



## **External USSD Channel Documentation.**

## Table of Contents

<b><u>EXTERNAL USSD CHANNEL .....</u></b>	<b><u>1</u></b>
<b><u>EXTERNAL USSD CHANNEL .....</u></b>	<b><u>3</u></b>
TECHNOLOGIES .....	3
CODING CONVENTIONS .....	3
<b><u>INSTALLATION .....</u></b>	<b><u>4</u></b>
DATABASE SETUP .....	4
USER INTERFACE SETUP .....	4
BUILDING A VIRTUAL ENVIRONMENT .....	4
SYNC YOUR DATABASE .....	5
START DJANGO SERVER .....	5
<b><u>INTEGRATION WITH RAPIDPRO .....</u></b>	<b><u>5</u></b>
<b><u>CHANNEL CONFIGURATIONS.....</u></b>	<b><u>8</u></b>
AGGREGATOR HANDLER CONFIGURATIONS .....	11
<b><u>RAPIDPRO USSD FLOW SETUP AND BEST PRACTICES .....</u></b>	<b><u>15</u></b>
<b><u>DEPLOYMENT (ENABLE LIVE SESSION TRACKING TABLE) .....</u></b>	<b><u>16</u></b>
<b><u>OTHER FEATURES .....</u></b>	<b><u>17</u></b>
DASHBOARD.....	17
USER PROFILE.....	18

# External USSD Channel

RapidPro External USSD channel is an open source software that uses RapidPro's Generic External Channel API to relay messages between RapidPro and USSD supported devices through USSD aggregator APIs that are configurable within the channel.

## Technologies

The minimum requirements and technologies for the channel service are the same as those for RapidPro: It is recommended to first have a grasp about the basics of how these technologies work and the programming languages they use before you can contribute to the code base.

### Frontend

- **Python - Django:**

The entire frontend is built in django, we try to use the latest version of django shortly in about 2 months when it's out.

### Backend

- **PostgreSQL:**

Although you can use other RDBMS together with Django as described [Here](#), We recommend and depend on PostgreSQL as our SQL server.

- **Redis:**

We use Redis as a reliable and fast broker to provide a locking mechanism during transactions between our Channel and RapidPro. It should be noted that the core logic of the system will not work when Redis is not installed.

## Coding Conventions

We generally try to adhere to the practices set forth by the Django, Python communities.

### Code Reviews & Pull Requests

All committed code must be code reviewed via a formal pull request. Whether it is a new feature work or bug fixes, all code should be written on a branch apart from master, then a pull request

should be opened to review those changes before committing to master. Any code on the master branch is considered ready for live deployment.

### Modularity of code

We try to keep related code together in less coupled modules that can be modified/removed from the core system at any time without grave effects on most parts of the system.

## Installation

In order to use the USSD External Channel Service, you must have a fully installed and working instance of the supported RapidPro system (i.e. post v5). Please follow the setup instructions found [Here](#):

Once RapidPro is up and running, you can install and configure the channel service as detailed below.

## Database Setup

Create a new database user called `ussd_user` for postgresSQL  
You can create one by running this command in your terminal

```
$createuser ussd_user --pwprompt -d
```

Create database named `ussd` by running the command below

```
$createdb ussd
```

If all is well with your PostgreSQL then your database is ready.

## User Interface Setup

Go on to clone the project [Here](#) by running the command below. You can create one by running this command in your terminal

```
$createdb ussd  
$git clone https://github.com/dsmagicug/rapidpro-external-ussd-channel.git  
$cd rapidpro-external-ussd-channel
```

## Building a virtual environment

This step is optional but it's highly recommended to use a virtual environment to run your External USSD channel installation. The pinned python dependencies for the project can be found in `pip-freeze.txt`. You can build the recommended environment as follows (from the root External USSD channel installation directory)

```
$ virtualenv -p python3 env
$ source env/bin/activate
$(env) $ pip install -r pip-freeze.txt
```

## sync your database

Still inside your project root directory, run the following command

```
$python manage.py migrate
```

Django countries requires you populate the Countries table with the latest country data using

```
$python manage.py update_countries_plus
```

## Start django server

Start the django (rocket) server by running the code below.

