MINIAPPS

Development Guide

API

How to create USSD service

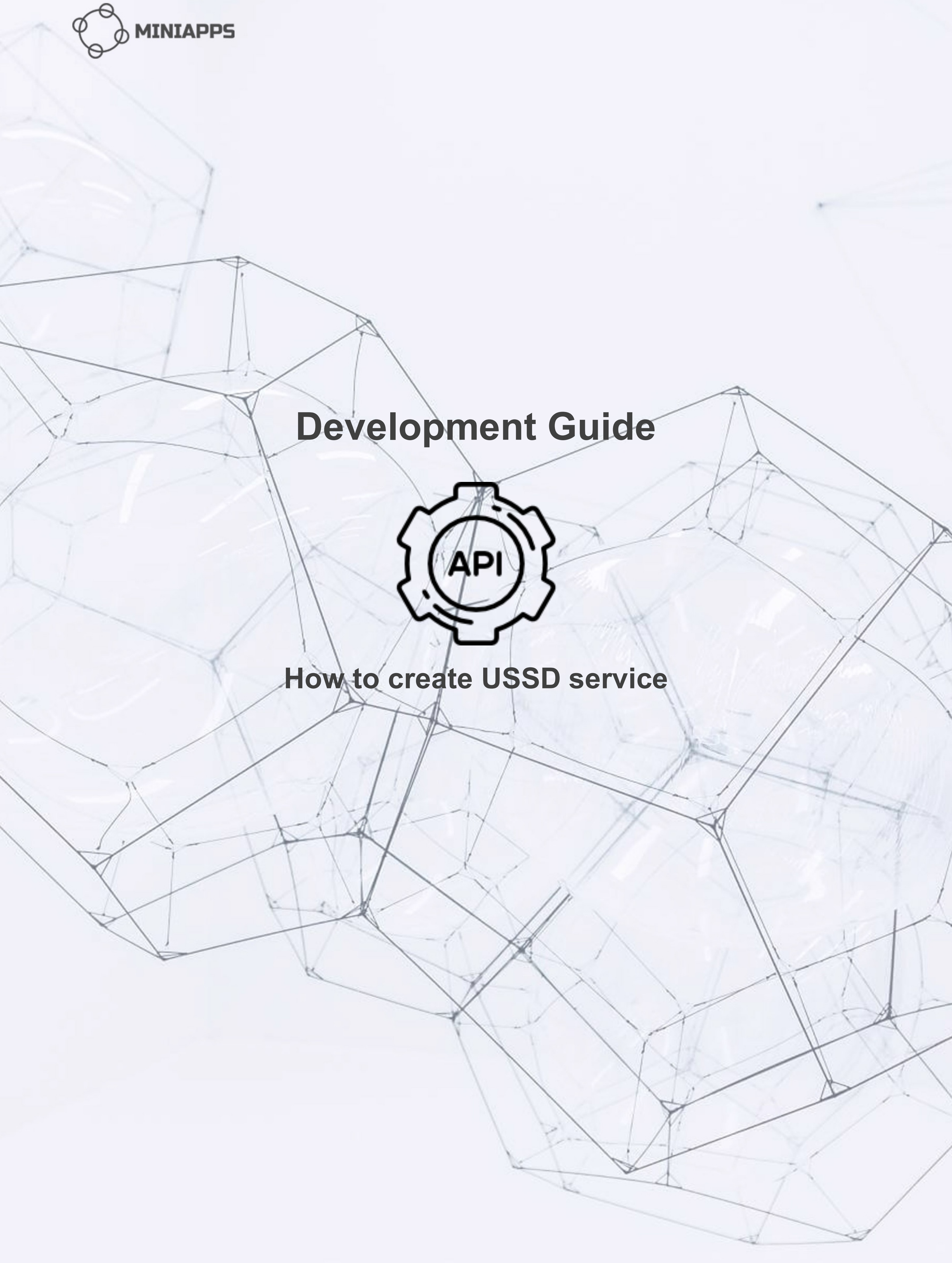


Table of Contents

1 Introduction

Creating mobile services is a complex process involving the following issues:

- variety of mobile devices;
- different screen sizes;
- lack of uniform standards and, consequently, very different interfaces and ways of processing and displaying information on mobile devices.

MiniApps platform is a solution that helps to save time and money needed for mobile service creation. There is no need for implementing low-level functionality to communicate with the network infrastructure (SMS centers, USSD centers, etc.) as well as mobile operators.Mobile service using MiniApps can be created in two different ways. The original service structure and its information content can be easy created via MiniApps platform constructor which allows creating in a graphical mode.The other way is to use a site of XML pages with additional functionality programmed in any convenient server side programming language. This option is used in the cases when the service needs to access databases, perform calculations, make choices depending on user data, etc.*This document describes the process of developing **USSD** services with MiniApps platform.*

2 What Is Mobile Service?

Mobile service is a set of static or dynamic information displayed on the screen of a mobile device in a specific order in accordance with user actions. It may contain textual and/or graphical information, it may request the subscriber’s data and use it in the system.

