SDP Solution API Reference (USSD, SOAP)

Change History

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

Issue 14 (2015-12-01)

This issue is the fourteen official release, and includes the following changes:
Add the redirectUssd interface.

Issue 13 (2015-11-20)

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

Issue 12 (2015-06-18)

This issue is the twelfth official release, and includes the following changes:
Add the error code **SVC4001** for sendUSSD.

Issue 11 (2015-01-12)

This issue is the eleventh official release, and includes the following changes:
Add the parameter **linkid** for notifyUssdReception.

Issue 10 (2014-05-04)

This issue is the tenth official release, and includes the following changes:
Change the description of **serviceId**.

Issue 09 (2014-04-26)

This issue is the ninth official release, and includes the following changes:
Modify the **Response** description of the **notifyUssdReception** interface and **notifyUssdAbort** interface.

Issue 08 (2014-03-28)

This issue is the eighth official release, and includes the following changes:
Add extensionInfo parameter in the **notifyUssdReception** interface and **notifyUssdAbort** interface.

Issue 07 (2014-03-10)

This issue is the seventh official release, and includes the following changes:
Modify ussdOpType parameter's description of the **notifyUssdReception** interface and **sendUSSD** interface.

Issue 06 (2014-01-24)

This issue is the sixth official release, and includes the following changes:
Modify some parameters' description of the **startUSSDNotification** interface and **notifyUssdReception** interface.

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.

Change History

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 and SOAPAction in the Overview.

Issue 02 (2013-08-05)

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

Updated Chapter 3.2 for adding **bundleID** field.

Issue 01 (2013-06-22)

This issue is the first official release.

Contents

01 Overview

1.1 API Functions

The SDP provides USSD capability application programming interfaces (APIs) for third-party applications (App for short) to connect to it and use its USSD capability to send and receive ussd messages. 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 USSD capability APIs provided by the SDP.

Table 1-1 Functions of USSD capability APIs

Function	Description	API
Receiving MO USSD messages	The App (functioning as the client) invokes an API to enable MO USSD message notification on the SDP (functioning as the server). The App uses this API to send MO routing information for receiving MO notifications to the SDP, and the App saves the MO routing information. After the enabling, the SDP sends MO USSD messages to the App based on the MO routing information when receiving the messages from the end user. 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 startUSSDNotification

1 Overview

Function	Description	API
	The SDP (functioning as the client) invokes an API to send MO USSD message to the App (functioning as the server). After the App successfully enables MO USSD message notification using the startUSSDNotification API, the SDP uses the notifyUssdReception API to send MO USSD message to the App when receiving the messages from the end user. 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 sends a response to the SDP within 30 seconds.	2.3 notifyUssdReception
	The SDP (functioning as the client) invokes an API to send abnormal USSD session ending requests to the App (functioning as the server). 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 sends a response to the SDP within 30 seconds.	notifyUssdAbort
	The App (functioning as the client) invokes an API to disable MO notification on the SDP (functioning as the server). This API is invoked by the App when it is to be brought offline. After the disabling, the SDP does not send MO USSD message to the App when receiving them from the end user. 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 stopUSSDNotification
Delivering USSD messages	The App (functioning as the client) invokes an API to send specified USSD 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 sendUSSD

1 Overview

Function	Description	API
	The App (functioning as the client) invokes an API to send USSD session ending requests 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.	3.3 sendUssdAbort

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 SOAP request in the following format:

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

Table 1-3 Request format

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

1.4 Response Format

Success Response Format

The SDP provides the SOAP 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 SOAP protocol.

Error Response Format

The SDP provides the SOAP API error responses in the following format:

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

      <detail>
         <parameter>…</parameter>
```

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

Table 1-5 Error response format

Element	Description
<soapenv:Envelope>	Root element in an error response, which specifies the namespace.
<soapenv:Body>	Error response body, contains the <soapenv:Fault> and <detail> elements. This element specifies the error code and error details.
<soapenv:Fault>	Error code and description. For details about error responses, see 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 USSD capability APIs.

Table 1-6 describes the namespaces of USSD capability APIs.

The namespace of data types used by the USSD capability APIs is **http://www.csapi.org/schema/parlayx/ussd/v1_0**.

Table 1-6 Namespaces of USSD capability APIs

Namespace	API
http://www.csapi.org/wsdl/osg/ussd/notification_manager/v1_0/	• startUSSDNotification • stopUSSDNotification
http://www.csapi.org/wsdl/parlayx/ussd/notification/v1_0	notifyUssdReception • notifyUssdAbort
http://www.csapi.org/wsdl/parlayx/ussd/send/v1_0	• sendUSSD • sendUssdAbort

1.6 SOAPAction

Leave the **SOAPAction** parameter empty.

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

