string	20	Conditional	<p>Reverse authentication ID for the App to authenticate the SDP.</p> <p>The ID is set by service Partners during registration. A service Partner can log in to the SDP management portal and query account information for the ID.</p> <p>This parameter is mandatory in a request sent to a service Partner who has configured authentication information during registration. This parameter can be left empty in a request sent to a Developer, an API Partner, or a service Partner who does not configure authentication information.</p> <p>[Example] 35000001</p>



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
spRevpassword	xsd: string	100	Conditional	<p>Reverse authentication key for the App to authenticate the SDP.</p> <p>The value is a character string encrypted. The encryption formula is as follows:</p> <ul style="list-style-type: none"> • SHA-256: spRevpassword = Base64(SHA-256(spRevId + Password + timeStamp)) • MD5: spRevpassword = MD5(spRevId + Password + timeStamp) <p>In the formula:</p> <ul style="list-style-type: none"> • spRevId and timeStamp: reverse authentication ID and timestamp. • Password: access password allocated by a service Partner to the SDP. A service Partner can obtain the password from the email notification received after successful registration. <p>This parameter is mandatory in a request sent to a service Partner who has configured authentication information during registration.</p> <p>This parameter can be left empty in a request sent to a Developer, an API Partner, or a service Partner who does not configure authentication information.</p> <p> NOTE</p> <p>To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks.</p> <p>[Example] 206D88BB7F3D154B130DD6E1E0B8828B</p>



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
spld	xsd: string	21	Mandatory	<p>Partner ID.</p> <p>The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID:</p> <ul style="list-style-type: none"> A service Partner and API Partner 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. <p>[Example] 000201</p>
serviceld	xsd: string	21	Conditional	<p>Service ID.</p> <p>The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID.</p> <p>The serviceld must be contained during invocation of a service interface developed by service partners and other partners, and must not be contained during invocation of a capability interface developed by API partners, other partners, and developers.</p> <p>[Example]</p> <p>35000001000001</p>
timeStamp	xsd: string	14	Conditional	<p>Timestamp (UTC time).</p> <p>The value is used in MD5 encryption of spRevpassword.</p> <p>This parameter is mandatory when the spRevpassword parameter is required.</p> <p>[Format]</p> <p>yyyyMMddHHmmss</p> <p>[Example] 20100731064245</p>



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
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] 100001200101110623021721000011

Message Body Parameters

Table 2-7 describes parameters in a **notifySmsReceptionRequest** message body.

Table 2-7 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 startSmsNotification API request. [Example] 12345
message	SmsMessage		Mandatory	SMS message information. This parameter contains the SMS message content, sender's mobile number/the fake ID, 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 2-8.

Table 2-8 describes the parameter structure of the SmsMessage type.



2 APIs for Receiving SMS Messages

Table 2-8 Parameter structure of the SmsMessage type

Parameter	Type	Length	Level of Requirement	Description
message	xsd:string	700	Mandatory	<p>SMS message content.</p> <p>[Format]</p> <p>[Command word] [Message body]</p> <p>In the format, [Command word] is optional. Its value can be a service ordering or subscription command word in a request sent to a service Partner, and is left empty in a request sent to a Developer or an API Partner.</p> <p>[Example] Hello world</p>
senderAddress	xsd:anyURI	30	Mandatory	<p>Mobile number or the fake ID of the sender.</p> <p>[Format]</p> <ul style="list-style-type: none"> • Mobile number: tel:[Prefix][Country code][Mobile number] <p>In the format, [Prefix] and [Country code] are optional. The value of [Prefix], if contained, can be +, +0, +00, 0, or 00.</p> <ul style="list-style-type: none"> • Fake ID: tel: [Prefix]- [opcoId]-[sequence]. <p>[Example]</p> <ul style="list-style-type: none"> • Mobile number: tel:8612312345678 • Fake ID: tel:f-245-11900000007639
smsServiceActivationNumber	xsd:anyURI	20	Mandatory	<p>Access code.</p> <p>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:</p> <ul style="list-style-type: none"> • A service Partner can log in to the SDP management portal and query service information. Service Partners can extend access codes allocated by carriers. In an extended access code, the prefix is allocated by carriers and the extension is defined by service Partners. • A Developer or an API Partner must contact the carrier. <p>[Format] tel:Access code</p> <p>[Example] tel:12321</p>



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
dateTime	xsd:dateTime	30	Optional	Date and time (UTC time) when the SDP receives the SMS message. [Format] yyyy-MM-ddTHH:mm:ss.SSSZ [Example] 2010-08-09T00:00:00.000+08:00

2.3.4 Response

The App functions as the server, processes **notifySmsReceptionRequest** messages received from the SDP, and sends **notifySmsReceptionResponse** 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/v3_1/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/v3



2 APIs for Receiving SMS Messages

In the format, *IP* and *Port* indicate the service IP address and Parlay X 3.0 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:loc="http://www.csapi.org/schema/parlayx/sms/notification_manager/v3_2/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:stopSmsNotification>
      <loc:correlator>12345</loc:correlator>
    </loc:stopSmsNotification>
  </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 2-9 describes parameters in a **stopSmsNotificationRequest** message header.

Table 2-9 Parameters in a stopSmsNotificationRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	<p>Partner ID.</p> <p>The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID:</p> <ul style="list-style-type: none"> • A service Partner and API Partner 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. <p>[Example] 000201</p>



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
spPassword	xsd: string	100	Conditional	<p>Authentication key for the SDP to authenticate partners. 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. The value is a character string encrypted. The encryption formula is as follows:</p> <ul style="list-style-type: none"> • SHA-256: spPassword = Base64(SHA-256(spld + Password + timeStamp)) • MD5: spPassword = MD5(spld + Password + timeStamp) <p>In the formula:</p> <ul style="list-style-type: none"> • spld and timeStamp: authentication ID and timestamp. • Password: access password allocated by the SDP to a partner. <ul style="list-style-type: none"> – A service Partner and API Partner 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. <p> NOTE To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks.</p> <p>[Example] e6434ef249df55c7a21a0b45758a39bb</p>



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
serviceld	xsd: string	21	Conditional	<p>Service ID.</p> <p>The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID.</p> <p>The serviceld must be contained during invocation of a service interface developed by service partners and other partners, and must not be contained during invocation of a capability interface developed by API partners, other partners, and developers.</p> <p>[Example] 35000001000001</p>
timeStamp	xsd: string	14	Conditional	<p>Timestamp (UTC time).</p> <p>The value is used in MD5 encryption of spPassword. This parameter is mandatory when the spPassword parameter is required.</p> <p>[Format] yyyyMMddHHmmss</p> <p>[Example] 20100731064245</p>

Message Body Parameters

Table 2-10 describes parameters in a **stopSmsNotificationRequest** message body.



2 APIs for Receiving SMS Messages

Table 2-10 Parameters in a stopSmsNotificationRequest message body

Parameter	Type	Length	Level of Requirement	Description
correlator	xsd:string	50	Mandatory	<p>Correlator ID that associates a startSmsNotificationRequest message with a stopSmsNotificationRequest message.</p> <p>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.</p> <p>The value must be the same as the value of correlator in the startSmsNotificationRequest message body.</p> <p>[Example] 12345</p>