Then visit <http://localhost:8000>

```
$python manage.py runserver
```

### Running alongside RapidPro on the same machine

In most cases, you may want to install both [RapidPro](#) and the External USSD channel on the same server instance.

To set up RapidPro follow the instructions [Here](#)

Since they are both Django applications, you have to run `python manage.py runserver` for each of them. This will return an error **Address already in use**, since the Django development server by default runs on port `8000`.

The trick here is starting each on a different port as described in the django documentation [Here](#).

For our example, we shall use port `5000` to run the External USSD Channel development server.

Inside the External USSD channel project directory, run

```
python manage.py 0.0.0.0:5000
```

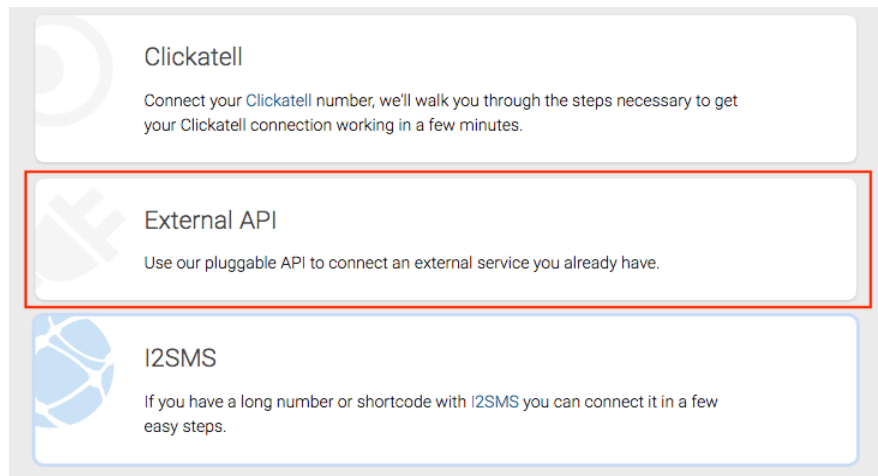
Visit <http://localhost:5000>

We recommend that you leave RapidPro use the default port `8000`

## Integration with RapidPro

For this step, You need an online RapidPro instance (it could be on the same server instance as the USSD channel on the same machine) as described [above](#).

Go to RapidPro and select External API as demonstrated in the image below.



*Figure 1: External API channel Option in RapidPro*

Proceed to the External API form as shown in the image below

### Connect External API

You can connect an external aggregator or messaging service to RapidPro using our external API. You can specify parameters to substitute in the URL or body by using these variables:

**text** the message of the text being sent  
**from** the phone number or address you have configured for this service  
**from\_no\_plus** the phone number or address you have configured for this service, with leading +s removed  
**to** the phone number or URN this message is addressed to  
**to\_no\_plus** the phone number or URN this message is addressed to, with leading +s removed  
**id** the unique ID of this message  
**quick\_replies** the quick replies for this message, formatted according to send method and content type

An example that would substitute variables in the URL:

```
http://myservice.com/send.php?from={{from}}&text={{text}}&to={{to}}&{{quick_replies}}
```

If using POST or PUT, you can specify the body of the request using the same variables.

The **quick\_replies** variable with method GET or content type URL Encoded will be replaced by **&quick\_reply=(reply)** for each quick reply.

After connecting your service we will provide URLs for the endpoints to call when you want to push a message to RapidPro or notify RapidPro of the delivery status of a message.