```
SOAPAction: ""
```

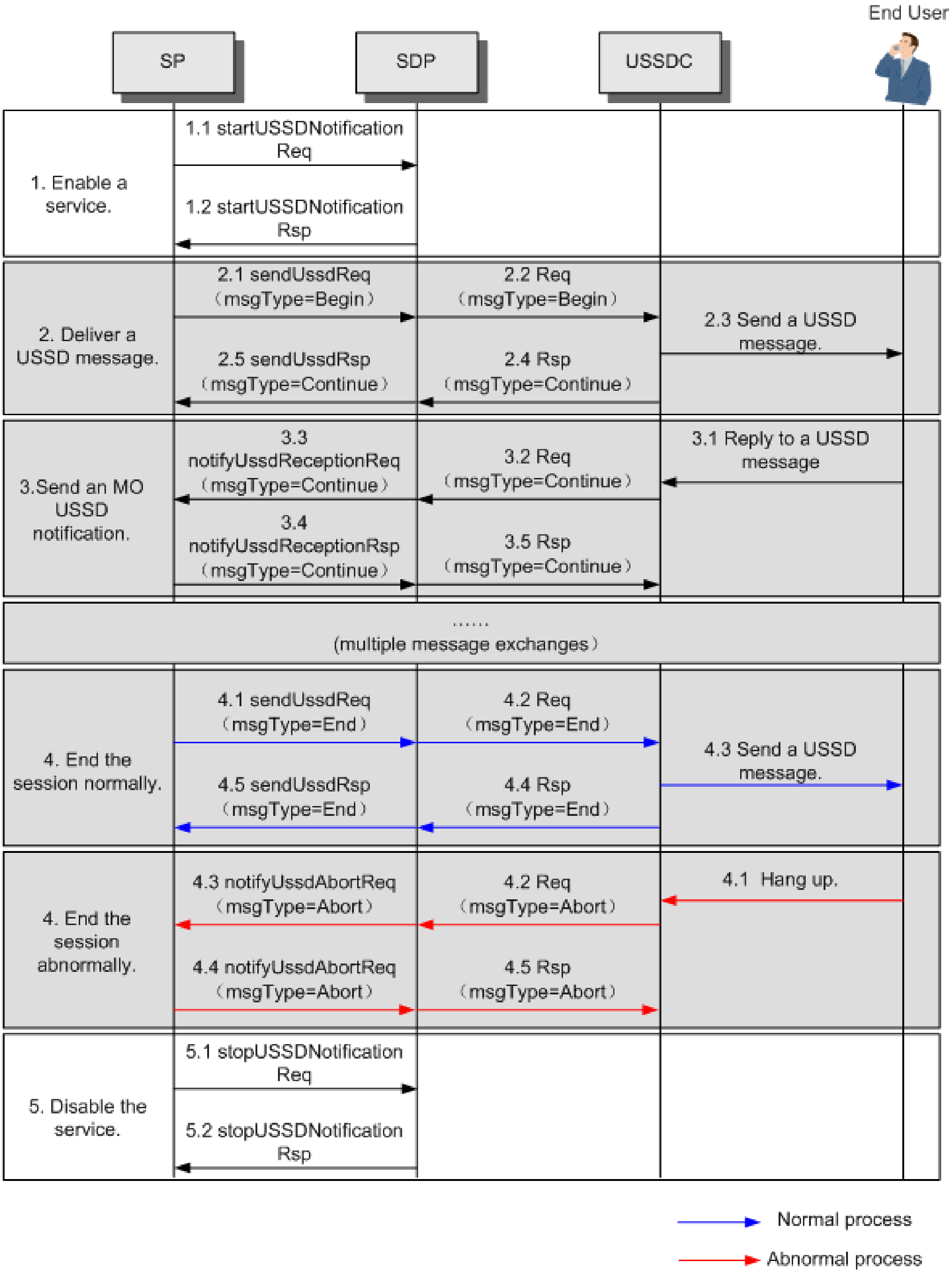
02

Interfaces for Receiving MO USSD messages

2.1 Process

Figure 2-1 shows the process of delivering USSD messages to a user who orders or subscribes to a USSD interactive service.

Figure 2-1 Process of delivering USSD messages (interactive service)


End User
SP
SDP
USSDC
1. Enable a service.
1.1 startUSSDNotification Req
1.2 startUSSDNotification Rsp
2. Deliver a USSD message.
2.1 sendUssdReq（msgType=Begin）
2.2 Req（msgType=Begin）
2.3 Send a USSD message.
2.5 sendUssdRsp（msgType=Continue）
2.4 Rsp（msgType=Continue）
3.Send an MO USSD notification.
3.3 notifyUssdReceptionReq（msgType=Continue）
3.2 Req（msgType=Continue）
3.1 Reply to a USSD message
3.4 notifyUssdReceptionRsp（msgType=Continue）
3.5 Rsp（msgType=Continue）
......（multiple message exchanges）
4. End the session normally.
4.1 sendUssdReq（msgType=End）
4.2 Req（msgType=End）
4.3 Send a USSD message.
4.5 sendUssdRsp（msgType=End）
4.4 Rsp（msgType=End）
4. End the session abnormally.
4.3 notifyUssdAbortReq（msgType=Abort）
4.2 Req（msgType=Abort）
4.1 Hang up.
4.4 notifyUssdAbortReq（msgType=Abort）
4.5 Rsp（msgType=Abort）
5. Disable the service.
5.1 stopUSSDNotification Req
5.2 stopUSSDNotification Rsp
Normal process
Abnormal process

The process is as follows:

- 1.1-1.2: A partner sends a USSD service enabling request to the SDP to enable a USSD service.

- 2.1-2.5: During service operation, an SP sends USSD messages to users who orders, subscribes to, or receives a USSD service as a gift. In the initial request, the **msgType** value is **Begin**, and the SDP creates a session between the SP and user.
- 3.1-3.5: The user sends replies based on the USSD message sent by the SP. The SDP sends the user's replies to the SP. A user can exchange multiple messages with an SP. In these messages, the **msgType** value is **continue**.
- 4.1-4.5: The USSD session ends.
  - If the USSD session ends normally, the SP sends a SendUSSD request where the **msgType** value is **End** to the SDP.
  - If a user hangs up before the session ends normally, the SDP sends the hang-up message to the SP, and the session ends.
- 5.1-5.2: An SP sends a USSD service disabling request to the SDP to disable the USSD service.

2.2 startUSSDNotification

2.2.1 API Function

The App (functioning as the client) invokes an API to enable MO USSD message notification on the SDP (functioning as the server).
The App uses this API to send MO routing information for receiving MO notifications to the SDP, and the App saves the MO routing information. After the enabling, the SDP sends MO USSD messages to the App based on the MO routing information when receiving the messages from the end user.
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 startUSSDNotification messages sent by the App to the SDP to enable the MO USSD message notification. The URI is provided by the SDP in the following format:
http://*IP*:*Port*/USSDNotificationManagerService/services/USSDNotificationManager
In the format, *IP* and *Port* indicate the service IP address and SOAP 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 **startUSSDNotificationRequest** message to the SDP to enable the MO USSD message notification.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/osg/ussd/notification_manager/v1_0/local">
   <soapenv:Header>
      <tns:RequestSOAPHeader xmlns:tns="http://www.huawei.com.cn/schema/common/v2_1">
         <tns:spId>000201</tns:spId>
      <tns:spPassword>e6434ef249df55c7a21a0b45758a39bb</tns:spPassword>
      <tns:serviceId>35000001000029</tns:serviceId>
      <tns:timeStamp>20100731064245</tns:timeStamp>
    </tns:RequestSOAPHeader>
  </soapenv:Header>
  <soapenv:Body>
    <loc:startUSSDNotification>
      <loc:reference>
        <endpoint>http://10.138.40.69:11400/xportal/services/NetworkNotify</endpoint>
        <interfaceName>notifyUssdReception</interfaceName>
        <correlator>123456</correlator>
      </loc:reference>
      <loc:ussdServiceActivationNumber>*1234*356#</loc:ussdServiceActivationNumber>
    </loc:startUSSDNotification>
  </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 2-1 describes parameters in a **startUSSDNotificationRequest** message header.

Table 2-1 Parameters in a startUSSDNotificationRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • 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. [Example] 000201

Parameter	Type	Length	Level of Requirement	Description
spPassword	xsd: string	100	Conditional	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: • SHA-256: spPassword = Base64(SHA-256(spId + Password + timeStamp)) • MD5: spPassword = MD5(spId + Password + timeStamp) In the formula: • **spId** and **timeStamp**: authentication ID and timestamp. • **Password**: access password allocated by the SDP to a partner. – 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. **NOTE** To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks. [Example] e6434ef249df55c7a21a0b45758a39bb
serviceId	xsd: string	21	Conditional	Service ID. 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. The **serviceId** 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. [Example] 35000001000029

Parameter	Type	Length	Level of Requirement	Description
timeStamp	xsd: string	14	Conditional	Time stamp (UTC time). The value is used in MD5 encryption of **spPassword**. This parameter is mandatory when the **spPassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245

Message Body Parameters

Table 2-2 describes parameters in a **startUSSDNotificationRequest** message body.

Table 2-2 Parameters in a startUSSDNotificationRequest message body

Parameter	Type	Length	Level of Requirement	Description
reference	common:SimpleReference	-	Mandatory	Reference. The App sends the App URL, API name, and correlator ID information to the SDP, which then uses the information for the specific status notification of the specific address. The **reference** parameter is of the **SimpleReference** type and contains multiple sub-parameters. For details about the **SimpleReference** type, see Table 2-3.
ussdServiceActivationNumber	xsd:anyURI	20	Mandatory	[Format] [Leading character]servicecode*serviceindicator1*serviceindicator2*…*serviceindicatorN[End character] The leading characters include one to three asterisks (*) or number signs (#). The end character is a number sign (#). [Example] *1234*356# – A service Partner can log in to the SDP management portal and query service information. Service Partners can extend service indicators 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.
criteria	xsd:string	50	Optional	Reserved for the SDP.

Table 2-3 describes the parameter structure of the SimpleReference type.

Table 2-3 Parameter structure of the SimpleReference type

Parameter	Type	Length	Level of Requirement	Description
endpoint	xsd:anyURI	512	Mandatory	URL from which the App obtains the upstream USSD message. [Example] http://10.138.38.139:9080/notify
interfaceName	xsd:string	20	Optional	Name of the notification interface. The value is defined by the interface invoker. This parameter can be left blank. [Example] notifyUssdReception
correlator	xsd:string	50	Mandatory	Correlator ID that associates a startUSSDNotificationRequest message with a stopUSSDNotificationRequest message. When the App sends a startUSSDNotificationRequest message to the SDP, the SDP records the correlator ID. When the App sends a stopUSSDNotificationRequest message to the SDP, the SDP disables the MO USSD messages notification based on the correlator ID. The value is a random number defined by a third party and must be unique. [Example] 12345

2.2.4 Response

The SDP functions as the server, processes **startUSSDNotification** messages received from the App, and sends **startUSSDNotificationResponse** messages to the App.
This topic provides a success response example and describes parameters in the response. If a request fails, the SDP sends an error response that contains an error code. For details about error responses, see API Error Responses.

Example

```
<soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<ns1:startUSSDNotificationResponse
xmlns:ns1="http://www.csapi.org/schema/osg/ussd/notification_manager/v1_0/local"/>
</soapenv:Body>
</soapenv:Envelope>
```

2.2.5 Error Codes

Table 2-4 describes **startUSSDNotification** error codes that the SDP may return upon an exception. For details about the error codes, see the *SDP Solution Error Code Reference*.

Table 2-4 startUSSDNotification error codes

Error Code	Description	Cause
SVC0002	TargetURL %1 is invalid.	The URL in the **endpoint** value in the request body is invalid.
	ServiceActivationNumber is null.	The **serviceactivationnumber** value in the request body is blank.
	Criteria %1 is invalid.	The **criteria** value in the request body is invalid.
	Criteria %1 is too long.	The length of the **criteria** value in the request body exceeds the specified range.
	SimpleReference is null.	The **simplereference** value in the request body is blank.
	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 invalid.
	Interface name is null in SimpleReference.	The **interface** value in the request body is invalid.
SVC0901	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 different from that set during SP registration with the SDP.
	Sp password is null!	The **password** value in the request header is blank.
	Sp password is not accepted!	The **password** value in the request header is different from that set by the SP in the SDP.
	SP %1 is in blacklist!	The **spId** value in the request header is blacklisted.
	The sp's Status is unknown.	An internal SDP service is abnormal.
	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.
	The sp's Status is forbidden.	The SP is in the forbidden state.

Error Code	Description	Cause
	The sp 's status is pause.	The SP is in the paused state.
	SP status is locked.	The SP is in the locked state.
	Service ID %1 is not exist!	The **serviceId** value in the request header does not exist in the SDP.
	Service ID is null!	The **serviceId** value in the request header is blank or does not exist in the SDP.
	Service ID %1 is invalid!	The **serviceId** value in the request header is in an incorrect format.
	Service %1 is in blacklist!	The **serviceId** value in the request header is blacklisted.
	The service status is configuring.	The service specified by **serviceId** in the request header is in the configuring state.
	The service status is suspended.	The service specified by **serviceId** in the request header is in the paused state.
	The service status is pre-deregistered.	The service specified by **serviceId** in the request header is in the pre-deregistered state.
	The service status is deregistered.	The service specified by **serviceId** 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.
	The ScfType %1 is inactive!	An internal error occurs in the SDP.
	The ScfType %1 is uninstalled!	An internal error occurs in the SDP.
	Timestamp is empty in soapheader.	The **timeStamp** value in the request header is blank.
	The authentication type is unknown!	An internal SDP service is abnormal.
	Authentication Failed, cause by SP,because of timestamp expired.	The **timeStamp** value in the request header has expired.
	local SP password is null!	An internal error occurs in the SDP.
SVC0905	Authentication Failed, cause by SP,because of timestamp is not valid.	The **timeStamp** value in the request body is in an incorrect format.

2.3 notifyUssdReception

2.3.1 API Function

The SDP (functioning as the client) invokes an API to send MO USSD message to the App (functioning as the server).
After the App successfully enables MO USSD message notification using the startUSSDNotification API, the SDP uses the notifyUssdReception API to send MO USSD message to the App when receiving the messages from the end user.
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 sends a response to the SDP within 30 seconds.

2.3.2 Request URI

The request URI is the destination URI of notifyUssdReception 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 **notifyUssdReceptionRequest** message to the App.

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>35000001000029</ns1:serviceId>
         <ns1:timeStamp>20100731064245</ns1:timeStamp>
         <ns1:linkid>12345678901111</ns1:linkid>
         <ns1:traceUniqueID>404092403801104031047140004003</ns1:traceUniqueID>
      </ns1:NotifySOAPHeader>
   </soapenv:Header>
   <soapenv:Body>
      <ns2:notifyUssdReception
xmlns:ns2="http://www.csapi.org/schema/parlayx/ussd/notification/v1_0/local">
         <ns2:msgType>0</ns2:msgType>
         <ns2:senderCB>320207133</ns2:senderCB>
         <ns2:receiveCB>0xFFFFFFFF</ns2:receiveCB>
         <ns2:ussdOpType>1</ns2:ussdOpType>
         <ns2:msIsdn>8613699991234</ns2:msIsdn>
         <ns2:serviceCode>2929</ns2:serviceCode>
         <ns2:codeScheme>17</ns2:codeScheme>
```

```
            <ns2:ussdString>*10086*01#</ns2:ussdString>
            <ns2:extenionInfo>
              <item>
                 <key></key>
                 <value></value>
              </item>
            </ns2:extensionInfo>
          </ns2:notifyUssdReception>
        </soapenv:Body>
      </soapenv:Envelope>
```

