USSD S-Gateway XML/TCP/IP Interface

User's Guide v1.7

Version control

Version	Date	Author	Comment
R0	2007/07/11	Alejandro Leib	Initial Version
V1.1	2009/02/04	Alejandro Leib	Added RequestShortCode API Call
V1.2	2014/11/13	Alejandro Leib	XML updates
V1.3	2015/03/09	Alejandro Leib	HA Support
V1.4	2015/04/20	Alejandro Leib	Generic Information on DialogId
V1.5	2016/01/18	Alejandro Leib	Origination replaced by Orgination_Reference and Operation by Op in XMLs.
V1.6	2016/01/19	Alejandro Leib	Annex B added
V1.7	2016/04/11	Alejandro Leib	UCS2 Support added

Notice

LeibICT

Montevideo, Uruguay Tel: +598 2 614 11 93 Fax: +598 2 203 66 54

http://www.leibict.com

Warranty

LeibICT Solutions keep a technical support department with the only purpose of providing efficient and reliable service.

All LeibICT products are warranted against defects in material and workmanship. The period of coverage and other warranty details are specified in the LeibICT terms and conditions warranty. In no event shall LeibICT be liable for incidental or consequential damages in connection with, or arising from use of any LeibICT product.

Table of Contents

Introduction

LeibICT USSD S-Gateway supports applications over MAP (Mobile Application Part) (phase 2 and 3), either mobile or USSD application originated, using the message set defined in the protocol. The mobile originated data are received by the USSD application and forwarded to the HLR (Home Location Register), where as the network originated messages are forwarded to the HLR at once.

This document presents a description of the XML/TCP/IP Interface supported and managed by the LeibICT's USSD S-Gateway.

Capacity

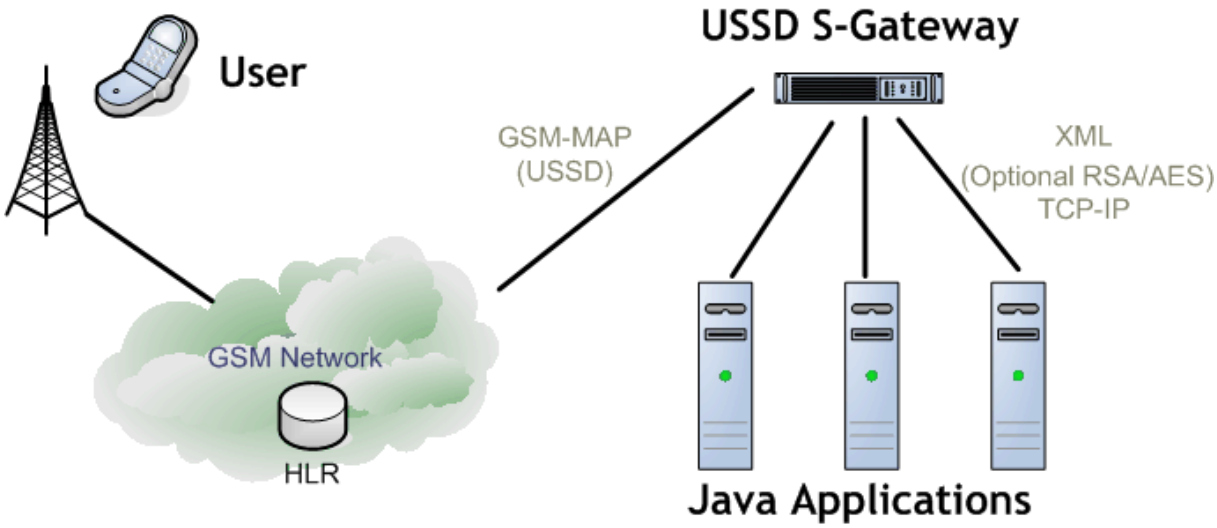
The current USSD S-Gateway version supports 65356 incoming and outgoing messages.

Each USSD transaction is identified by a dialog identifier (field named: *DialogueId*), represented by a 32bits integer. This parameter must be included in each dialogue message so the USSD application must be capable of controlling such parameter. For each outgoing message coming from the application, the latter must include a *Dialogue Id* between 0 and 65356.

The RSA encryption and decryption process is very high cpu consuming, setting up the right key length is needed to depending of the traffic estimated.