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 4 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/receive/v3_1/local"/>
    </soapenv:Body>
  </soapenv:Envelope>
```

2.4.5 Error Codes

Table 2-11 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 2-11 stopSmsNotification error codes

Error Code	Description	Cause
SVC0001	Deleting SMS notification fail, the record is %1	An internal SDP service is abnormal.



2 APIs for Receiving SMS Messages

Error Code	Description	Cause
SVC0002	Correlator is null.	The Correlator value in the request body is blank.
	Correlator %1 has a invalid format.	The Correlator value in the request body is invalid.
SVC0901	SPID is null!	The SPID value in the request header is blank or does not exist in the SDP.
	SPID %1 is invalid!	The SPID value in the request header is in an incorrect format.
	SPID %1 is not exist!	The SP specified by spId in the request header does not exist in the SDP.
	SP ip is null!	The IP address in the request header is blank.
	Sp ip %1 is not accepted!	The IP address in the request header is incorrect.
	Sp password is null!	The password value in the request header is blank.
	Timestamp is empty in soapheader.	The timeStamp value in the request header is blank.
	local SP password is null!	An internal error occurs in the SDP.
	Sp password is not accepted!	The password value in the request header is incorrect.
	The Sp has not logged in.	The Token value in the request header does not exist in the SDP and the SP needs to log in first.
	SP %1 is in blacklist!	The SP account is in the blacklist.
	The sp's Status is pre-deregistered.	The SP is in the pre-deregistered state.
	The sp's Status is deregistered.	The SP is in the deregistered state.
	SP status is locked.	The SP is in the locked state.
	The sp's Status is deregistered.	The SP is in the forbidden state.
	The sp 's status is pause.	The SP is in the paused state.
	The sp's Status is unknown.	An internal SDP service is abnormal.
	Service ID %1 is invalid!	The serviceID value in the request header is incorrect.



2 APIs for Receiving SMS Messages

Error Code	Description	Cause
	Service ID %1 is not existed!	The servicelD value in the request header does not exist in the SDP.
	Service %1 is in blacklist!	The servicelD value in the request header in the blacklist.
	The service status is configuring.	The service specified by servicelD in the request header is in the configuring state.
	The service status is suspended.	The service specified by servicelD in the request header is in the paused state.
	The service status is pre-deregistered.	The service specified by servicelD in the request header is in the pre-deregistered state.
	The service status is deregistered.	The service specified by servicelD in the request header is in the deregistered state.
	The service status is unknown.	An internal SDP service is abnormal.
	The API %1 is not existed.	This SP does not have the permission for using the API.
	The API status is disabled.	This SP does not have the permission for using the API.

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/v3

In the format, **IP** and **Port** indicate the service IP address and Parlay X 3.0 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.



2 APIs for Receiving SMS Messages

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:loc="http://www.csapi.org/schema/parlayx/sms/receive/v3_1/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:getReceivedSms>
      <loc:registrationIdentifier>1111</loc:registrationIdentifier>
    </loc:getReceivedSms>
  </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 2-12 describes parameters in a **getReceivedSmsRequest** message header.

Table 2-12 Parameters in a getReceivedSmsRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	<p>Partner ID.</p> <p>The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID:</p> <ul style="list-style-type: none"> A service Partner and API Partner 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. <p>[Example] 000201</p>



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
spPassword	xsd: string	100	Conditional	<p>Authentication key for the SDP to authenticate partners.</p> <p>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.</p> <p>The value is a character string encrypted. The encryption formula is as follows:</p> <ul style="list-style-type: none"> • SHA-256: spPassword = Base64(SHA-256(spId + Password + timeStamp)) • MD5: spPassword = MD5(spId + Password + timeStamp) <p>In the formula:</p> <ul style="list-style-type: none"> • spId and timeStamp: authentication ID and timestamp. • Password: access password allocated by the SDP to a partner. <ul style="list-style-type: none"> – A service Partner and API Partner 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. <p> NOTE</p> <p>To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks.</p> <p>[Example] e6434ef249df55c7a21a0b45758a39bb</p>



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
serviceld	xsd: string	21	Conditional	<p>Service ID.</p> <p>The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID.</p> <p>The serviceld must be contained during invocation of a service interface developed by service partners and other partners, and must not be contained during invocation of a capability interface developed by API partners, other partners, and developers.</p> <p>[Example] 35000001000001</p>
timeStamp	xsd: string	14	Conditional	<p>Timestamp (UTC time).</p> <p>The value is used in MD5 encryption of spPassword.</p> <p>This parameter is mandatory when the spPassword parameter is required.</p> <p>[Format] yyyyMMddHHmmss</p> <p>[Example] 20100731064245</p>

Message Body Parameters

Table 2-13 describes parameters in a **getReceivedSmsRequest** message body.



2 APIs for Receiving SMS Messages

Table 2-13 Parameters in a `getReceivedSmsRequest` message body

Parameter	Type	Length	Level of Requirement	Description
registrationIdentifier	xsd:string	20	Mandatory	<p>Access code.</p> <p>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:</p> <ul style="list-style-type: none"> A service Partner can log in to the SDP management portal and query service information. Service Partners can extend access codes allocated by carriers. In an extended access code, the prefix is allocated by carriers and the extension is defined by service Partners. A Developer or an API Partner must contact the carrier. <p>[Example] 1111</p>

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 4 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/v3_1/local">
      <ns1:result>
        <message>Hello world</message>
        <senderAddress>tel:8612312345678</senderAddress>
        <smsServiceActivationNumber>tel:1111</smsServiceActivationNumber>
        <dateTime>2010-08-09T00:00:00.000+08:00</dateTime>
      </ns1:result>
    </ns1:getReceivedSmsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```



2 APIs for Receiving SMS Messages

Message Body Parameters

Table 2-14 describes parameters in a **getReceivedSmsResponse** message body.

Table 2-14 Parameters in a getReceivedSmsResponse message body

Parameter	Type	Length	Level of Requirement	Description
result	SmsMessage[0..unbounded]		Optional	<p>SMS message content.</p> <p>This parameter contains the SMS message content, sender's mobile number/the fake ID, access code, and time information.</p> <p>The result parameter is of the SmsMessage type and contains multiple sub-parameters. For details about the SmsMessage type, see Table 2-15.</p>

Table 2-15 describes the parameter structure of the SmsMessage type.

Table 2-15 Parameter structure of the SmsMessage type

Parameter	Type	Length	Level of Requirement	Description
