Building the Internet of Things

Sierra Wireless Cloud & Connectivity Services

USSD API Developer's guide

Version 1.6

Table of Contents

1. Introduction

This document is the developer's guide of the Sierra Wireless USSD API solution. It provides the reader with details on:

- The USSD Pull feature
- The USSD Push feature

Note:
Sierra Wireless reserves the right of applying a restriction of use through filters on the messages content in case of offending terms being used. Furthermore, use of this service for spamming purposes is forbidden in all its manifestations. As a consequence, Sierra Wireless may suspend the access to the service at any time.

For further information, please contact the Sierra Wireless technical support at the following address: support-cloudconnectivity@sierrawireless.com

2. USSD Push API

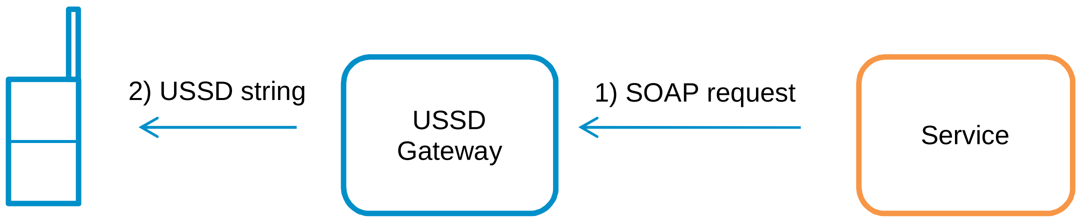
2.1 Introduction

The USSD Push refers to a USSD message being pushed from the network (customer service) to a Sierra Wireless subscriber's module / handset.

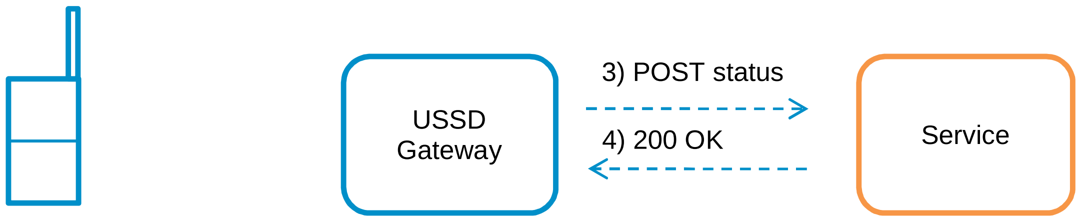
The USSD Push feature is made available to customers via the Sierra Wireless SOAP provisioning API, which aims at offering an HTTP public programing interface exposed through the following URL: https://api.mobiquithings.com/MBQTUssdApiOnlyService.wsdl

2.2 Description

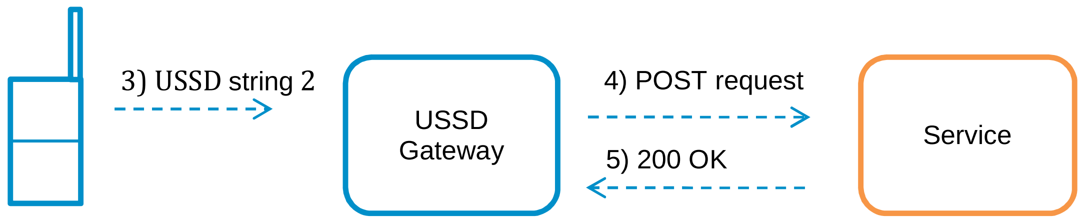
The following schema depicts a simple USSD Push scenario.



Optionally:



Or:



In the Push scenario, a customer service will invoke a SOAP request which will trigger a USSD emission from the USSD Gateway to a subscriber's module.

2.3 Access control

The user needs opening an authentication session before being able to issue any other command.

In order to achieve this goal, a login / password authentication will be required.

Note: The list of users authorized to open a session is defined in advance.

A successful login will provide the user with an authentication token, which will have to be passed to all subsequent requests.

This token is called “sessionAuthId” in the rest of this document, and is defined as a string from 20 to 50 characters.

Sessions are limited in time in case of idle activity from the user:

- Once a session has been opened, a lifespan of the session is determined
- For each new transaction, the session lifespan is extended
- If no transaction has been performed for 60 minutes, the session is terminated and a new session will have to be opened

2.3.1 Introduction

The following functions allow controlling the access to the web service:

Name	Description
login	Open the session

2.3.2 login

2.3.2.1 Description

This function allows the session to be opened.