Typical system structure

The LeibICT USSD S-Gateway provides Load Balancing and Fault-Tolerance of applications:



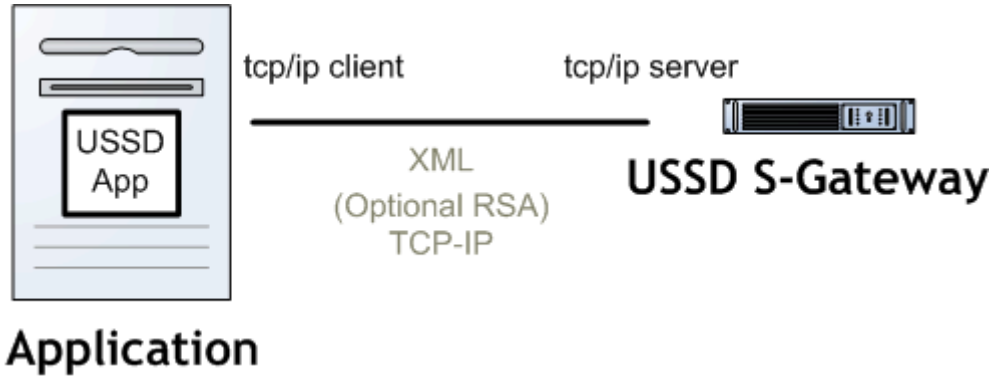
Two USSD S-Gateways can be paired to provide Fault-Tolerance on the gateway and GSM Network level.

TCP/IP Architecture

The XML/TCP/Interface consists in opened port and several XML messages being transmitted and received.

The USSD S-Gateway may send pings (a specific XML messages) within an internal pattern.

Also the application should ping the GW to test the connection.



Each XML messages is transferred using a 32bit unsigned integer valued that represents the length in octets of the XML string.

The XML string can potentially have the ending zero (\0) octet or not.

The XML/TCP/Interafce is free to discard any non-complian package without notification to the application.

XML Messages

.1 Ping/Pong

The first message the server will send is a ping, it must be replied with a pong:

<siggw><msg type="ping"/></siggw>

<siggw><msg type="pong"/></siggw>

.2 Request ShortCode

Optionally, the application can request the traffic for a specific shortcode.

<siggw>

<msg type="hello">

<id v="0"/>

*<ussd_shortcode_request v="*123#"/>*

</msg>

</siggw>

This is useful only on multi-shortcode, multi-application scenario.

For High Avilability GW scenario, the application must request the shortcode to both Gws with the call:

<siggw>

<msg type="hello">

<id v="0"/>

*<ussd_shortcode_request_ha v="*123#"/>*

<version v="1"/>

</msg>

</siggw>

.3 Process USSD Request (invoke)

Once a subscriber starts a transaction the first messages contains the MSISDN of the subscriber and the short code entered.

```
<siggw>
        <msg type="begin">
                <user id="0"/>
                <dialog id="1"/>
                <context v="04000001001302"/>
                <user_info>
                        <object_identifier v="04000001010101"/>
                        <map op="open">
                                <param      name="destination_reference"      number="5989999909"
nai="1" npi="1"/>
                                <param      name="origination_reference"      number="5989925603"
nai="1" npi="1"/>
                        </map>
                </user_info>
                <component type="invoke" id="1" op="process_ussd_request">
                        <param name="data_coding_scheme" v="15"/>
                        <param name="ussd_string" v="2A31323323"/>
                </component>
        </msg>
</siggw>
```

The USSD interaction is drived by a "dialog" each dialog is identified by a number (in the example the first "1" in red)

.4 USSD Request (invoke)

The menus and prompts are sent to the subscriber using the next type of messages.

```
<siggw>
        <msg type="continue">
                <user id="0"/>
                <dialog id="1"/>
                <context v="04000001001302"/>
                <user_info>
                        <map op="accept"/>
                </user_info>
                <component type="invoke" id="2" op="ussd_request">
                        <param name="data_coding_scheme" v="15"/>
                        <param                                          name="ussd_string"
v="4669727374204D656E750A312E20746573740A322E20696D7075740A332E20746F2065786974"/>
                </component>
        </msg>
</siggw>
```

The invoke id (in the example, the "2" in red) must be increased every time during the same dialog.