URN Type  
Phone number  
The type of URNs handled by this channel

Number  
The phone number or that this channel will send from

Handle  
The Twitter handle that this channel will send from

Address  
The external address that this channel will send from

Country  
Afghanistan  
The country this phone number is used in

Method  
HTTP POST  
What HTTP method to use when calling the URL

Encoding  
Default Encoding

Figure 2: External API channel Option in RapidPro

Head back to the External USSD Channel Web interface when you are sure the above steps.

# Channel Configurations

- Visit the web interface of RapidPro External USSD Channel system
- Create an account under Register if you haven't
- Login.

USSD CHANNEL

tester

MENU

- DASHBOARD
- CHANNEL CONFIGS
- HANDLERS
- DOCUMENTATION
- USER PROFILE
- LOGOUT

### USSD CHANNEL CONFIGURATIONS

Generating RapidPro SEND URL

SEND URL: `https://keeyas-MBP.lan/adaptor/send-url`

Please Enter your URL here, copy the generated SEND URL for use in the Send URL field on RapidPro's External API configuration form

`https://keeyas-MBP.lan`

Next

© COPYRIGHT 2020-2020 DIGITAL SOLUTIONS ALL RIGHTS RESERVED

Figure 3: Channel Configuration form

Continue to CHANNEL CONFIGS on the side menu (as shown in the image above) to configure a channel closely following the instructions on the Step by step form. Note that you will need to configure the following for your channel to work properly.

## 1. Send URL

This is a URL which RapidPro will call when sending an outgoing message. This URL will be generated as you input your local server URL required.

The format of the SEND URL will be like this; `http://yourmachine/adaptor/send-url`. Copy this link, head to RapidPro's External API form and paste it under the SEND URL field as shown below.



Send URL

The URL we will call when sending messages, with variable substitutions

Request Body

The request body if any, with variable substitutions (only used for PUT or POST)

MT Response check

The content that must be in the response to consider the request successful

Submit

Figure 4: RapidPro external service conf Instructions screen

- Fill the rest of RapidPro's External API configuration form and then submit it. You will be presented with instructions on how to setup an external service (The USSD External Channel) similar to the image below.

### External API Configuration

\*255#  
Uganda

To finish configuring your connection you'll need to set the following callback URLs on your service or aggregator.

Send URL

When we need to send an outgoing message it will make a POST to this URL with the parameters 'text', 'to', 'from', 'channel' and 'id'

EXAMPLE

```
POST http://ussd-channel.familyconnect.co.ug/channel-config/adaptor/send-url
id=1241244&text=Love+is+patient.+Love+is+kind.&to=%2B250788123123&to_no_plus=250788123123&from=%2A255%23&from_no_plus=%2A255%23&channel=2&session_status={{session_status}}
```

Received URL

When a new message is received by your service, it should notify us with a POST to the following URL, passing the following parameters: 'from' and 'text'. Callers can optionally also send a 'date' parameter in ISO-8601 (ex: 2012-04-23T18:25:43.511Z) format to specify the time the message was received.

EXAMPLE

```
POST https://app.familyconnect.co.ug/c/ex/499ca20c-882e-4ede-b7a8-65c03045f159/receive
from=%2B250788123123&text=Love+is+patient.+Love+is+kind.&date=2012-04-23T18:25:43.511Z
```

Figure 5: Receive URL required from RapidPro at step4 below

**NOTE:** Before submitting RapidPro's External API configuration form, make sure you add cgi params.

`session_status={{session_status}}&to_no_plus={{to_no_plus}}&from_no_plus={{from_no_plus}}`

to the Request Body field. These are required when rapidPro is making a request to the USSD external channel in addition to the

default `id={{id}}&text={{text}}&to={{to}}&from={{from}}&channel={{channel}}`.

Your configuration will not work if these cgi-params are not set at this level since the channel uses `session_status` within the request from RapidPro to tell if the interaction is still ongoing or completed so as it initiates the right USSD menu to the end user device. If you are confused, just copy

`id={{id}}&text={{text}}&to={{to}}&to_no_plus={{to_no_plus}}&from={{from}}&from_no_plus={{from_no_plus}}&channel={{channel}}&session_status={{session_status}}` and paste it in Request Body field.

