# TracPro Documentation

***Release 1.5.0***

**UNICEF**

February 27, 2017

TracPro is a minimal, generic dashboard for polls in any RapidPro org, such as for a real-time monitoring service for education. TracPro is for simple dashboards that help to create information loops, not for advocacy or flashy visuals. TracPro is open source, anyone can build their own dashboard. Deployment requires advanced Linux systems administration experience, but once running it can be configured and used by anyone.

TracPro was built for UNICEF by Nyaruka and is currently maintained by Caktus Group.

CHAPTER 1

# Technical Overview

## Concepts

- *Polls* are RapidPro flows. Click on the name of a poll to see a summary of the responses over time and dates it was conducted.
- *PollRuns* are the dates a poll was conducted. Click on a poll date to see a summary of just the responses for that date, participation among reporter groups, send a message to contacts, individual responses, and export responses.
- *Contacts* are the same as in RapidPro, they are the respondents of polls.
- *Supervisors* could be, for example, a provincial monitoring officer in a ministry. They are meant to see a particular region or multiple regions, and encourage participation by seeing who is responding, and restart polls. Supervisors also manage contacts.
- *Administrators* setup TracPro for an org and can manage everything supervisors can, in addition to managing supervisors.
- *Regions* are administrative areas, and can be anything, for example a city, county, or state in the USA. They must be configured as contact groups in RapidPro first. Regions can be made hierarchical in TracPro. For example, a supervisor who has access only to Kampala, would not have access to all of Africa's regional data, and a second supervisor who had access to Africa, would have access to data for all sub-regions within Africa.
- *Reporter groups* are types of contacts, such as head teachers, or grade 5 teachers. They must be configured as contact groups in RapidPro first.
- *Recent Indicators* allow users to compare baseline poll results to follow up poll results over time. For instance, if a poll was sent out to gather enrollment numbers at the start of the term, that could be selected as the baseline poll, and a follow up poll could be the poll sent out over the term asking for attendance numbers. Users may also spoof poll data to create sample results to chart on Recent Indicators.
- *Inbox Messages* allow users to view and send unsolicited messages, that is, messages outside of a flow, to contacts.

## Features

**Administrator accounts**: Configure a dashboard for any org, such as choosing polls, reporter groups, and regions to make available.

**Supervisor management**: Administrators can easily add, remove, and configure supervisors. Administators set up any desired access to particular regions for supervisors. Supervisors have access to data for their regions and to Homepage Poll data, Contacts, Inbox Messages and Recent Indicators.

**Contact management**: Supervisors can create, manage, and remove contacts, including changing their region, reporter group, facility and language, and all those changes are synchronized in the background with RapidPro.

**Start polls**: Supervisors can start any poll brought into TracPro from the Homepage.

**Send messages**: Send messages about polls to non-responsive contacts, responsive contacts or all contacts of a group. Select a Poll, select a date that the poll ran from the list of dates it was conducted on (PollRuns), choose the Participation tab, and click the button to "Send Message..." and choose whether to send it to respondents or non-respondents. These messages can be viewed from the Message Log.

**Activity**: See the most active regions and the most active reporter groups.

**Participation**: See participation of every contact in all polls on one page.

**Right-to-left (RTL) language support**: Any language can be displayed, whether left-to-right or right-to-left with content switching correctly too.

**Easy translations**: TracPro uses Transifex.com, the easy way to add a translation. Anyone can add translations that are then available for others to use in their TracPro dashboard.

**Automatic charts**: Simple analytics are automatically created because TracPro and RapidPro know what the type of response is for a flow step, i.e. numeric, multiple choice, open-ended text, or audio recording from IVR. Responses can be seen for time-series or for one poll issue.

**Export as CSV**: Export data from TracPro into CSV format, which can easily be imported into Microsoft Excel or other tools.

**Edit poll names**: Change any poll description.

**Mobile-ready**: TracPro works in smartphone and tablet browsers.

CHAPTER 2

# Getting Started

The following steps take you through the process of creating a new organization in TracPro. If you are using your local development installation, then ensure that both the web server and Celery are running.

## Preparing RapidPro

Before setting up your TracPro organization, one should ensure that the RapidPro organization has the following:

- A set of contact groups representing geographical regions, e.g. *Kigali*, *Florida*
- A set of contact groups representing reporting groups, e.g. *Males*, *Teachers*

Obviously you will also want to define some flows in RapidPro which are suitable for running as polls.

## Creating An Organization In TracPro

- Navigate to http://localhost:8000/
- Log in as a superuser
- Navigate to http://localhost/manage/user/ and add a new administrator user account
- Navigate to http://localhost/manage/org/ and click *Add* to create a new organization
- Include the newly created user as an administrator
- Use the API token provided by your RapidPro organization. If you don't know it then visit the API explorer.
- Save new org and navigate to http://SUBDOMAIN.localhost:8000/ where SUBDOMAIN is the subdomain of your new organization
- Log in as the new administrator user

## Configuring An Organization

There won't be much to see until you tell TracPro about which flows and groups to use.

- Navigate to **Administration > Polls** and click **Select** to select which flows in RapidPro will be used as polls in TracPro

- Navigate to **Administration > Reporter Groups** and click **Select** to select which contact groups in RapidPro will be used as reporter groups in TracPro
- Navigate to **Administration > Regions** and click **Select** to select which contact groups in RapidPro will be used as regions in TracPro. This will trigger a fetch of all contacts from those groups.

## Fetching old runs

If a new poll is added, TracPro will only track runs made after the poll has been added. If you need to fetch older runs, then there is a button which allows you to do this.

- Navigate to *http://SUBDOMAIN.yourtracprodomain/*.
- Navigate to **Administration > Organization** and click **Fetch runs** (on the right side near the top).
- Enter how many days in the past the fetch should go. For example, to fetch runs from the last two weeks, enter 14.
- Click **Submit**.

