USSD S-Gateway USSD C/C++ API

User's Guide v1.1

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

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 USSD C/C++ API supported and managed by the LeibICT's USSD S-Gateway.

The USSD C/C++ API offers a high abstraction layer for C/C++ Applications witch:

1) Controls the TCP/IP connection, including reconnection.
2) Implements the RSA encryption and decryption.
3) Implements the XML creation and parsing.
4) Controls the USSD/MAP protocol details.

Capacity

The current USSD S-Gateway version supports 16384 incoming and outgoing messages by stack, with capability of two stacks running independently, so a total capacity of 32678 incoming and outgoing simultaneous messages

Each USSD transaction is identified by a dialog identifier (field named: *DialogueId*), represented by 16 bits. 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 16383, where as when it receives a message, it must answer using the same *Dialogue Id* of the incoming message, which will be a digit between 16384 and 32768.

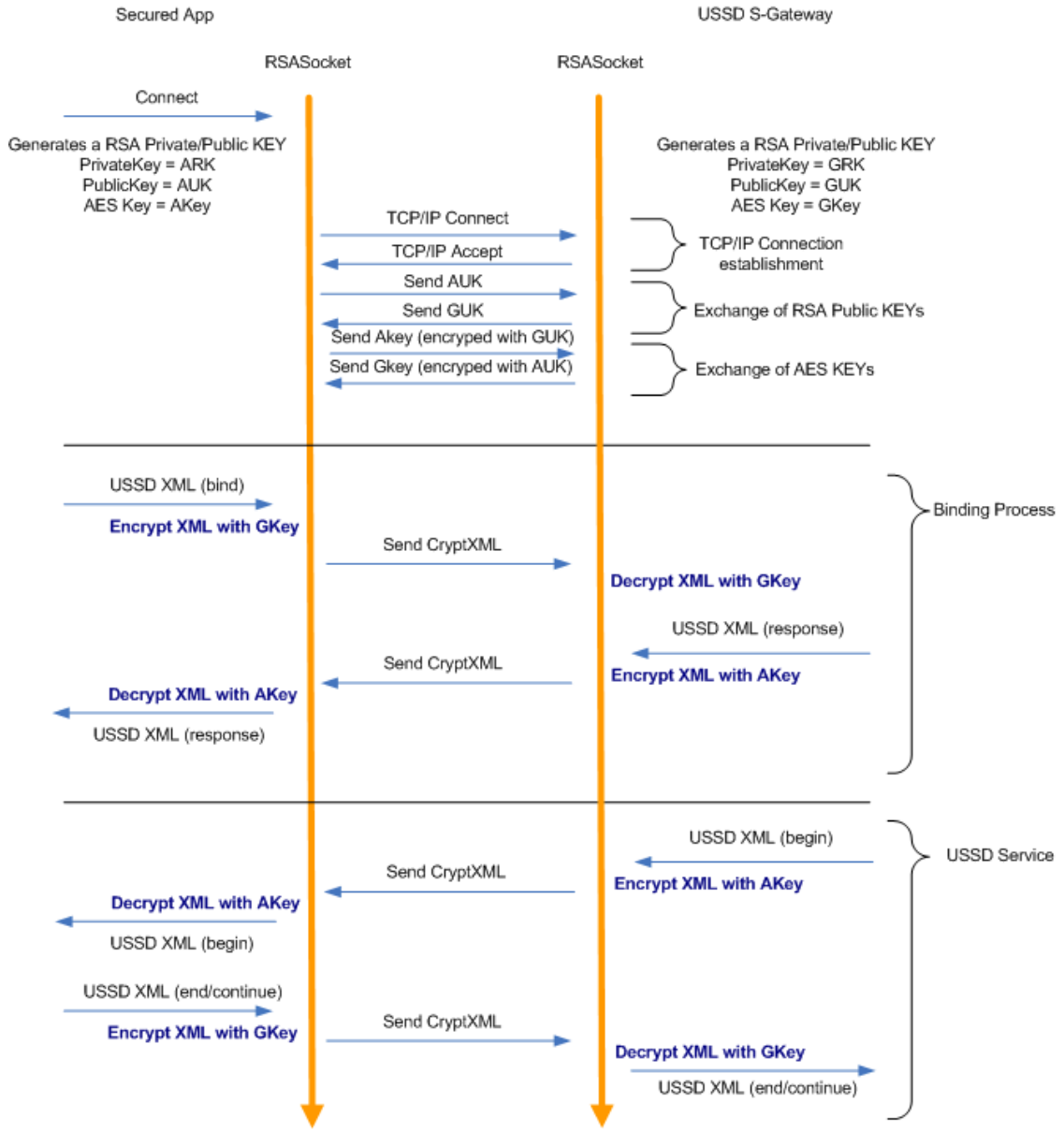
The RSA encryption and decryption process is very high cpu consuming, setting up the right key length is needed to depending of the traffic estimated.