message	xsd:string	700	Mandatory	<p>SMS message content.</p> <p>[Example] Hello world</p>
senderAddress	xsd:anyURI	30	Mandatory	<p>Mobile number or the fake ID of the sender.</p> <p>[Format]</p> <ul style="list-style-type: none"> Mobile number: tel:[Prefix][Country code][Mobile number] In the format, <i>[Prefix]</i> and <i>[Country code]</i> are optional. The value of <i>[Prefix]</i>, if contained, can be +, +0, +00, 0, or 00. Fake ID: tel: [Prefix]- [opcoid]-[sequence]. <p>[Example]</p> <ul style="list-style-type: none"> Mobile number: tel:8612312345678 Fake ID: tel:f-245-11900000007639



2 APIs for Receiving SMS Messages

Parameter	Type	Length	Level of Requirement	Description
smsServiceActivationNumber	xsd:anyURI	20	Mandatory	<p>Access code.</p> <p>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:</p> <ul style="list-style-type: none"> A service Partner can log in to the SDP management portal and query service information. Service Partners can extend access codes allocated by carriers. In an extended access code, the prefix is allocated by carriers and the extension is defined by service Partners. A Developer or an API Partner must contact the carrier. <p>[Format] tel:Access code [Example] tel:1111</p>
dateTime	xsd:dateTime	30	Optional	<p>Date and time when the SDP receives the SMS message.</p> <p>[Format] yyyy-MM-ddTHH:mm:ss.SSSZ [Example] 2010-08-09T00:00:00.000+08:00</p>

2.5.5 Error Codes

Table 2-16 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 2-16 getReceivedSms error codes

Error Code	Description	Cause
SVC0901	SPID is null!	The SPID value in the request header is blank or does not exist in the SDP.
	SPID %1 is invalid!	The SPID value in the request header is in an incorrect format.
	SPID %1 is not exist!	The SP specified by spid in the request header does not exist in the SDP.
	SP ip is null!	The IP address in the request header is blank.
	Sp ip %1 is not accepted!	The IP address in the request header is incorrect.



2 APIs for Receiving SMS Messages

Error Code	Description	Cause
	Sp password is null!	The password value in the request header is blank.
	Timestamp is empty in soapheader.	The timeStamp value in the request header is blank.
	local SP password is null!	An internal error occurs in the SDP.
	Sp password is not accepted!	The password value in the request header is incorrect.
	The Sp has not logged in.	The Token value in the request header does not exist in the SDP and the SP needs to log in first.
	SP %1 is in blacklist!	The SP account is in the blacklist.
	The sp's Status is pre-deregistered.	The SP is in the pre-deregistered state.
	The sp's Status is deregistered.	The SP is in the deregistered state.
	SP status is locked.	The SP is in the locked state.
	The sp's Status is deregistered.	The SP is in the forbidden state.
	The sp 's status is pause.	The SP is in the paused state.
	The sp's Status is unknown.	An internal SDP service is abnormal.
	Service ID %1 is invalid!	The serviceld value in the request header is incorrect.
	Service ID %1 is not existed!	The serviceld value in the request header does not exist in the SDP.
	Service %1 is in blacklist!	The serviceld value in the request header in the blacklist.
	The service status is configuring.	The service specified by serviceld in the request header is in the configuring state.
	The service status is suspended.	The service specified by serviceld in the request header is in the paused state.
	The service status is pre-deregistered.	The service specified by serviceld in the request header is in the pre-deregistered state.
	The service status is deregistered.	The service specified by serviceld in the request header is in the deregistered state.
	The service status is unknown.	An internal SDP service is abnormal.



2 APIs for Receiving SMS Messages

Error Code	Description	Cause
	The API %1 is not existed.	This SP does not have the permission for using the API.
	The API status is disabled.	This SP does not have the permission for using the API.

03

APIs for Sending SMS Messages

3.1 Process

A third party uses the SDP to deliver an SMS message and requires a status report. The third party can receive the status report from the SDP in the Notify or Get mode.

- Notify mode: The third party uses the SDP to automatically receive the status report from the SMSC.
- Get mode: The third party invokes the SDP interface to obtain the status report. The SDP stores a status report for 48 hours by default.

The Notify mode is different from the Get mode in the following aspects:

- When the Notify mode is used, the third party set the notification address of upstream SMS messages status report in the following situations:
 - The SP invokes the **sendSms** interface of the SDP, the message body of the **sendSms** request contains the **receiptRequest** field.
 - The SP invokes the **startDeliveryReceiptNotification** interface of the SDP to enable the status report notification function.
- When the Get mode is used, the third party must invoke the **getSmsDeliveryStatus** interface to obtain the status report in 48 hours after invoking the **sendSms** interface. After 48 hours, the SDP deletes the outdated status report. If the third party requires a longer status report storage duration in the SDP, contact and sign the SLA with the carrier.

Notify Mode

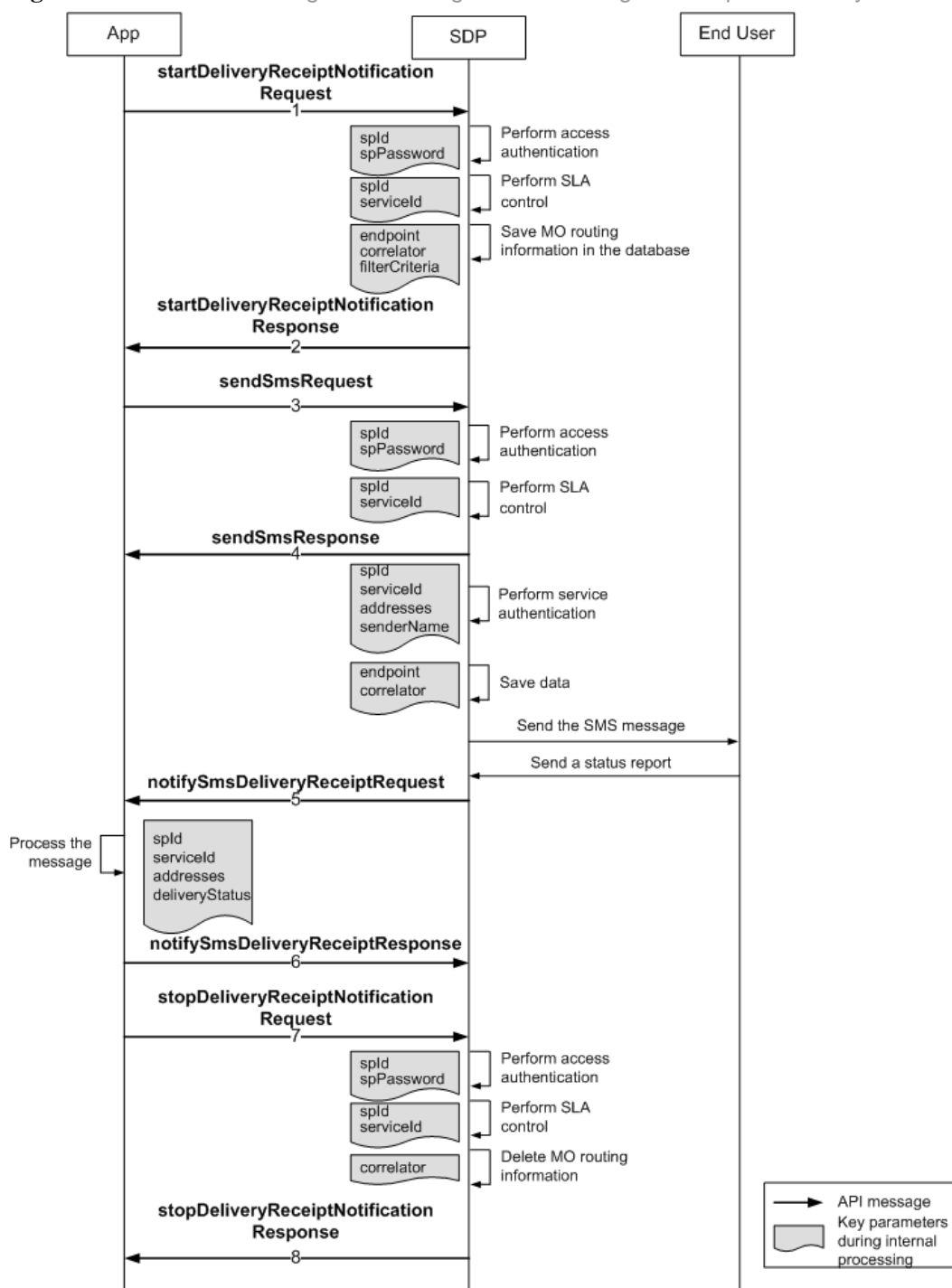
The process of the App receiving SMS messages status report in Notify mode consists of the following main steps:

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

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



3 APIs for Sending SMS Messages

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



3 APIs for Sending SMS Messages

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	<ul style="list-style-type: none">• The App sends a request to the SDP to subscribe to MO SMS message status report 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	<ul style="list-style-type: none">• 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.
5–6	<ul style="list-style-type: none">• The SDP receives a status report and sends a notification of the status report to the App.• The App parses the notification and sends a response to the SDP.
7–8	<ul style="list-style-type: none">• The App sends a request to the SDP to unsubscribe from MO SMS message notification status report 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 3-2 shows the process of the App sending SMS messages and receiving status reports in Get mode.

3 APIs for Sending SMS Messages

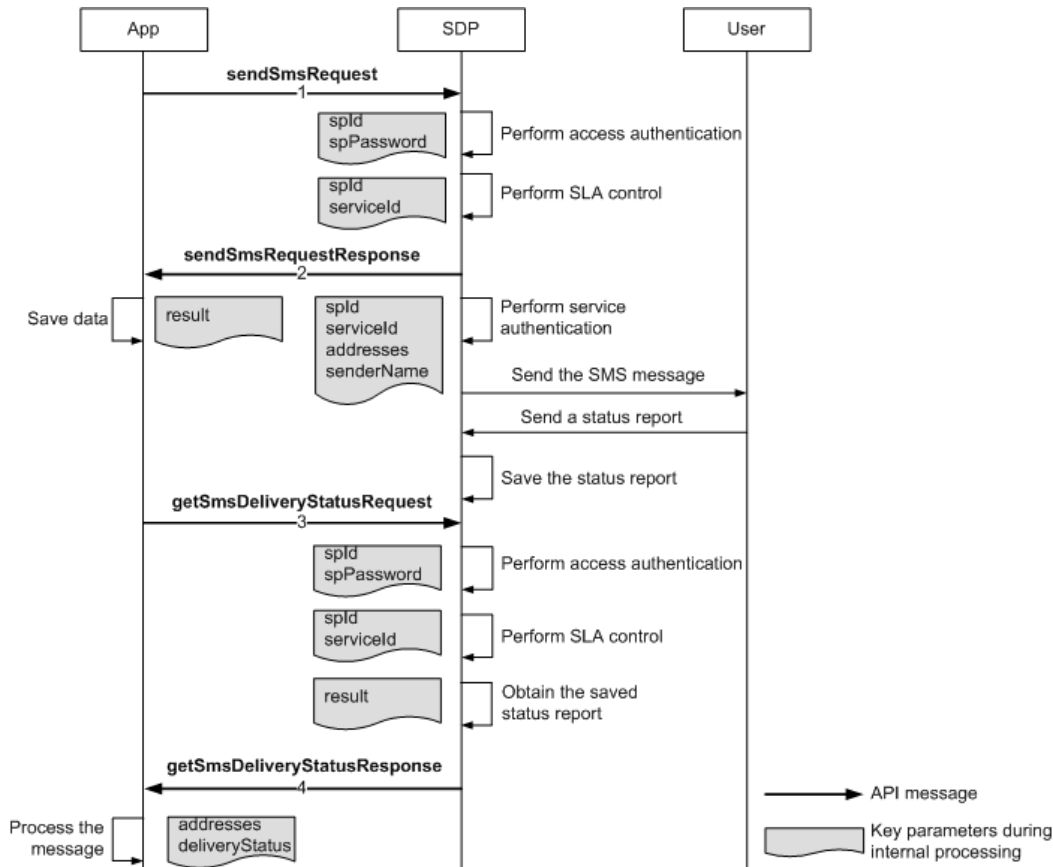
Figure 3-2 Process of sending SMS messages and receiving status reports in Get mode

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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> The SDP receives a status report from the user and saves the report for a 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 APIs for Sending SMS Messages

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/v3

In the format, **IP** and **Port** indicate the service IP address and Parlay X 3.0 port number of the API provided by the SDP. Contact carriers to obtain the IP address and port number.

3.2.3 Request

Example

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

- When a service Partner delivers an on-demand service message and obtains the status report in the Get mode:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:loc="http://www.csapi.org/schema/parlayx/sms/send/v3_1/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>
      <OA>8612312345678</OA>
      <FA>8612312345678</FA>
      <linkid>12345678901111</linkid>
    </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:sendSms>
  </soapenv:Body>
</soapenv:Envelope>
```