3. On the above instructions page in RapidPro, we are ONLY interested in the RECEIVE URL. Copy it and head back to our USSD channel form.
4. Click Next and paste the URL there. (make sure the <https://app.rapidpro.io> part of this URL is substituted with the correct link to your RapidPro instance and please leave the other part `/c/ex/...../receive` untouched).
5. Click Next to configure the Channel's Max-timeout (default=10s). This is the time in seconds our channel shall wait for a response from RapidPro. It will time out if RapidPro does not reply within that time interval.
6. Click Next to configure the Trigger word (default=USSD). This is a keyword to trigger a given flow execution in Rapidpro when a contact starts a new session when it isn't involved with any RapidPro flows yet.  
Please **note** that in RapidPro, a keyword can be set to trigger only one flow at a time. So we advise that if multiple choices of flows are available to be used in USSD, The Rapidpro flow creator should create a flow which routes to other flows then assign this keyword to that flow. i.e a User on USSD will be presented with options of which flow to participate in then get routed accordingly. This way one keyword can trigger the router flow, and routes will be made according to user replies.
7. Submit the form to save the channel configurations.

# Aggregator Handler Configurations

Since different USSD aggregator APIs have different Request/Response formats, this channel standardizes these formats into one understood by RapidPro.

This is achieved through configuring handlers for each of the aggregator APIs that you want to use.

To set up a Handler;

Go to **HANDLERS** on your side menu, click **Add Handler** button. You will be presented with a form (as shown in the image below).

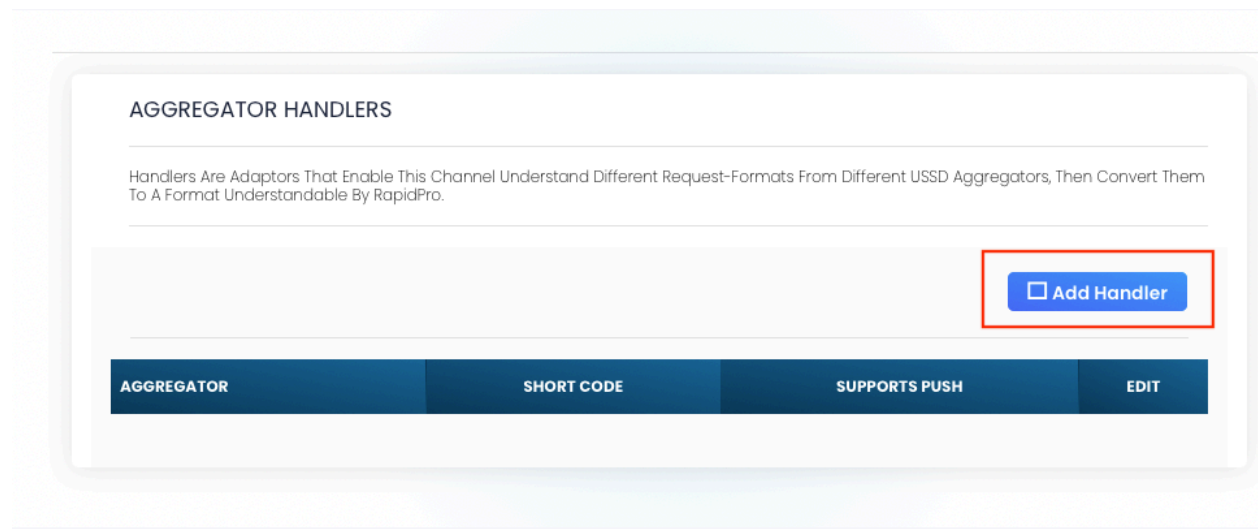


Figure 6: Add Handler Button

## REGISTER A NEW AGGREGATOR HANDLER

A handler enables this channel understand different request-formats from different USSD aggregators, then convert them to a format understandable by RapidPro.