.5 USSD Request (result)

The result of a menu or prompt is received by the next type of messages:

```
<siggw>
        <msg type="continue">
                <user id="0"/>
                <dialog id="1"/>
                <component type="return_l" id="2" op="ussd_request">
                        <param name="data_coding_scheme" v="15"/>
                        <param name="ussd_string" v="32"/>
                </component>
        </msg>
</siggw>
```

.6 Process USSD Request (result)

To end a ussd dialog the next message is used:

```
<siggw>
        <msg type="end">
                <user id="0"/>
                <dialog id="1"/>
                <component type="return_l" id="1" op="process_ussd_request">
                        <param name="data_coding_scheme" v="15"/>
                        <param                                          name="ussd_string"
v="596F7520656E746572656420223222"/>
                </component>
        </msg>
</siggw>
```

The final text to the subscriber is sent in red.

Note: the conext and suser information must be sent if the end correspond to the first anwser.

.7 Any Time Interrogation (invoke)

This command applies to LBS Gateway only.

To request the location of a subscriber:

```
<siggw>
        <msg type="begin">
                <user id="0"/>
                <dialog id="1"/>
                <context v="04000001001D03"/>
                <component type="invoke" id="0" op="any_time_interrogation">
                        <param name="subscriber_identity">
                                <param   name="msisdn"   number="18763808880"   nai="1"
npi="1"/>
                        </param>
                        <param name="requested_info">
                                <param name="location_information"/>
                        </param>
                        <param  name="gsm_scf_address"  number="18763808847"  nai="1"
npi="1"/>
                </component>
        </msg>
</siggw>
```

.8 Any Time Interrogation (result)

The successful result to the ATI invoke:

```
<siggw>
        <msg type="end">
                <user id="0"/>
                <dialog id="0"/>
                <context v="04000001001D03"/>
                <component type="return_l" id="0" op="any_time_interrogation">
                        <param name="subscriber_info">
                                <param name="location_information">
                                        <param name="cell_glob_id_or_lai">
                                                <param
name="cell_glob_id_or_serv_area_fixed_length" mcc="506" mnc="563" lca="1" cid="4093"/>
                                        </parm>
                                </param>
                        </param>
                </component>
        </msg>
</siggw>
```

.9 USSD Notify (invoke)

```
<siggw>
        <msg type="begin">
                <user id="0"/>
                <dialog id="2"/>
                <destination id="0"/>
                <context v="04000001001302"/>
                <user_info>
                        <object_identifier v="04000001010101"/>
                        <map op="open">
                                <param                          name="destination_reference"
number="59899256037"  nai="1" npi="1"/>
                                <param name="origination_reference" number="59899998931"
nai="1" npi="1"/>
                        </map>
                </user_info>
                <component type="invoke" id="1" op="unstructured_ss_notify">
                        <param name="data_coding_scheme" v="15"/>
                        <param name="ussd_string" v="746573742074657874"/>
                        <param name="msisdn" number="59899256037" nai="1" npi="1"/>
                </component>
        </msg>
</siggw>
```

.10 Abort

In case the subscribers end the dialog the next messages is received:

```
<siggw>
        <msg type="abort">
                <user id="0"/>
                <dialog id="1"/>
        </msg>
</siggw>
```

.11 Association Status

Association Status message is a broadcast to all applications informing of a status change in an SCTP link.

```
<siggw>
        <msg type="sctp_association_status">
                <association id="1" name="HLR"/>
                <status v="1"/>
        </msg>
</siggw>
```

Status values 1 is InService, 0 is Out Of Service.

.12 Route Status

Route Status message is a broadcast to all applications informing of a status change in a MTP3 route.

```
<siggw>
        <msg type="mtp_route_status">
                <route id="1" name="Route to HLR"/>
                <status v="1"/>
        </msg>
</siggw>
```

Status values 1 is InService, 0 is Out Of Service.

.13 Other Messages

The application must support receving undocumented messages, simply discarding them as warnings.

Its unaceptable an application that hangs, slow downs, or become unstable by adding new parameters to the XML messages documented or adding new XML messages.

In the other direction, the Gateway supports receiving malformed, corrupted and out of sequence messages: the Gatway will not hang, slow down or become unstable.