Message Header Parameters

Table 2-5 describes parameters in a **notifyUSSDReceptionRequest** message header.

Table 2-5 Parameters in a notifyUSSDReceptionRequest message header

Parameter	Type	Length	Level of Requirement	Description
spRevId	xsd: string	20	Conditional	Reverse authentication ID for the App to authenticate the SDP. The ID is set by service Partners during registration. A service Partner can log in to the SDP management portal and query account information for the ID. 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. [Example] 35000001

Parameter	Type	Length	Level of Requirement	Description
spRevpassword	xsd: string	100	Conditional	Reverse authentication key for the App to authenticate the SDP. The value is a character string encrypted. The encryption formula is as follows: • SHA-256: spRevpassword = Base64(SHA-256(spRevId + Password + timeStamp)) • MD5: spRevpassword = MD5(spRevId + Password + timeStamp) In the formula: • **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. 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. **NOTE** To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks. [Example] 206D88BB7F3D154B130DD6E1E0B8828B
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • A service 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. • An API Partner must contact the carrier. [Example] 000201

Parameter	Type	Length	Level of Requirement	Description
serviceId	xsd: string	21	Conditional	Service ID. The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID. The **serviceId** 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. [Example] 35000001000029
timeStamp	xsd: string	14	Conditional	Timestamp (UTC time). The value is used in MD5 encryption of **spRevpassword**. This parameter is mandatory when the **spRevpassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245
linkid	xsd:string	20	Conditional	Service order ID. The ID is automatically generated by the SDP when a user on-demand a service in the SDP. This parameter is mandatory during service on-demand by USSD 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] 404092403801104031047140004003

Message Body Parameters

Table 2-6 describes parameters in a **notifyUSSDReceptionRequest** message body.

Table 2-6 Parameters in a notifyUssdReceptionRequest message body

Parameter	Type	Length	Level of Requirement	Description
msgType	xsd:int	20	Mandatory	Message type. - 0: Begin - 1: Continue - 2: End The first request sent by users in a USSD session is of the **Begin** type. Other requests are of the **Continue** type. [Example] 0
senderCB	xsd:string	10	Mandatory	Initiator session ID. The value is generated when the SDP uses the **notifyUssdReceptionRequest** interface to send requests for the first time. In a valid session, the **senderCB** values in the subsequent **notifyUssdReceptionRequest** messages are the same as that in the first **notifyUssdReceptionRequest** message. [Example] 320207133
receiveCB	xsd:int	10	Mandatory	Recipient session ID. The value is generated by SPs. - In the first message (**msgType**=**0**) sent by the SDP, the value is **0xFFFFFFFF**. - In the subsequent messages, the value is the same as the **senderCB** value in the first **sendUSSDRequest** message. [Example] 0xFFFFFFFF

Parameter	Type	Length	Level of Requirement	Description
ussdOpType	xsd:int	2	Mandatory	USSD operation type. • 1: Request • 2: Notify • 3: Response • 4: Release The mapping between the **ussdOpType** and **msgType** values in **notifyUssdReceptionRequest** is as follows: • msgType=0: ussdOpType=1 • msgType=1: ussdOpType=3 • msgType=2: – ussdOpType=3 (A user initiates the USSD session.) – ussdOpType=4 (An partner initiates the USSD session.) [Example] 1
msIsdn	xsd:string	21	Mandatory	Mobile number or the fake ID of the sender. [tel:]*[Prefix][Country code]Mobile number*. In the format, **[tel:]**, *[Prefix]*, and *[Country code]* are optional. The value of *[Prefix]* can be **+**, **+0**, **+00**, **0**, or **00**. [Example] • Mobile number: tel:8612312345678 • Fake ID: tel:f-245-11900000007639
serviceCode	xsd:string	20	Mandatory	MO service access code. The value is distributed by the carrier. [Example] 2929
codeScheme	xsd:int	2	Mandatory	Coding mode of the **ussdString** value. • 15: 7-digit coding mode • 68: 8-digit coding mode • 17: 16-digit coding mode Different USSDC processing capabilities support different coding modes. The specific coding mode is obtained by the carrier. [Example] 15

Parameter	Type	Length	Level of Requirement	Description
ussdString	xsd:string	160	Mandatory	Message content sent by the user. [Format] [Leading character]servicecode*command word1*command word 2*…*command word N[End character] The leading characters include one to three asterisks (*) or number signs (#). The end character is a number sign (#). [Example] *10086*01#
extenionInfo	NamedParameterList	-	Conditional	Extended field. The SDP does not support this parameter. **NOTE** It's required for Liberia, but not for other opcos.

2.3.4 Response

The App functions as the server, processes the request messages received from the SDP, and sends the response messages to the SDP.
The response is constructed based on the WSDL specification by the partner that provides the App.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/ussd/notification/v1_0/local">
   <soapenv:Header/>
   <soapenv:Body>
      <loc:notifyUssdReceptionResponse>
         <loc:result>0</loc:result>
      </loc:notifyUssdReceptionResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

2.3.5 Error Codes

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

2.4 notifyUssdAbort

2.4.1 API Function

The SDP (functioning as the client) invokes an API to send abnormal USSD session ending requests to the App (functioning as the server).

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 sends a response to the SDP within 30 seconds.

2.4.2 Request URI

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

2.4.3 Request

The SDP functions as the client and sends a **notifyUssdAbortRequest** message to the App.

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>35000001000029</ns1:serviceId>
      <ns1:timeStamp>20100731064245</ns1:timeStamp>
      <ns1:traceUniqueID>404092403801104031047140004003</ns1:traceUniqueID>
    </ns1:NotifySOAPHeader>
  </soapenv:Header>
  <soapenv:Body>
    <ns2:notifyUssdAbort
xmlns:ns2="http://www.csapi.org/schema/parlayx/ussd/notification/v1_0/local">
      <ns2:senderCB>320207133</ns2:senderCB>
      <ns2:receiveCB>220207133</ns2:receiveCB>
      <ns2:abortReason>The end user cancels.</ns2:abortReason>
      <ns2:extenionInfo>
        <item>
          <key></key>
          <value></value>
        </item>
      </ns2:extensionInfo>
    </ns2:notifyUssdAbort>
  </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 2-7 describes parameters in a **notifyUssdAbortRequest** message header.

Table 2-7 Parameters in a notifyUssdAbortRequest message header

Parameter	Type	Length	Level of Requirement	Description
spRevId	xsd: string	20	Conditional	Reverse authentication ID for the App to authenticate the SDP. The ID is set by service Partners during registration. A service Partner can log in to the SDP management portal and query account information for the ID. 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. [Example] 35000001
spRevpassword	xsd: string	100	Conditional	Reverse authentication key for the App to authenticate the SDP. The value is a character string encrypted. The encryption formula is as follows: • SHA-256: spRevpassword = Base64(SHA-256(spRevId + Password + timeStamp)) • MD5: spRevpassword = MD5(spRevId + Password + timeStamp) In the formula: • **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. 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. **NOTE** To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks. [Example] 206D88BB7F3D154B130DD6E1E0B8828B

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • A service 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. • An API Partner must contact the carrier. [Example] 000201
serviceId	xsd: string	21	Conditional	Service ID. The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID. The **serviceId** 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. [Example] 35000001000029
timeStamp	xsd: string	14	Conditional	Timestamp (UTC time). The value is used in MD5 encryption of **spRevpassword**. This parameter is mandatory when the **spRevpassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245
traceUniqueID	xsd:string	30	Mandatory	Transaction ID. 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] 404092403801104031047140004003

Message Body Parameters

Table 2-8 describes parameters in a **notifyUssdAbortRequest** message body.