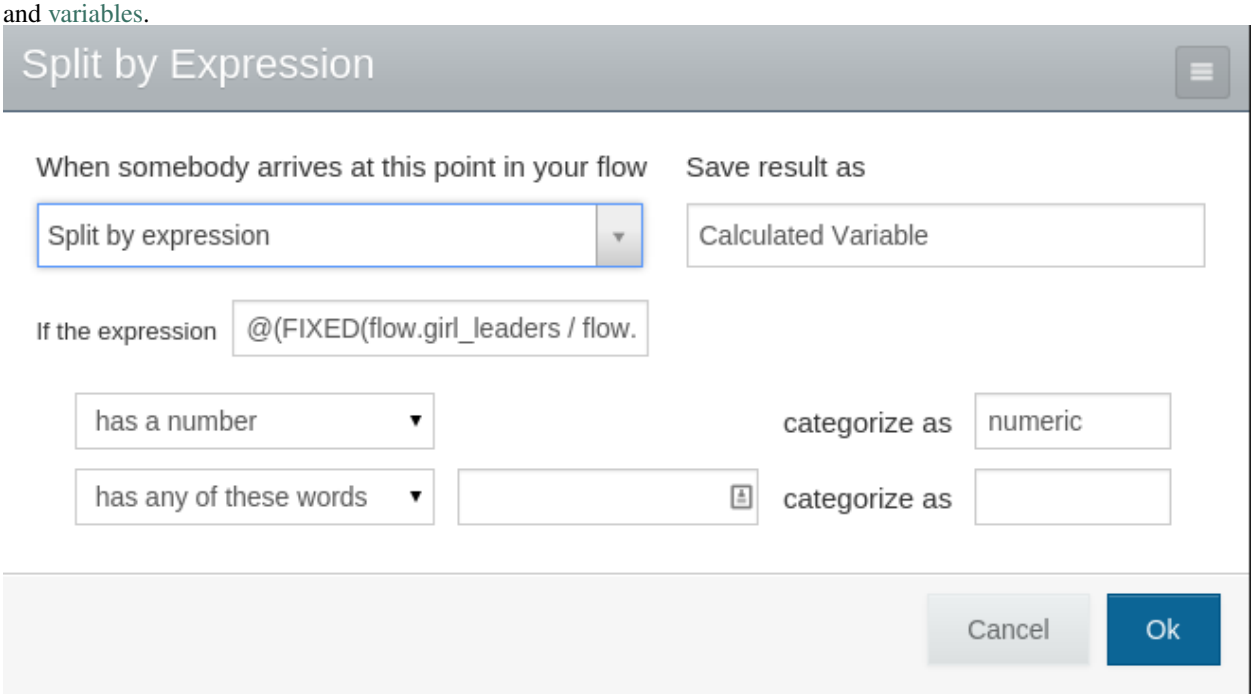
**One should use this command with caution as it could potentially try to download a very high number of runs**

## Displaying Calculated Values on Tracpro

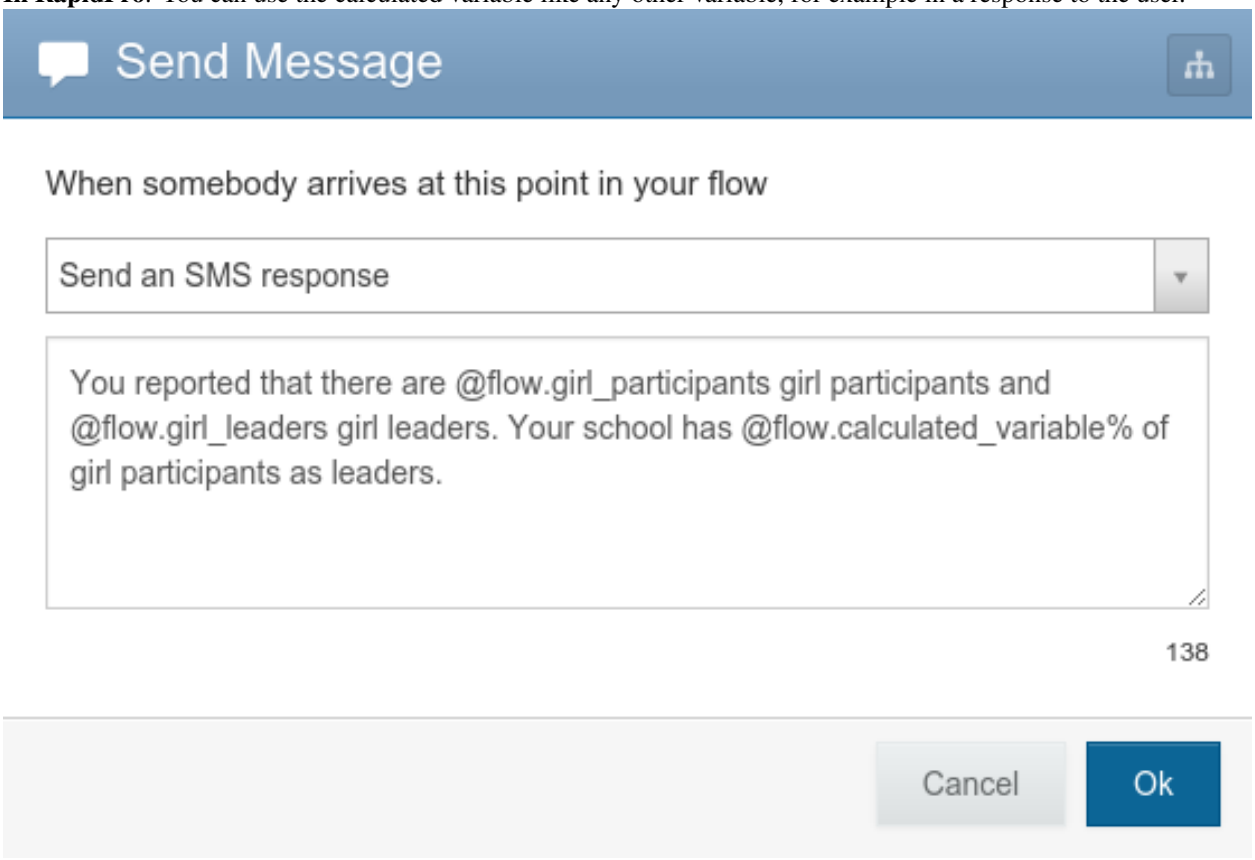
You can display values that are calculated from user responses on Tracpro. We reference an `example flow` that uses a calculated variable from results of a structured message form.