Aggregator:

e.g. DMARK

USSD aggregator whose requests will be handled by this handler.

Short code:

e.g. 255

The USSD shortcode provided by the aggregator above.

Request format:

```
{{short_code=ussdServiceCode}}, {{session_id=transactionId}}, {{from=msisdn}}, {{to=msisdn}},  
{{text=ussdRequestString}}, {{date=creationTime}}
```

The format of the request string from aggregator's USSD API e.g.  
in `{{from=msisdn}}`, the values of key `msisdn` in the request will be converted  
and assigned to key `from` before being sent to RapidPro by the channel.

Figure 7: Register Handler Form

Below are the fields on the above form and what they require

### 1. Aggregator

The name of the USSD aggregator that you want to use for example DMARK or [Africa's Talking](#))

### 2. Shotcode

The shortcode provided by your aggregator e.g 348.

### 3. Request format

This is a very important field which must be set correctly in order for the channel to safely standardize aggregator Requests/Responses.. The template format is provided already in that textarea as shown below.

```
{{short_code=ussdServiceCode}}, {{session_id=transactionId}},  
{{from=msisdn}}, {{text=ussdRequestString}}, {{date=creationTime}}
```

Each is a combination of two parameters, on either sides of the equal sign(=).  
The one on the left represents the format understood by RapidPro and our External channel which MUST not change.  
The one on the right hand side represents the format in which the aggregator requests delivers that value to our channel. for example

```
{{short_code=ussdServiceCode}}, {{session_id=transactionId}},  
{{from=msisdn}}, {{text=ussdRequestString}}
```

means the request from the aggregator API in this case say DMARK will arrive as

```
{"ussdServiceCode":"257","transactionId":123456789,  
"msisdn":"25678xxxxxx","ussdRequestString":"Hello world"}
```

the above format will then be converted into

```
{"short_code":"257", "session_id":123456789, "from":"25678xxxxxx",  
"text":"Hello World"}
```

which is understood by Rapidpro and our Channel. for example if a particular aggregator represents short\_code as serviceCode, the combination to map this will be {{short\_code=serviceCode}}. Please refer to your respective aggregator USSD Docs for their formats.

Note that our channel and RapidPro have the following standard formats

<b><u>session_id</u></b>	for USSD session ID
<b><u>short_code</u></b>	for USSD short codes e.g. 348
<b><u>from</u></b>	for the phone number in the Request.
<b><u>text</u></b>	for the content(message) in the request i.e. User replies.
<b><u>date</u></b>	(Optionally set) for the time string in the request. Note that apart from date, your request format has to cater for all the rest by mapping them to their equivalents in the USSD request from a given aggregator API.

#### 4. Response content type

this is how the response to the aggregator API should be encoded (default is application/json) (refer to your aggregator's USSD Docs for such information )

#### 5. Response method

Specifies whether your aggregator expects a POST, GET or PUT method in the response.

## 6. Signal reply string

this is a keyword in the response to your aggregator's API that is used to signal further interaction in the USSD session, i.e. a USSD menu with an inputbox for the end user to reply. (Refer to aggregator's USSD Docs for this keyword)

## 7. Signal end string

this is the Keyword your aggregator uses in the response string to indicate end of USSD Session in order to send a USSD prompt without an inputbox to the end user device. (Refer to aggregator's USSD Docs for this keyword)

## 8. Response format

This field indicates the format expected by the aggregator's API in the response sent by our channel. two options are so far provided.

- Is Key Value

(Default) means the aggregator API will accept the message and signal response string(seen above) from our channel if its in a json like key-value string e.g

```
{"responseString":"Hello User how are you":  
"signal":"request"}
```

- Starts With

means the aggregator expects a string in the response body that starts with a keyword to signal end or request for interaction as seen above.

## 9. Response Structure

if option 1 (Is Key Value) in 8 above is chosen, a textarea appear expecting entries similar to ones discussed under Request Format in (3) above.