RSA/AES Secured mode

The USSD S-Gateway may be configured to use secure AES and RSA PKI Encryption. This mode assures a totally secure communication, valid for banking applications over non-secured lans.

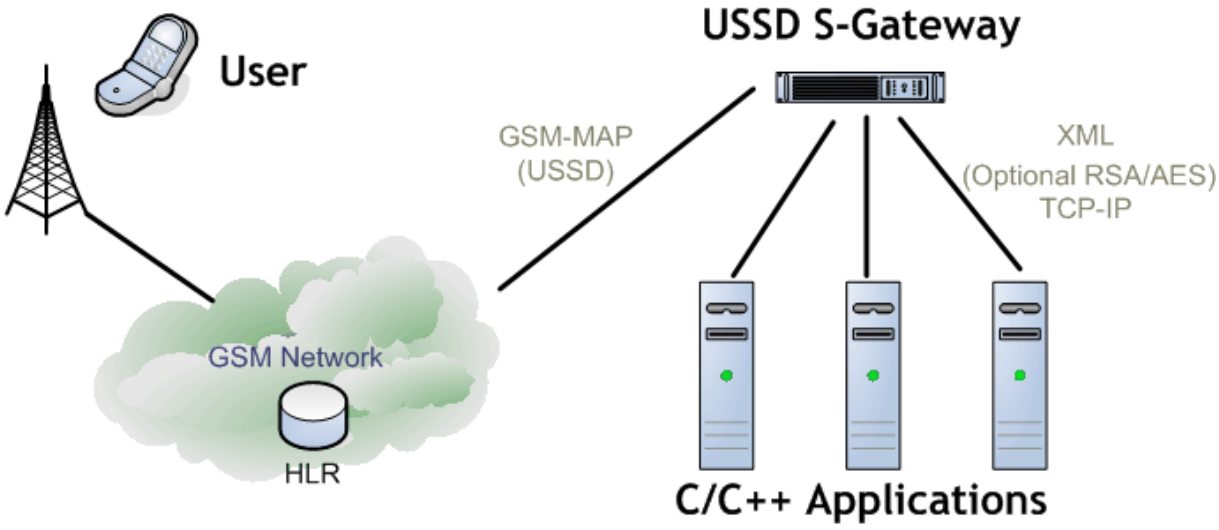
Due to the asymmetric load of the operations of encryption and decryption, the connection transmission and reception can be secured using different key length, or even without it.

For example, if your application vulnerable point is information presentation, you can encrypt the reception with 4096 RSA and 256 AES bits keys, and maintain the transmission unsecured. Otherwise if your application critical information is user input, you can encrypt the transmission.