Table 2-8 Parameters in a notifyUssdAbortRequest message body

Parameter	Type	Length	Level of Requirement	Description
senderCB	xsd:string	10	Mandatory	Initiator session ID. The value is generated when the SDP uses the **notifyUssdReceptionRequest** interface to send requests for the first time. The value is the same as the **senderCB** value in the **notifyUssdReceptionRequest** request. [Example] 320207133
receiveCB	xsd:int	10	Mandatory	Recipient session ID. The value is generated by an SP and is sent to the SDP using the **senderCB** field in the **sendUSSDRequest** message. [Example] 0xFFFFFFFF
abortReason	xsd:string	160	Mandatory	USSD session termination cause. [Example] The end user cancels.
extenionInfo	NamedParameterList	-	Conditional	Extended field. The SDP does not support this parameter. **NOTE** It's required for Liberia, but not for other opcos.

2.4.4 Response

The App functions as the server, processes the request messages received from the SDP, and sends the response messages to the SDP.
The response is constructed based on the WSDL specification by the partner that provides the App.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/ussd/notification/v1_0/local">
   <soapenv:Header/>
   <soapenv:Body>
      <loc:notifyUssdAbortResponse/>
   </soapenv:Body>
</soapenv:Envelope>
```

2.4.5 Error Codes

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

2.5 stopUSSDNotification

2.5.1 API Function

The App (functioning as the client) invokes an API to disable MO notification on the SDP (functioning as the server). This API is invoked by the App when it is to be brought offline.

After the disabling, the SDP does not send MO USSD message to the App when receiving them from the end user.

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 stopUSSDNotification messages sent by the App to the SDP to disenable the MO USSD message notification. The URI is provided by the SDP in the following format:

http://*IP*:*Port*/USSDNotificationManagerService/services/USSDNotificationManager

In the format, *IP* and *Port* indicate the service IP address and SOAP 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 **stopUSSDNotificationRequest** message to the SDP to disenable the MO USSD message notification.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/osg/ussd/notification_manager/v1_0/local">
  <soapenv:Header>
    <tns:RequestSOAPHeader xmlns:tns="http://www.huawei.com.cn/schema/common/v2_1">
      <tns:spId>000201</tns:spId>
      <tns:spPassword>e6434ef249df55c7a21a0b45758a39bb</tns:spPassword>
      <tns:serviceId>35000001000029</tns:serviceId>
      <tns:timeStamp>20100731064245</tns:timeStamp>
    </tns:RequestSOAPHeader>
  </soapenv:Header>
  <soapenv:Body>
    <loc:stopUSSDNotification>
      <loc:correlator>123456</loc:correlator>
      </loc:stopUSSDNotification>
   </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

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

Table 2-9 Parameters in a stopUSSDNotificationRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • 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. [Example] 000201

Parameter	Type	Length	Level of Requirement	Description
spPassword	xsd: string	100	Conditional	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: • SHA-256: spPassword = Base64(SHA-256(spId + Password + timeStamp)) • MD5: spPassword = MD5(spId + Password + timeStamp) In the formula: • **spId** and **timeStamp**: authentication ID and timestamp. • **Password**: access password allocated by the SDP to a partner. – 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. **NOTE** To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks. [Example] e6434ef249df55c7a21a0b45758a39bb
serviceId	xsd: string	21	Conditional	Service ID. 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. The **serviceId** 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. [Example] 350000010000029

Parameter	Type	Length	Level of Requirement	Description
timeStamp	xsd: string	14	Conditional	Time stamp (UTC time). The value is used in MD5 encryption of **spPassword**. This parameter is mandatory when the **spPassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245

Message Body Parameters

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

Table 2-10 Parameters in a stopUSSDNotificationRequest message body

Parameter	Type	Length	Level of Requirement	Description
correlator	xsd:string	50	Mandatory	Correlator ID that associates a startUSSDNotificationRequest message with a stopUSSDNotificationRequest message. When the App sends a startUSSDNotificationRequest message to the SDPthe SDP records the correlator ID. When the App sends a stopUSSDNotificationRequest message to the SDPthe SDP disables the MO USSD message notification based on the correlator ID. It matches the **correlator** parameter in the startUSSDNotification message. [Example] 123456

2.5.4 Response

The SDP functions as the server, processes **stopUSSDNotification** messages received from the App, and sends **stopUSSDNotificationResponse** 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 5 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:stopUSSDNotificationResponse
xmlns:ns1="http://www.csapi.org/schema/osg/ussd/notification_manager/v1_0/local"/>
```

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

2.5.5 Error Codes

Table 2-11 describes **stopUSSDNotification** 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 stopUSSDNotification 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 %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 different from that set during SP registration with the SDP.
	Sp password is null!	The **password** value in the request header is blank.
	Sp password is not accepted!	The **password** value in the request header is different from that set by the SP in the SDP.
	SP %1 is in blacklist!	The **spId** value in the request header is blacklisted.
	The sp's Status is unknown.	An internal SDP service is abnormal.
	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.
	The sp's Status is forbidden.	The SP is in the forbidden state.
	The sp 's status is pause.	The SP is in the paused state.
	SP status is locked.	The SP is in the locked state.
	Service ID %1 is not exist!	The **serviceId** value in the request header does not exist in the SDP.

Error Code	Description	Cause
	Service ID is null!	The **serviceId** value in the request header is blank or does not exist in the SDP.
	Service ID %1 is invalid!	The **serviceId** value in the request header is in an incorrect format.
	Service %1 is in blacklist!	The **serviceId** value in the request header is blacklisted.
	The service status is configuring.	The service specified by **serviceId** in the request header is in the configuring state.
	The service status is suspended.	The service specified by **serviceId** in the request header is in the paused state.
	The service status is pre-deregistered.	The service specified by **serviceId** in the request header is in the pre-deregistered state.
	The service status is deregistered.	The service specified by **serviceId** 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.
	The ScfType %1 is inactive!	An internal error occurs in the SDP.
	The ScfType %1 is uninstalled!	An internal error occurs in the SDP.
	Timestamp is empty in soapheader.	The **timeStamp** value in the request header is blank.
	The authentication type is unknown!	An internal SDP service is abnormal.
	Authentication Failed, cause by SP,because of timestamp expired.	The **timeStamp** value in the request header has expired.
	local SP password is null!	An internal error occurs in the SDP.
SVC0905	Authentication Failed, cause by SP,because of timestamp is not valid.	The **timeStamp** value in the request body is in an incorrect format.

2 Interfaces for Receiving MO USSD messages

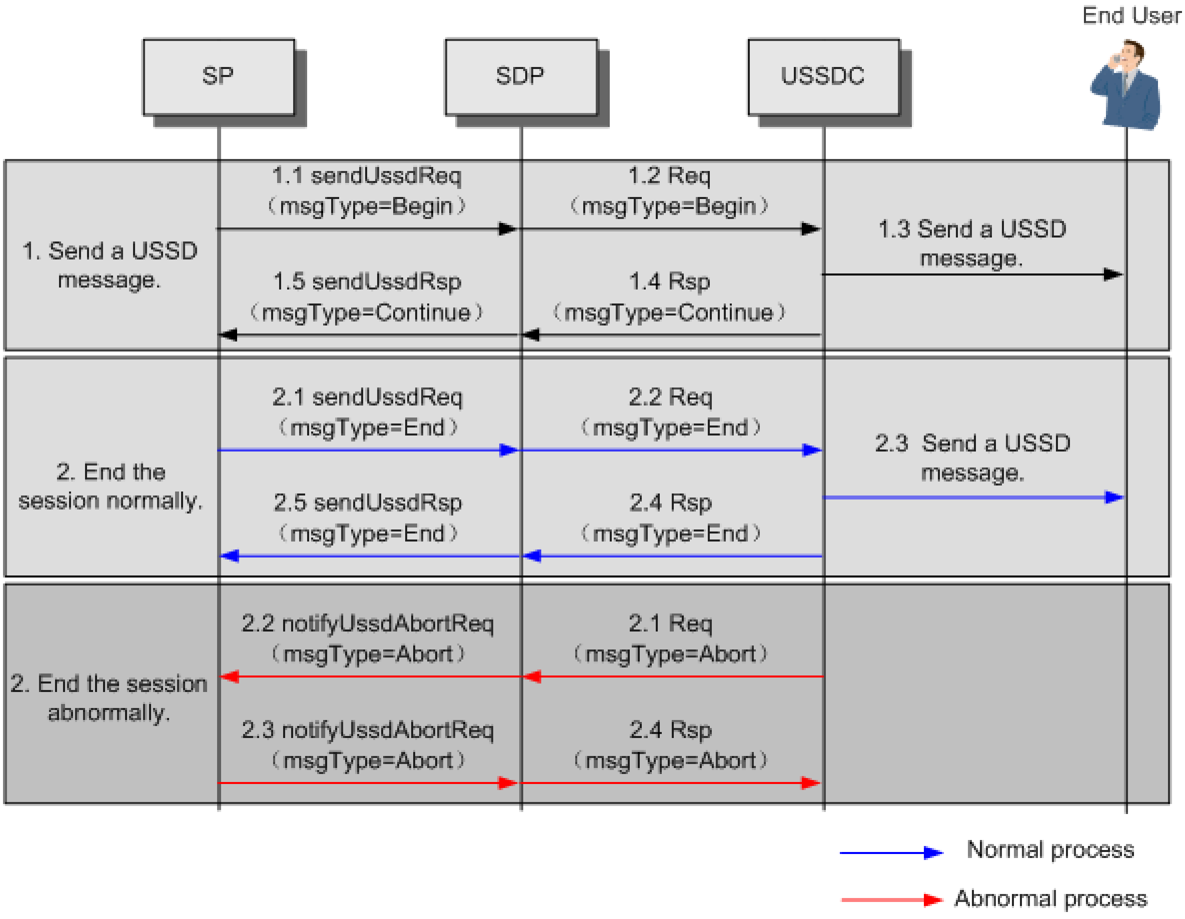
04

Interfaces for Switching USSD session

3.1 Process

Figure 3-1 shows the process of delivering USSD messages to a user who orders or subscribes to a USSD service.