With similar logic as in (3) above, our channel understands as follows

- text

for the text(message) in the response

- action

for the signal keywords i.e. (the ones that signal end of session or more interaction as discussed above).

for example, if aggregator A's expected response is in this format;

```
{"responseString":"Hello User how are  
you":"signal":"Signal_keyword"}
```

the entry in this field should then be

```
{{text=responseString}}, {{action=signal}}
```

## 10. Push support

Boolean to specify whether your aggregator supports USSD PUSH protocol i.e MT USSD sessions (Default=False)

Submit this form and enjoy the power of RapidPro through USSD. You can configure multiple handlers for multiple aggregators but each aggregator must have one handler depending on the format of their requests/responses.

# RapidPro USSD Flow Setup and best Practices

Although the channel does not require much changes in the normal RapidPro SMS flows, for a swift experience using this channel, you may want to revisit your SMS flow designs to best serve USSD. Below we present you our recommended best practices.

- Create a RapidPro trigger that corresponds to the **Trigger word** that was specified during step 6 under Channel config above.
- If you want a single shortcode to support multiple flows, create a single flow that routes to other flows and assign the trigger (create in above) to that flow. for example, if You have 3 flows (Register flow, Login flow, and Survey flow), you can create an extra flow say **Link flow** and route the user accordingly as below.
  1. Register
  2. Login
  3. Survey
- Ensure that there are no open nodes in the flows, for example if a node (a step-in flow) requires a user to provide options as below;
  1. Yes
  2. No
  3. Opt out
  4. Back to main menu

Your flow should have a node that handles a scenario where a user enters a 5 or anything undesirable. This is normally provided by RapidPro using the route called **Other**. Make sure **Other** is handled and not left open. i.e. you can create a node that notifies the user of a

wrong entry and routes back to waiting another entry. Failure to handle other may result into bad channel behaviour which can be frustrating.

- When designing your flows for USSD, make sure the user does not have to enter long responses whenever you can. e.g. provide options to pick from as numbers (1,2,3,4,5....) like in the above examples.
- Its good practice to provide an option for Cancelling and routing back to the previous step or back to main menu in your flows. For example
  1. Yes
  2. No
  3. Back
  4. Back to main menu
  5. Cancel (opt out)
- Lastly, RapidPro provides a flow feature which expires inactive contacts after a given period of time, make sure you specify this period during or after flow creation to avoid scenarios where a contact who dialed a shortcode a month ago and probably forgot where they stopped, comes back and picks up from where they left off. You may want to set the flows to restart these cases afresh.

## Deployment (enable Live session tracking table)

We have looked at setting up a development server for this channel above. let us look at how quickly you can set up an instance for production.

The External Channel applications uses [Django Channels](#)

Channels is a project that takes Django and extends its abilities beyond HTTP - to handle WebSockets, chat protocols, IoT protocols, and more. It's built on a Python specification called [ASGI](#)

In order to support live session table on the dashboard i.e. whenever a user initiates a USSD session, it can be visible in realtime on the graph and table on the dashboard. This ability uses websockets which if not deployed well may disable the feature.

Note that this has no negative effects on the overall performance of the system. It just grants you the ability to watch USSD sessions as they are initiated in realtime. If you are excited about this feature, we offer you an easy way of getting started.

- The system comes with [Daphne](#) which is a HTTP, HTTP2 and WebSocket protocol server for ASGI and ASGI-HTTP, developed to power Django Channels.
- The system has an `asgi.py` file under `project_folder/core/` which is ready to get you started quickly. What you need to do is simply run `daphne -b 0.0.0.0 -p 8000 core.asgi:application` inside your project directory and you are good to go. You can



```
run daphne user a python virtual environment by simply using venv/bin/daphne -b 0.0.0.0 -p 8000 core.asgi:application
```

You can run this as a daemon if you want using [systemd](#) or similar projects on Linux or macOS.