2.3.2.2 Parameters

Name	Description	Type	Required
userLogin	Login provided by Sierra Wireless	string	yes
userPwd	Password provided by Sierra Wireless (when initializing the customer account)	string	yes

2.3.2.3 Response elements

Name	Description	Type
requestId	Request identifier	integer
sessionAuthId	Current session authentication token	string

2.3.2.4 Examples

Request sample

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/”
xmlns:api=”https://api.mobiquithings.com/”>
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <api:login>
         <userLogin>test_login</userLogin>
         <userPwd>test_pwd</userPwd>
      </api:login>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response sample

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/”
xmlns:ns1=”https://api.mobiquithings.com/”>
   <SOAP-ENV:Body>
      <ns1:loginResponse>
         <requestId>112704929</requestId>
         <sessionAuthId>437571012</sessionAuthId>
      </ns1:loginResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2.4 USSD Broker

2.4.1 Introduction

The user can take advantage of the API to send USSD messages from his own service to the subscriber’s module / handset.

This is commonly known as network initiated USSD, also called “USSD Push” by opposition to mobile originated USSD, also called “USSD Pull”.

The subscription is then charged like any other subscription, and the CDR are integrated to the usual flow.

The following functions allow managing the USSD Broker service:

Name	Description
pushUssd	Send an USSD message

2.4.2 pushUssd

2.4.2.1 Description

This function allows sending a USSD message to a Sierra Wireless subscriber.

So as to push a USSD message via the API, we need to:

- Log in (refer to login)
- Input the destination MSISDN internationally formatted, e.g. 33641620872, which identifies the contract that will be charged
- Input the text of the message
- Input the application ID: this will be a numerical identifier, e.g. 774, that will be associated to an application HTTPS URL. This is the network equivalent to the USSD code concept in the mobile world.
  The identifier will allow a response from the module or a status from the USSD Gateway to be routed to the application

So as to be able to receive a response:

- Use the notification and final flags as required

Note :

- The association between an application ID and an URL will have to be set up by the Sierra Wireless support team before starting to use this service.
- The URL of the originating application will have to be HTTPS based, for security reasons.
- The application ID can be any number, e.g. 12345, provided it was agreed with the Sierra Wireless support team.

2.4.2.2 Parameters

Name	Description	Type	Required
sessionAuthId	Current session authentication token	string	yes
toMsisdn	Destination MSISDN internationally formatted	string	yes
ussdString	Text to be received on the subscriber module / handset. The string will have to be plain text (UTF8) if 7 bit GSM encoding is used (the default), or a hexadecimal string if a binary encoding is chosen	string	yes
appId	Application identifier. Do not use your own identifier since there must exist a mapping between this ID and the customer service URL in the USSD Gateway. Typically, the customer is advised to use his USSD short code	integer	yes
notification	“true” if the request does not expect a response, “false” otherwise. Default is true	boolean	
final	“true” if the request terminates the dialog, “false” otherwise. Default is true	boolean	
encoding	Encoding scheme (see DCS 3GPP 23.038). Default is 7 bit GSM encoding. The following values are allowed: **Value** – **Description** 15 – 7 bit GSM encoding 68 – 8 bit encoding	integer	

	17	UCS 2 encoding		

2.4.2.3 Response elements

Name	Description	Type
requestId	Request identifier	integer
sessionId	Dialog identifier	integer

2.4.2.4 Examples

Request sample

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="https://api.mobiquithings.com/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <api:pushUssd>
         <sessionAuthId>437571012</sessionAuthId>
         <toMsisdn>44649955679</toMsisdn>
         <ussdString>Your request has been processed</ussdString>
         <appId>774</appId>
         <notification>false</notification>
         <final>true</final>
         <encoding>15</encoding>
      </api:pushUssd>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response sample

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="https://api.mobiquithings.com/">
   <SOAP-ENV:Body>
      <ns1:pushUssdResponse>
         <requestId>112707536</requestId>
         <sessionId>13708</sessionId>
      </ns1:pushUssdResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2.5 POST Request

2.5.1 Description

In case a response from the module is required by the push request, this response will be forwarded to the customer service as a POST request.

A dialog may then establish between the subscriber’s module and the customer HTTPS service.

<u>Note:</u>
In order to be able to route the request to the customer service, the pushUssd() appId parameter will have to be set, and mapped to a proper HTTPS URL.