- When a service Partner delivers a gift service message and obtains the status report in the Notify mode:



3 APIs for Sending SMS Messages

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/sms/send/v3_1/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>
      <OA>8612312345678</OA>
      <FA>8612312345678</FA>
      <presentid>22345678901113</presentid>
    </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>12345</correlator>
      </loc:receiptRequest>
    </loc:sendSms>
  </soapenv:Body>
</soapenv:Envelope>
```

- When an API partner user or a developer sends an SMS message and obtains the status report in the Get mode:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/sms/send/v3_1/local">
  <soapenv:Header>
    <RequestSOAPHeader xmlns="http://www.huawei.com.cn/schema/common/v2_1">
      <RequestSOAPHeader>
        <spId>000201</spId>
        <spPassword>e6434ef249df55c7a21a0b45758a39bb</spPassword>
        <bundleID>256000039</bundleID>
        <timestamp>20100731064245</timestamp>
        <OA>8612312345678</OA>
        <FA>8612312345678</FA>
      </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:sendSms>
    </soapenv:Body>
  </soapenv:Envelope>
```




3 APIs for Sending SMS Messages

```
</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	<p>Partner ID.</p> <p>The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID:</p> <ul style="list-style-type: none">• A service Partner and an API Partner 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. <p>[Example]</p> <p>000201</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
spPassword	xsd: string	100	Conditional	<p>Authentication key for the SDP to authenticate partners.</p> <p>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.</p> <p>The value is a character string encrypted. The encryption formula is as follows:</p> <ul style="list-style-type: none"> • SHA-256: spPassword = Base64(SHA-256(spId + Password + timeStamp)) • MD5: spPassword = MD5(spId + Password + timeStamp) <p>In the formula:</p> <ul style="list-style-type: none"> • spId and timeStamp: authentication ID and timestamp. • Password: access password allocated by the SDP to a partner. <ul style="list-style-type: none"> – A service Partner and API Partner 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. <p> NOTE</p> <p>To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks.</p> <p>[Example] e6434ef249df55c7a21a0b45758a39bb</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
serviceld	xsd: string	21	Conditional	<p>Service ID.</p> <p>The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID.</p> <p>The serviceld must be contained during invocation of a service interface developed by service partners and other partners, and must not be contained during invocation of a capability interface developed by API partners, other partners, and developers.</p> <p>[Example] 35000001000001</p>
bundleID	xsd: string	21	Conditional	<p>Bundle ID.</p> <p>When SDP creates a capability bundle, SDP allocates a bundleID to capability bundle.</p> <p>The bundleID must not be contained during invocation of a service interface developed by service partners and other partners, and must be contained during invocation of a capability interface developed by API partners, other partners, and developers.</p> <p>[Example] 256000039</p>
timeStamp	xsd: string	14	Conditional	<p>Timestamp (UTC time).</p> <p>The value is used in MD5 encryption of spPassword.</p> <p>This parameter is mandatory when the spPassword parameter is required.</p> <p>[Format] yyyyMMddHHmmss</p> <p>[Example] 20100731064245</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
OA	xsd:string	30	Conditional	<p>Mobile number or the fake ID of the service originator.</p> <p>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.</p> <ul style="list-style-type: none"> In a service Partner'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 a service Partner's request for sending an SMS message to a gift recipient, the value is the mobile number of the gift sender. In an API Partner's or a Developer's request, the value is the mobile number of the message recipient. <p>[Example]</p> <ul style="list-style-type: none"> Mobile number: 8612312345678 Fake ID: f-245-11900000007639
FA	xsd:string	30	Conditional	<p>Mobile number or the fake ID of the charged party. The value must be the same as the value of OA.</p>
linkid	xsd:string	20	Conditional	<p>Service order ID.</p> <p>The ID is automatically generated by the SDP when a user orders a service in the SDP.</p> <p>This parameter is mandatory during on-demand service delivery by SMS message.</p> <p>The SDP sends the value to service Partners as follows in different scenarios:</p> <ul style="list-style-type: none"> 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. <p>[Example]</p> <p>12345678901111</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
presentid	xsd:string	15	Conditional	<p>Service gift ID.</p> <p>The ID is automatically generated by the SDP when a user sends a service to another user as a gift on the SDP.</p> <p>This parameter is mandatory in an service Partner's request for sending an SMS message to a gift recipient, and can be left empty in an API Partner's or a Developer's request.</p> <p>The SDP invokes the assignPresentToUser API to send the value to service Partners.</p> <p>[Example]</p> <p>22345678901113</p>
oauth_token	xsd:string	20	Conditional	<p>Token authentication code synchronized by the MDSP to the SP if the SP has obtained the user consent.</p> <p>It must present if user confirmation is required.</p>

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	<p>Mobile number or the fake ID of the message recipient.</p> <p>[Format]</p> <ul style="list-style-type: none"> Mobile number: tel:[Prefix][Country code][Mobile number] <p>In the format, <i>[Prefix]</i> is optional. The value of <i>[Prefix]</i>, if contained, can be +, +0, +00, 0, or 00.</p> <ul style="list-style-type: none"> Fake ID: tel: [Prefix]-[opcoId]-[sequence]. <p>[Example]</p> <ul style="list-style-type: none"> Mobile number: tel:8612312345678 Fake ID: tel:f-245-11900000007639



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
senderName	xsd:string	20	Mandatory	<p>Name of the message sender, which is displayed on users' terminals.</p> <ul style="list-style-type: none"> In a service Partner's request, the value is a service access code obtained from carriers before service release. In an API Partner's or a Developer's request, the value is an access code allocated by carriers during capability product purchase. <p>[Example] 321123</p>
message	xsd:string	700	Mandatory	<p>Content of the SMS message to be sent.</p> <p>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.</p> <p>[Example] Hello World.</p>
receiptRequest	common:SimpleReference		Conditional	<p>Information required for status report notification.</p> <p>The App sends the App URL, API name, and correlator ID information to the SDP, which then uses the information for status report notification.</p> <p>The receiptRequest parameter is of the SimpleReference type and contains multiple sub-parameters. For details about the SimpleReference type, see Table 3-5.</p>

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	<p>URL of the App for receiving status reports.</p> <p>[Example] http://10.138.38.139:9080/notify</p>