Typical system structure

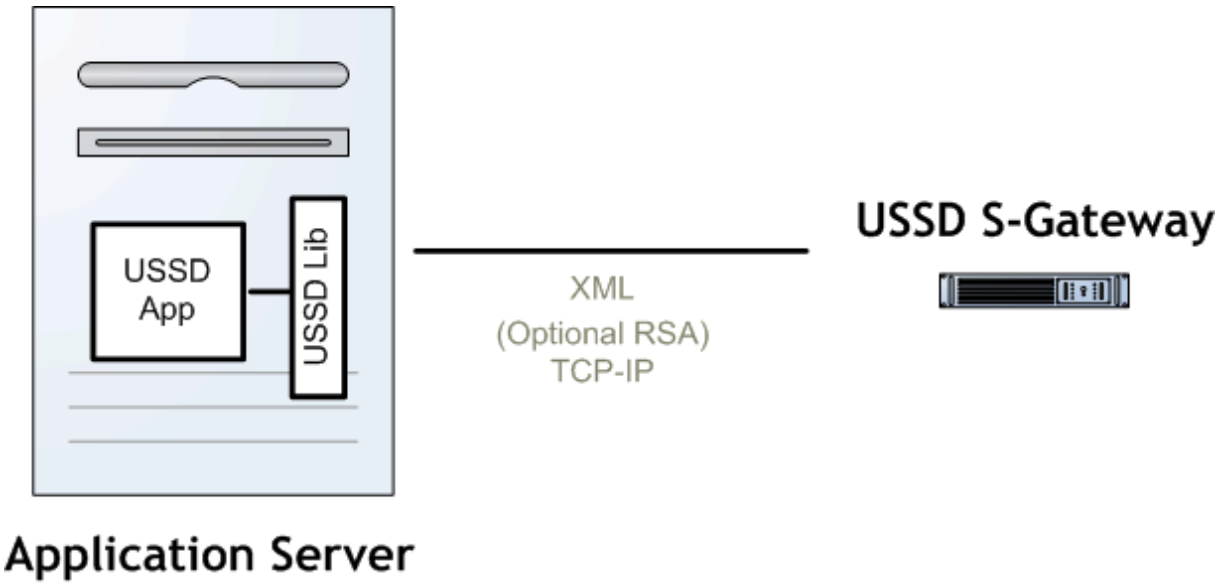
The LeibICT USSD S-Gateway provides Load Balancing and Fault-Tolerance of applications:



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

Software Framework

The USSD C/C++ API Library consists in a group of dynamically linked libraries developed in C++ that are loaded into the Application at run time.



API Architecture

The USSD C/C++ API Library enables programmers to access the services provided by USSD S-Gateway and protocol using a C/C++ function and callbacks based interface.

The USSD C/C++ API – USSD S-Gateway communication mechanism used is transparent to the programmer, as all the socket management is performed by the USSD Library.

When the USSD Library is initialized by calling **initialize** and **connect**, a worker thread is started which is responsible for socket management.

Events received from USSD S-Gateway are reported to the user application from this thread using a callback mechanism.

The USSD Library is thread-safe from the user's point of view. However, the user is responsible for protecting any of his own data structures accessible from multiple threads.

The USSD Library worker thread only delivers one event at a time. That is, it does not notify the user application of a new event until the user returns control from the current event handler. However this means that the user needs to avoid using blocking or lengthy operations within callback functions.

API Functions

The API is centralized in a class: CUSSD, these are the API functions :

1. initialize

int CUSSD::initialize(USSDCallbacks));*

Description

The initialize function initializes the USSD Library and declares the call-backs locations.

Output

Return values:

0 – An error occurred during initialization

1 – Initialization successfully

2. ussd_requestShortCode

int CUSSD::ussd_requestShortCodet(String shortCode);

Input

Parameter	Description	Example
shortCode	Interest USSD ShortCode	*123#

Description

The ussd_ requestShortCode should be called before the connect function to indicate the USSD S-Gateway of a traffic filter.

Output

Return values:

0 – An error occurred

1 – The command was successfully received by the USSD S-Gateway

3. connect

int CUSSD::connect(String addr,int port,int rsaRcv, int rsaSnd);

Input

Parameter	Description	Example
addr	IP or Domain Name of the USSD S-Gateway	10.0.0.1
por	TCP port of the USSD S-Getway	5454
rsaRcv	RSA Security used for reception	0- Unsecured 1- RSA 512bit 2- RSA 1024bit 3- RSA 2048bit 4- RSA 4096bit
rsaSnd	RSA Security used for transmission	Same as rsaRcv

Description

The connect function tries to connect the USSD S-Gateway. If it success and there is a network failure, there is no need to call this function again, the USSD Library will try to reconnect automatically.

Output

Return values:

0 – An error occurred during socket connection establishment or while securing it.