The POST request will contain the following parameters:

Name	Description	Type	Example
msisdn	Originator MSISDN (x-www-form-urlencoded/charset utf8 encoding)	string	+33641620872
ussdstring	Text input on the subscriber module / handset (x-www-form-urlencoded / charset UTF8)	string	anything
sessionid	USSD dialog identifier	integer	13708
encoding	Encoding scheme (see DCS 3GPP 23.038). Default is 7 bit GSM encoding. The following values are allowed: **Value** | **Description** 15 | 7 bit GSM encoding 68 | 8 bit encoding 17 | UCS 2 encoding	integer	

2.5.2 POST Request example

```
POST https://customerserverurl
Content-Type: application/x-www-form-urlencoded
…
msisdn=+33641620872&ussdstring=*774#string1#&sessionid=13481608&ussdcode=774&encodi
ng=15
```

2.6 Response to POST Request

2.6.1 Description

The response expected from the customer service will have to use the following parameters:

Name	Description	Type	Required

ussdstring	Text to be received on the subscriber module / handset. The string will have to be x-www-form-urlencoded / UTF8 if 7 bit GSM encoding is used (the default), or a hexadecimal string if a binary encoding is chosen.	string	Yes
notification	"true" if the request does not expect a response, "false" otherwise. Default is false	boolean	
final	"true" if the request terminates the dialog, "false" otherwise. Default is true	boolean	
encoding	Encoding scheme (see DCS 3GPP 23.038). Default is 7 bit GSM encoding. The following values are allowed:	integer	
	Value — **Description**		
	15 — 7 bit GSM encoding		
	68 — 8 bit encoding		
	17 — UCS 2 encoding		

2.6.2 Response example

```
HTTP/1.1 200 OK
Content-Type: application/x-www-form-urlencoded
…
final=false&notification=false&ussdstring=string2&sessionid=13481608
```

2.7 POST Status

2.7.1 Description

A POST status request may be returned to the customer service when the dialog is terminated.

E.g.

- No response is expected from the module, or
- A response is expected from the module, but not sent (e.g. dialog cancelled by the module).

Note: In order to be able to route the status to the customer service, the pushUssd() appId parameter will have to be set, and mapped to a proper HTTPS URL.

The POST status will contain the following parameters:

Name	Description	Type	Example
status	Termination status. This status will be : • greater than or equal to 0 if no error • negative in case of error. Please refer to Section 4 for details on error codes	integer	-39
sessionid	USSD dialog identifier	integer	13708

2.7.2 POST Status example

```
POST https://customerserverurl
Content-Type: application/x-www-form-urlencoded
…
status=-39&final=true&msisdn=+33641620872&sessionid=13481608
```

2.8 PHP sample code

The PHP example shown below is a scenario where the user wishes to push a USSD message expecting a response to be sent back.

The user then performs the following steps:

- Initialize a SOAP client in WSDL mode
- Log in
- Push the USSD

```
<?php
// ***********************************************************************
//
//                         Sierra Wireless
//                 Copyright 2013 All Rights Reserved.
//
//     These computer program listings and specifications, herein,
//     are the property of Sierra Wireless and shall not be
//     reproduced or copied or used in whole or in part as the basis
//     for manufacture or sale of items without written permission.
//
// @description  Sierra Wireless Subscription SOAP API pushUssd sample.
//
// ***********************************************************************

// New SOAP Client
$client = new
SoapClient('https://api.mobiquithings.com/MBQTUssdApiOnlyService.wsdl',
           array('trace' => 1,
           'encoding' => 'UTF-8',
           'features' => SOAP_SINGLE_ELEMENT_ARRAYS));
try {

    // Get a new session authentication ID
    $result = $client->login('myLogin', 'myPassword');
    $sessionAuthId = $result['sessionAuthId'];

    // Send a USSD message to '33641620872'
    $result = $client->pushUssd($sessionAuthId,
                                '33641620872',
                                'Your request has been processed',
                                774, // application ID
                                false, // response expected from device
                                false); /* do not terminate the dialog after
                                           first response sent */
    var_dump($result);

} catch (SoapFault $fault) {
    trigger_error("SOAP Fault: (faultcode: {$fault->faultcode}, faultstring:
{$fault->faultstring})",
                  E_USER_ERROR);
}
```

In case the user did not expect a response from the destination, we could simply have used:

```
$result = $client->pushUssd($sessionAuthId,
                            '33641620872',
                            'Your request has been processed',
                            774);
```

And the result is:

```
array(2) {
  ["requestId"] =>  int(112707536)
  ["sessionId"] =>  int(13708)
}
```

2.9 Java sample code

This section illustrates the steps involved for sending a USSD message in Java using the command line.

Pre-requisites:

- Install the Oracle JDK
- Generate the proxy from the WSDL file

Source code steps:

- Instantiate the MBQTUssdApiOnlyService web service
- Log in
- Push the USSD

2.9.1 Generating the WSDL proxy

This step will generate the Java artefacts inside a Java package named com.mobiquithings.api by importing https://api.mobiquithings.com/MBQTUssdApiOnlyService.wsdl, and compile them.

It needs to be done only once.

We will need this package to compile the client.

```
wsimport -p com.mobiquithings.api -b bindings.xml
https://api.mobiquithings.com/MBQTUssdApiOnlyService.wsdl -keep
```

-b <bindings> Specifies some bindings customization

-p Specifies the package name

-keep <pkg> Keeps the generated java files

See http://docs.oracle.com/javase/8/docs/technotes/tools/windows/wsimport.html for more information.