3 APIs for Sending SMS Messages

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] 12345

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 4 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/v3_1/local">
      <ns1:result>100001200301111029065714000141</ns1:result>
    </ns1:sendSmsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Message Body Parameters

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



3 APIs for Sending SMS Messages

Table 3-6 Parameters in a sendSmsResponse message body

Parameter	Type	Length	Level of Requirement	Description
result	xsd:string	30	Mandatory	<p>A string of 30 digits.</p> <p>When the App invokes the getSmsDeliveryStatus API to obtain status reports, the request must contain this parameter value.</p> <p>[Example]</p> <p>100001200301111029065714000141</p>

3.2.5 Error Codes

Table 3-7 describes sendSms error codes that the SDP may return upon an exception. For details about the error codes, see the *SDP Solution Error Code Reference*.

Table 3-7 sendSms error codes

Error Code	Description	Cause
SVC0001	Sending message fail, message ID is %1, reason is %2	An internal SDP service is abnormal.
	The message has been licence controled.	The SDP resource usage is too high.
SVC0002	Destination Addresses length is 0.	The addresses value in the request body is blank.
	address %1 is invalid.	The mobile numbers or the fake ID in the OA , FA , and addresses values of the request body are in an incorrect format.
	Some destination address in addresses is null.	The addresses value in the request body is blank. Example: <loc:addresses></loc:addresses>
	%1 is unknown destination address	The mobile numbers in the OA , FA , and addresses values of the request body are incorrect.
	SimpleReference is null.	The receiptRequest value in the request body is blank.
	Endpoint is invalid in SimpleReference.	The endpoint value in the request body is invalid.
	Endpoint is null in SimpleReference.	The endpoint value in the request body is blank.



3 APIs for Sending SMS Messages

Error Code	Description	Cause
	Correlator is null in SimpleReference.	The correlator value in the request body is blank.
	Correlator %1 has a invalid format.	The correlator value in the request body is in an incorrect format.
	Message contains sensitive word %1.	Message contains sensitive word.
	Interface name is null in SimpleReference.	The interfacename value in the request body is blank.
SVC0280	SMS max length.	The size of the SMS message in the request body exceeds the threshold.
SVC0901	SPID is null!	The SPID value in the request header is blank or does not exist in the SDP.
	SPID %1 is invalid!	The SPID value in the request header is in an incorrect format.
	SPID %1 is not exist!	The SP specified by spid in the request header does not exist in the SDP.
	SP ip is null!	The IP address in the request header is blank.
	Sp ip %1 is not accepted!	The IP address in the request header is incorrect.
	Sp password is null!	The password value in the request header is blank.
	Timestamp is empty in soapheader.	The timeStamp value in the request header is blank.
	local SP password is null!	An internal error occurs in the SDP.
	Sp password is not accepted!	The password value in the request header is incorrect.
	The Sp has not logged in.	The Token value in the request header does not exist in the SDP and the SP needs to log in first.
	SP %1 is in blacklist!	The SP account is in the blacklist.
	The sp's Status is pre-deregistered.	The SP is in the pre-deregistered state.
	The sp's Status is deregistered.	The SP is in the deregistered state.
	SP status is locked.	The SP is in the locked state.
	The sp's Status is deregistered.	The SP is in the forbidden state.



3 APIs for Sending SMS Messages

Error Code	Description	Cause
	The sp 's status is pause.	The SP is in the paused state.
	The sp's Status is unknown.	An internal SDP service is abnormal.
	Service ID %1 is invalid!	The serviceld value in the request header is incorrect.
	Service ID %1 is not existed!	The serviceld value in the request header does not exist in the SDP.
	Service %1 is in blacklist!	The serviceld value in the request header in the blacklist.
	The service status is configuring.	The service specified by serviceld in the request header is in the configuring state.
	The service status is suspended.	The service specified by serviceld in the request header is in the paused state.
	The service status is pre-deregistered.	The service specified by serviceld in the request header is in the pre-deregistered state.
	The service status is deregistered.	The service specified by serviceld in the request header is in the deregistered state.
	The service status is unknown.	An internal SDP service is abnormal.
	The API %1 is not existed.	This SP does not have the permission for using the API.
	The API status is disabled.	This SP does not have the permission for using the API.
SVC0905	OA is null !	The OA value in the request header is blank.
	OA %1 is invaild!	The OA value in the request header is invalid.
	OA %1 is unknown!	The OA value in the request header is invalid.
	FA is null !	The FA value in the request header is blank.
	FA %1 is invaild!	The FA value in the request header is invalid.
	FA %1 is unknown!	The FA value in the request header is invalid.
POL0003	MaxDestAddr exceed.	The number of mobile numbers specified by addresses in the request body exceeds the threshold in the SLA.
POL0904	SP level gross control not pass.	The number of requests delivered by the SP exceeds the signed TPS.



3.3 startDeliveryReceiptNotification

3.3.1 API Function

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

The startDeliveryReceiptNotification API sends the routing information for the App to receive MO SMS message status report notifications to the SDP. When receiving the API request, the SDP saves the MO routing information of the App. After MO SMS message status report notification is subscribed, the SDP sends MO SMS messages status reports 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.

3.3.2 Request URI

The request URI is the destination URI of startDeliveryReceiptNotification messages sent by the App to the SDP to enable MO SMS message status report notification. The URI is provided by the SDP in the following format:

http://IP:Port/SmsNotificationManagerService/services/SmsNotificationManager/v3

In the format, *IP* and *Port* indicate the service IP address and Parlay X 3.0 port number of the API provided by the SDP. Contact carriers to obtain the IP address and port number.

3.3.3 Request

The App functions as the client and sends a **startDeliveryReceiptNotificationRequest** message to the SDP to enable MO SMS message status report notification.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:loc="http://www.csapi.org/schema/parlayx/sms/notification_manager/v3_2/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:startDeliveryReceiptNotification>
      <loc:reference>
        <endpoint>http://10.138.38.139:9080/notify</endpoint>
        <interfaceName>notifyDeliveryReceiptReception</interfaceName>
        <correlator>12345</correlator>
      </loc:reference>
    </loc:startDeliveryReceiptNotification>
  </soapenv:Body>
</soapenv:Envelope>
```