**In RapidPro**: After you have built your flow to collect variables that you wish to use for the calculation, add a "Split by Expression" step.

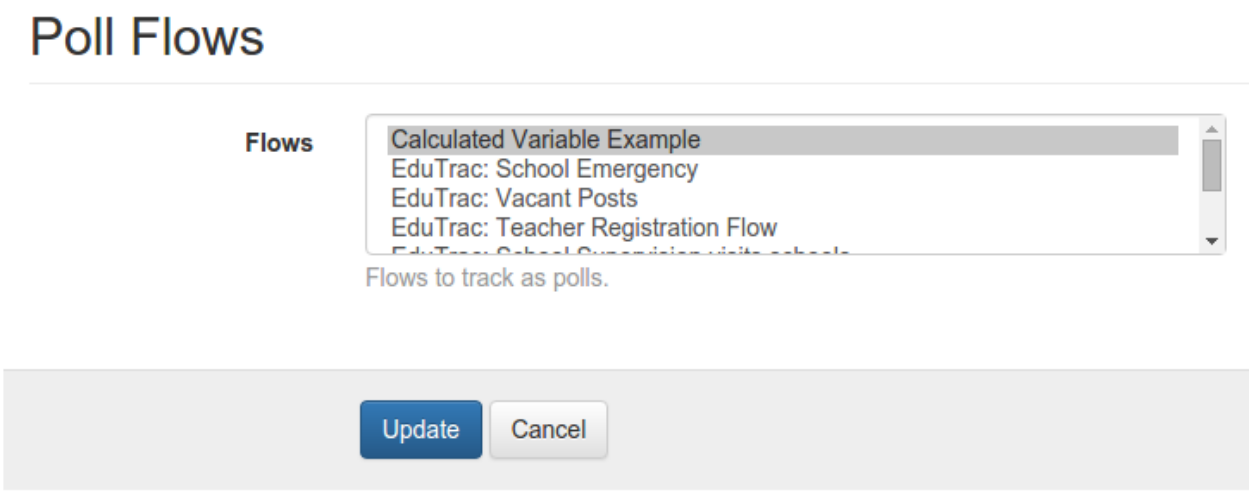
Save the result as a particular variable name, for example *Calculated Variable*. Write the expression to calculate the variable, for example *@(flow.girl_leaders / flow.girl_participants * 100)*. You have access to basic math, expressions, and variables.



**In RapidPro**: You can use the calculated variable like any other variable, for example in a response to the user.



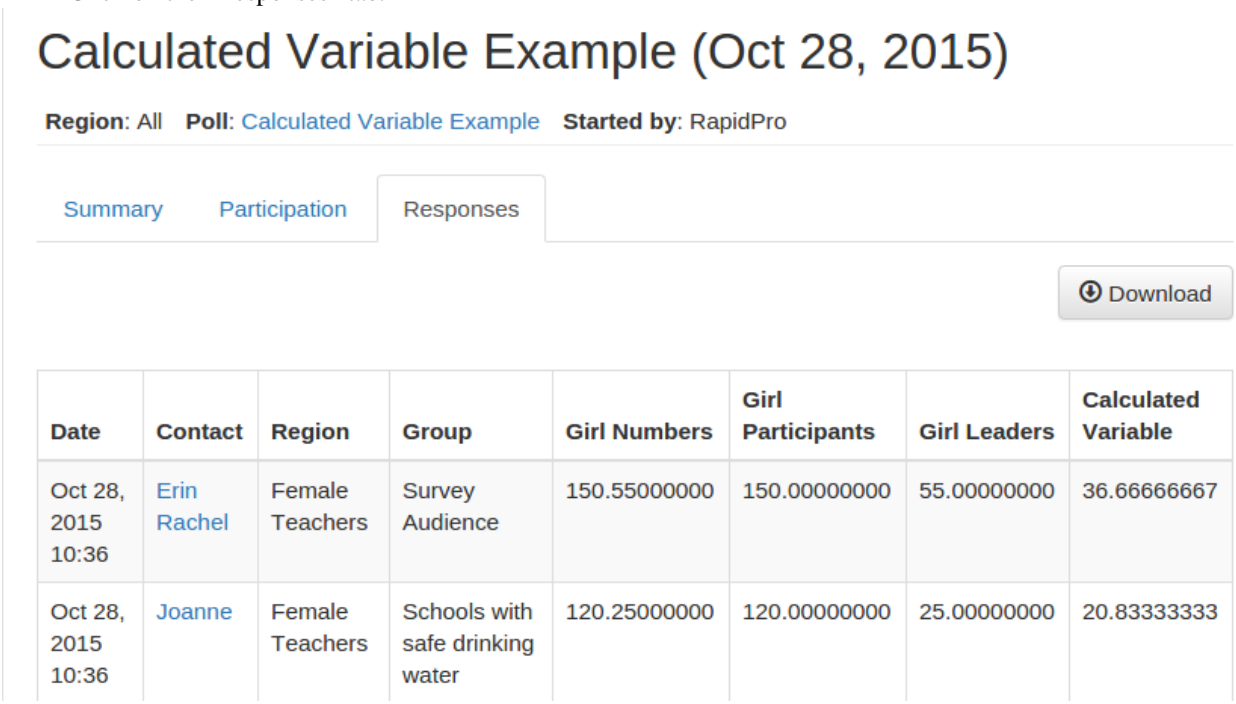

**In TracPro**: To view on Tracpro, ensure that your poll has been added through the Poll Flows chooser.




**In TracPro**: Your calculated variable will be available as a Question on Tracpro. You can view and use it exactly as you would any other Question.