The bindings.xml contents are:

```
<jaxws:bindings xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                xmlns:jaxws="http://java.sun.com/xml/ns/jaxws"

wsdlLocation="https://api.mobiquithings.com/MBQTUssdApiOnlyService.wsdl">

  <jaxws:bindings
node="wsdl:definitions/wsdl:portType[@name='MBQTUssdApiOnlyService']/wsdl:operation
[@name='pushUssd']">

    <jaxws:parameter

part="wsdl:definitions/wsdl:message[@name='pushUssdRequest']/wsdl:part[@name='final
']"
       childElementName="final" name="finalParam" />
  </jaxws:bindings>
</jaxws:bindings>
```

Note:
The customization done through bindings.xml is required so as to work around the following error:

[ERROR] Invalid operation "pushUssd", can't generate java method. Parameter: part "final" in wsdl:message "pushUssdRequest", is a java keyword. Use customization to change the parameter name or change the wsdl:part name.

It simply renames the "final" element by "finalParam".

2.9.2 Source code: PushUssd.java

```
// ***********************************************************************
//
//                          Sierra Wireless
//                 Copyright 2015 All Rights Reserved.
//
//     These computer program listings and specifications, herein,
//     are the property of Sierra Wireless and shall not be
//     reproduced or copied or used in whole or in part as the basis
//     for manufacture or sale of items without written permission.
//
// @description  Sierra Wireless Subscription SOAP API: PushUssd Java sample
//
// ***********************************************************************

import com.mobiquithings.api.MBQTUssdApiOnlyService;
import com.mobiquithings.api.MBQTUssdApiOnlyService_Service;
import java.math.BigInteger;
import javax.xml.ws.Holder;

public class PushUssd {

    public static void main(String[] args) {

        MBQTUssdApiOnlyService_Service mbqtServiceImpl = new
MBQTUssdApiOnlyService_Service();
        MBQTUssdApiOnlyService mbqtService =
mbqtServiceImpl.getMBQTUssdApiOnlyServiceSOAP();

        final Holder<BigInteger> requestId = new Holder<BigInteger>();
        final Holder<String> sessionAuthId = new Holder<String>();

        mbqtService.login("myLogin", "myPassword", requestId, sessionAuthId);
        System.out.println("sessionAuthId="+sessionAuthId.value);

        final Holder<BigInteger> sessionId = new Holder<BigInteger>();
        mbqtService.pushUssd(sessionAuthId.value,
                             "33641620872",
                             "Your request has been processed"+,
                             BigInteger.valueOf(774),
                             false/*notification*/,
                             false/*final*/,
                             BigInteger.valueOf(15),
                             requestId,
                             sessionId);
        System.out.println("sessionId="+sessionId.value);
    }
}
```

2.9.3 Compiling and executing

This step reads the client source code and the proxy classes, and produces a .class bytecode.

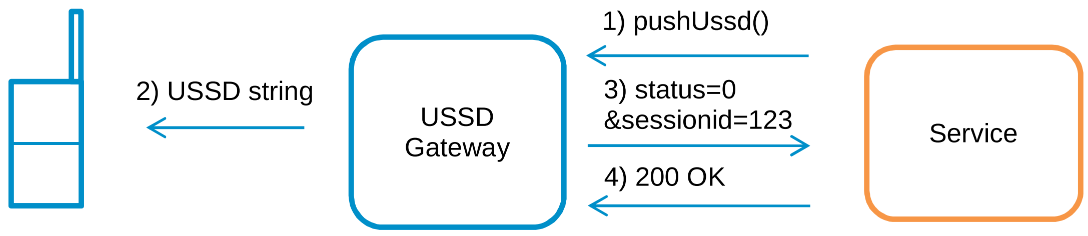
```
javac -classpath . PushUssd.java
```

The client can be tested by executing the .class obtained.

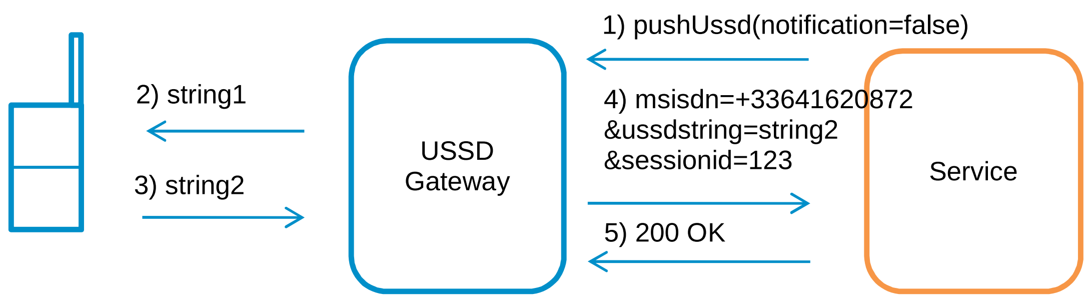
```
java -classpath . PushUssd

sessionAuthId=4177454004
sessionId=13926487
```