3 APIs for Sending SMS Messages

```
<loc:filterCriteria>report</loc:filterCriteria>
</loc:startDeliveryReceiptNotification>
</soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 3-8 describes parameters in a **startDeliveryReceiptNotificationRequest** message header.

Table 3-8 Parameters in a startDeliveryReceiptNotificationRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	<p>Partner ID.</p> <p>The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID:</p> <ul style="list-style-type: none"> • A service Partner and API Partner 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. <p>[Example] 000201</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
spPassword	xsd: string	100	Conditional	<p>Authentication key for the SDP to authenticate partners.</p> <p>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.</p> <p>The value is a character string encrypted. The encryption formula is as follows:</p> <ul style="list-style-type: none"> • SHA-256: spPassword = Base64(SHA-256(spId + Password + timeStamp)) • MD5: spPassword = MD5(spId + Password + timeStamp) <p>In the formula:</p> <ul style="list-style-type: none"> • spId and timeStamp: authentication ID and timestamp. • Password: access password allocated by the SDP to a partner. <ul style="list-style-type: none"> – A service Partner and API Partner 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. <p> NOTE</p> <p>To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks.</p> <p>[Example] e6434ef249df55c7a21a0b45758a39bb</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
serviceld	xsd: string	21	Conditional	<p>Service ID.</p> <p>The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID.</p> <p>The serviceld must be contained during invocation of a service interface developed by service partners and other partners, and must not be contained during invocation of a capability interface developed by API partners, other partners, and developers.</p> <p>[Example]</p> <p>35000001000001</p>
timeStamp	xsd: string	14	Conditional	<p>Time stamp (UTC time).</p> <p>The value is used in MD5 encryption of spPassword.</p> <p>This parameter is mandatory when the spPassword parameter is required.</p> <p>[Format] yyyyMMddHHmmss</p> <p>[Example] 20100731064245</p>

Message Body Parameters

Table 3-9 describes parameters in a **startDeliveryReceiptNotificationRequest** message body.

Table 3-9 Parameters in a startDeliveryReceiptNotificationRequest message body

Parameter	Type	Length	Level of Requirement	Description
reference	common:SimpleReference	-	Mandatory	<p>Reference.</p> <p>For the detailed reference information, see Table 3-10.</p>
filterCriteria	xsd:string	50	Conditional	<p>Filtering keyword.</p> <p>After the setting, the SDP uses the access code and command word to match an MO route and sends the status reports to the SP.</p> <p>[Example] report</p>

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



3 APIs for Sending SMS Messages

Table 3-10 Parameter structure of the SimpleReference type

Parameter	Type	Length	Level of Requirement	Description
endpoint	xsd:anyURI	512	Mandatory	Notification address of an MO SMS message status report. The value is the address for receiving status reports. [Example] http://10.138.38.139:9080/notify
interfaceName	xsd:string	20	Conditional	Name of the notification interface. The value is defined by the interface invoker. This parameter can be left blank. [Example] notifyDeliveryReceipt
correlator	xsd:string	50	Mandatory	ID that associates the startDeliveryReceiptNotificationRequest message with a stopDeliveryReceiptNotification message. When a third party sends a stopDeliveryReceiptNotification to the SDP, the SDP stops the corresponding MO SMS status report notification based on this parameter. The value is a random number defined by a third party and must be unique. [Example] 12345

3.3.4 Response

The SDP functions as the server, processes **startDeliveryReceiptNotificationRequest** messages received from the App, and sends **startDeliveryReceiptNotificationResponse** 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 4 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:startDeliveryReceiptNotificationResponse
xmlns:ns1="http://www.csapi.org/schema/parlayx/sms/receive/v3_1/local"/>
  </soapenv:Body>
</soapenv:Envelope>
```



3 APIs for Sending SMS Messages

3.3.5 Error Codes

Table 3-11 describes startDeliveryReceiptNotification error codes that the SDP may return upon an exception. For details about the error codes, see the *SDP Solution Error Code Reference*.

Table 3-11 startDeliveryReceiptNotification error codes

Error Code	Description	Cause
SVC0901	SPID is null!	The SPID value in the request header is blank or does not exist in the SDP.
	SPID %1 is invalid!	The SPID value in the request header is in an incorrect format.
	SPID %1 is not exist!	The SP specified by spId in the request header does not exist in the SDP.
	SP ip is null!	The IP address in the request header is blank.
	Sp ip %1 is not accepted!	The IP address in the request header is incorrect.
	Sp password is null!	The password value in the request header is blank.
	Timestamp is empty in soapheader.	The timeStamp value in the request header is blank.
	local SP password is null!	An internal error occurs in the SDP.
	Sp password is not accepted!	The password value in the request header is incorrect.
	The Sp has not logged in.	The Token value in the request header does not exist in the SDP and the SP needs to log in first.
	SP %1 is in blacklist!	The SP account is in the blacklist.
	The sp's Status is pre-deregistered.	The SP is in the pre-deregistered state.
	The sp's Status is deregistered.	The SP is in the deregistered state.
	SP status is locked.	The SP is in the locked state.
	The sp's Status is deregistered.	The SP is in the forbidden state.
	The sp 's status is pause.	The SP is in the paused state.
	The sp's Status is unknown.	An internal SDP service is abnormal.
	Service ID %1 is invalid!	The serviceID value in the request header is incorrect.



3 APIs for Sending SMS Messages

Error Code	Description	Cause
	Service ID %1 is not existed!	The servicelD value in the request header does not exist in the SDP.
	Service %1 is in blacklist!	The servicelD value in the request header in the blacklist.
	The service status is configuring.	The service specified by servicelD in the request header is in the configuring state.
	The service status is suspended.	The service specified by servicelD in the request header is in the paused state.
	The service status is pre-deregistered.	The service specified by servicelD in the request header is in the pre-deregistered state.
	The service status is deregistered.	The service specified by servicelD in the request header is in the deregistered state.
	The service status is unknown.	An internal SDP service is abnormal.
	The API %1 is not existed.	This SP does not have the permission for using the API.
	The API status is disabled.	This SP does not have the permission for using the API.

3.4 notifySmsDeliveryReceipt

3.4.1 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 sendSms API to send SMS messages to users. Requests of the 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.

3.4.2 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.

3.4.3 Request

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



3 APIs for Sending SMS Messages

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>35000001</ns1:spRevId>
      <ns1:spRevpassword>206D88BB7F3D154B130DD6E1E0B8828B</ns1:spRevpassword>
      <ns1:spId>000201</ns1:spId>
      <ns1:serviceId>35000001000001</ns1:serviceId>
      <ns1:timeStamp>20100731064245</ns1:timeStamp>
      <ns1:traceUniqueID>100001200101110623021721000011</ns1:traceUniqueID>
    </ns1:NotifySOAPHeader>
  </soapenv:Header>
  <soapenv:Body>
    <ns2:notifySmsDeliveryReceipt
      xmlns:ns2="http://www.csapi.org/schema/parlayx/sms/notification/v3_1/local">
      <ns2:correlator>12345</ns2:correlator>
      <ns2:deliveryStatus>
        <address>tel:8612312345678</address>
        <deliveryStatus>DeliveredToTerminal</deliveryStatus>
      </ns2:deliveryStatus>
    </ns2:notifySmsDeliveryReceipt>
  </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 3-12 describes parameters in a **notifySmsDeliveryReceiptRequest** message header.

Table 3-12 Parameters in a notifySmsDeliveryReceiptRequest message header

Parameter	Type	Length	Level of Requirement	Description
spRevId	xsd: string	20	Conditional	<p>Reverse authentication ID for the App to authenticate the SDP.</p> <p>The ID is set by service Partners during registration. A service Partner can log in to the SDP management portal and query account information for the ID.</p> <ul style="list-style-type: none"> This parameter is mandatory in a request sent to a service Partner who has configured authentication information during registration. This parameter can be left empty in a request sent to a Developer, an API Partner, or a service Partner who does not configure authentication information. <p>[Example] 35000001</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
spRevpassword	xsd: string	100	Conditional	<p>Reverse authentication key for the App to authenticate the SDP.</p> <p>The value is a character string encrypted. The encryption formula is as follows:</p> <ul style="list-style-type: none"> • SHA-256: spRevpassword = Base64(SHA-256(spRevId + Password + timeStamp)) • MD5: spRevpassword = MD5(spRevId + Password + timeStamp) <p>In the formula:</p> <ul style="list-style-type: none"> • spRevId and timeStamp: reverse authentication ID and timestamp. • Password: access password allocated by a service Partner to the SDP. A service Partner can obtain the password from the email notification received after successful registration. <ul style="list-style-type: none"> – This parameter is mandatory in a request sent to a service Partner who has configured authentication information during registration. – This parameter can be left empty in a request sent to a Developer, an API Partner, or a service Partner who does not configure authentication information. <p> NOTE</p> <p>To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks.</p> <p>[Example] 206D88BB7F3D154B130DD6E1E0B8828B</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
spld	xsd: string	21	Mandatory	<p>Partner ID.</p> <p>The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID:</p> <ul style="list-style-type: none"> • A service Partner and can API Partner 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. <p>[Example] 000201</p>
serviceld	xsd: string	21	Conditional	<p>Service ID.</p> <p>The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID.</p> <p>The serviceld must be contained during invocation of a service interface developed by service partners and other partners, and must not be contained during invocation of a capability interface developed by API partners, other partners, and developers.</p> <p>[Example]</p> <p>35000001000001</p>
timeStamp	xsd: string	14	Conditional	<p>Timestamp (UTC time).</p> <p>The value is used in MD5 encryption of spRevpassword.</p> <p>This parameter is mandatory when the spRevpassword parameter is required.</p> <p>[Format]</p> <p>yyyyMMddHHmmss</p> <p>[Example] 20100731064245</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
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] 100001200101110623021721000011

Message Body Parameters

Table 3-13 describes parameters in a **notifySmsDeliveryReceiptRequest** message body.