The service can be delivered to the mobile device using the following technologies:

1. ***USSD***. In this case, the subscriber communicates with the service in text only mode, with a font of the same color and shape. A strictly limited number of characters can be displayed on the screen of the mobile device at one time.
2. ***SMS***. SMS can be used as a substitute for USSD, in the case when information needs to be stored on the phone (links, phone numbers, etc.).

Both access modes or combination of them can be used for the service.

3 Creating a Mobile Service

Creating a mobile service is divided into two main steps: implementing initial service structure (described in this chapter), and connecting service sources to MiniApps (Connecting Service).

3.1 Description of Service Structure and Logic

The first step in creating a service is to define the functionality and the content of the service, to prepare the documents describing the behavior of the service, menu pages and their sequence, and to choose the access points to the service (USSD, SMS).

It is important to determine whether the service is static or dynamic.

- ***Static service*** is displayed in the same way to all users, regardless of the time of day, subscriber data or other factors. For example, information about cities in Russia is of static nature: time zones, population, or size do not change.
- ***Dynamic service*** is displayed differently depending on various conditions. Information or the menu structure which the service displays may depend on the date, subscriber data, choices previously made, etc. For example, in the case of account balance, service response depends on the subscriber’s number which is different for everyone.

Depending on the service, whether it is static or dynamic, implementation can go in different directions.

3.1.1 Example of Financial Service

Let us see the process of service development using the example of a banking service. Imagine that a bank offers its customers a service allowing users to manage their money via mobile phone. The customer may open a “mobile” bank account which is associated with his telephone number, transfer a certain amount of money from his main bank account to this account, and make money transfers using only the mobile device. For example, he can pay for merchandise in stores, transferring the money from his mobile account to the account of the store.

Let us describe the service functionality. The client can:

- check the current balance of his mobile account;
- transfer the money to another account;
- find out what the current bank offers are;
- find out contact details.

Connection to the service will be possible through ***USSD***.

Diagram below describes the service menu, the way it looks when the service is entered via USSD, and the logic of switches between pages. Each block of the diagram is a separate page of the service.

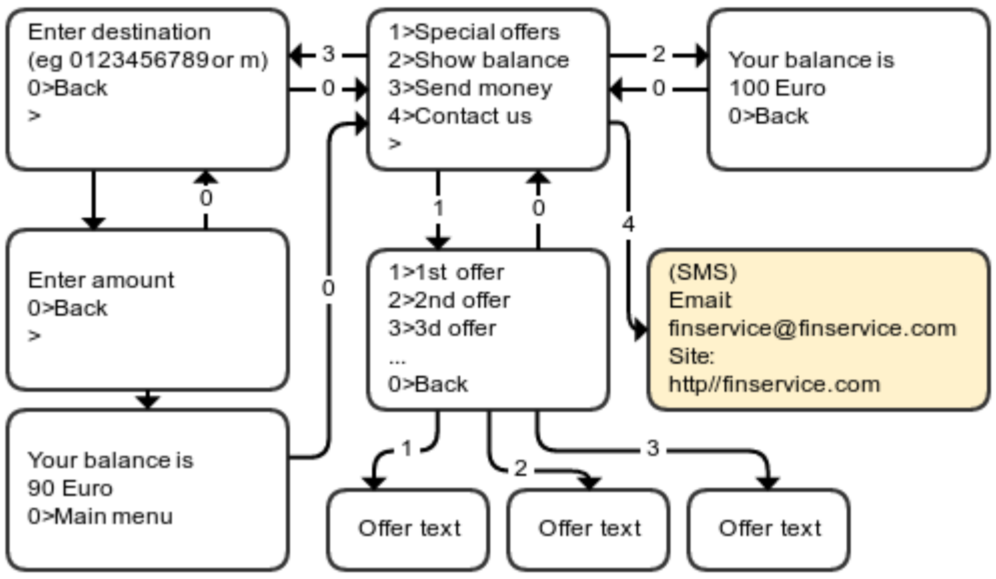


Diagram 1. *Financial Service*

The sections ***Show Balance*** and ***Send Money*** are ***dynamic*** and will vary from client to client. The sections ***Special Offers*** and ***Contact us*** are ***static*** and will be the same for everyone.