Dialog ID

The USSD/LBS Gateway uses a DialogID in order to identify a transaction.

The DialogID is a integer value between 0 and 65536.

Intcoming and Outgoing transactions have different selection algorithm and ranges as described next:

.1 Incoming transactions

Incoming transactions have a DialogID between 32768 and 65536.

The Gateway will select a temporary DialogID while the dialog exists, after the dialog is closed, the Gateway is free to reuse the same DialogID.

The selection algorithm sequencialy incrases one unit from the minimum value up to the maximum value and then restart from minimum again.

.2 Outgoing transactions

Outgoing transactions have a DialogID between 0 and 32768.

The application staring and outgoing transaction must use one of these dialogIDs and its free to use sequencial, increasing, decreasing, random or any special algorigthm.

If multiple applications are starting outgoint transactions at the same time they must be able to not cloide.

Colission Avoidance could be as simple as Application A uses DialogId from 0 to 1000, Application B uses DialogId from 1000 to 2000, and so on.

Smart algorithm could be much simplier, for example: if all applications reply to a incoming transaction (which is defined by the USSDGW and its unique at any moment) you can use:

Ougoing DialogID = Incoming DialogID – 32768.

So for example, if a USSD transaction comes to the application on dialog 32800, its safe to generate a Outgoing ATI request on dialog 32, as there is no other application having the Incoming Dialog 32800, there will be no other application using Outgoing dialog 32.

Component ID

The Component ID identifies a pair of Invoke/Return operations withing the same Dialog.

The component exist for a single reason: one side of the transaction could easily start multiple invokes, and the another party could reply out of order.

For example:

App A starts Dialog ID 1 to App B

App A starts Invokes 1 on Dialog ID 1 to App B

App A starts Invokes 2 on Dialog ID 1 to App B

App A starts Invokes 3 on Dialog ID 1 to App B

App B reply Return 1 on Dialog ID 1 to App A

App B reply Return 3 on Dialog ID 1 to App A

App B reply Return 2 on Dialog ID 1 to App A

App B ends Dialog ID 1 with App A

In this example, the 2nd reply is the result from the 3rd invoke.

Type of Number (TON)

These are the possible values for TON field:

TON	VALUE (Binary)
Unknown	00000000
International	00000001
National	00000010
Network Specific	00000011
Subscriber Number	00000100
Alphanumeric	00000101
Abbreviated	00000110

Numbering Plan Indicator (NPI)