Table 3-13 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] 12345
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 3-14.

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



3 APIs for Sending SMS Messages

Table 3-14 Parameter structure of the DeliveryInformation type

Parameter	Type	Length	Level of Requirement	Description
Address	xsd:anyURI	30	Mandatory	<p>Mobile number or the fake ID of the sender. [Format]</p> <ul style="list-style-type: none"> Mobile number: 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. Fake ID: tel: [Prefix]- [opcoid]-[sequence]. [Example] Mobile number: tel:8612312345678 Fake ID: tel:f-245-11900000007639
deliveryStatus	DeliveryStatus	40	Mandatory	<p>Status description. [Enumerated values of DeliveryStatus]</p> <ul style="list-style-type: none"> 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. <p>[Example] DeliveredToTerminal</p>

3.4.4 Response

The App functions as the server, processes **notifySmsDeliveryReceiptRequest** messages received from the SDP, and sends **notifySmsDeliveryReceiptResponse** 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/v3_1/local">
  <soapenv:Header/>
  <soapenv:Body>
    <loc:notifySmsDeliveryReceiptResponse/>
  </soapenv:Body>
</soapenv:Envelope>
```



3 APIs for Sending SMS Messages

3.4.5 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.

3.5 stopDeliveryReceiptNotification

3.5.1 Function

The App (functioning as the client) invokes the stopDeliveryReceiptNotification API to unsubscribe from MO SMS message status report 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 status report notification is unsubscribed, the SDP does not send MO SMS messages status reports 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.

3.5.2 Request URI

The request URI is the destination URI of **stopDeliveryReceiptNotificationRequest** 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/v3

In the format, *IP* and *Port* indicate the service IP address and Parlay X 3.0 port number of the API provided by the SDP. Contact carriers to obtain the IP address and port number.

3.5.3 Request

The App functions as the client and sends a **stopDeliveryReceiptNotificationRequest** message to the SDP to disable MO SMS message status report notification.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:loc="http://www.csapi.org/schema/parlayx/sms/notification_manager/v3_2/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:stopDeliveryReceiptNotification>
      <loc:correlator>12345</loc:correlator>
    </loc:stopDeliveryReceiptNotification>
  </soapenv:Body>
</soapenv:Envelope>
```



3 APIs for Sending SMS Messages

Message Header Parameters


Table 3-15 describes parameters in a **stopDeliveryReceiptNotificationRequest** message header.

Table 3-15 Parameters in a stopDeliveryReceiptNotificationRequest message header

Parameter	Type	Length	Level of Requirement	Description
spld	xsd: string	21	Mandatory	<p>Partner ID.</p> <p>The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID:</p> <ul style="list-style-type: none">• A service Partner and API Partner 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. <p>[Example] 000201</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
spPassword	xsd: string	100	Conditional	<p>Authentication key for the SDP to authenticate partners.</p> <p>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.</p> <p>The value is a character string encrypted. The encryption formula is as follows:</p> <ul style="list-style-type: none"> • SHA-256: spPassword = Base64(SHA-256(spId + Password + timeStamp)) • MD5: spPassword = MD5(spId + Password + timeStamp) <p>In the formula:</p> <ul style="list-style-type: none"> • spId and timeStamp: authentication ID and timestamp. • Password: access password allocated by the SDP to a partner. <ul style="list-style-type: none"> – A service Partner and API Partner 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. <p> NOTE</p> <p>To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks.</p> <p>[Example] e6434ef249df55c7a21a0b45758a39bb</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
serviceld	xsd: string	21	Conditional	<p>Service ID.</p> <p>The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID.</p> <p>The serviceld must be contained during invocation of a service interface developed by service partners and other partners, and must not be contained during invocation of a capability interface developed by API partners, other partners, and developers.</p> <p>[Example] 35000001000001</p>
timeStamp	xsd: string	14	Conditional	<p>Time stamp (UTC time).</p> <p>The value is used in MD5 encryption of spPassword.</p> <p>This parameter is mandatory when the spPassword parameter is required.</p> <p>[Format] yyyyMMddHHmmss</p> <p>[Example] 20100731064245</p>

Message Body Parameters

Table 3-16 describes parameters in a **stopDeliveryReceiptNotificationRequest** message body.

Table 3-16 Parameters in a stopDeliveryReceiptNotificationRequest message body

Parameter	Type	Length	Level of Requirement	Description
correlator	xsd:string	50	Conditional	<p>ID that associates the startDeliveryReceiptNotificationRequest message with a stopDeliveryReceiptNotification message.</p> <p>The value is defined by a third party and is sent to the SDP when the startDeliveryReceiptNotification interface is invoked.</p> <p>[Example] 12345</p>

3.5.4 Response

The SDP functions as the server, processes **stopDeliveryReceiptNotificationRequest** messages received from the App, and sends **stopDeliveryReceiptNotificationResponse** messages to the App.