- To view the numeric responses, including those for the calculated variable:
- View the Poll.

- Click on the “Dates” tab.
- Select the date of the poll you wish to see.
- Click on the “Responses” tab.



Calculated Variable Example (Oct 28, 2015)

**Region**: All **Poll**: Calculated Variable Example **Started by**: RapidPro

Summary | Participation | Responses

Download

| Date | Contact | Region | Group | Girl Numbers | Girl Participants | Girl Leaders | Calculated Variable |
|---|---|---|---|---|---|---|---|
| Oct 28, 2015 10:36 | Erin Rachel | Female Teachers | Survey Audience | 150.55000000 | 150.00000000 | 55.00000000 | 36.66666667 |
| Oct 28, 2015 10:36 | Joanne | Female Teachers | Schools with safe drinking water | 120.25000000 | 120.00000000 | 25.00000000 | 20.83333333 |

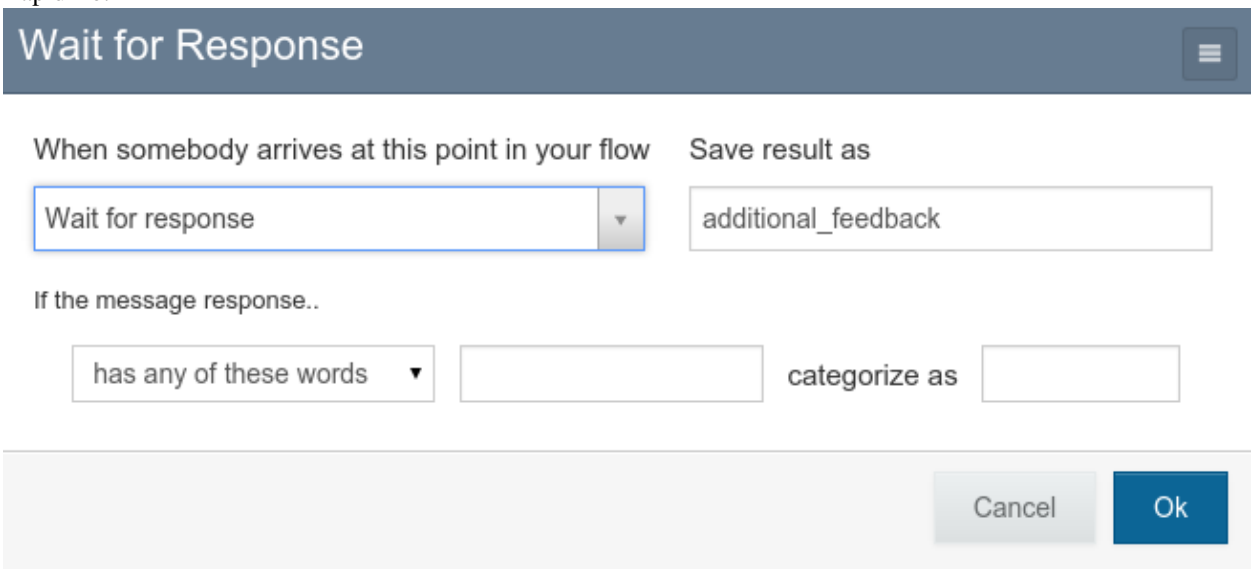
CHAPTER 3

# Question types

TracPro questions are classified as either categorical or open-ended. Question type is determined by how the flow variable is set up on RapidPro.

## Open-ended questions

To set up an open-ended question, do not change the default choices in the "If the message response..." section on RapidPro.