Using this example, we should consider all possible options for service implementation:

1. The static part in XML, Section Static Service in XML
2. The dynamic part in XML + programming language (PHP was used), Dynamic service
3. The static part in XML in conjunction with the dynamic part in the XML + PHP, Combining the Static and Dynamic Parts in XML

3.2 Static Service in XML

The service itself and the logic of its work are realized in the form of a website. Each page of the website is one page of the service on the mobile device. Website pages are written in a specific XML format. A complete description of the format can be found at: MiniApps XML format specification
Let us see the static part of the financial service (highlighted in the Diagram 2).

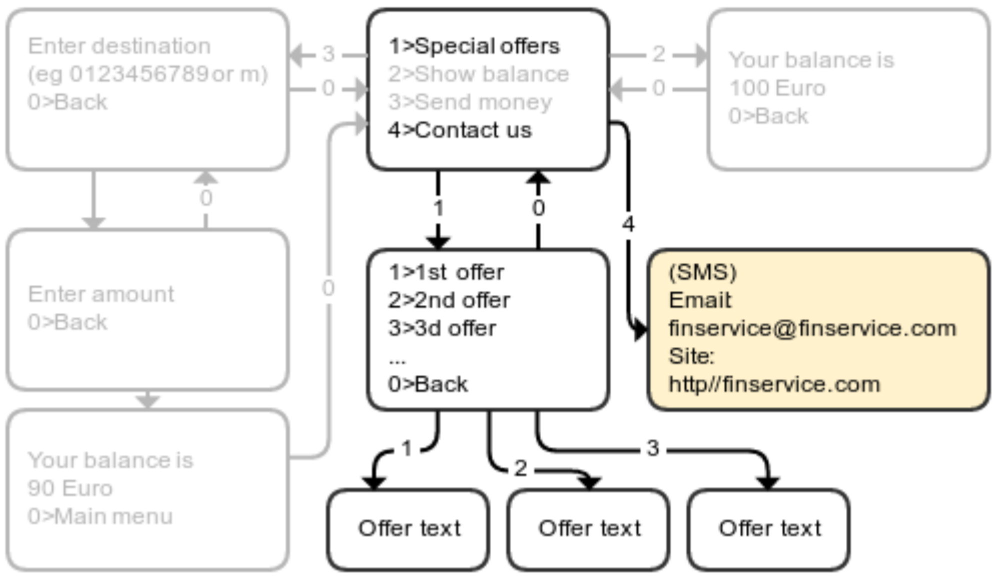


Diagram 2. *Static Part of the* ***Financial Service***.

Since the service is created as a website, we describe the start page of the service in the file named index.xml. Thus, it will be automatically defined as the first page of the website.

Code Block 1 index.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<page version="2.0">
    <navigation>
        <link accesskey="1" pageId="b01.xml">Special Offers</link>
<!—the items "2. Show Balance" and "3. Send Money" are still missing -->
        <link accesskey="4" pageId="b04.xml">Contact Us</link>
    </navigation>
</page>
```

<?xml version="1.0" encoding="UTF-8"?> is the announcement of XML, an optional line that specifies the version of the XML standard and the charset.

<page version="2.0"></page> defines the beginning and the end of the service page; the parameter version specifies the version of the XML format of the page.The block **<navigation></navigation>** contains menu items. Each menu item is located in the block **<link></link>**.The parameters of the tag **<link>** are:

- ***accesskey***; it can assume numeric values. It specifies the code to be entered on a mobile device for navigating this menu item. This parameter is mandatory for compiling USSD commands.

- ***pageId***; this is a file describing the page which will be reached through this menu item.

- ***protocol***; it defines a list of access points in which this menu item will be visible. If this parameter is not specified, the item will be visible in all access points.

In the phone this page will look as follows:

```
1>Special Offers
4>Contact Us
```

Special offers page:

Code Block 2 b01.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<page version="2.0">
    <navigation>
        <link accesskey="1" pageId="c01.xml">Campaign for credit card
holders</link>
     <link accesskey="2" pageId="c02.xml">Bring a friend - get $100
coupon</link>
        <link accesskey="0" pageId="b01.xml" type="back">Back</link>
</navigation>
</page>
```

Parameter ***type="back"*** in the tag **<link>** sets type ***back*** for the menu item "Back." This parameter is optional; it is used to determine the instruction for return, which will be executed when the user clicks the button of return on the telephone.

```
1>Campaign for credit card holders
2>Bring a friend - get $100 coupon
0>Back
```

The Contact Us page:

Code Block 3 b04.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<page version="2.0">
    <div>
         Information sent to you by SMS
    </div>
    <div type="sms">
        Email: finservice@finservice.com, Site: http://finservice.com
    </div>
</page>
```

The text from the first block **<div></div>** will be displayed on the mobile device as plain text.The text from the second block **<div></div>** will be delivered in the form of a SMS simultaneously upon entering this page; the parameter ***type="sms"*** is specified for this.It should be noted that the link in the text for ***SMS*** is not enclosed in the tag **<a></a>**, that is, it will be displayed as plain text. However, many phone models support recognition of links in SMS, thus it can possibly be displayed not as plain text, but as a link.

USSD	Information sent to you by SMS
SMS	Email: finservice@finservice.com, Site: http://finservice.com

The page Offer 1:

Code Block 4 c01.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<page version="2.0">
 <title>Campaign for credit card holders</title>
    <div>Credit card holders may raise credit limit up to 30%</div>
    <navigation>
        <link accesskey="0" pageId="b01.xml" type="back">Back</link>
    </navigation>
</page>
```

```
Credit card holders may raise credit limit up to 30%
0>Back
```

The static part of the service is defined. The entire set of created files is uploaded on the web server, and we can start use the service.The appearance of the uploaded files:

Code Block 5 File system

```
../service_site_directory/
b01.xml
b04.xml
c01.xml
index.xml
```

3.3 Dynamic service

Dynamic functionality is created using a programming language best suited for the task. The example of “Financial Service” is implemented using the language PHP.

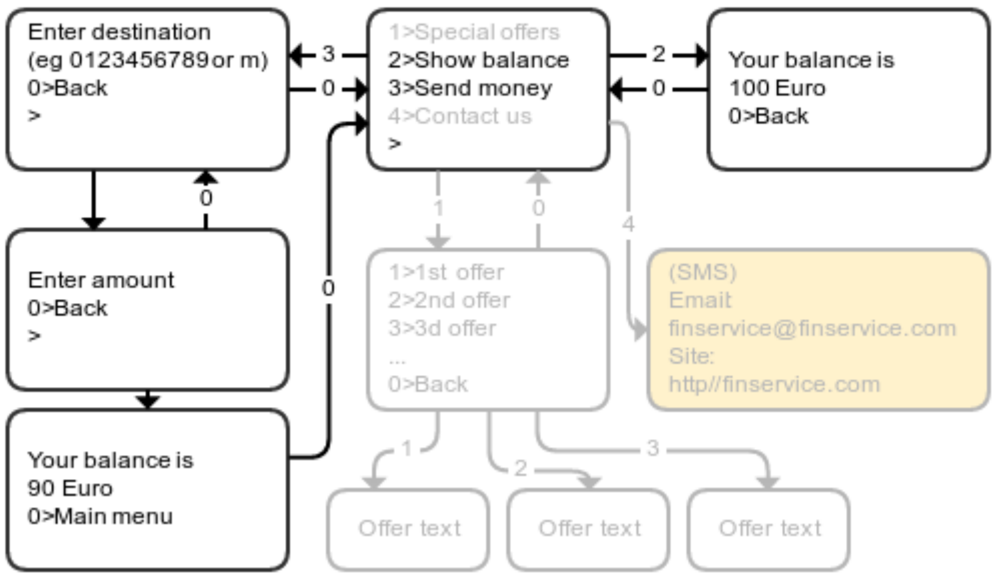


Diagram 4**. Financial Service: **Dynamic Part

For implementing dynamic functionality, a small database was created that will store the records of this type [phone number; account balance]. When the subscriber makes an inquiry about his balance or transfers money, the service will access this database. Config.php defines the database settings and the values of variables.

Code Block 6 Config.php

```
<?php
// ** Variables definition ** //
define('DB_NAME', 'wp_fs');     // The name of the database
define('DB_USER', 'root');      // Your MySQL username
define('DB_PASSWORD', 'root'); // ...and password
define('DB_HOST', 'localhost');    // 99% chance you won't need to change
this value
define('DB_CHARSET', 'utf8');
define('DB_COLLATE', '');
define('DEFAULT_BALANCE', 100);
?>
```

Common.php describes the functions required for working with the balance.