1 – Connection and security established successfully

4. ussd_request

int CUSSD::ussd_request(int dialogId,String prompt);

Input

Parameter	Description	Example
dialogId	Dialog of the current USSD Service	1
prompt	String containing a prompo for the user	"Enter the PIN"

Description

The ussd_request function sends a message to the user trough the USSD S-Gateway requesting some input.

Output

Return values:

0 – An error occurred

1 – The command was successfully received by the USSD S-Gateway

5. ussd_end

int CUSSD::ussd_end(int dialogId,String info);

Input

Parameter	Description	Example
dialogId	Dialog of the current USSD Service	1
info	String containing the last message to the user	"Have a nice day"

Description

The ussd_end function sends a message to the user trough the USSD S-Gateway finalizing the dialog.

Output

Return values:

0 – An error occurred

1 – The command was successfully received by the USSD S-Gateway

6. pong

int CUSSD::pong();

Description

The pong function sends a pong to the USSD S-Gateway. The pong function must be the response to the ping call-back. If the application does not respond the ping with a pong, the USSD S-Gateway may disconnect the application.

The intervals between pings are defined in the USSD S-Gateways and cannot be determined in the Application side.

Output

Return values:

0 – An error occurred

1 – The command was successfully received by the USSD S-Gateway

API Callbacks

These are the API Callbacks:

1. ussd_service

void ussd_service(int dialogId,String shortCode,String phoneNumber) {}

Output

Parameter	Description	Example
dialogId	Dialog of the current USSD Service	1
shortCode	String containing the shortcode entered by the user	"*123#"
phoneNumber	String containing the phone number of the user	"888-123-123"

Description

The *ussd_service* call-back is called when a new ussd service is started by the User.

2. ussd_response

void ussd_response(int dialogId,String str) {}

Output

Parameter	Description	Example
dialogId	Dialog of the current USSD Service	1
str	String containing the user response of a request.	"1"

Description

The *ussd_response* call-back is called when a user responses a request.

3. ussd_end

void ussd_end(int dialogId) {}

Output

Parameter	Description	Example
dialogId	Dialog of the current USSD Service	1

Description

The *ussd_end* call-back is called when the user, the GSM network or the USSD S-Gateway ends the dialog.

4. disconnected

void disconnected() {}

Description

The *disconnected* call-back is called when USSD Library gets disconnected from the USSD S-Gateway.

5. connected

void connected() {}

Description

The *connected* call-back is called when USSD Library gets connected again to the USSD S-Gateway.

6. ping

void ping() {}

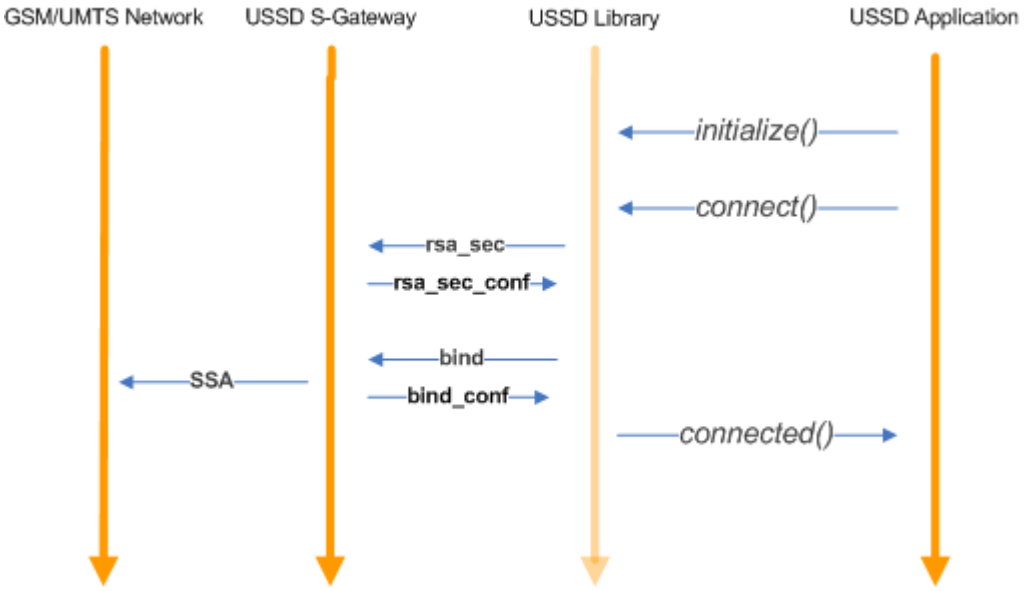
Description

The *ping* call-back is called when USSD S-Gateway sends a ping to the Application.