3 APIs for Sending SMS Messages

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 4 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:stopDeliveryReceiptNotificationResponse
      xmlns:ns1="http://www.csapi.org/schema/parlayx/sms/receive/v3_1/local"/>
    </soapenv:Body>
  </soapenv:Envelope>
```

3.5.5 Error Codes

Table 3-17 describes stopDeliveryReceiptNotification error codes that the SDP may return upon an exception. For details about the error codes, see the *SDP Solution Error Code Reference*.

Table 3-17 stopDeliveryReceiptNotification error codes

Error Code	Description	Cause
SVC0002	Correlator is null.	The Correlator value in the request body is blank.
	Correlator %1 has a invalid format.	The Correlator value in the request body is invalid.
SVC0901	SPID is null!	The SPID value in the request header is blank or does not exist in the SDP.
	SPID %1 is invalid!	The SPID value in the request header is in an incorrect format.
	SPID %1 is not exist!	The SP specified by spId in the request header does not exist in the SDP.
	SP ip is null!	The IP address in the request header is blank.
	Sp ip %1 is not accepted!	The IP address in the request header is incorrect.
	Sp password is null!	The password value in the request header is blank.
	Timestamp is empty in soapheader.	The timeStamp value in the request header is blank.
	local SP password is null!	An internal error occurs in the SDP.



3 APIs for Sending SMS Messages

Error Code	Description	Cause
	Sp password is not accepted!	The password value in the request header is incorrect.
	The Sp has not logged in.	The Token value in the request header does not exist in the SDP and the SP needs to log in first.
	SP %1 is in blacklist!	The SP account is in the blacklist.
	The sp's Status is pre-deregistered.	The SP is in the pre-deregistered state.
	The sp's Status is deregistered.	The SP is in the deregistered state.
	SP status is locked.	The SP is in the locked state.
	The sp's Status is deregistered.	The SP is in the forbidden state.
	The sp 's status is pause.	The SP is in the paused state.
	The sp's Status is unknown.	An internal SDP service is abnormal.
	Service ID %1 is invalid!	The servicelD value in the request header is incorrect.
	Service ID %1 is not existed!	The servicelD value in the request header does not exist in the SDP.
	Service %1 is in blacklist!	The servicelD value in the request header in the blacklist.
	The service status is configuring.	The service specified by servicelD in the request header is in the configuring state.
	The service status is suspended.	The service specified by servicelD in the request header is in the paused state.
	The service status is pre-deregistered.	The service specified by servicelD in the request header is in the pre-deregistered state.
	The service status is deregistered.	The service specified by servicelD in the request header is in the deregistered state.
	The service status is unknown.	An internal SDP service is abnormal.
	The API %1 is not existed.	This SP does not have the permission for using the API.
	The API status is disabled.	This SP does not have the permission for using the API.



3.6 *getSmsDeliveryStatus*

3.6.1 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 `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.

3.6.2 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/V3`

In the format, *IP* and *Port* indicate the service IP address and Parlay X 3.0 port number of the API provided by the SDP. Contact carriers to obtain the IP address and port number.

3.6.3 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:loc="http://www.csapi.org/schema/parlayx/sms/send/v3_1/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:getSmsDeliveryStatus>
    <loc:requestIdentifier>100001200301110907091036000031</loc:requestIdentifier>
  </loc:getSmsDeliveryStatus>
</soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 3-18 describes parameters in a `getSmsDeliveryStatusRequest` header.



3 APIs for Sending SMS Messages

Table 3-18 Parameters in a getSmsDeliveryStatusRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	<p>Partner ID.</p> <p>The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID:</p> <ul style="list-style-type: none">• A service Partner and API Partner 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. <p>[Example] 000201</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
spPassword	xsd: string	100	Conditional	<p>Authentication key for the SDP to authenticate partners.</p> <p>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.</p> <p>The value is a character string encrypted. The encryption formula is as follows:</p> <ul style="list-style-type: none"> • SHA-256: spPassword = Base64(SHA-256(spId + Password + timeStamp)) • MD5: spPassword = MD5(spId + Password + timeStamp) <p>In the formula:</p> <ul style="list-style-type: none"> • spId and timeStamp: authentication ID and timestamp. • Password: access password allocated by the SDP to a partner. <ul style="list-style-type: none"> – A service Partner and API Partner 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. <p> NOTE</p> <p>To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks.</p> <p>[Example] e6434ef249df55c7a21a0b45758a39bb</p>



3 APIs for Sending SMS Messages

Parameter	Type	Length	Level of Requirement	Description
serviceld	xsd: string	21	Conditional	<p>Service ID.</p> <p>The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID.</p> <p>The serviceld must be contained during invocation of a service interface developed by service partners and other partners, and must not be contained during invocation of a capability interface developed by API partners, other partners, and developers.</p> <p>[Example] 35000001000001</p>
timeStamp	xsd: string	14	Conditional	<p>Timestamp (UTC time).</p> <p>The value is used in MD5 encryption of spPassword.</p> <p>This parameter is mandatory when the spPassword parameter is required.</p> <p>[Format] yyyyMMddHHmmss</p> <p>[Example] 20100731064245</p>

Message Body Parameters

Table 3-19 describes parameters in a **getSmsDeliveryStatusRequest** message body.

Table 3-19 Parameters in a getSmsDeliveryStatusRequest message body

Parameter	Type	Length	Level of Requirement	Description
registrationIdentifier	xsd:string	30	Mandatory	<p>ID of an SMS message request.</p> <p>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.</p> <p>[Example] 100001200301111029065714000141</p>



3 APIs for Sending SMS Messages

3.6.4 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 4 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/v3_1/local">
      <ns1:result>
        <address>tel:8612312345678</address>
        <deliveryStatus>DeliveredToTerminal</deliveryStatus>
      </ns1:result>
    </ns1:getSmsDeliveryStatusResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Message Body Parameters

Table 3-20 describes parameters in a **getSmsDeliveryStatusResponse** message body.

Table 3-20 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 3-21.

Table 3-21 describes the parameter structure of the DeliveryInformation type.



3 APIs for Sending SMS Messages

Table 3-21 Parameter structure of the DeliveryInformation type

Parameter	Type	Length	Level of Requirement	Description
Address	xsd:anyURI	30	Mandatory	<p>Mobile number or the fake ID of the sender.</p> <p>[Format]</p> <ul style="list-style-type: none"> Mobile number: tel:[Prefix][Country code][Mobile number] In the format, <i>[Prefix]</i> and <i>[Country code]</i> are optional. The value of <i>[Prefix]</i>, if contained, can be +, +0, +00, 0, or 00. Fake ID: tel: [Prefix]- [opcoid]-[sequence]. <p>[Example]</p> <ul style="list-style-type: none"> Mobile number: tel:8612312345678 Fake ID: tel:f-245-11900000007639
deliveryStatus	DeliveryStatus	40	Mandatory	<p>Status description.</p> <p>[Enumerated values of DeliveryStatus]</p> <ul style="list-style-type: none"> 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. <p>[Example] DeliveredToTerminal</p>

3.6.5 Error Codes

Table 3-22 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 3-22 getSmsDeliveryStatus error codes

Error Code	Description	Cause
SVC0901	SPID is null!	The SPID value in the request header is blank or does not exist in the SDP.



3 APIs for Sending SMS Messages

Error Code	Description	Cause
	SPID %1 is invalid!	The SPID value in the request header is in an incorrect format.
	SPID %1 is not exist!	The SP specified by spId in the request header does not exist in the SDP.
	SP ip is null!	The IP address in the request header is blank.
	Sp ip %1 is not accepted!	The IP address in the request header is incorrect.
	Sp password is null!	The password value in the request header is blank.
	Timestamp is empty in soapheader.	The timeStamp value in the request header is blank.
	local SP password is null!	An internal error occurs in the SDP.
	Sp password is not accepted!	The password value in the request header is incorrect.
	The Sp has not logged in.	The Token value in the request header does not exist in the SDP and the SP needs to log in first.
	SP %1 is in blacklist!	The SP account is in the blacklist.
	The sp's Status is pre-deregistered.	The SP is in the pre-deregistered state.
	The sp's Status is deregistered.	The SP is in the deregistered state.
	SP status is locked.	The SP is in the locked state.
	The sp's Status is deregistered.	The SP is in the forbidden state.
	The sp 's status is pause.	The SP is in the paused state.
	The sp's Status is unknown.	An internal SDP service is abnormal.
	Service ID %1 is invalid!	The servicelD value in the request header is incorrect.
	Service ID %1 is not existed!	The servicelD value in the request header does not exist in the SDP.
	Service %1 is in blacklist!	The servicelD value in the request header in the blacklist.
	The service status is configuring.	The service specified by servicelD in the request header is in the configuring state.



3 APIs for Sending SMS Messages

Error Code	Description	Cause
	The service status is suspended.	The service specified by serviceld in the request header is in the paused state.
	The service status is pre-deregistered.	The service specified by serviceld in the request header is in the pre-deregistered state.
	The service status is deregistered.	The service specified by serviceld in the request header is in the deregistered state.
	The service status is unknown.	An internal SDP service is abnormal.
	The API %1 is not existed.	This SP does not have the permission for using the API.
	The API status is disabled.	This SP does not have the permission for using the API.
POL0904	SP level gross control not pass.	The number of requests delivered by the SP exceeds the signed TPS.

04

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.



4 API Error Responses

Table 4-1 Parameters in a service error response

Parameter Type		Level of Requirement	Description
faultcode	xsd:string	Mandatory	Result code. [Format] SVCABCD In the format, SVC identifies a service error response, and <i>ABCD</i> is a number ranging from 0001 to 9999. [Example] SVC0001
faultstring	xsd:string	Mandatory	Error description. The value can contain the variable <i>%#</i> in definition. When sending a response, the server replaces the variable <i>%#</i> 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



NOTE

In the **Level of Requirement** column, **Mandatory** indicates that a parameter is always mandatory, **Conditional** indicates that a parameter is mandatory or optional in specified conditions, and **Optional** indicates that a parameter is always optional.

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">
```



4 API Error Responses

```
<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] POLABCD 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 .



4 API Error Responses

Parameter Type		Level of Requirement	Description
variables	xsd:string [0..unbounded]	Conditional	Value of the variable defined in the value of faultstring . [Example] GroupAddr

**NOTE**

In the **Level of Requirement** column, **Mandatory** indicates that a parameter is always mandatory, **Conditional** indicates that a parameter is mandatory or optional in specified conditions, and **Optional** indicates that a parameter is always optional.