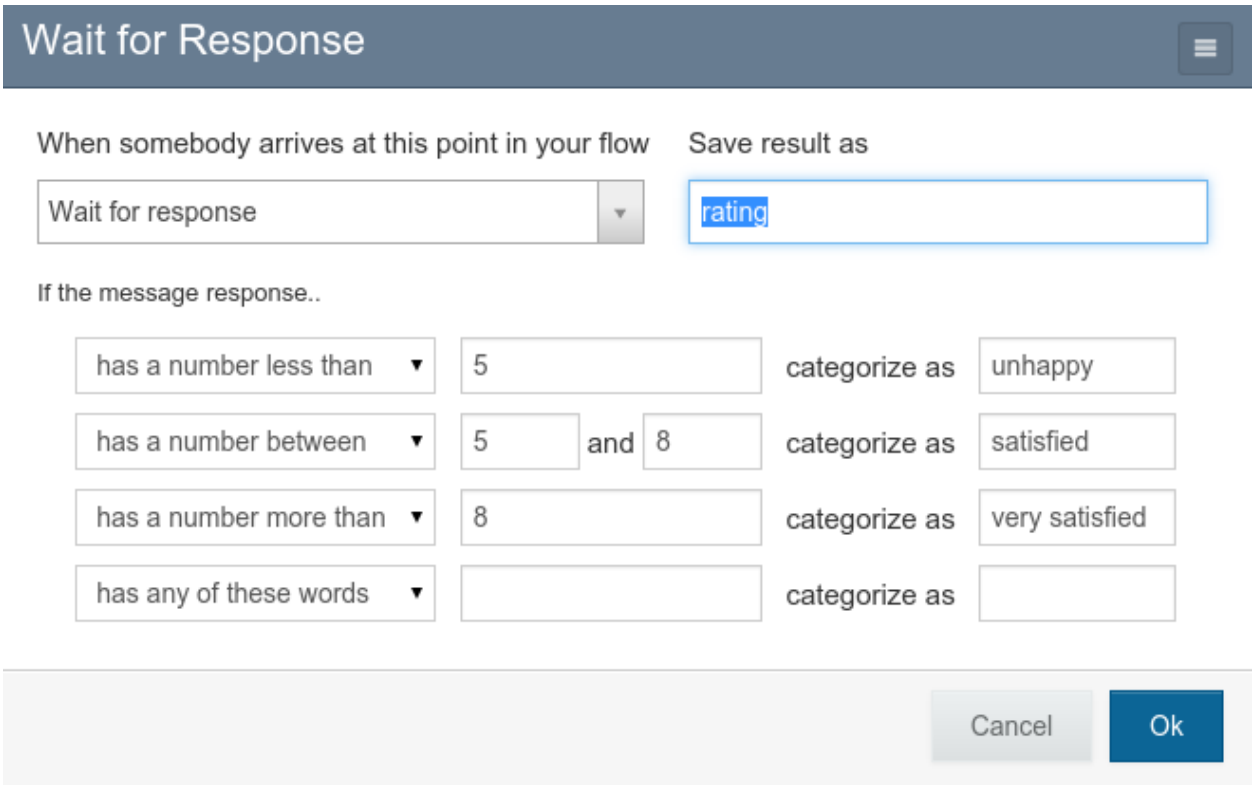



Your question will look like this on the flow editor:


Data for open-ended questions is displayed as a word cloud.

## Categorical questions

To set up a categorical question, define categories for your responses on RapidPro.




Your question will look like this on the flow editor:


Data for categorical questions is displayed as a pie chart.

CHAPTER 4

# Desigining Flows

This section addresses additional design considerations for users who are building RapidPro flows that will be imported as TracPro polls.

See the RapidPro knowledge base for detailed instructions on how to use the RapidPro flow editor tool.

## Rules

Rules are the directives that categorize a contact's responses to an SMS message. A rule set is the ordered group of rules that are applied to a response; the first applicable rule determines the response's category and therefore the next step the user will visit in the flow.

RapidPro flow steps correspond with TracPro questions. TracPro stores the rules that are associated with the flow step in order to determine all answer categories that are available for the question. In the case of numeric questions, TracPro uses the rules to categorize calculated values, for example to determine the category of the question's average answer.

## Numeric rules

Several rule types address numeric input:

If the message response...

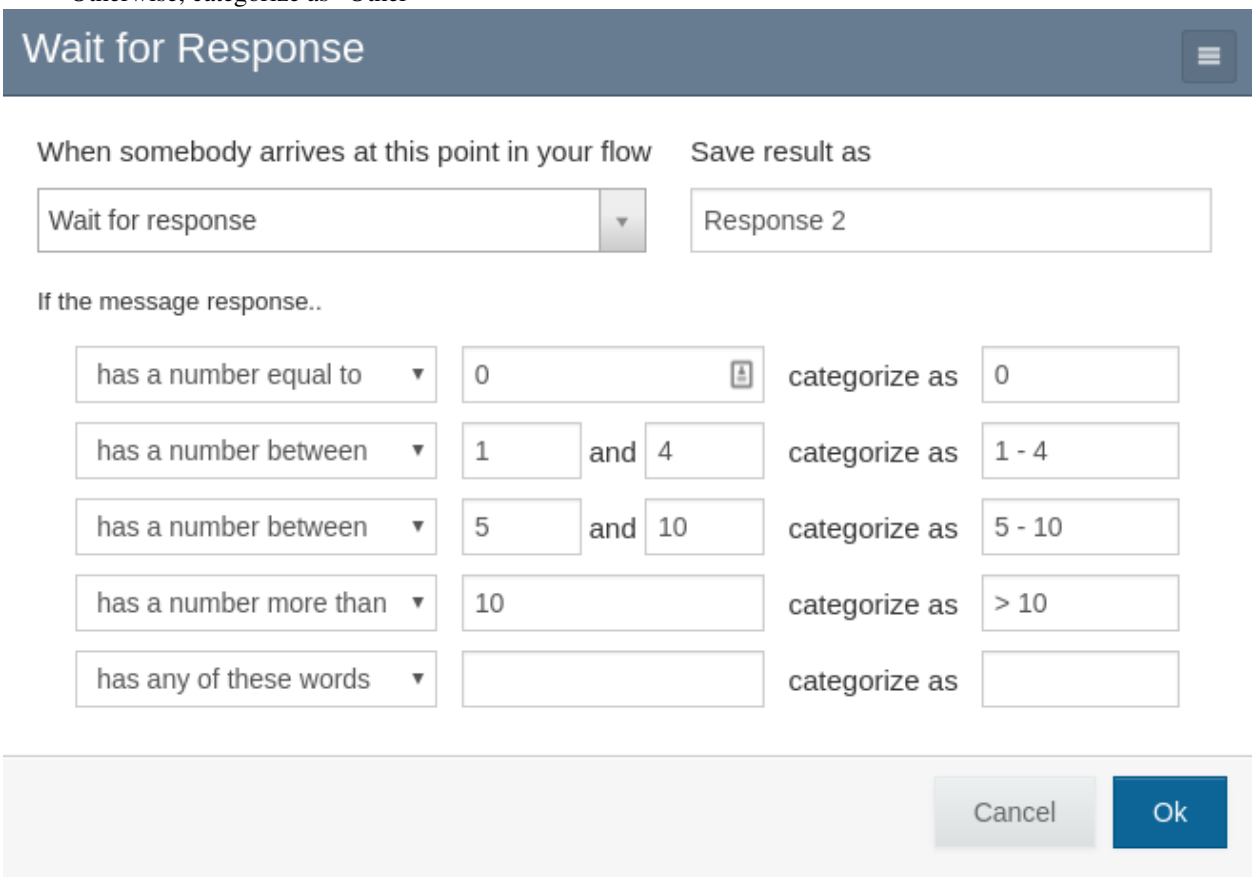
- is a number
- has a number equal to X
- has a number greater than X
- has a number less than X
- has a number between X and Y

The *between* test is inclusive of its endpoints - that is, the rule *has a number between 5 and 10* would match the inputs "5" and "10". The *greater than* and *less than* tests are not inclusive - that is, neither *has a number greater than 5* nor *has a number less than 5* would match the input "5".