Figure 3-1 Process of delivering USSD messages


End User
SP
SDP
USSDC
1. Send a USSD message.
1.1 sendUssdReq（msgType=Begin）
1.2 Req（msgType=Begin）
1.3 Send a USSD message.
1.4 Rsp（msgType=Continue）
1.5 sendUssdRsp（msgType=Continue）
2. End the session normally.
2.1 sendUssdReq（msgType=End）
2.2 Req（msgType=End）
2.3 Send a USSD message.
2.4 Rsp（msgType=End）
2.5 sendUssdRsp（msgType=End）
2. End the session abnormally.
2.1 Req（msgType=Abort）
2.2 notifyUssdAbortReq（msgType=Abort）
2.3 notifyUssdAbortReq（msgType=Abort）
2.4 Rsp（msgType=Abort）
Normal process
Abnormal process

The process is as follows:

- 1.1-1.5: During service operation, an SP sends USSD messages to users who orders, subscribes to, or receives a USSD service as a gift. In the initial request, the **msgType** value is **Begin**, and the SDP creates a session between the SP and user.
- 2.1-2.5: The USSD session ends.
  - If the USSD session ends normally, the SP sends a SendUSSD request where the **msgType** value is **End** to the SDP.
  - If a user hangs up before the session ends normally, the SDP sends the hang-up message to the SP, and the session ends.

3.2 sendUSSD

3.2.1 API Function

The App (functioning as the client) invokes an API to send specified USSD messages to theSDP (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 sendUSSD messages sent by the App to the SDP. The URI is provided by the SDP in the following format:
http://*IP*:*Port*/SendUssdService/services/SendUssd
In the format, *IP* and *Port* indicate the service IP address and SOAP port number of the API provided by the SDP. Contact carriers to obtain the IP address and port number.

3.2.3 Request

The App functions as the client and sends a **sendUSSDRequest** message to the SDP.

Example

- When a service Partner delivers an on-demand service message:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/ussd/send/v1_0/local">
  <soapenv:Header>
    <tns:RequestSOAPHeader xmlns:tns="http://www.huawei.com.cn/schema/common/v2_1">
      <tns:spId>000201</tns:spId>
      <tns:spPassword>e6434ef249df55c7a21a0b45758a39bb</tns:spPassword>
      <tns:serviceId>35000001000029</tns:serviceId>
      <tns:timeStamp>20100731064245</tns:timeStamp>
      <tns:OA>8613300000010</tns:OA>
```

```
         <tns:FA>8613300000010</tns:FA>
         <tns:linkid>12345678901111</tns:linkid>
      </tns:RequestSOAPHeader>
   </soapenv:Header>
   <soapenv:Body>
      <loc:sendUssd>
         <loc:msgType>0</loc:msgType>
         <loc:senderCB>306909975</loc:senderCB>
         <loc:receiveCB/>
         <loc:ussdOpType>1</loc:ussdOpType>
         <loc:msIsdn>8633699991234</loc:msIsdn>
         <loc:serviceCode>2929</loc:serviceCode>
         <loc:codeScheme>68</loc:codeScheme>
         <loc:ussdString>please select: Menuplease</loc:ussdString>
      </loc:sendUssd>
   </soapenv:Body>
</soapenv:Envelope>
```

- When a service Partner delivers a gift service message:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/ussd/send/v1_0/local">
   <soapenv:Header>
      <tns:RequestSOAPHeader xmlns:tns="http://www.huawei.com.cn/schema/common/v2_1">
         <tns:spId>000201</tns:spId>
         <tns:spPassword>e6434ef249df55c7a21a0b45758a39bb</tns:spPassword>
         <tns:serviceId>35000001000029</tns:serviceId>
         <tns:timeStamp>20100731064245</tns:timeStamp>
         <tns:OA>8613300000010</tns:OA>
         <tns:FA>8613300000010</tns:FA>
         <tns:presentid>12345678901111</tns:presentid>
      </tns:RequestSOAPHeader>
   </soapenv:Header>
   <soapenv:Body>
      <loc:sendUssd>
         <loc:msgType>0</loc:msgType>
         <loc:senderCB>306909975</loc:senderCB>
         <loc:receiveCB/>
         <loc:ussdOpType>1</loc:ussdOpType>
         <loc:msIsdn>8633699991234</loc:msIsdn>
         <loc:serviceCode>2929</loc:serviceCode>
         <loc:codeScheme>68</loc:codeScheme>
         <loc:ussdString>please select: Menuplease</loc:ussdString>
      </loc:sendUssd>
   </soapenv:Body>
</soapenv:Envelope>
```

- When an API Partner or a Developer sends a USSD message:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" 
xmlns:loc="http://www.csapi.org/schema/parlayx/ussd/send/v1_0/local">
  <soapenv:Header>
    <tns:RequestSOAPHeader xmlns:tns="http://www.huawei.com.cn/schema/common/v2_1">
      <tns:spId>000201</tns:spId>
      <tns:spPassword>e6434ef249df55c7a21a0b45758a39bb</tns:spPassword>
      <tns:bundleID>256000039</tns:bundleID>
      <tns:timeStamp>20100731064245</tns:timeStamp>
      <tns:OA>8613300000010</tns:OA>
      <tns:FA>8613300000010</tns:FA>
    </tns:RequestSOAPHeader>
  </soapenv:Header>
  <soapenv:Body>
    <loc:sendUssd>
      <loc:msgType>0</loc:msgType>
      <loc:senderCB>306909975</loc:senderCB>
      <loc:receiveCB/>
      <loc:ussdOpType>1</loc:ussdOpType>
      <loc:msIsdn>8633699991234</loc:msIsdn>
      <loc:serviceCode>2929</loc:serviceCode>
      <loc:codeScheme>68</loc:codeScheme>
      <loc:ussdString>please select: Menuplease</loc:ussdString>
    </loc:sendUssd>
  </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 3-1 describes parameters in a **sendUSSDRequest** message header.

Table 3-1 Parameters in a sendUSSDRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • 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. [Example] 000201

Parameter	Type	Length	Level of Requirement	Description
spPassword	xsd: string	100	Conditional	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: • SHA-256: spPassword = Base64(SHA-256(spId + Password + timeStamp)) • MD5: spPassword = MD5(spId + Password + timeStamp) In the formula: • **spId** and **timeStamp**: authentication ID and timestamp. • **Password**: access password allocated by the SDP to a partner. – 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. 📖 **NOTE** To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks. [Example] e6434ef249df55c7a21a0b45758a39bb

Parameter	Type	Length	Level of Requirement	Description
serviceId	xsd: string	21	Conditional	Service ID. The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID. The **serviceId** 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. [Example] 35000001000029
bundleID	xsd: string	21	Conditional	Bundle ID. When SDP creates a capability bundle, SDP allocates a bundleID to the capability bundle. 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. [Example] 256000039
timeStamp	xsd: string	14	Conditional	Time stamp (UTC time). The value is used in MD5 encryption of **spPassword**. This parameter is mandatory when the **spPassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245

Parameter	Type	Length	Level of Requirement	Description
OA	xsd:string	30	Conditional	Mobile number or the fake ID of the service originator. This parameter is mandatory in a request for sending a single SMS message, and can be left empty in a request for sending bulk SMS messages. • In 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 or the fake ID of the user. In a service Partner's request for sending an SMS message to a gift recipient, the value is the mobile number or the fake ID of the gift sender. • In an API Partner's or a Developer's request, the value is the mobile number or the fake ID of the message recipient. [Example] • Mobile number: 8612312345678 • Fake ID: f-245-11900000007639
FA	xsd:string	30	Conditional	Mobile number or the fake ID of the charged party. The value must be the same as the value of **OA**.
linkid	xsd:string	20	Conditional	Service order ID. The ID is automatically generated by the SDP when a user orders a service in the SDP. This parameter is mandatory during on-demand service delivery by SMS message. The SDP sends the value to service Partners as follows in different scenarios: • Invokes the ServiceOnDemand API to send the value when a user orders a service on the SDP portals. • Invokes the notifySmsReception API to send the value when a user orders a service by sending an SMS message. [Example] 12345678901111

Parameter	Type	Length	Level of Requirement	Description
presentid	xsd:string	15	Conditional	Service gift ID. The ID is automatically generated by the SDP when a user sends a service to another user as a gift on the SDP. This parameter is mandatory in a 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. The SDP invokes the assignPresentToUser API to send the value to service Partners. [Example] 22345678901113

Message Body Parameters

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

Table 3-2 Parameters in a sendUSSDRequest message body

Parameter	Type	Length	Level of Requirement	Description
msgType	xsd:int	2	Mandatory	Message type. • 0: Begin • 1: Continue • 2: End The value varies according to the session initiator. • For a session initiated by a user: The value is **1**. • For a session initiated by a third party: – The value is **0** in the first message. – The value is **2** in the last message. – The value is **1** in other messages. [Example] 0