2.10 Use cases

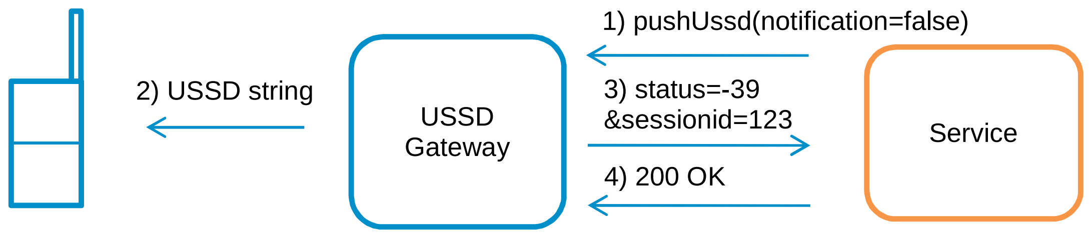
appId mapped to URL (e.g. USSD code) and no response expected



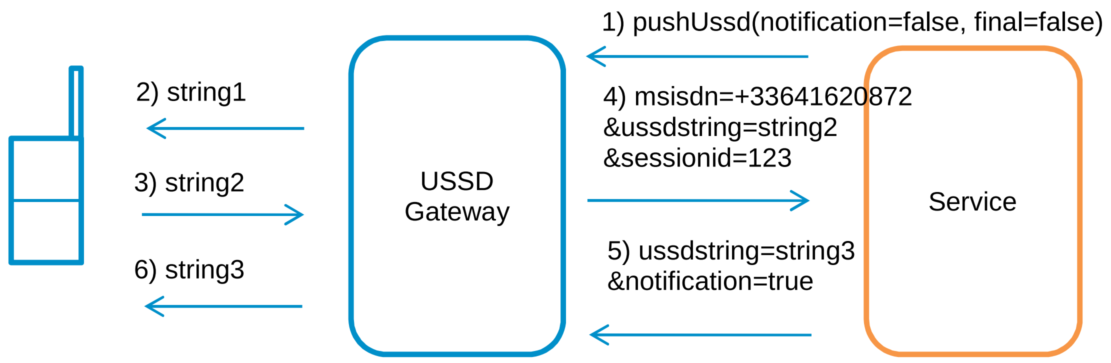
appId mapped to URL (e.g. USSD code) and response expected and sent back


1) pushUssd(notification=false)
2) string1
3) string2
USSD Gateway
4) msisdn=+33641620872 &ussdstring=string2 &sessionid=123
Service
5) 200 OK

appId mapped to URL (e.g. USSD code) and response expected but not sent back



appId mapped to URL (e.g. USSD code) and dialogue


1) pushUssd(notification=false, final=false)
2) string1
3) string2
6) string3
USSD Gateway
4) msisdn=+33641620872 &ussdstring=string2 &sessionid=123
Service
5) ussdstring=string3 &notification=true

2.11 USSD Push setup requirements

From the customer

- Provide some application identifiers to be mapped to an HTTPS URL in case a USSD dialog is to be established during the push scenario.

From Sierra Wireless

- Provide a login / password in order to be able to use the SOAP API.
- Provide a consumption profile that enables USSD usage.
- Setup the application ID to HTTPS URL mapping.

3. USSD Pull API

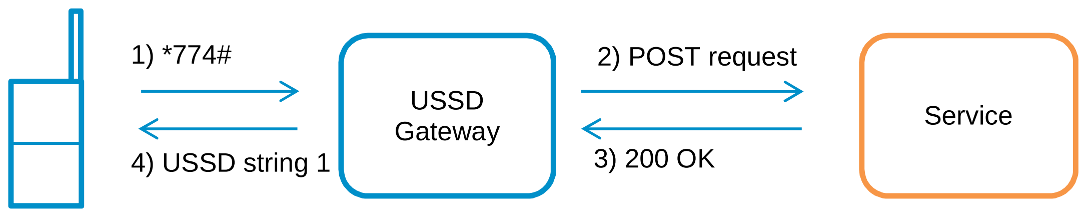
3.1 Introduction

The USSD Pull refers to a USSD message being sent in from a Sierra Wireless subscriber's module / handset to a customer HTTPS service.