When numeric input is expected, flow designers should determine the minimum and maximum valid values and ensure that all possible numeric within the range is covered. Even if you only expect integer responses, you should design rules that can categorize arbitrary values, for example the average answer value.

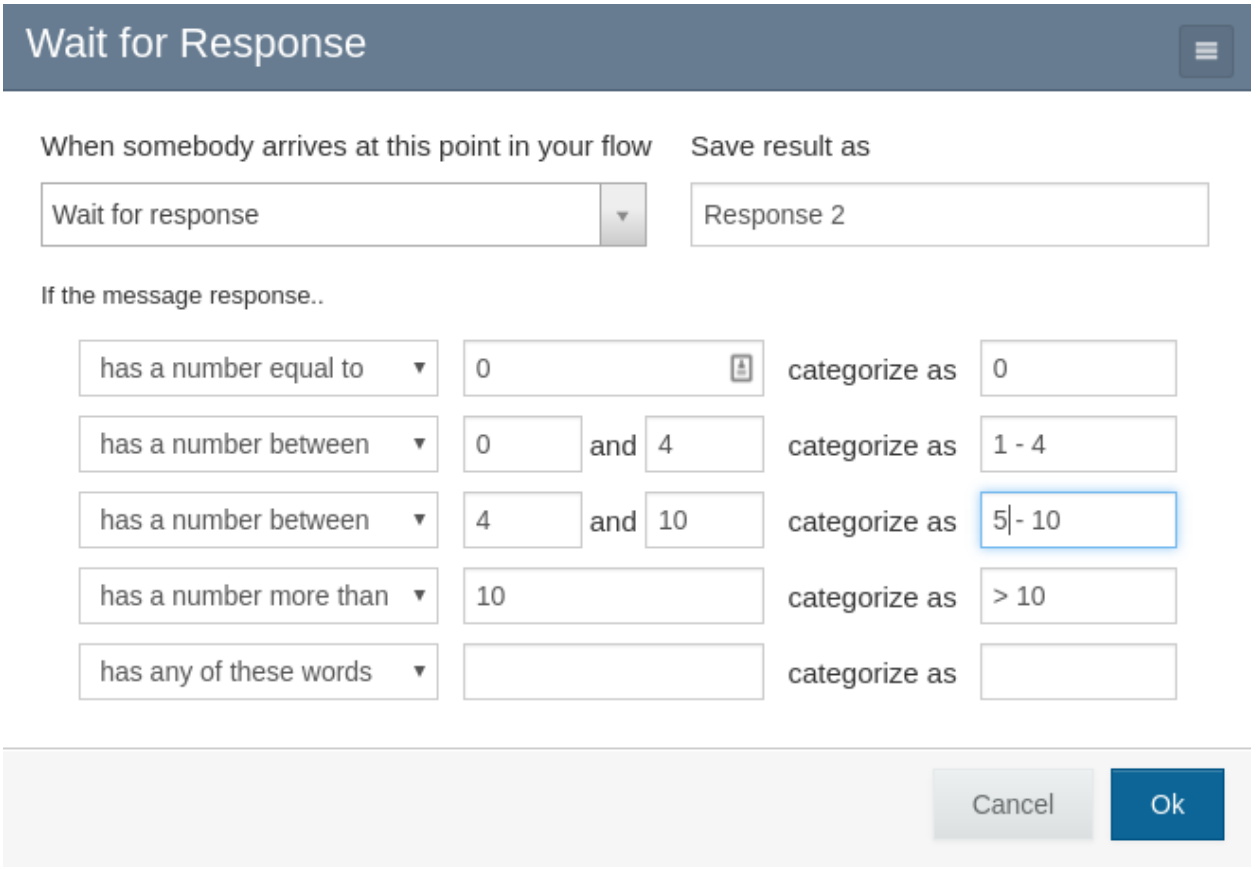
Consider this common rule set description:

- If the response is a number equal to 0, then categorize as “zero”
- If the response is a number between 1 and 4, then categorize as “1-4”
- If the response is a number between 5 and 10, then categorize as “5-10”
- If the response is a number greater than 10, categorize as “11+”
- Otherwise, categorize as “Other”




This set-up will work well when collecting data from contacts. However, if the average value calculated on TracPro is 0.5 or 4.5, then it will be categorized as “Other”. To prevent this situation while maintaining the intended meaning for integer input, adjust the categories above:

- If the response is a number equal to 0, then categorize as “zero”
- If the response is a number between 0 and 4, then categorize as “1-4”
- If the response is a number between 4 and 10, then categorize as “5-10”
- If the response is a number greater than 10, then categorize as “11+”
- Otherwise, categorize as “Other”


Wait for Response
When somebody arrives at this point in your flow
Wait for response
Save result as
Response 2
If the message response..
has a number equal to
0
categorize as
0
has a number between
0
and
4
categorize as
1 - 4
has a number between
4
and
10
categorize as
5 - 10
has a number more than
10
categorize as
> 10
has any of these words
categorize as
Cancel
Ok

CHAPTER 5

# Local Development Environment Setup

Use this guide to bootstrap your local development environment.

**NOTE:** These instructions are written with Ubuntu in mind. Some adjustments may be needed for Mac setup. To begin you should have the following applications installed on your local development system:

- Python 2.7
- pip >= 1.5
- virtualenv >= 1.10
- virtualenvwrapper >= 3.0
- Postgres >= 9.3
- git >= 1.7

1. Install the LESS CSS precompiler and the CoffeeScript JavaScript compiler using `npm`. First, install `nodejs` which comes with `npm`:

```
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:chris-lea/node.js
sudo apt-get update
sudo apt-get install nodejs
```

   Then, use `npm` to install `less` and `coffee`:

```
sudo npm install less coffee-script -g
```

   **NOTE:** You may already have `less`, `coffee` or `npm` installed. Before running installation commands, use `which [program_name]` to see if the path to the program's executable is known.