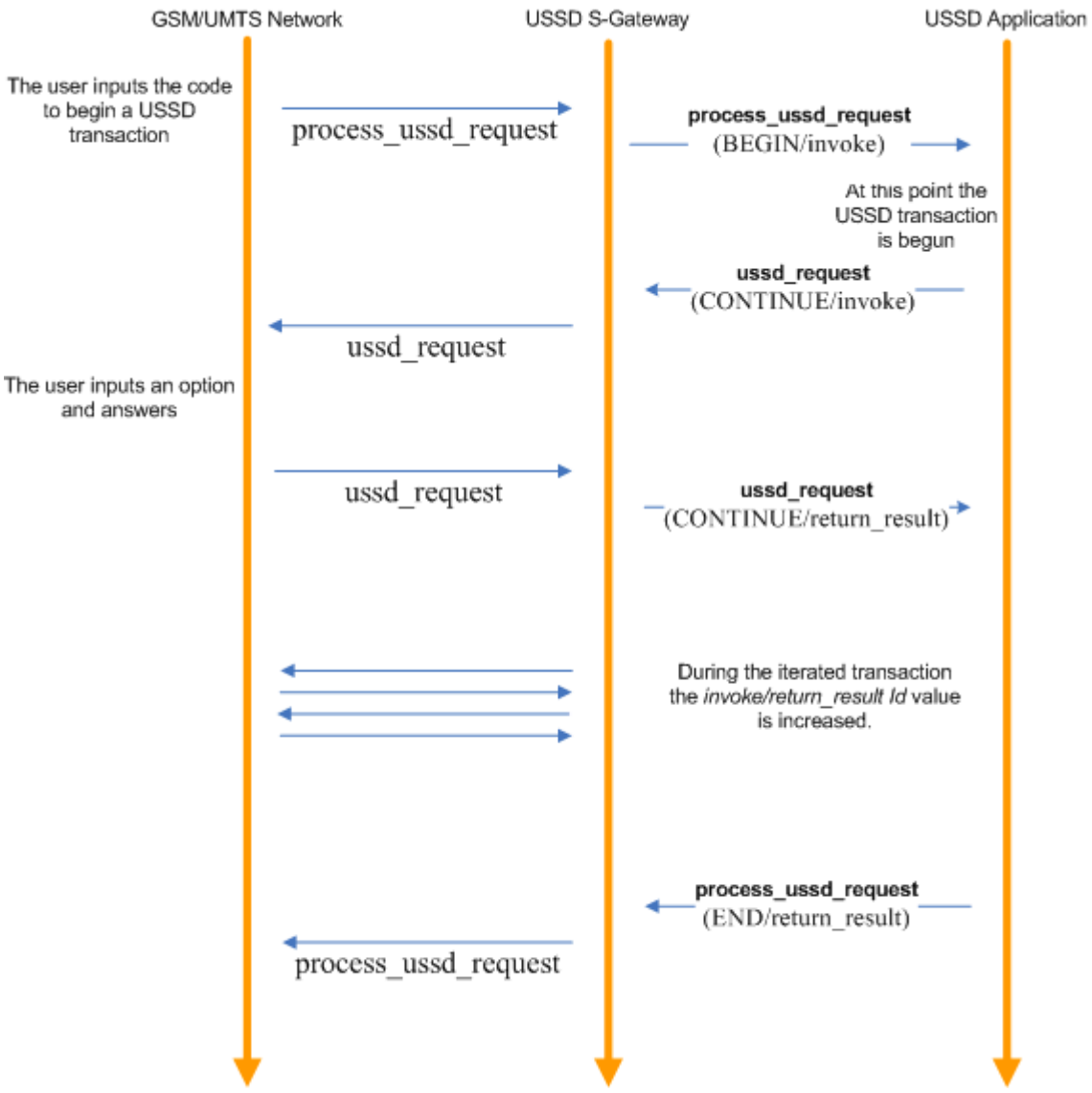
These are the possible values for NPI field:

NPI	VALUE (Binary)
Unknown	00000000
ISDN (E163/E164)	00000001
Data (X.121)	00000011
Telex (F.69)	00000100
Land Mobile (E.212)	00000110
National	00001000
Private	00001001
ERMES	00001010
Internet (IP)	00001110