Code Block 7 Common.php

```
<?php
require_once 'config.php';
function getConnection()
{
    $con = mysql_connect(DB_HOST, DB_USER, DB_PASSWORD)
        or die('Could not connect: ' . mysql_error());
    if(!mysql_select_db(DB_NAME))
        die('Could not select database');
    return $con;
}
function createAccount($abonent)
{
    $con = getConnection();
           $query   =   "INSERT   INTO   `accounts`(`abonent`,
`balance`)   values($abonent,
".DEFAULT_BALANCE.")";
    if(!mysql_query($query, $con))
        die('createAccount: Query failed: ' . mysql_error());
}
function getBallance($abonent)
{
    $con = getConnection();
    $query = "SELECT `balance` FROM `accounts` WHERE `abonent` =
$abonent";
    $result = mysql_query($query, $con)
        or die('getBallance: Query failed: ' . mysql_error());
    if(mysql_num_rows($result) == 0)
    {
        return -1;
    }
    $data = mysql_fetch_array($result);
    return $data['balance'];
}
function setBalance($abonent, $balance)
{
    $con = getConnection();
    $query = "UPDATE `accounts` SET `balance`= $balance WHERE `abonent` =
$abonent";
    if(!mysql_query($query, $con))
        die('setBalance: Query failed: ' . mysql_error());
}
function sendMoney($abonent, $amount, $dest)
{
    $con = getConnection();
    $curBalance = getBallance($abonent);
    if($dest === $abonent)
    {
        return "You can't send money to yourself";
    }
    if($curBalance < $amount)
    {
        return "Not enough money";
    }
    $newBalance = $curBalance - $amount;
    if($dest !== "m" && $dest !== "M")
    {
        $destBalance = getBallance($dest);
        if($destBalance == -1)
        {
            return "Account $dest not found";
        }
        setBalance($dest, $destBalance + $amount);
    }
    $message = "Operation successful.";
    if($newBalance == 0)
    {
        $newBalance = DEFAULT_BALANCE;
        $message .= " Your account has been recharged.";
    }
    setBalance($abonent, $newBalance);
    $message .= " Your balance is $newBalance";
    return $message;
}
?>
```

Now it is necessary to describe service pages for showing balance and transferring money.

Code Block 8 show-balance.php

```
<?php
require_once 'common.php';
if(!isset($_POST['abonent']))
    die("Not enough parameters");
$abonent = $_POST['abonent'];
$balance = getBallance($abonent);
if($balance == -1)
{
    createAccount($abonent);
    $balance = DEFAULT_BALANCE;
}
header('Content-Type: text/xml');
print '<?xml version="1.0" encoding="UTF-8"?>';
?>
<page version="2.0">
    <title>Balance</title>
    <div>
        Your balance is <?=$balance?>
    </div>
    <navigation>
     <link accesskey="0" pageId="a01.xml" type="back">Back</link>
    </navigation>
</page>
```

MiniApps sends the subscriber number and the protocol (USSD) used by the subscriber for access, on the part of the service.The balance is requested from the database using the subscriber number. If the record needed is not there yet, a record is created [subscriber number; default balance]. On the service page we draw the value of balance.The link ***Back*** leads to the file ***a01.xml***, described in Static Service in XML of this document. This file specifies the main service menu.On a mobile device, this page will appear as follows:

```
BalanceYour balance is 1000>Back
```

For transferring money, the subscriber should enter the number of another subscriber who has the entry in the database, or the letter “***m***” for a predefined transfer, as well as the transfer amount.Since USSD supports the input of only one answer on each page, for entering the values ***Destination*** and ***Amount***, we will create two different pages, consecutive to each other, using the protocol USSD.

Code Block 9 send-money.php

```
<?php
header('Content-Type: text/xml');
print '<?xml version="1.0" encoding="UTF-8"?>';
?>
<page version="2.0">
    <div protocol="ussd">
    <input navigationId="form" title="Enter Destination" name="dest"/>
    </div>
    <navigation id="form" protocol="ussd">
        <link pageId="amount.php"></link>
    </navigation>
    <navigation>
        <link accesskey="0" pageId="a01.xml" type="back">Back</link>
</navigation>
</page>
```

<input> specifies input fields for the values;
the parameter ***navigationId*** specifies the identifier of the navigation block for the given field or for the form of several fields;
parameter ***title*** is a caption next to the field;
parameter ***name*** is the name of the variable that conveys the value.After entering the value ***Destination*** using ***USSD*** protocol, you are transferred to the page ***amount.php***.

```
Enter Destination:0>Back
```

Code Block 10 amount.php

```
<?php
$dest = $_REQUEST['dest'];
header('Content-Type: text/xml');
print '<?xml version="1.0" encoding="UTF-8"?>';
?>
<page version="2.0">
    <div protocol="ussd">
        <input navigationId="form" title="Enter Destination" name="dest"
value="<?php echo
$dest?>" type="hidden"/><br/>
        <input navigationId="form" title="Enter Amount" name="amount"
type="Number"/>
    </div>
    <navigation id="form" protocol="ussd">
        <link pageId="transaction.php" ></link>
    </navigation>
    <navigation>
        <link accesskey="0" pageId="send-money.php"
type="back">Back</link>
        <link accesskey="1" pageId="a01.xml">Main menu</link>
    </navigation>
</page>
```

To avoid losing the value ***Destination***, a hidden field ***type=”hidden”*** is used; you can use a session mechanism or both. After entering the value “Amount” you are transferred to the page transaction.php.

```
Enter Amount:0>Back
```

Processing the input data, transferring the money:

Code Block 11 transaction.php

```
<?php
require_once 'common.php';
$abonent = $_REQUEST['abonent'];
$amount = $_REQUEST['amount'];
$dest = $_REQUEST['dest'];
$message = sendMoney($abonent, $amount, $dest);
header('Content-Type: text/xml');
print '<?xml version="1.0" encoding="UTF-8"?>';
?>
<page version="2.0">
    <title protocol="wap java">Send money</title>
    <div><?php echo $message?></div>
    <navigation>
        <link accesskey="0" pageId="send-money.php"
type="back">Back</link>
        <link accesskey="1" pageId="a01.xml">Main menu</link>
    </navigation>
</page>
```

The appearance of the mobile device (example):

Operation successful. Your balance is 90.0>Back1>Main menu

Code Block 12 File system

```
../service_site_directory/
    amount.php
    common.php
    config.php
    send-money.php
    show-balance.php
    transaction.php
    other files
...
```

3.4 Combining the Static and Dynamic Parts in XML

It is possible to change the main service menu, created in XML in Static Service in XML by adding transitions to dynamic service items – ***Send money*** and ***Show balance***.

Code Block 13 a01.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<page version="2.0">
    <title protocol="wap java">Financial Service</title>
 <navigation>
        <link accesskey="1" pageId="b01.xml">Special Offers</link>
        <link accesskey="2" pageId="show-balance.php">Show Balance</link>
        <link accesskey="3" pageId="send-money.php">Send Money</link>
        <link accesskey="4" pageId="b04.xml">Contact Us</link>
    </navigation>
</page>
```

It is possible to combine the static and the dynamic service files and upload them on the web server. The service source is ready and it is possible to start its using.

Code Block 14 File system