Parameter	Type	Length	Level of Requirement	Description
senderCB	xsd:string	10	Mandatory	Initiator session ID. The value is generated when an SP uses the sendUSSDRequest interface to send requests for the first time. In a valid session, the **senderCB** values in the subsequent sendUSSDRequest and sendUssdAbortRequest messages are the same as that in the first sendUSSDRequest message. [Example] 320207133
receiveCB	xsd:int	10	Mandatory	Recipient session ID. The value is generated by the SDP. ● In the first message (**msgType**=**0**) sent by the SDP, the value is **0xFFFFFFFF**. ● In the subsequent messages, the value is the same as the **senderCB** value in the first notifyUssdReception message. [Example] 0xFFFFFFFF

Parameter	Type	Length	Level of Requirement	Description
ussdOpType	xsd:int	2	Mandatory	USSD operation type. • 1: Request • 2: Notify • 3: Response • 4: Release The mapping between the **ussdOpType** and **msgType** values in **sendUSSDRequest** is as follows: • msgType=0: – ussdOpType=1 (A user must reply after receiving a message.) – ussdOpType=2 (A user cannot view the delivered message and does not need to reply.) • msgType=1: – ussdOpType=1 (A user must reply after receiving a message.) – ussdOpType=2 (A user cannot view the delivered message and does not need to reply.) • msgType=2: – ussdOpType=3 (A user initiates the USSD session.) – ussdOpType=4 (An partner initiates the USSD session.) [Example] 1
msIsdn	xsd:string	30	Mandatory	Mobile number or the fake ID of the message recipient. [Format] • Mobile number: tel:[Prefix][Country code][Mobile number] In the format, *[Prefix]* is 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

Parameter	Type	Length	Level of Requirement	Description
serviceCode	xsd:string	20	Mandatory	Name of the message sender, which is displayed on the user terminal. • For an SP, the value is the access code of an SP service. A partner must obtain the access code from the carrier before releasing a service. • For an API Partner user or a Developer, the value is a partner access code allocated by the carrier. The carrier allocates an access code when an enterprise user or a developer purchases a capability product. [Example] 321123
codeScheme	xsd:int	2	Mandatory	Coding mode of the **ussdString** value. • 15: 7-digit coding mode • 68: 8-digit coding mode • 17: 16-digit coding mode Set the value to the coding mode supported by the USSDC. Obtain t he value from the carrier. [Example] 15
ussdString	xsd:string	160	Mandatory	USSD message content received by the user. [Example] Please vote for xxx.
endpoint	xsd:anyURI	512	Conditional	Notification address of an MO USSD message. If a third party does not use the startUSSDNotification interface to set the notification address, this parameter must be contained in the sendUSSD message. [Example] http://10.138.38.139:9080/notify
extenionInfo	NamedParameterList	-	Conditional	Extended field. The SDP does not support this parameter.

3.2.4 Response

The SDP functions as the server, processes **sendUSSD** messages received from the App, and sends **sendUSSDResponse** 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 5 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:sendUssdResponse xmlns:ns1="http://www.csapi.org/schema/parlayx/ussd/send/v1_0/local">
      <ns1:result>0</ns1:result>
    </ns1:sendUssdResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Message Body Parameters

Table 3-3 describes the parameter in a sendUSSDResponse message body.

Table 3-3 Parameters in a sendUSSDResponse message body

Parameter	Type	Length	Level of Requirement	Description
result	xsd:string	30	Mandatory	Value **0** indicates success. For the error codes, see 3.2.5 Error Codes.

3.2.5 Error Codes

Table 3-4 describes **sendUSSD** error codes that the SDP may return upon an exception. For details about the error codes, see the *SDP Solution Error Code Reference*.

Table 3-4 sendUSSD error codes

Error Code	Description	Cause
SVC0001	Get addr route info failed!the baseScfType = %1,the sourceAddr = %2,the destAddr = %3.	An internal SDP service is abnormal.
	Waiting for response timed out, message type is %1	An internal SDP service is abnormal.
	Received failed response, message type is %1, error is %2	An internal SDP service is abnormal.
SVC0002	msisdn %1 is invalid format.	The **msisdn** value in the request body is in an incorrect format.
	msisdn %1 is too long.	The **msisdn** value in the request body is too long.
	msisdn is null.	The **msisdn** value in the request body is blank.

Error Code	Description	Cause
	msgType is null.	The **msgType** field is blank in the request body.
	msgType %1 is invalid.	The **msgType** value in the request body is in an incorrect format.
	ussdOpType is null.	The **ussdOpType** value in the request body is blank.
	codeSchema is null.	The **codeSchema** value in the request body is blank.
	Cannot find route policy, the policy identifier is %1.	An internal SDP service is abnormal.
	codeSchema %1 is invalid.	The **codeSchema** value in the request body is in an incorrect format.
	senderCB is null.	The **senderCB** value in the request body is blank.
	Cannot find USSD transaction, receiverCB is %1.	The session times out, and the SDP cannot match values for **receiverCB**.
	senderCB %1 in message cannot match to the one %2 in original USSD transaction.	The session times out, and the SDP cannot match values for **senderCB**.
	senderCB %1 is invalid format.	The **senderCB** value in the request body is in an incorrect format.
	ussdString is null.	The **ussdString** value in the request body is blank.
	receiveCB %1 is invalid format.	The **receiveCB** value in the request body is in an incorrect format.
	receiveCB is null.	The **receiveCB** value in the request body is blank.
SVC0901	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 different from that set during SP registration with the SDP.
	Sp password is null!	The **password** value in the request header is blank.

Error Code	Description	Cause
	Sp password is not accepted!	The **password** value in the request header is different from that set by the SP in the SDP.
	SP %1 is in blacklist!	The **spId** value in the request header is blacklisted.
	The sp's Status is unknown.	An internal SDP service is abnormal.
	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.
	The sp's Status is forbidden.	The SP is in the forbidden state.
	The sp 's status is pause.	The SP is in the paused state.
	SP status is locked.	The SP is in the locked state.
	Service ID %1 is not exist!	The **serviceId** value in the request header does not exist in the SDP.
	Service ID is null!	The **serviceId** value in the request header is blank or does not exist in the SDP.
	Service ID %1 is invalid!	The **serviceId** value in the request header is in an incorrect format.
	Service %1 is in blacklist!	The **serviceId** value in the request header is blacklisted.
	The service status is configuring.	The service specified by **serviceId** in the request header is in the configuring state.
	The service status is suspended.	The service specified by **serviceId** in the request header is in the paused state.
	The service status is pre-deregistered.	The service specified by **serviceId** in the request header is in the pre-deregistered state.
	The service status is deregistered.	The service specified by **serviceId** 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.

Error Code	Description	Cause
	The ScfType %1 is inactive!	An internal error occurs in the SDP.
	The ScfType %1 is uninstalled!	An internal error occurs in the SDP.
	Timestamp is empty in soapheader.	The **timeStamp** value in the request header is blank.
	The authentication type is unknown!	An internal SDP service is abnormal.
	Authentication Failed, cause by SP,because of timestamp expired.	The **timeStamp** value in the request header has expired.
	local SP password is null!	An internal error occurs in the SDP.
SVC0905	Authentication Failed, cause by SP,because of timestamp is not valid.	The **timeStamp** value in the request body is in an incorrect format.
SVC4001	The subscriber is in blacklist	The subscriber is in blacklist

3.3 sendUssdAbort

3.3.1 API Function

The App (functioning as the client) invokes an API to send USSD session ending requests to theSDP (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.

3.3.2 Request URI

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

http://*IP*:*Port*/SendUssdService/services/SendUssd

In the format, *IP* and *Port* indicate the service IP address and SOAP 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 **sendUssdAbortRequest** message to the SDP.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/ussd/send/v1_0/local">
  <soapenv:Header>
```

```
    <tns:RequestSOAPHeader xmlns:tns="http://www.huawei.com.cn/schema/common/v2_1">
      <tns:spId>000201</tns:spId>
      <tns:spPassword>e6434ef249df55c7a21a0b45758a39bb</tns:spPassword>
      <tns:serviceId>35000001000029</tns:serviceId>
      <tns:timeStamp>20100731064245</tns:timeStamp>
      <tns:OA>8613300000010</tns:OA>
      <tns:FA>8613300000010</tns:FA>
    </tns:RequestSOAPHeader>
  </soapenv:Header>
  <soapenv:Body>
    <loc:sendUssdAbort>
      <loc:senderCB>306909975</loc:senderCB>
      <loc:receiveCB>286652700</loc:receiveCB>
      <loc:abortReason>sp abort</loc:abortReason>
    </loc:sendUssdAbort>
  </soapenv:Body>
</soapenv:Envelope>
```

Message Header Parameters

Table 3-5 describes parameters in a **sendUssdAbortRequest** message header.

Table 3-5 Parameters in a sendUssdAbortRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • 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. [Example] 000201

Parameter	Type	Length	Level of Requirement	Description
spPassword	xsd: string	100	Conditional	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: • SHA-256: spPassword = Base64(SHA-256(spId + Password + timeStamp)) • MD5: spPassword = MD5(spId + Password + timeStamp) In the formula: • **spId** and **timeStamp**: authentication ID and timestamp. • **Password**: access password allocated by the SDP to a partner. – 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. **NOTE** To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks. [Example] e6434ef249df55c7a21a0b45758a39bb

Parameter	Type	Length	Level of Requirement	Description
serviceId	xsd: string	21	Conditional	Service ID. The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID. The **serviceId** 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. [Example] 35000001000029
timeStamp	xsd: string	14	Conditional	Time stamp (UTC time). The value is used in MD5 encryption of **spPassword**. This parameter is mandatory when the **spPassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245

Message Body Parameters

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

Table 3-6 Parameters in a sendUssdAbortRequest message body

Parameter	Type	Length	Level of Requirement	Description
senderCB	xsd:string	10	Mandatory	Initiator session ID. The value is generated when an SP uses the sendUSSDRequest interface to send requests for the first time. The value is the same as the **senderCB** value in the first sendUssdRequest message. [Example] 320207133
receiveCB	xsd:int	10	Mandatory	Recipient session ID. The value is generated when the SDP uses the notifyUssdReceptionRequest interface to send requests for the first time. The value is the same as the **senderCB** value in the first

Parameter	Type	Length	Level of Requirement	Description
				notifyUssdReceptionRequest message. [Example] 420207133
abortReason	xsd:string	160	Mandatory	USSD session termination cause. [Example] SP abort the session.

3.3.4 Response

The SDP functions as the server, processes **sendUssdAbort** messages received from the App, and sends **sendUssdAbortResponse** 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 5 API Error Responses.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/ussd/send/v1_0/local">
   <soapenv:Header/>
   <soapenv:Body>
      <loc:sendUssdAbortResponse/>
   </soapenv:Body>
</soapenv:Envelope>
```