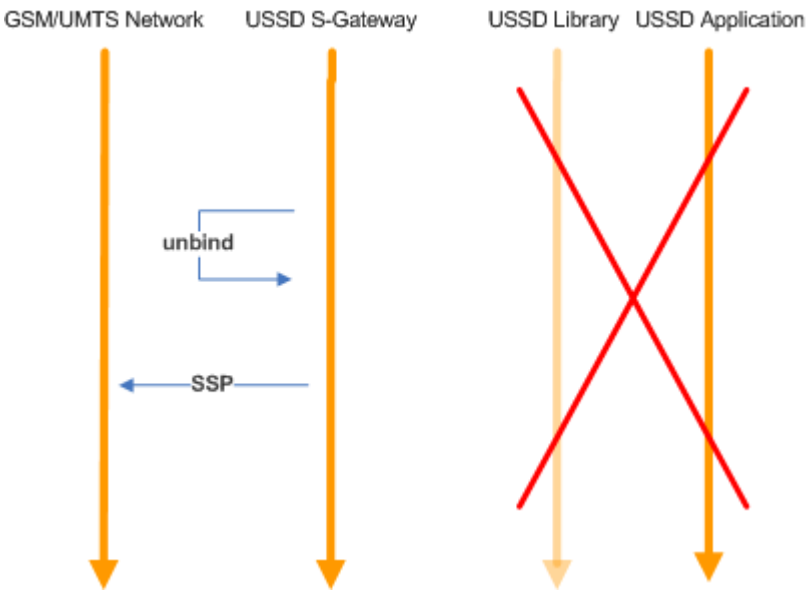
Example Message Sequences

The fallowing chapter gives example message sequences for interacting with USSD C/C++ API.