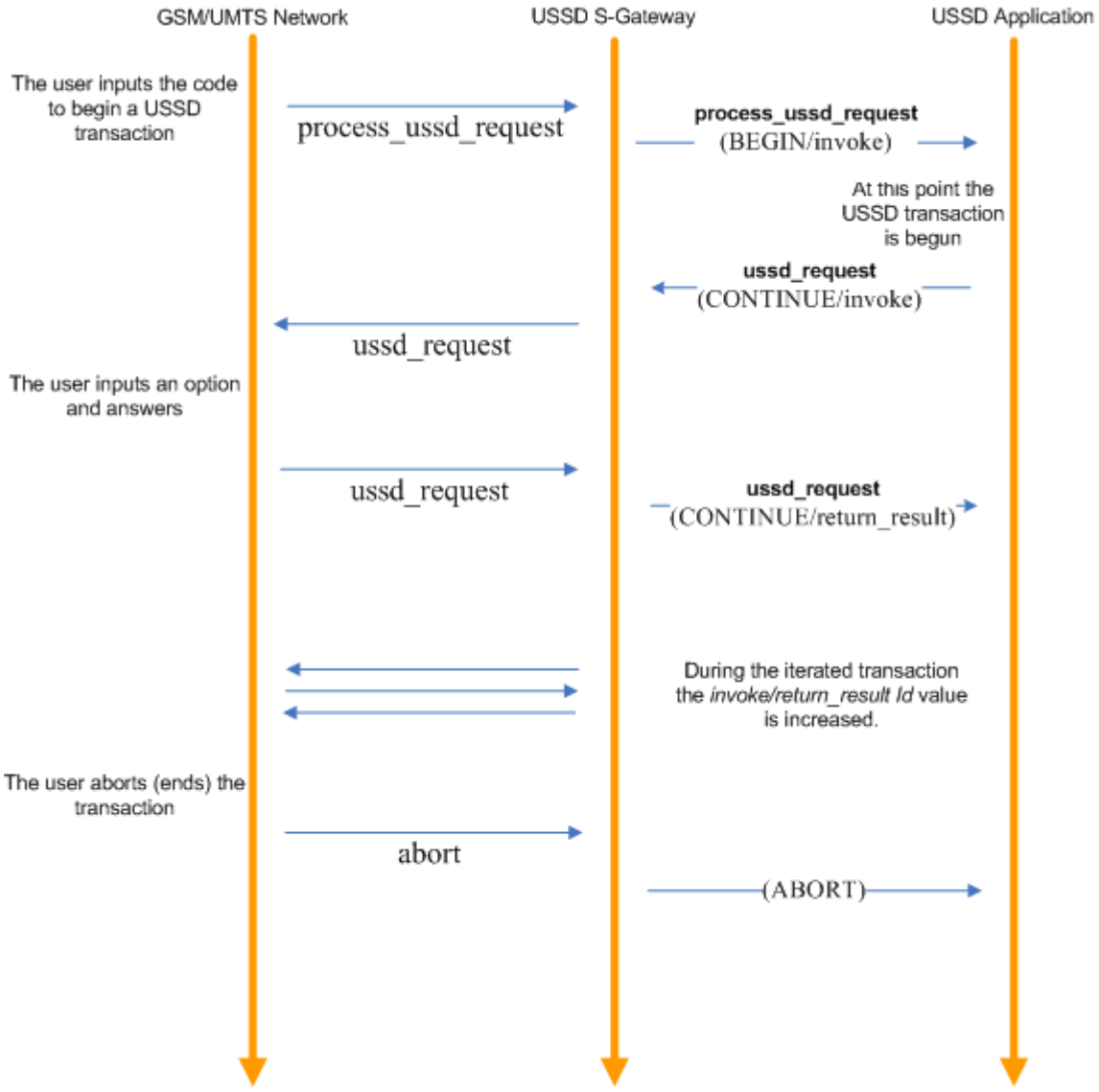
Example Message Sequences

The fallowing chapter gives example message sequences for interacting with USSDJNI Java API.

.1 User Initiated USSD Transaction Ended by Application



.2 User Initiated USSD Transaction Aborted by User



Length of the USSD String

In GSM 0902 160 octets is defined as the maximum length for the USSD string. Due to underlying signalling layers the maximum length of the USSD string depends on the message:

USSD operation	Max length
Begin, Invoke ProcessUSSDRequest	133
End, Result ProcessUSSDRequest	160
First Continue, Invoke USSDRequest in mobile initiated dialogue	154
Begin, Invoke USSDRequest	144
First Continue, Result USSDRequest in network initiated dialogue	154
Other messages	160

UCS2 max string length is limited to 60 to 80 characters depending on the USSD Operation.

UCS2 Support (UTF16)

UCS2 (a previous form of UTF-16 Unicode) is supported using a small diference in the XML:

1) DCS (Data Coding Scheme) must be switched to 17

2) The text must be sent in Hexa format using the ussd_string_ex XML tag.

To send this examle (hello in Arabic) مرحبا

The XML example is:

```
<siggw>
        <msg type="continue">
                <user id="0"/>
                <dialog id="1"/>
                <context v="04000001001302"/>
                <user_info>
                        <map op="accept"/>
                </user_info>
                <component type="invoke" id="2" op="ussd_request">
                        <param name="data_coding_scheme" v="17"/>
                        <param name="ussd_string_ex" v="06450631062D06280627"/>
                </component>
        </msg>
</siggw>
```

"06450631062D06280627" means:

0645 = ARABIC LETTER MEEM
0631 = ARABIC LETTER REH
062D = ARABIC LETTER HAH
0628 = ARABIC LETTER BEH
0627 = ARABIC LETTER ALEF

For full UCS2 character list please request LeibICT - UCS-2 Code Chart.pdf documentation.

Annex A – 3GPP 29.002 Errors

This is a (sorted by error code) list of errors described on the 3GPP 29.002 standard.