3.3.5 Error Codes

Table 3-7 describes **sendUssdAbort** 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 sendUssdAbort error codes

Error Code	Description	Cause
SVC0001	Get addr route info failed!the baseScfType = %1,the sourceAddr = %2,the destAddr = %3.	An internal SDP service is abnormal.
	Waiting for response timed out, message type is %1	An internal SDP service is abnormal.
	Received failed response, message type is %1, error is %2	An internal SDP service is abnormal.
SVC0002	abortReason is null.	The **abortReason** value in the request body is blank.

Error Code	Description	Cause
	Cannot find route policy, the policy identifier is %1.	An internal SDP service is abnormal.
	senderCB is null.	The **senderCB** value in the request body is blank.
	Cannot find USSD transaction, receiverCB is %1.	The session times out, and the SDP cannot match values for **receiverCB**.
	senderCB %1 in message cannot match to the one %2 in original USSD transaction.	The session times out, and the SDP cannot match values for **senderCB**.
	senderCB %1 is invalid format.	The **senderCB** value in the request body is in an incorrect format.
	receiveCB %1 is invalid format.	The **receiveCB** value in the request body is in an incorrect format.
	receiveCB is null.	The **receiveCB** value in the request body is blank.
SVC0901	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 different from that set during SP registration with the SDP.
	Sp password is null!	The **password** value in the request header is blank.
	Sp password is not accepted!	The **password** value in the request header is different from that set by the SP in the SDP.
	SP %1 is in blacklist!	The **spId** value in the request header is blacklisted.
	The sp's Status is unknown.	An internal SDP service is abnormal.
	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.
	The sp's Status is forbidden.	The SP is in the forbidden state.
	The sp 's status is pause.	The SP is in the paused state.
	SP status is locked.	The SP is in the locked state.
	Service ID %1 is not exist!	The **serviceId** value in the request header does not exist in the SDP.
	Service ID is null!	The **serviceId** value in the request header is blank or does not exist in the SDP.

Error Code	Description	Cause
	Service ID %1 is invalid!	The **serviceId** value in the request header is in an incorrect format.
	Service %1 is in blacklist!	The **serviceId** value in the request header is blacklisted.
	The service status is configuring.	The service specified by **serviceId** in the request header is in the configuring state.
	The service status is suspended.	The service specified by **serviceId** in the request header is in the paused state.
	The service status is pre-deregistered.	The service specified by **serviceId** in the request header is in the pre-deregistered state.
	The service status is deregistered.	The service specified by **serviceId** 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.
	The ScfType %1 is inactive!	An internal error occurs in the SDP.
	The ScfType %1 is uninstalled!	An internal error occurs in the SDP.
	Timestamp is empty in soapheader.	The **timeStamp** value in the request header is blank.
	The authentication type is unknown!	An internal SDP service is abnormal.
	Authentication Failed, cause by SP,because of timestamp expired.	The **timeStamp** value in the request header has expired.
	local SP password is null!	An internal error occurs in the SDP.
SVC0905	Authentication Failed, cause by SP,because of timestamp is not valid.	The **timeStamp** value in the request body is in an incorrect format.

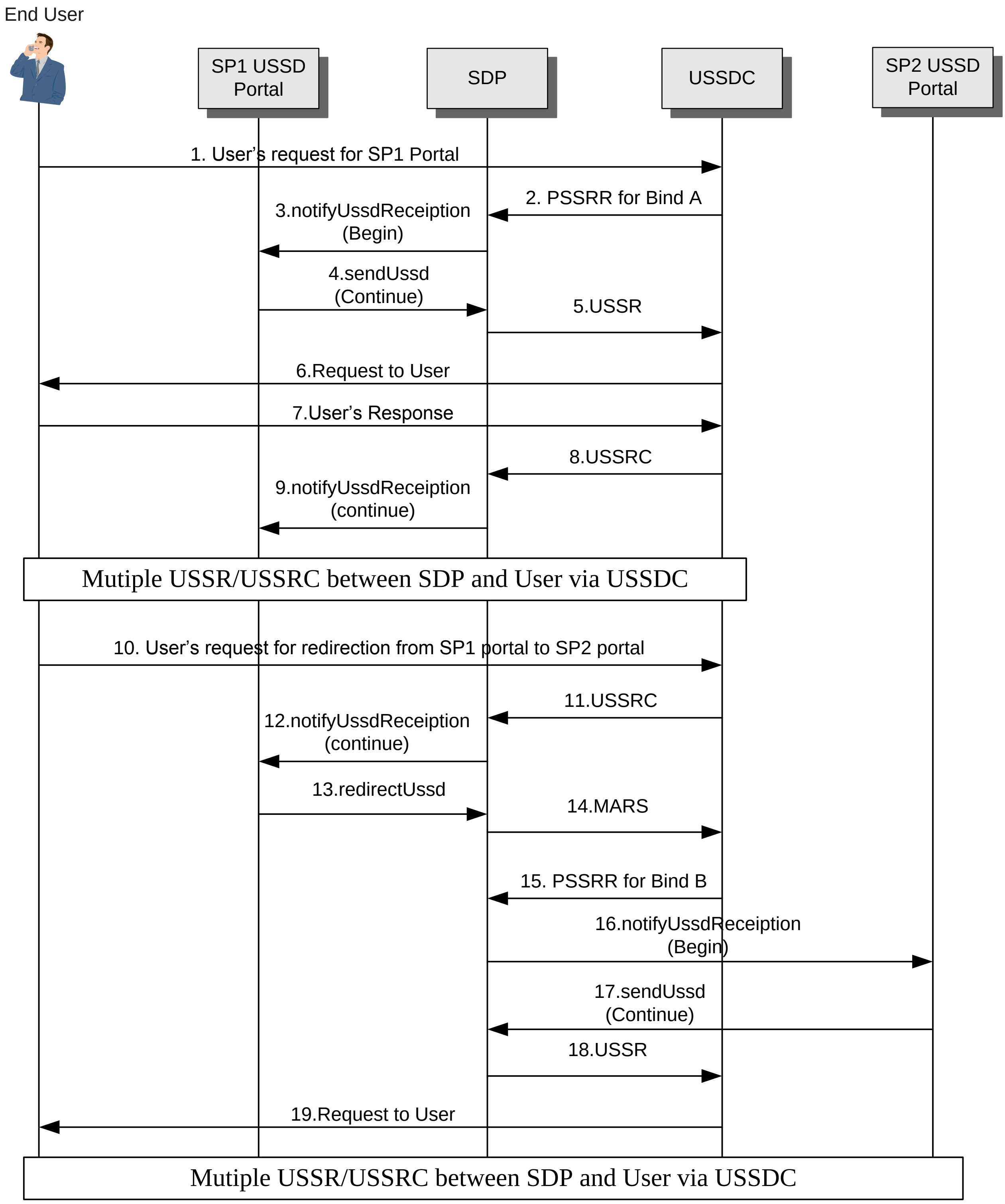
04

Interfaces for Switching USSD session

4.1 Process

Figure 4-1 shows the process of switching USSD session from one USSD portal to another USSD portal.

Figure 4-1 Process of switching USSD session


End User
SP1 USSD Portal
SDP
USSDC
SP2 USSD Portal
1. User's request for SP1 Portal
2. PSSRR for Bind A
3.notifyUssdReceiption (Begin)
4.sendUssd (Continue)
5.USSR
6.Request to User
7.User's Response
8.USSRC
9.notifyUssdReceiption (continue)
Mutiple USSR/USSRC between SDP and User via USSDC
10. User's request for redirection from SP1 portal to SP2 portal
11.USSRC
12.notifyUssdReceiption (continue)
13.redirectUssd
14.MARS
15. PSSRR for Bind B
16.notifyUssdReceiption (Begin)
17.sendUssd (Continue)
18.USSR
19.Request to User
Mutiple USSR/USSRC between SDP and User via USSDC

The process is as follows:

- 1-9: Communication between User and SP1 Portal
- 10-16: User requests to direct the session from SP1 portal to SP2 portal
- 17-19: Communication between User and SP2 Portal

4.2 redirectUSSD

4.2.1 API Function

The App (functioning as the client) invokes the redirectUssd API exposed by the SDP (functioning as the server) to redirect the USSD session from one portal to another portal.
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.