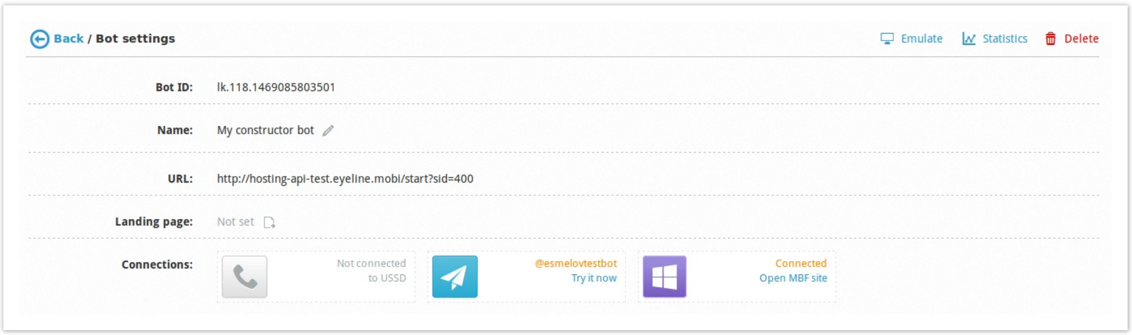
```
../service_site_directory/
    amount.php
    common.php
    config.php
    send-money.php
    show-balance.php
    transaction.php
    b01.xml
    b04.xml
    c01.xml
    index.xml
```

4 Connecting Service

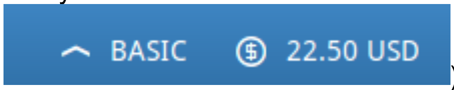
This chapter describes how to connect ***C2S*** number and then use it as user's access point to service. C2S number is a ***Call to Service*** phone number. By dialing this number a user receives USSD menu for communication with a service.

4.1 C2S number renting

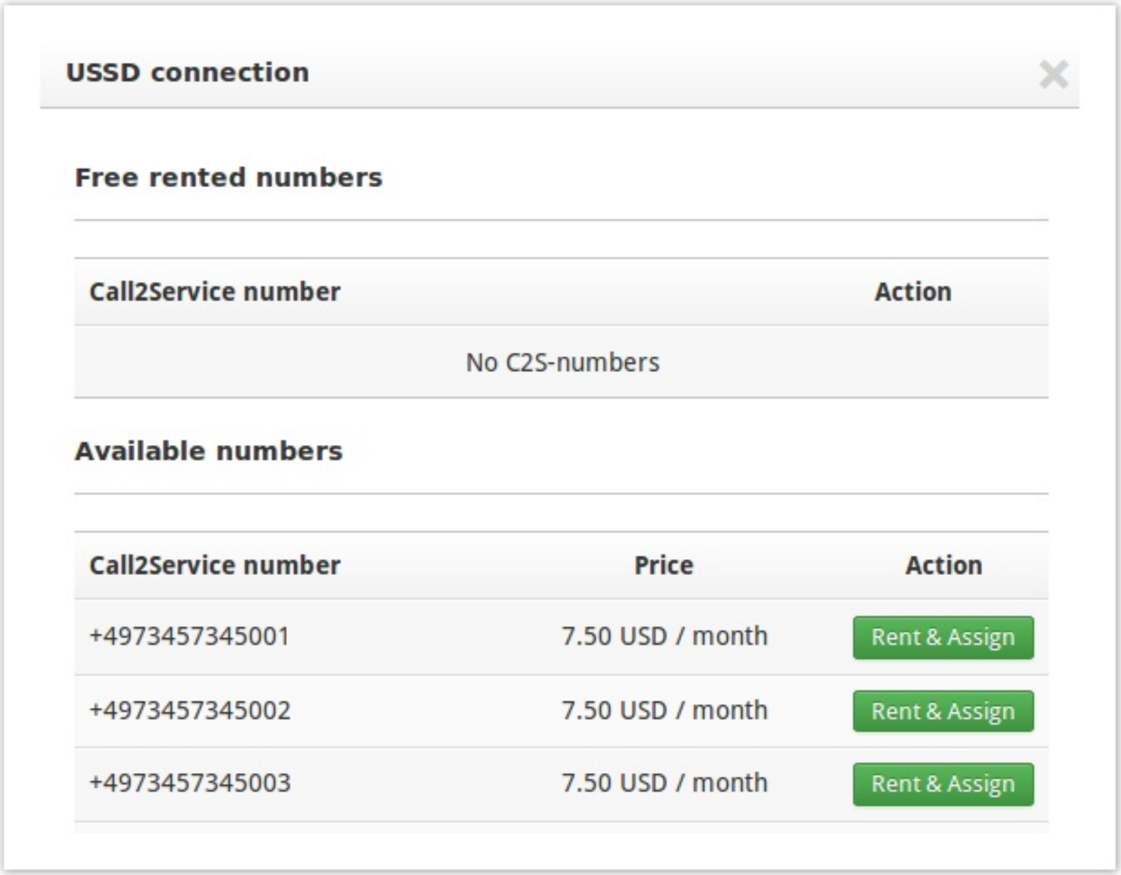
Log in to MiniApps using the link https://dev.miniapps.run. You will see all your created services (in the other words ***bots*** - in terms of MiniApps). Select the one you need. The bot details will appear:



Bot ID:	lk.118.1469085803501
Name:	My constructor bot
URL:	http://hosting-api-test.eyeline.mobi/start?sid=400
Landing page:	Not set
Connections:	Not connected to USSD; @esmelovtestbot Try it now; Connected Open MBF site

You will see a gray handset icon with ***Not connected to USSD*** text hint next to it. Click on the icon and a window, where you can rent C2S numbers, will appear (please notice that you must have a positive balance in your wallet to rent a C2S number. You can always check the amount at the top of the page 

):



USSD connection

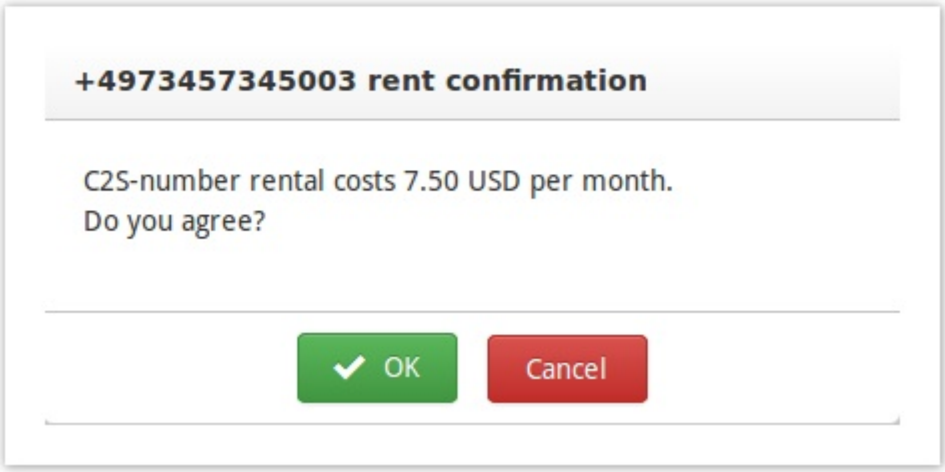
Free rented numbers

Call2Service number	Action
No C2S-numbers	

Available numbers

Call2Service number	Price	Action
+4973457345001	7.50 USD / month	Rent & Assign
+4973457345002	7.50 USD / month	Rent & Assign
+4973457345003	7.50 USD / month	Rent & Assign

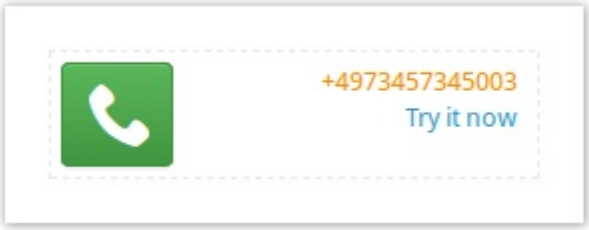
You can see the price for every available number. Find a suitable number and click ***Rent & Assign*** button. Then confirm renting with ***OK*** button.