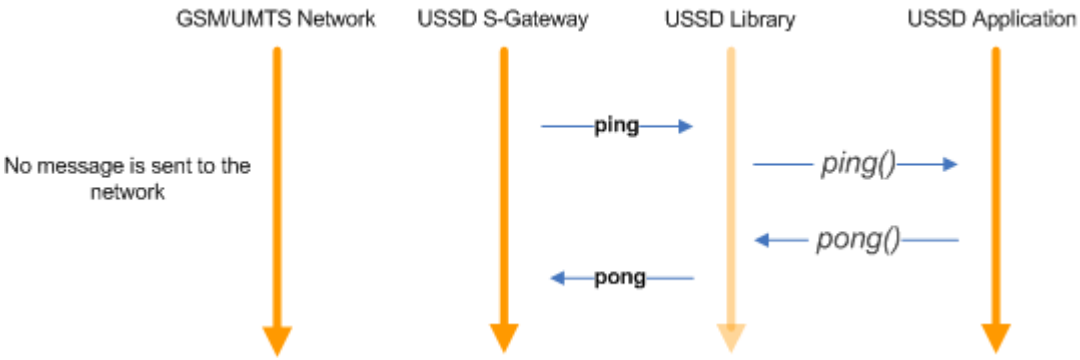
1. Succesfull Application Connection



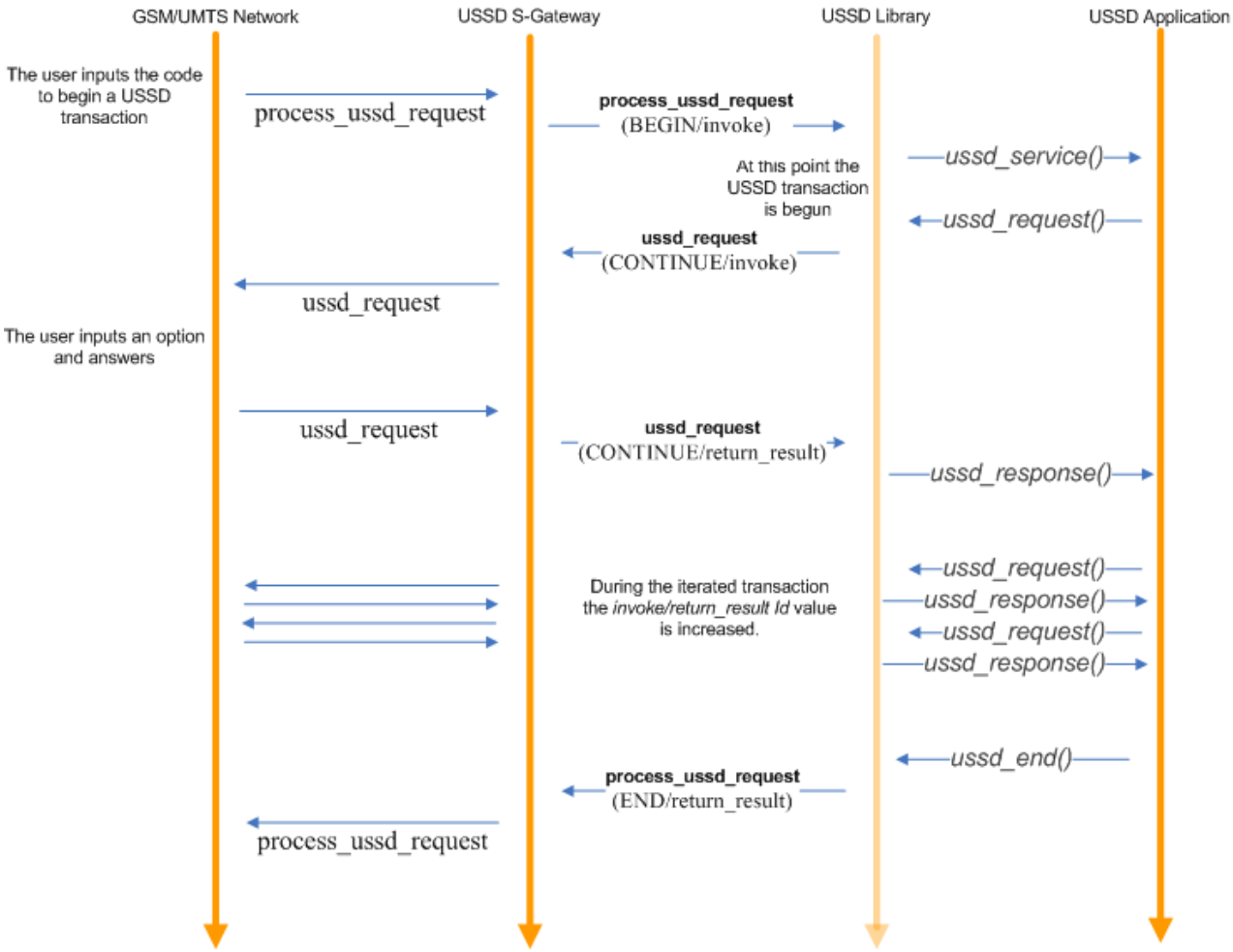
2. Application Disconnection



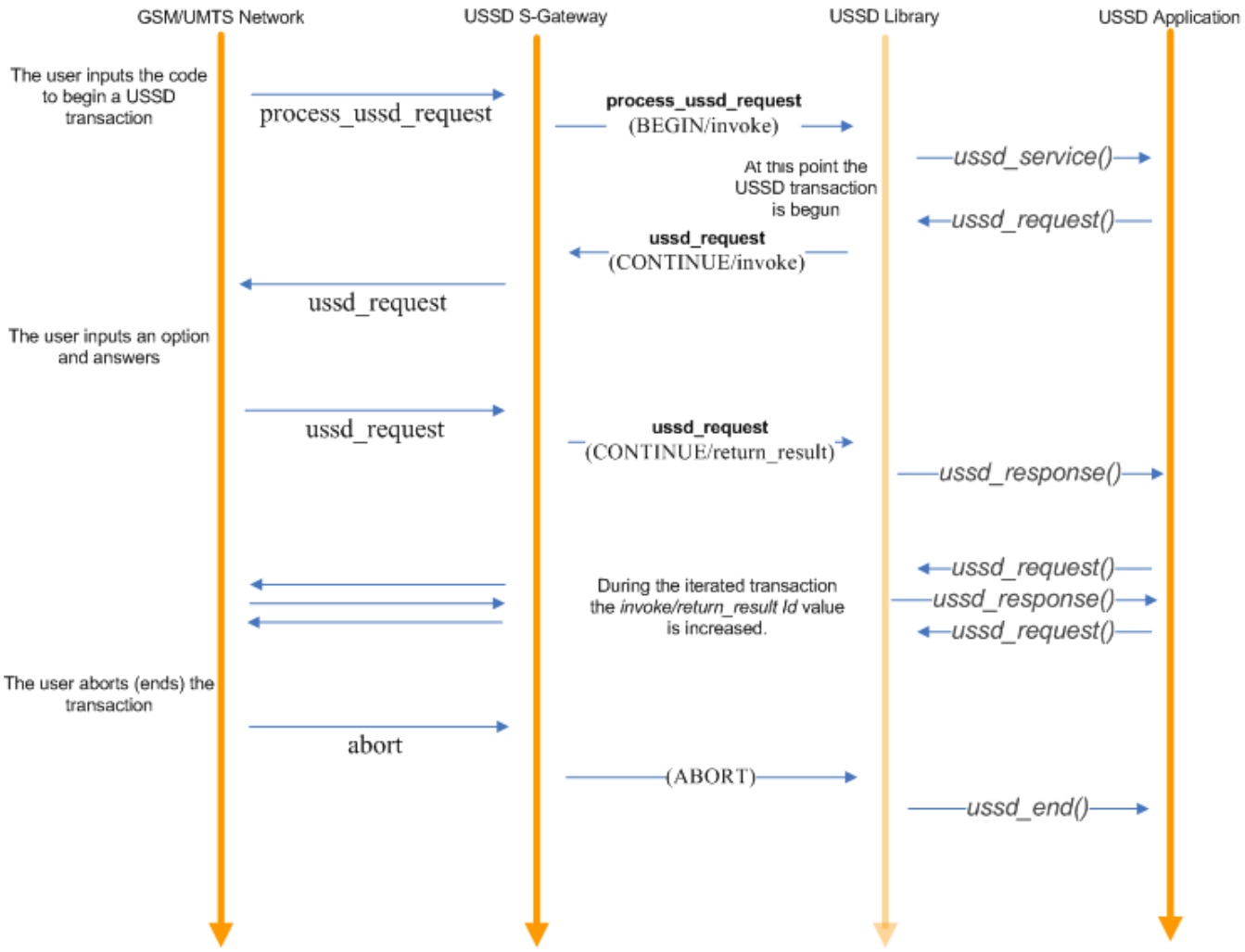
3. Ping-Pong


GSM/UMTS Network
USSD S-Gateway
USSD Library
USSD Application
No message is sent to the network
ping
ping()
pong()
pong

4. User Initiated USSD Transaction Ended by Application



5. User Initiated USSD Transaction Aborted by User



Length of the USSD String

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

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

USSD Configuration

The USSD C/C++I Library has on Windows a registry based configuration and on Linux a file based configuration.

Windows USSD:

HKEY_LOCAL_MACHINE\SOFTWARE\LeibICT\USSD

Linux USSD:

./USSD.conf

Field	Description	Default Value
Log	Enable or Disable standard logs	1
logDebug	Enable or Disable debug logs	0
logDir	Directory where the logs are placed	c:\logs\
logErrorsFileName	Name of the file for errors logs	USSDErrors.log
logFileName	Name of the file for standard and debug logs	USSD.log
logTraffic	Enable or Diable traffic logs	0
logTrafficFileName	Name of the file for traffic logs	USSDTraffic.log