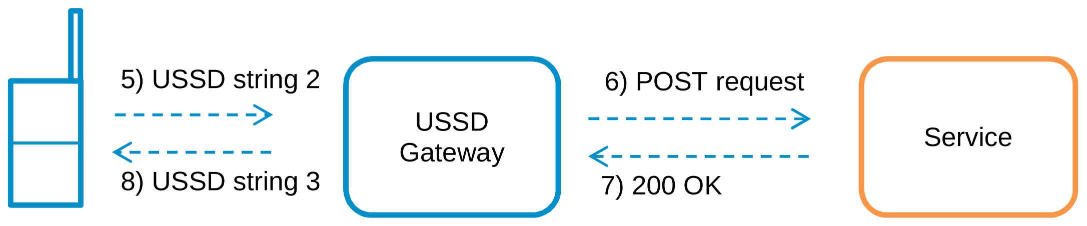
The USSD Pull feature is made available to customers via the availability of a USSD code mapped to a remote HTTPS address.

3.2 Description

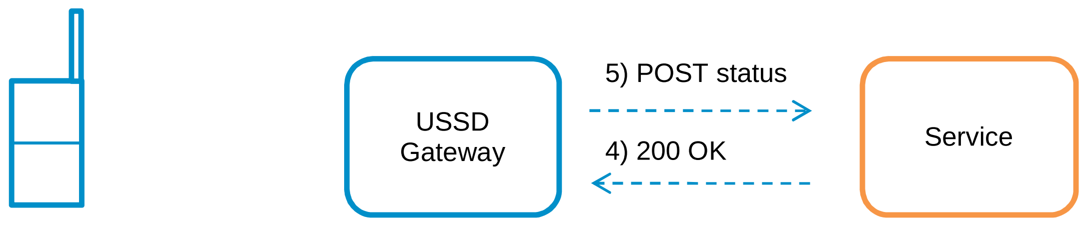
The following schema depicts a simple USSD Pull scenario.



Optionally:



Or:



In the Pull scenario, a subscriber's module will emit a USSD short string, e.g. *774#, which will trigger an HTTPS POST request to a customer URL previously mapped to this USSD code.

The customer service will then reply **in the body of the HTTPS POST response**, which will be transferred to the subscriber’s module by the USSD Gateway.

The dialog may then continue as long as parameters transmitted in the response require it.

3.3 POST Request

3.3.1 Description

The first POST request that will be sent to the customer service will contain the following parameters:

Name	**Description**	**Type**	**Example**
ussdstring	Text input on the subscriber module / handset (x-www-form-urlencoded / charset UTF8)	string	*774#
sessionid	USSD dialog identifier	Integer	13708
msisdn	Originator MSISDN (x-www-form-urlencoded/charset utf8 encoding)	string	+33641621425
ussdcode	USSD short string	string	774
encoding	Encoding scheme (see DCS 3GPP 23.038). Default is 7 bit GSM encoding. The following values are allowed:	integer	15

Value	**Description**
15	7 bit GSM encoding
68	8 bit encoding
17	UCS 2 encoding

The following POST requests if any will contain the following parameters:

Name	**Description**	**Type**	**Example**
ussdstring	Text input on the subscriber module / handset (x-www-form-urlencoded / charset UTF8)	string	anything
sessionid	USSD dialog identifier	Integer	13708

msisdn	Originator MSISDN (x-www-form-urlencoded/charset utf8 encoding)	string	+33641621425

3.3.2 POST Request example

```
POST https://customerserverurl
Content-Type: application/x-www-form-urlencoded
…
msisdn=+33641620872&ussdstring=*774#string1#&sessionid=13481608&ussdcode=774&encodi
ng=15
```

3.4 Response to POST Request

3.4.1 Description

The response expected from the customer service will have to use the following parameters:

Name	Description		Type	Required
ussdstring	Text to be received on the subscriber module / handset. The string will have to be x-www-form-urlencoded / UTF8 if 7 bit GSM encoding is used (the default), or a hexadecimal string if a binary encoding is chosen.		string	yes
notification	"true" if the request does not expect a response, "false" otherwise. Default is false		boolean	
final	"true" if the request terminates the dialog, "false" otherwise. Default is true		boolean	
encoding	Encoding scheme (see DCS 3GPP 23.038). Default is 7 bit GSM encoding. The following values are allowed:		integer	
	Value	**Description**		
	15	7 bit GSM encoding		
	68	8 bit encoding		
	17	UCS 2 encoding		