+4973457345003 rent confirmation

C2S-number rental costs 7.50 USD per month.
Do you agree?

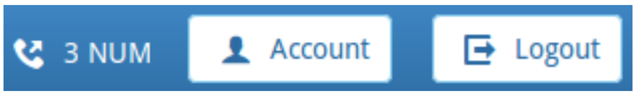
A message: ***Call2Service number was successfully assigned*** will appear. The bot information page is updated. Now you can see that the handset icon is active:



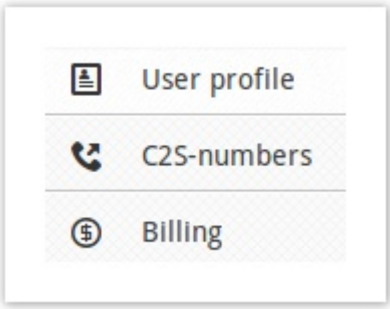
From now on, your bot can use USSD channel to communicate with users.

5 C2S number management

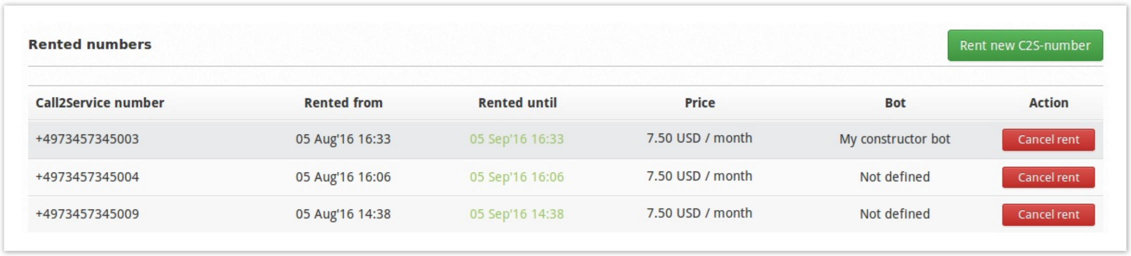
To see all your C2S numbers, click the ***Account*** button in the upper right corner of the window:



Select C2S-numbers tab:



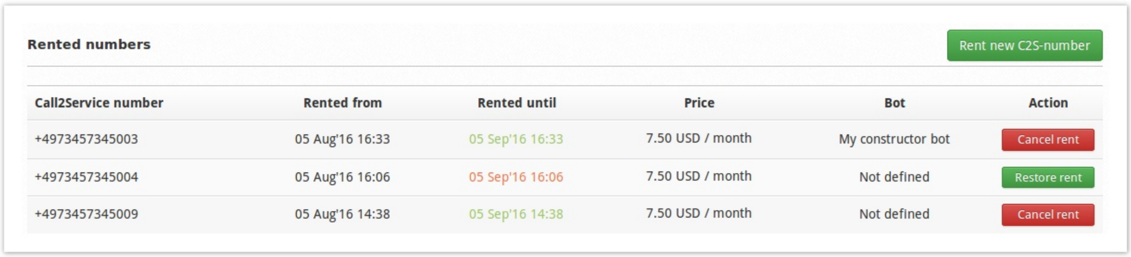
You will see all your rented numbers:



Rented numbers — Rent new C2S-number

Call2Service number	Rented from	Rented until	Price	Bot	Action
+4973457345003	05 Aug'16 16:33	05 Sep'16 16:33	7.50 USD / month	My constructor bot	Cancel rent
+4973457345004	05 Aug'16 16:06	05 Sep'16 16:06	7.50 USD / month	Not defined	Cancel rent
+4973457345009	05 Aug'16 14:38	05 Sep'16 14:38	7.50 USD / month	Not defined	Cancel rent

In the ***Bot*** column you will see if a number is assigned to a bot. If you don't the number anymore, click on the ***Cancel rent*** red button opposite the number:



Rented numbers — Rent new C2S-number

Call2Service number	Rented from	Rented until	Price	Bot	Action
+4973457345003	05 Aug'16 16:33	05 Sep'16 16:33	7.50 USD / month	My constructor bot	Cancel rent
+4973457345004	05 Aug'16 16:06	05 Sep'16 16:06	7.50 USD / month	Not defined	Restore rent
+4973457345009	05 Aug'16 14:38	05 Sep'16 14:38	7.50 USD / month	Not defined	Cancel rent

The number will be available for renting again only until the date stated in the column ***Rental until***.