2. Clone the repo and check out the `develop` branch:

```
$ git clone git@github.com:rapidpro/tracpro.git
$ cd tracpro
$ git checkout develop
```

3. Create a virtual environment using Python 2.7 and install the project requirements:

```
# Check that you have python2.7 installed
$ which python2.7
$ mkvirtualenv tracpro -p `which python2.7`
(tracpro)$ $VIRTUAL_ENV/bin/pip install -r $PWD/requirements/dev.txt
```

4. Create a local settings file:

```
(tracpro)$ cp tracpro/settings/local.example.py tracpro/settings/local.py
```

You may edit this file to make settings changes that are local to your machine. This file is listed in the .gitignore file and should never be checked into GitHub.

5. This project uses django-dotenv to manage environment variables. Configure the environment variable Django uses to locate the project settings file:

```
(tracpro)$ echo "DJANGO_SETTINGS_MODULE=tracpro.settings.local" >> .env
```

6. Create a Postgres database and run the initial migrate:

```
(tracpro)$ createdb -E UTF-8 tracpro
(tracpro)$ python manage.py migrate
```

7. NOTE: TracPro uses Smartmin for permissions-based object scaffolding. If you make changes to a permission in GROUP_PERMISSIONS in tracpro/settings/base.py, you are required to migrate the database in order for that permission to take effect.

   (tracpro)$ python manage.py migrate

8. Background tasks. To run background tasks, you'll also need to start celery:

```
(tracpro)$ celery -A tracpro worker -B -l info
```

9. Subdomain Setup

   TracPro uses subdomains to determine which organization a user is currently accessing. For example, if you create an organization with the subdomain **testing**, you should configure that as an alias for localhost. On a UNIX-like system you would edit /etc/hosts as follows:

```
127.0.0.1   localhost. testing.localhost
```

10. RapidPro Integration

    The default development settings file connects to app.rapidpro.io.To integrate with a different RapidPro instance, either edit this file or create a new settings file.

11. Create Super User

    If creating a super user, be sure to select a valid password. TracPro enforces an 8 character minimum password.

12. Run the development server and navigate to localhost:8000:

```
python manage.py runserver
```

CHAPTER 6

# Running Tests

TracPro uses Travis CI to automatically build and test code upon each push to GitHub.

For the Travis build to pass, all tests should pass, code coverage should be greater than 75%, and no Flake8 errors should exist.

To run the Django tests for all `tracpro` apps, run this command:

```
python manage.py test
```

You can also run the tests with coverage and check the code coverage results:

```
coverage run manage.py test
coverage report
```

To see HTML output of the coverage results (which is usually easier to read), run `coverage html` after running tests with coverage, then navigate to the file `htmlcov/index.html` (relative to the project root) in your browser.

To check PEP8 and pyflakes compliance:

```
flake8
```

Errors & their locations will be output; no output indicates success.

CHAPTER 7

# Translation

This project uses the standard Django translation mechanisms.

## What goes in version control?

Message files are located in the `locale/` directory. Files for each available language are in subdirectories named according to ISO language codes.

The `django.po` file in each language directory contains all translatable messages, plus the translation if available. Django reads translated messages at runtime from the `django.mo` file, which is the compiled version of the `.po` file. We commit both files to GitHub for ease of tracking changes and making reverts if needed.

## Update translations

A developer should update message files when Python or template code with translatable messages is changed or when a new language is added. **NOTE:** To minimize noise, we generally update translation files just once per release cycle.

1. Create or update the message files:

```
python manage.py makemessages --ignore 'conf/*' \
                              --ignore 'docs/*' \
                              --ignore 'requirements/*' \
                              --no-location --no-obsolete -l en
```

2. Ensure that changes look reasonable using `git diff`. E.g., if translations have vanished, figure out why before proceeding.
3. Edit these files to add translations, if or when able.
4. Compile the message files:

```
python manage.py compilemessages
```

   If you get any errors due to badly formatted translations, work with your translators to fix the errors and start this process over.

5. Commit these changes to GitHub.

## Add a new language

1. Add the language to the `LANGUAGES` setting. `tracpro/settings/base.py`.
2. If the language is written from right to left, add the language code to `RTL_LANGUAGES`.
3. Create an empty directory, named with the language's two-letter ISO code, in the `locale/` directory.
4. Update the translation files as described above.
5. Commit these changes to GitHub.

CHAPTER 8

# Changelog

Tracpro's version is incremented upon each merge to master according to our production deployment process.

We recommend reviewing the release notes and code diffs before upgrading between versions.

## v1.5.0 (released 2017-02-27)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.5.0...develop

- Many-to-many contact to group relation update in the back-end
- Allow admins to set super user status on the front end
- Web interface to manage fetch runs

## v1.4.3 (released 2016-04-06)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.4.2...develop

- Update docs structure.
- Remove outdated docs.
- Update deployment process notes.
- Reduce log level for "Scheduled task" message.
- Allow for custom templates and staticfiles directories.

## v1.4.2 (released 2016-04-04)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.4.1...v1.4.2

- Reduce log levels

## v1.4.1 (released 2016-03-29)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.4.0...v1.4.1

- Fix placement of lock release

## v1.4.0 (released 2016-03-28)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.3.3...v1.4.0

- Migrations to move RapidPro uuid unique constraint to unique_together with another model field (*org* for Contact, Region, and Group models; *pollrun* for Response)

## v1.3.3 (released 2016-03-28)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.3.2...v1.3.3

- Implement "backoff" for OrgTasks that fail
- Ensure that cache key timeout is set properly in OrgTask
- Do not use @task decorator on class-based task

## v1.3.2 (released 2016-03-23)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.3.1...v1.3.2

- Add debug logging for OrgTask

## v1.3.1 (released 2016-03-23)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.3.0...v1.3.1

- Fix formatting errors in this changelog
- Return *None* if *SoftTimeLimitExceeded* is raised during *OrgTask*
- Run *pipconflictchecker* on Travis builds
- Fail before running Travis tests if there are missing migrations
- Increase the *hard_time_limit* value for Org tasks