Notes:
This API is only available for South Africa.

4.2.2 Request URI

The request URI is the destination URI of sendUSSD messages sent by the App to the SDP. The URI is provided by the SDP in the following format:
http://*IP*:*Port*/RedirectUssdService/services/redirectUssd
In the format, *IP* and *Port* indicate the service IP address and SOAP port number of the API provided by the SDP. Contact carriers to obtain the IP address and port number.

4.2.3 Request

The App functions as the client and sends a **redirectUSSDRequest** message to the SDP.

Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/ussd/redirect/v1_0/local">
   <soapenv:Header>
      <tns:RequestSOAPHeader xmlns:tns="http://www.huawei.com.cn/schema/common/v2_1">
         <tns:spId>000201</tns:spId>
         <tns:spPassword>e6434ef249df55c7a21a0b45758a39bb</tns:spPassword>
         <tns:timeStamp>20100731064245</tns:timeStamp>
      </tns:RequestSOAPHeader>
   </soapenv:Header>
   <soapenv:Body>
      <loc:redirectUssd>
         <loc:senderCB>12312312</loc:senderCB>
         <loc:receiveCB>4546457</loc:receiveCB>
```

```
            <loc:msIsdn>86121312121</loc:msIsdn>
            <loc:serviceCode>*123*45#</loc:serviceCode>
            <loc:extenionInfo>
               <loc:item>
                  <loc:key>key1</loc:key>
                  <loc:value>value1</loc:value>
               </loc:item>
            </loc:extenionInfo>
         </loc:redirectUssd>
      </soapenv:Body>
   </soapenv:Envelope>
```

Message Header Parameters

Table 4-1 describes parameters in a **redirectUSSDRequest** message header.

Table 4-1 Parameters in a redirectUSSDRequest message header

Parameter	Type	Length	Level of Requirement	Description
spId	xsd: string	21	Mandatory	Partner ID. The ID is automatically allocated by the SDP to partners after successful registration. To obtain the ID: • 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. [Example] 000201

Parameter	Type	Length	Level of Requirement	Description
spPassword	xsd: string	100	Conditional	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: • SHA-256: spPassword = Base64(SHA-256(spId + Password + timeStamp)) • MD5: spPassword = MD5(spId + Password + timeStamp) In the formula: • **spId** and **timeStamp**: authentication ID and timestamp. • **Password**: access password allocated by the SDP to a partner. – 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. **NOTE** To retain features of earlier versions, the SP uses the MD5 algorithm in the connection to the SDP, which might cause security risks. [Example] e6434ef249df55c7a21a0b45758a39bb

Parameter	Type	Length	Level of Requirement	Description
serviceId	xsd: string	21	Conditional	Service ID. The ID is automatically allocated by the SDP to services after successful release. Partner can log in to the SDP Management Portal and query service information for the ID. The **serviceId** 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. [Example] 35000001000029
timeStamp	xsd: string	14	Conditional	Time stamp (UTC time). The value is used in MD5 encryption of **spPassword**. This parameter is mandatory when the **spPassword** parameter is required. [Format] *yyyyMMddHHmmss* [Example] 20100731064245

Message Body Parameters

Table 4-2 describes parameters in a **redirectUssdRequest** message body.

Table 4-2 Parameters in a redirectUSSDRequest message body

Parameter	Type	Length	Level of Requirement	Description
senderCB	xsd:string	10	Mandatory	Initiator session ID. The value is generated when an SP uses the sendUSSDRequest interface to send requests for the first time. In a valid session, the **senderCB** values in the subsequent sendUSSDRequest and sendUssdAbortRequest messages are the same as that in the first sendUSSDRequest message. [Example] 320207133

Parameter	Type	Length	Level of Requirement	Description
receiveCB	xsd:int	10	Mandatory	Recipient session ID. The value is generated by the SDP. • In the first message (**msgType**=**0**) sent by the SDP, the value is **0xFFFFFFFF**. • In the subsequent messages, the value is the same as the **senderCB** value in the first notifyUssdReception message. [Example] 0xFFFFFFFF
msIsdn	xsd:string	30	Mandatory	Mobile number or the fake ID of the message recipient. [Format] • Mobile number: tel:[Prefix][Country code][Mobile number] In the format, *[Prefix]* is 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
serviceCode	xsd:string	20	Mandatory	Name of the message sender, which is displayed on the user terminal. • For an SP, the value is the access code of an SP service. A partner must obtain the access code from the carrier before releasing a service. • For an API Partner user or a Developer, the value is a partner access code allocated by the carrier. The carrier allocates an access code when an enterprise user or a developer purchases a capability product. [Example] 321123
extenionInfo	NamedParameterList	-	Conditional	Extended field. Reserved.

4.2.4 Response

The SDP functions as the server, processes **redirectUSSD** messages received from the App, and sends **redirectUSSDResponse** 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 5 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:redirectUssdResponse
xmlns:ns1="http://www.csapi.org/schema/parlayx/ussd/redirect/v1_0/local">
      <ns1:result>0</ns1:result>
    </ns1:redirectUssdResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Message Body Parameters

Table 4-3 describes the parameter in a redirctUSSDResponse message body.

Table 4-3 Parameters in a redirectUSSDResponse message body

Parameter	Type	Length	Level of Requirement	Description
result	xsd:string	30	Mandatory	Value **0** indicates success. For the error codes, see 4.2.5 Error Codes.

4.2.5 Error Codes

Table 4-4 describes **redirectUSSD** error codes that the SDP may return upon an exception. For details about the error codes, see the *SDP Solution Error Code Reference*.

Table 4-4 sendUSSD error codes

Error Code	Description	Cause
SVC0001	Get addr route info failed!the baseScfType = %1,the sourceAddr = %2,the destAddr = %3.	An internal SDP service is abnormal.
	Waiting for response timed out, message type is %1	An internal SDP service is abnormal.
	Received failed response, message type is %1, error is %2	An internal SDP service is abnormal.

Error Code	Description	Cause
SVC0002	msisdn %1 is invalid format.	The **msisdn** value in the request body is in an incorrect format.
	msisdn %1 is too long.	The **msisdn** value in the request body is too long.
	msisdn is null.	The **msisdn** value in the request body is blank.
	Cannot find route policy, the policy identifier is %1.	An internal SDP service is abnormal.
	senderCB is null.	The **senderCB** value in the request body is blank.
	Cannot find USSD transaction, receiverCB is %1.	The session times out, and the SDP cannot match values for **receiverCB**.
	senderCB %1 in message cannot match to the one %2 in original USSD transaction.	The session times out, and the SDP cannot match values for **senderCB**.
	senderCB %1 is invalid format.	The **senderCB** value in the request body is in an incorrect format.
	receiveCB %1 is invalid format.	The **receiveCB** value in the request body is in an incorrect format.
	receiveCB is null.	The **receiveCB** value in the request body is blank.
SVC0901	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 different from that set during SP registration with the SDP.
	Sp password is null!	The **password** value in the request header is blank.
	Sp password is not accepted!	The **password** value in the request header is different from that set by the SP in the SDP.
	SP %1 is in blacklist!	The **spId** value in the request header is blacklisted.
	The sp's Status is unknown.	An internal SDP service is abnormal.

Error Code	Description	Cause
	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.
	The sp's Status is forbidden.	The SP is in the forbidden state.
	The sp 's status is pause.	The SP is in the paused state.
	SP status is locked.	The SP is in the locked state.
	Service ID %1 is not exist!	The **serviceId** value in the request header does not exist in the SDP.
	Service ID is null!	The **serviceId** value in the request header is blank or does not exist in the SDP.
	Service ID %1 is invalid!	The **serviceId** value in the request header is in an incorrect format.
	Service %1 is in blacklist!	The **serviceId** value in the request header is blacklisted.
	The service status is configuring.	The service specified by **serviceId** in the request header is in the configuring state.
	The service status is suspended.	The service specified by **serviceId** in the request header is in the paused state.
	The service status is pre-deregistered.	The service specified by **serviceId** in the request header is in the pre-deregistered state.
	The service status is deregistered.	The service specified by **serviceId** 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.
	The ScfType %1 is inactive!	An internal error occurs in the SDP.
	The ScfType %1 is uninstalled!	An internal error occurs in the SDP.
	Timestamp is empty in soapheader.	The **timeStamp** value in the request header is blank.
	The authentication type is unknown!	An internal SDP service is abnormal.

Error Code	Description	Cause
	Authentication Failed, cause by SP,because of timestamp expired.	The **timeStamp** value in the request header has expired.
	local SP password is null!	An internal error occurs in the SDP.
SVC0905	Authentication Failed, cause by SP,because of timestamp is not valid.	The **timeStamp** value in the request body is in an incorrect format.
SVC4002	destinationServiceCode is invalid / menu redirection failed	serviceCode is wrong
SVC4003	No existing session exists	Cannot find ussd session
SVC4004	API not allowed in OpCo	Operation is not available for the specific opco.

05

API Error Responses

5.1 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 5-1 describes parameters in a service error response.

Table 5-1 Parameters in a service error response

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

5.2 Policy Error Response

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

Example

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

```
          <messageId>POL0006</messageId>
          <text>GroupAddr is not supported</text>
          <variables>GroupAddr</variables>
        </ns1:PolicyException>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Parameter Description

Table 5-2 describes parameters in a policy error response.

Table 5-2 Parameters in a policy error response

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