There are also alternative ASGI servers that you can use for serving Channels as described [here](#).

Another important resource on this can be found [here](#)

You can as well deploy the system using wsgi and it will still work, except you won't get a live session tracking table on your dashboard. Under this setup, you can only see new USSD sessions after reloading the page. But the graphs will still give you a relatively better update of what's happening.

Thank You

Good luck.

## Other Features

The USSD External Channel web interface has other features. Below we look at them.

### Dashboard

The dashboard provides a summary of live USSD sessions in two forms as shown in the image below.

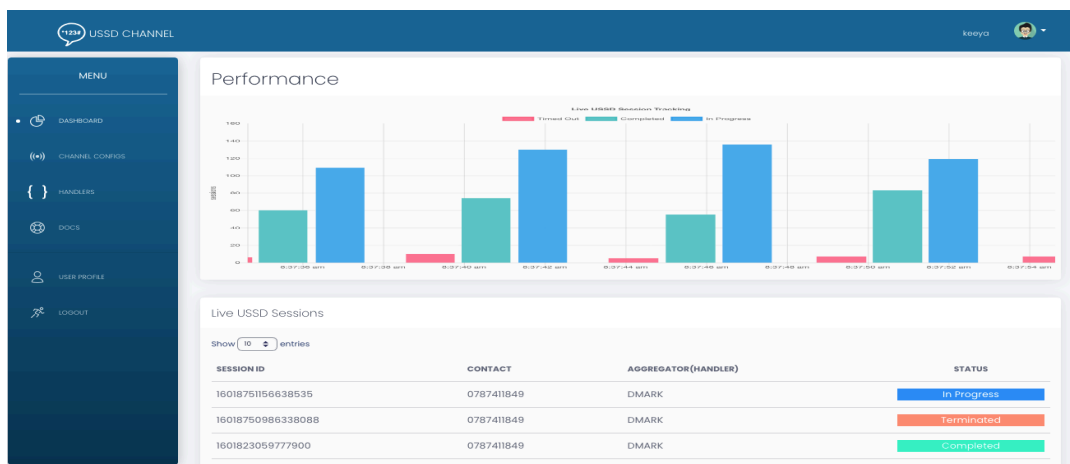


Figure 8: A screenshot of the Dashboard

## 1. Bar Graph

This represents sessions on an x-axis (Timestamp in intervals of 5 secs) and y-axis (No. of sessions).

The key of this graph clearly shows that sessions are tracked by status i.e.

- Timed Out: **Red**
- Completed: **Green**
- In Progress: **Blue**

**Attention** is needed when the red Bar is taller or the only visible bar. This means sessions are timing out in at the USSD External channel side. i.e. The channel can not reach RapidPro or RapidPro is not responding to the requests our channel forwards to it. Please check if your RapidPro instance is online or if it is, check if it's replying to the send-url you set during channel configurations.

You may also want to check if the send-url you set is correct.

## 2. Tabular Format

The table displays Live USSD Sessions, every new session is added to the top of the table. Whereas the Bar Graph represents only three session statuses, the table adds another status on top of the 3 seen above. This is the **Terminated** status.

Like in the Bar Graph, **Red** calls for Attention but **Orange** should not raise any concerns. This is because USSD session termination is normal, a USSD session may expire on the end User's device or they may choose to cancel it by pressing the **CANCEL** button on their USSD enabled devices.

Once this happens, the next time a this user initiates a new USSD, the last one is labelled as terminated. Therefore, for every terminated session, it means a user was resumed from where they stopped in a given flow.

# User Profile

This section handles Current user profile management. for example, editing username, email and changing the password.

The Admin user (super user) is capable of the following extra activities;

### 1. Enabling/Disabling other users.

Disabled users will not be able to log into the system.

### 2. Inviting Users by email.

The Admin clicks, Invite User, then a form will appear (as shown in the image below) requesting for an invite email address to which on invite, a new account will be created. This account will have a randomly generated password. and a username which will be the

first part of the email address before @.