1 unknownSubscriber
3 unknownMSC
5 unidentifiedSubscriber
7 unknownEquipment
6 absentSubscriberSM
8 roamingNotAllowed
9 illegalSubscriber
10 bearerServiceNotProvisioned
11 teleserviceNotProvisioned
12 illegalEquipment
13 callBarred
14 forwardingViolation
15 cug-Reject
16 illegalSS-Operation
17 ss-ErrorStatus
18 ss-NotAvailable
19 ss-SubscriptionViolation
20 ss-Incompatibility
21 facilityNotSupported
22 ongoingGroupCall
25 noHandoverNumberAvailable
26 subsequentHandoverFailure
27 absentSubscriber
28 incompatibleTerminal
29 shortTermDenial
30 longTermDenial
31 subscriberBusyForMT-SMS
32 sm-DeliveryFailure
33 messageWaitingListFull
34 systemFailure
35 dataMissing
36 unexpectedDataValue
37 pw-RegistrationFailure
38 negativePW-Check
39 noRoamingNumberAvailable
40 tracingBufferFull
42 targetCellOutsideGroupCallArea
43 numberOfPW-AttemptsViolation
44 numberChanged
45 busySubscriber
46 noSubscriberReply
47 forwardingFailed
48 or-NotAllowed

49 ati-NotAllowed
50 noGroupCallNumberAvailable
51 resourceLimitation
52 unauthorizedRequestingNetwork
53 unauthorizedLCSClient
54 positionMethodFailure
58 unknownOrUnreachableLCSClient
59 mm-EventNotSupported
60 atsi-NotAllowed
61 atm-NotAllowed
62 informationNotAvailable
71 unknownAlphabet
72 ussd-Busy

Annex B - Mobile Terminated Dialog Example

USSD Notify XML Request and Responses:

USSD Notify (1st Invoke begin) (from Application to Gateway):

```
<siggw>
    <msg type="begin">
        <user id="0" />
        <dialog id="2" />
        <destination id="0" />
        <context v="04000001001302" />
        <user_info>
            <object_identifier v="04000001010101" />
            <map op="open">
                <param name="destination_reference" number="598612532166"
nai="1" npi="1" />
                <param name="origination_reference" number="598610000015"
nai="1" npi="1" />
            </map>
        </user_info>
        <component type="invoke" id="0" op="ussd_request">
            <param name="data_coding_scheme" v="15" />
                                                <param    name="ussd_string"
v="576520686176652069 6E74726F647563656420 6E6577206665617475726 5732C20696620
796F752061726520696E746572657374696E6720706C65617365207265706C7920776974682
031" />
            <param name="msisdn" number="252612532166" nai="1" npi="1" />
        </component>
    </msg>
</siggw>
```

ussd_request: 1st continue *GW-to-App (User Input Response)

```
<siggw>
        <msg type="continue">
                <stack id="0"/>
                <user id="0"/>
                <dialog id="2"/>
                <context v="04000001001302"/>
                <component type="return_l" id="0" op="ussd_request">
                        <param name="data_coding_scheme" v="15"/>
                        <param name="ussd_string" v="31"/>
                </component>
        </msg>
</siggw>
```

ussd_request: 2nd continue *App-to-GW (Prompt User Input)

```
<siggw>
    <msg type="continue">
        <user id="0" />
        <dialog id="2" />
        <component type="invoke" id="2" op="ussd_request">
            <param name="data_coding_scheme" v="15" />
                                                    <param    name="ussd_string"
v="53657276696365733A5C6E312E20536572766963652031 5C6E322E205365727669636520
325C6E332E2045786974" />
        </component>
    </msg>
</siggw>
```

ussd_request: 2nd continue *GW-to-App (User Input Response)

```
<siggw>
        <msg type="continue">
                <stack id="0"/>
                <user id="0"/>
                <dialog id="2"/>
                <component type="return_l" id="2" op="ussd_request">
                        <param name="data_coding_scheme" v="15"/>
                        <param name="ussd_string" v="31"/>
                </component>
        </msg>
</siggw>
```

unstructured_ss_notify: last_end *App-to-GW (Notify User)

```
<siggw>
    <msg type="end">
        <user id="0" />
        <dialog id="2" />
        <component type="invoke" id="0" op="unstructured_ss_notify">
            <param name="data_coding_scheme" v="15" />
                                                    <param    name="ussd_string"
v="5468616E6B7320666F72207573696E67205365727669636520312E" />
        </component>
    </msg>
</siggw>
```