Service example:
The PHP code shown below is a simplistic example of what can be executed when posting a request to the customer HTTPS address.

```
<?php
// ***********************************************************************
//
//                           Sierra Wireless
//                  Copyright 2013 All Rights Reserved.
//
//     These computer program listings and specifications, herein,
//     are the property of Sierra Wireless and shall not be
//     reproduced or copied or used in whole or in part as the basis
//     for manufacture or sale of items without written permission.
//
// @description  Sierra Wireless USSD API HTTP POST response sample.
//
// ***********************************************************************

$now = date("Y-m-d H:i:s");
header("HTTP/1.1 200 OK");
$response = array('ussdstring='.urlencode("test response @ $now"),
                  'notification=false',
                  'final=false');
print(implode('&', $response));
exit(0);
```

3.4.2 response example

```
HTTP/1.1 200 OK
Content-Type: application/x-www-form-urlencoded
…
final=false&notification=false&ussdstring=string2&sessionid=13481608
```

3.5 POST Status

3.5.1 Description

A POST status request may be returned to the customer service when the dialog is terminated.

E.g. A response is expected from the module, but not sent (e.g. dialog cancelled by the module).

The POST status will contain the following parameters:

Name	Description	Type	Example
status	Termination status. This status will be : • greater than or equal to 0 if no error • negative in case of error. Please refer to Section 4 for details on error codes	integer	-39

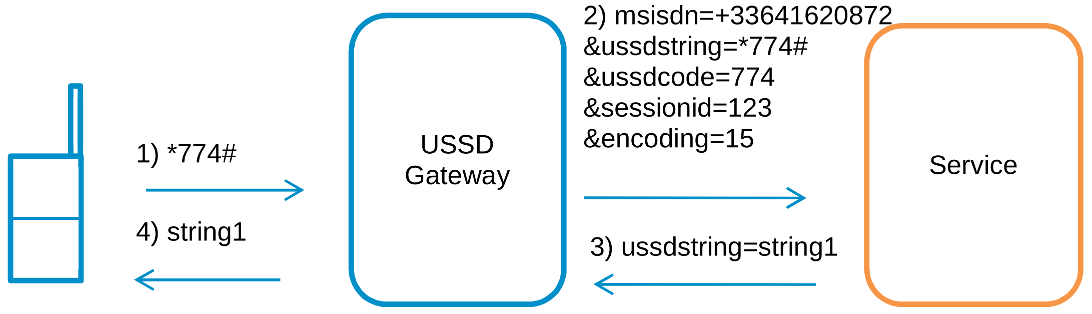
sessionid	USSD dialog identifier	integer	13708

3.5.2 POST Status example

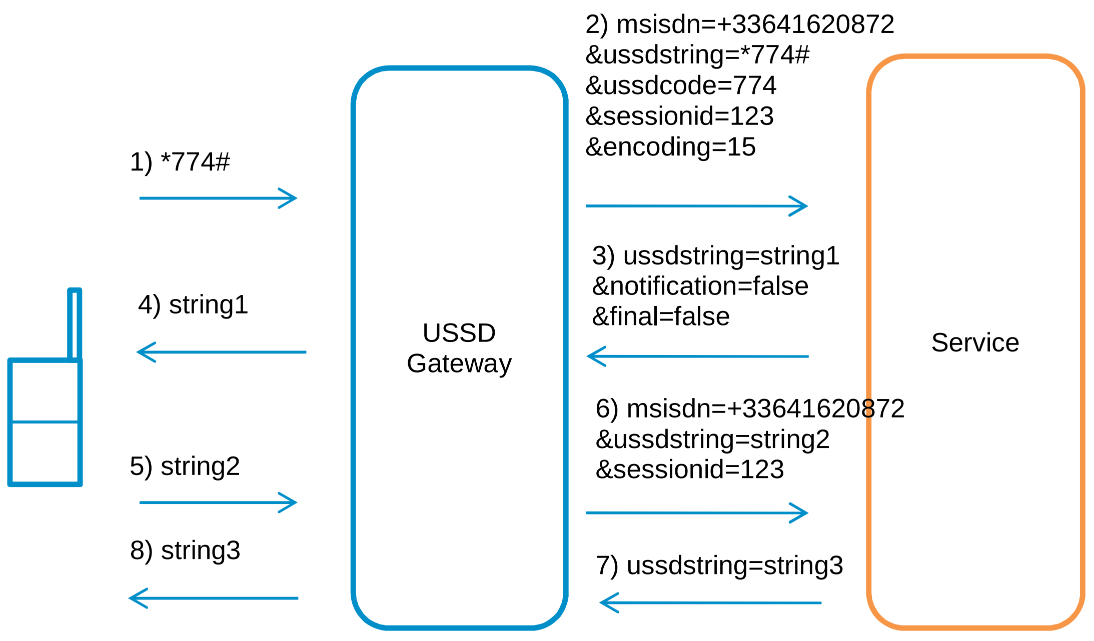
```
POST https://customerserverurl
Content-Type: application/x-www-form-urlencoded
…
status=-39&final=true&msisdn=+33641620872&sessionid=13481608
```