An email will then be sent to the address specified, with a link to the USSD External channel login page and the auto generated credentials. The invited user can use these to login for the first time and can change their credentials once logged in.

Note that these users won't be Admins, so they can only change their account profiles, other options like Invite User, Enable/Disable users won't be accessible to them.

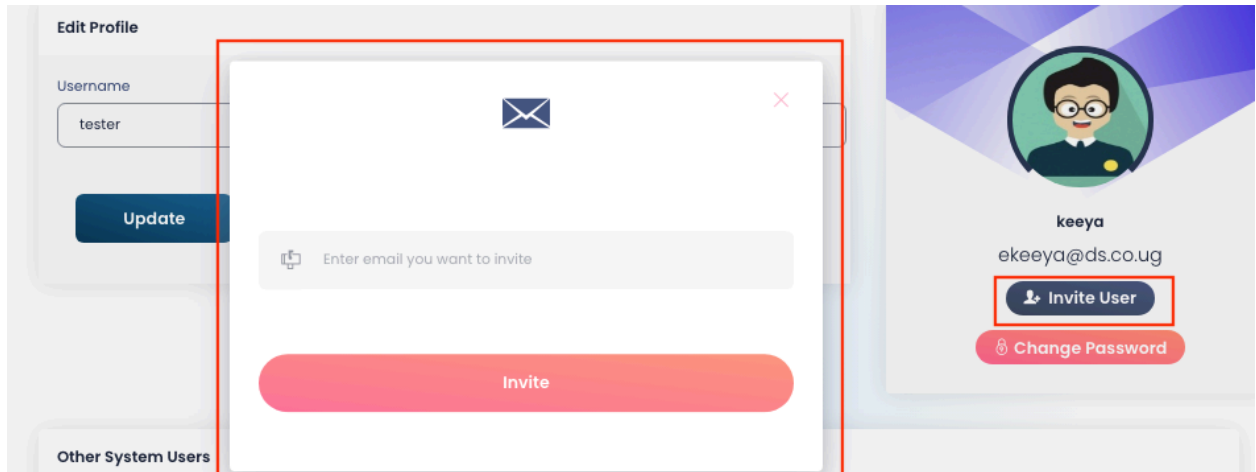



Figure 9: Invite user by email

Note that for emails to be sent out, and SMTP server credentials must be provided in `core/settings.py` below are the SMTP settings required for feature to work

```
HOST = "http://link_to_external_channel.app" (this is your USSD  
External channel server link which will be sent in the email)  
EMAIL_HOST = "smtp.domain.com" (smtp server host e.g for gmail it's  
smtp.gmail.com)  
EMAIL_PORT = 25 (SMTP port e.g 25, 465, 587)  
EMAIL_HOST_USER = "example@app.com" (your email address or username  
on this smtp server e.g a gmail user can set this to  
tester@gmail.com )  
EMAIL_HOST_PASSWORD = "xxxxx"(Your email password)
```

keeya

Edit Profile

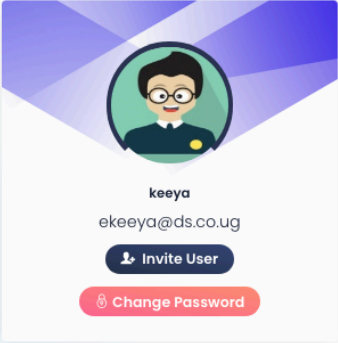
Username

tester

Email address

tester@ds.co.ug

Update



keeya

ekeeya@ds.co.ug

Invite User

Change Password

Other System Users

USERNAME	EMAIL	ENABLED?
ekeeya	ekeeya@ds.co.ug	Enabled
ksberry	ksberry@ds.co.ug	Disabled
kmavin	kmavin@gmail.com	Disabled

Figure 10: User profile page for Admin users