## v1.3.0 (released 2016-03-22)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.2.1...v1.3.0

**Infrastructure**

- Update to Django==1.8.11
- Update versions on many third-party packages (excluding forks)
- Updated the Caktus smartmin fork
- Serve library scripts and stylesheets from */static/libs/* rather than CDNs
- Ensure all test classes inherit from *TracProTest*, which ensures that critical features are mocked

**Features & Bugfixes**

- Fix email prefix on deployed environments
- Add *from __future__ import unicode_literals* to all files
- Only show responses from active contacts on charts for baseline, poll detail, and pollrun detail
- Don't abbreviate big numbers on charts (1,000,000 rather than 1M)
- Add user documentation about designing flows
- **Add *Boundary* model to *tracpro.groups***
  - Sync with RapidPro
  - Add endpoint to retrieve all boundaries for an Org
- Add *boundary* foreign key to *Region* and allow setting the *boundary* on the Region list page
- Add contact data field filters to PollRun detail page & pass applicable filters to PollRun detail page when clicking on a data point on the Poll detail page.
- Store ruleset on the *Question* model
- Add ability to categorize arbitrary (numeric) values
- Display results on a map

## v1.2.1 (released 2016-03-21)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.2.0...v1.2.1

- Fix EMAIL_HANDLER
- Add *django* logger
- Prevent Celery from hijacking the root logger

## v1.2.0 (released 2016-03-14)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.1.1...v1.2.0

- **Settings changes:**
  - Update *LOGGING* to reflect sending logs to *syslog*
  - Utility for grabbing settings from the environment
  - Utility for falling back to Django default settings
  - Email configuration
  - Remove unused *HOSTNAME* setting
  - Misc. settings tweaks related to deployment.

## v1.1.1 (released 2016-03-01)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.1.0...v1.1.1

- Updated to Django==1.8.10 from Django==1.8.6
- Send Celery task error emails.
- Limit InboxMessages fetch to the past 7 days.
- Use relativedelta where possible.
- Update 404 page template.

## v1.1.0 (released 2016-02-24)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.0.4...v1.1.0

Many changes, including:

- Break out deployment-related assets into a private repo.
- **Update Celery task structure.**
  - **Note:** Existing tasks are probably very backed up. After deploy, purge all existing tasks (see Celery FAQ).
- Chart enhancements on Poll detail and PollRun detail pages.
- Filters on Recent Indicators, Poll detail, and PollRun detail pages.

## v1.0.4 (never released)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.0.3...v1.0.4

- Update versions of Celery-related packages.

## v1.0.3 (released 2015-11-30)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.0.2...v1.0.3

- Bug fix for clearing spoof data. See #100.
- Release notes added for ReadTheDocs builds

## v1.0.2 (released 2015-11-25)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.0.1...v1.0.2

- Don't paginate results on responses CSV export.
- Show participant count in participant column on PollRun ByPoll page.

## v1.0.1 (released 2015-11-25)

Code diff: https://github.com/rapidpro/tracpro/compare/v1.0.0...v1.0.1

- Updated contact sync to run every 30 minutes, rather than every 5.

## v1.0.0 (released 2015-11-19)

Code diff: https://github.com/rapidpro/tracpro/compare/v0.0.51...v1.0.0

- Add documentation to ReadTheDocs.
- Upgrade version requirements.
  - **Note:** Due to a change in structure for *django-celery*, you will need to run *python manage.py migrate djcelery –fake-initial* before running new migrations.
- Add *prod_db_to_staging* Fabric command.
- Fix *hostname* in *manage_run* Fabric command so that it now runs without error.
- Require that source is updated before updating pip requirements during deploy.
  - **Note:** Pip requirements were sometimes being updated before the source code was updated. If you have this issue before updating to v1.0.0, run the deploy again to solve.
- Add deadsnakes Python 2.7 to deploy environment.
  - **Note:** An SSL dependency requires Python 2.7.9 or greater. If your deployment is using a lower version, destroy the virtual environment before your next deploy so that it is rebuilt.
- Add org config option to show/hide spoof data. See #92.
  - **Note:** A migration sets the default to False for all orgs except "Caktus".
- Fix unicode bug when setting a Contact DataField value. See #88.
- Use `django.utils.dateparser` rather than `dateutil` when parsing datetimes for DataFields. See #88.
- Fix org languages bugs. See #91.

CHAPTER 9

# Deployment Process

Deployment and provisioning are currently managed in a separate repo. Here we discuss the basic outline of our basic development and deployment practices.

## Production deployment

1. **Before merging changes to the master branch,** the deployment process must be tested from end to end on staging.
    - Restore current production data, media, and code (the current `master` branch) to the staging machine.
    - Deploy the release branch to staging.
    - Confirm that changes are behaving as expected, and troubleshoot the deploy process until you are confident.
2. Update the release notes (including notes about one-off deployment requirements, if needed) and add the current date as the release date. Commit these changes and push to GitHub.
3. Finalize the version number according to semantic versioning, and change the version state to `"final"`. Version must be updated in `tracpro/__init__.py` and `docs/conf.py`.
    - The micro version is incremented with backwards-compatible bugfixes.
    - The minor version is incremented with backwards-compatible features.
    - The major version is incremented with incompatible changes.

    Commit these changes and push to GitHub.
4. Merge all changes to the `master` branch, ensure that these changes are pushed to GitHub, and confirm that the Travis build has passed.
5. Check out the updated `master` branch locally, and create a release tag:

```
git tag -a vX.Y.Z -m "Released YYYY-MM-DD" && git push origin --tags
```

6. Copy the release notes to the GitHub releases interface.
7. Deploy the master branch to production.
8. Merge the `master` branch into `develop`:

```
git checkout develop
git merge origin/master
git push origin develop
```

9. On the `develop` branch, increment the micro version and change the code state to `"dev"`. Commit these changes and push to GitHub.
10. Run ReadTheDocs builds for the new release & the latest develop.