3.6 Use cases

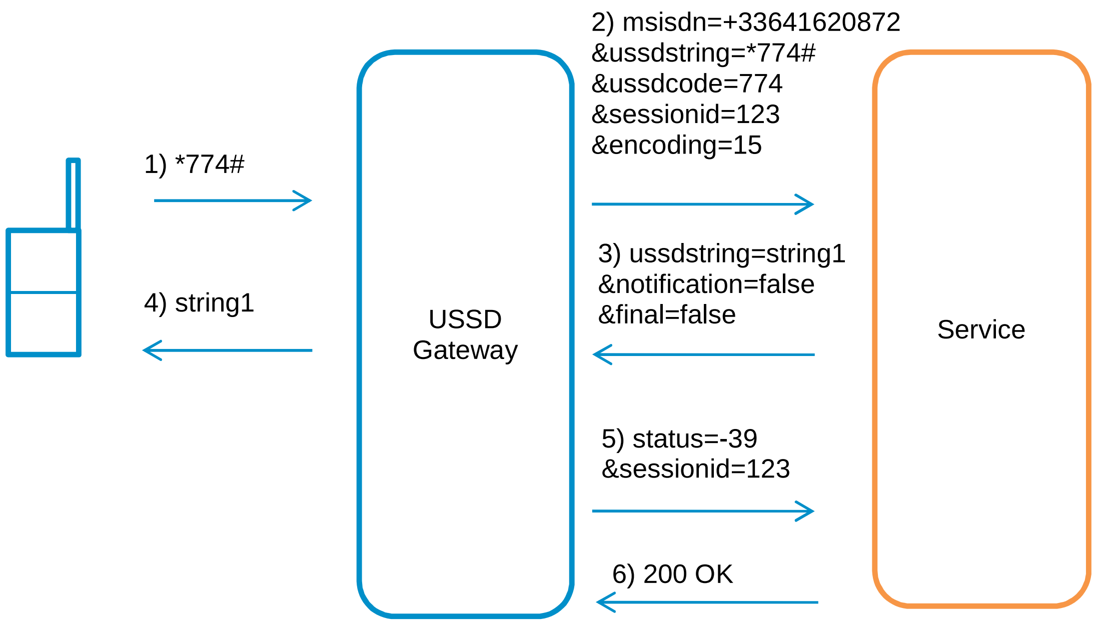
Simple dialog


1) *774#
4) string1
USSD Gateway
2) msisdn=+33641620872 &ussdstring=*774# &ussdcode=774 &sessionid=123 &encoding=15
3) ussdstring=string1
Service

Response expected and sent back


1) *774#
4) string1
5) string2
8) string3
USSD Gateway
2) msisdn=+33641620872 &ussdstring=*774# &ussdcode=774 &sessionid=123 &encoding=15
3) ussdstring=string1 &notification=false &final=false
6) msisdn=+33641620872 &ussdstring=string2 &sessionid=123
7) ussdstring=string3
Service

Response expected but not sent back


1) *774#
4) string1
USSD Gateway
2) msisdn=+33641620872 &ussdstring=*774# &ussdcode=774 &sessionid=123 &encoding=15
3) ussdstring=string1 &notification=false &final=false
5) status=-39 &sessionid=123
6) 200 OK
Service

3.7 USSD Pull setup requirements

From the customer

- Provide some USSD short codes to be mapped to an HTTPS URL.

From Sierra Wireless

- Provide a consumption profile that enables USSD usage.
- Setup the USSD short code to HTTPS URL mapping.

4. USSD API Error Codes

When an error occurs during a call to the USSD API, the status parameter will be negative and provide the error code. The list of available errors is cited here for reference only in order to qualify the error and provide more details explaining why the USSD is not yet processed.

Error Code	**Error Message**	**Details**
4	Default error	-
6	Bad Parameter	Failed because a "*malformed*" information. This could be data with a bad value or a DCS not supported by the module (eg UCS instead of 7 bit)
8	*Timeout - No Response from Peer*	Failed because a "*timeout*" error. There is no response from the module or a lost message.
14	*Unspecified - System failure*	Failed because issue on '*system failure*' error on network or module
15	*Unsupported*	The module is prohibited, or does not support USSD.
20	*Missing subscriber*	The destination subscriber is not registered into the network.
21	*Unreachable subscriber*	The destination subscriber is not reachable in the destination mobile network : Recipient module is turned off, or temporary without coverage.
27	*Missing parameter - Data missing*	Failed because a "missing parameter". Data not present in the message.
39	*Rejected*	The USSD dialog has been interrupted or canceled.
40	*Subscriber busy*	USSD session already on going