Chatbot Service

Introduction

Whatsapp_PGR chatbot service is a chatbot which provides functionality to the user to access PGR module services like file complaint, track complaint, notifications from whatsapp. Currently citizen has three options to start conversation scan QR code, give missed call or directly send message to configured whatsapp number.

Requirements

- Prior Knowledge of Java/J2EE
- Prior Knowledge of SpringBoot
- Prior Knowledge of REST APIs and related concepts like path parameters, headers, JSON etc.
- Prior Knowledge of PostgresSQL
- Prior Knowledge of Kafka and related concepts like Producer, Consumer, Topic, Kafka Streams etc

Features:

- File PGR complaint
- Track PGR complaint
- Support images when filing complaints
- Notifications to citizen when employee performs any action on complaint
- Put user interactions on elastic search for Telemetry

New Whatsapp Provider Integrations

Whatsapp provider is a third party service which works in the middle of a user's whatsapp client and chatbot server. All messages coming/going to/from user pass through whatsapp provider. Chatbot calls whatsapp provider to send messages to user. When a user responds with any whatsapp message the whatsapp provider calls Chatbot service's configured endpoint with details ex:- user sent message, sender's number etc. If any new whatsapp provider is to be

used with chatbot, code must be written to convert provider's incoming messages to the format that chatbot understands and also final output from chatbot should be converted to whatsapp provider's API request format. We discuss these next.

Currently as whatsapp provider we are using ValueFirst. The documentation about their APIs can be found from below files

ValueFirst_API Format - Incoming WA 1.1.pdf
797 KB
ValueFirst_Whatsapp_Doc_Version_1.0.3 (2).pdf
1 MB

Input to Chabot

Example - Refer following class in chatbot "org.egov.chat.xternal.responseformatter.ValueFirst.ValueFirstResponseFormatter"
Chatbot input JSON Format:
{"user":{"mobileNumber":"xxxxxxxxx","userId":"uuid string"},
"message":{"contentType":"text","rawInput":"mseva"},"extraInfo":{"recipient":"91874496011},"timestamp":1584172688961}
Explanation for fields:

- user:- stores information about sender of message
- mobileNumber:- Mobile number of sender
    - userId: user uuid of sender (from user service)
- Message: represents user sent message details
    - contentType:- type of message ex:- text, image
    - rawInput:- the actual message body sent by the sender
- extraInfo
    - Object to keep extra information that may be used while sending response to user in last stage, ex - source mob number. All the details which is not mentioned should be kept here. Chatbot will ignore if any new value stored here
    - Timestamp:- timestamp when message was received

Send chatbot output message to user

Example: Refer following class in chatbot service

“org.egov.chat.xternal.responseformatter.ValueFirst.ValueFirstResponseFormatter”

The object that we sent to the chatbot as input is filled with many details which are for internal working of chatbot. Out of these fields the fields which are useful while calling whatsapp provider’s send API and their paths in chatbot output are as follows -

- FieldInfo - Mobile number of sender (who sent message to chatbot)
  Field Path - /user/mobileNumber
- FieldInfo - Response message from chatbot to user
  Field Path - /response/text
- FieldInfo - Chatbot response type
  Field Path - /response/type

Chatbot Configurations

There are two types of configurations for chatbot states:-

- Configuration for each state in chatbot
- Graph adjacency list configuration:- to define flow between chatbot states

Configuration for each state

- name: name of state,ex:- pgr.create.tenantId
- description : description about state
- nodeType : step or branch.
  - Step:- only one possible next state
  - Branch:- more than one possible next state
- message: localisation code of the message which will be asked when user enters this state
- Nodes: define all other chat states from which user sent input is required to be read in current state
- validationRequired : whether user provided value to be validated

- Values: -List of options to be shown to the user in numerics in case of branch node type. The config must have entry with option text -name of state. Based on the user provided value next state will be decided.

Ex:- “values : ["File a Complaint", "Track Complaint Status"]

File a Complaint : pgr.create.tenantId

Track Complaint Status : pgr.track.end”

- typeOfValues : optional field
    - FixedSetValues:- The options in the question would be in numbers, and only numeric input corresponding to shown options is allowed
- displayValuesAsOptions: optional, true in case of FixedSet values
- numberPrefixLocalizationCode: When user is asked ordered options in numerics

Localisation code for prefix to numbers ex:- **Type** 1 for complaint

Type 2 for tracking

- numberPostfixLocalizationCode: When user is asked ordered options in numerics

Localisation code for postfix to numbers ex:- Type 1 **for** complaint

Type 2 **for** tracking

- values :
    - batchSize : Maximum number of ordered options shown to the user at a time to choose from in case of Fixed values
    - class : Fetcher class to fetch options list to the user ex:- tenantValueFetcher, localityValueFetcher etc. All these classes are defined in “org.egov.chat.xternal.valuefetch” package.
    - params : values to be picked from chatbot object while processing current state ex:- userInfo, user’s mobile number
- matchAnswerThreshold: fuzz search matching parameter for validations of user sent messages ,ex:- 80

- errorMessage: localisation code of error message which would be shown to the user if he provides invalid input

Graph adjacency list configuration

This configuration defines flow between different chatbot states. For each state it is defined what could be the possible set of next states. The first state is always “root”.

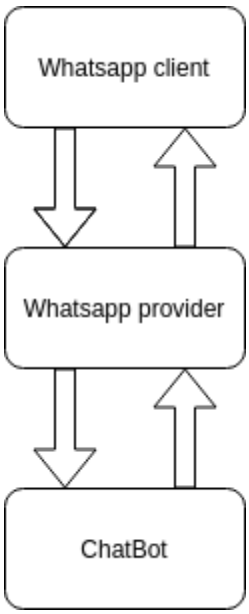
All the states defined in adjacency list configurations should be defined as per the configurations defined in last section. All the entries in this config fields should be present in name field of states. The structure of adjacency list configuration is as follows:-

Current state, next state1, next state2 ……

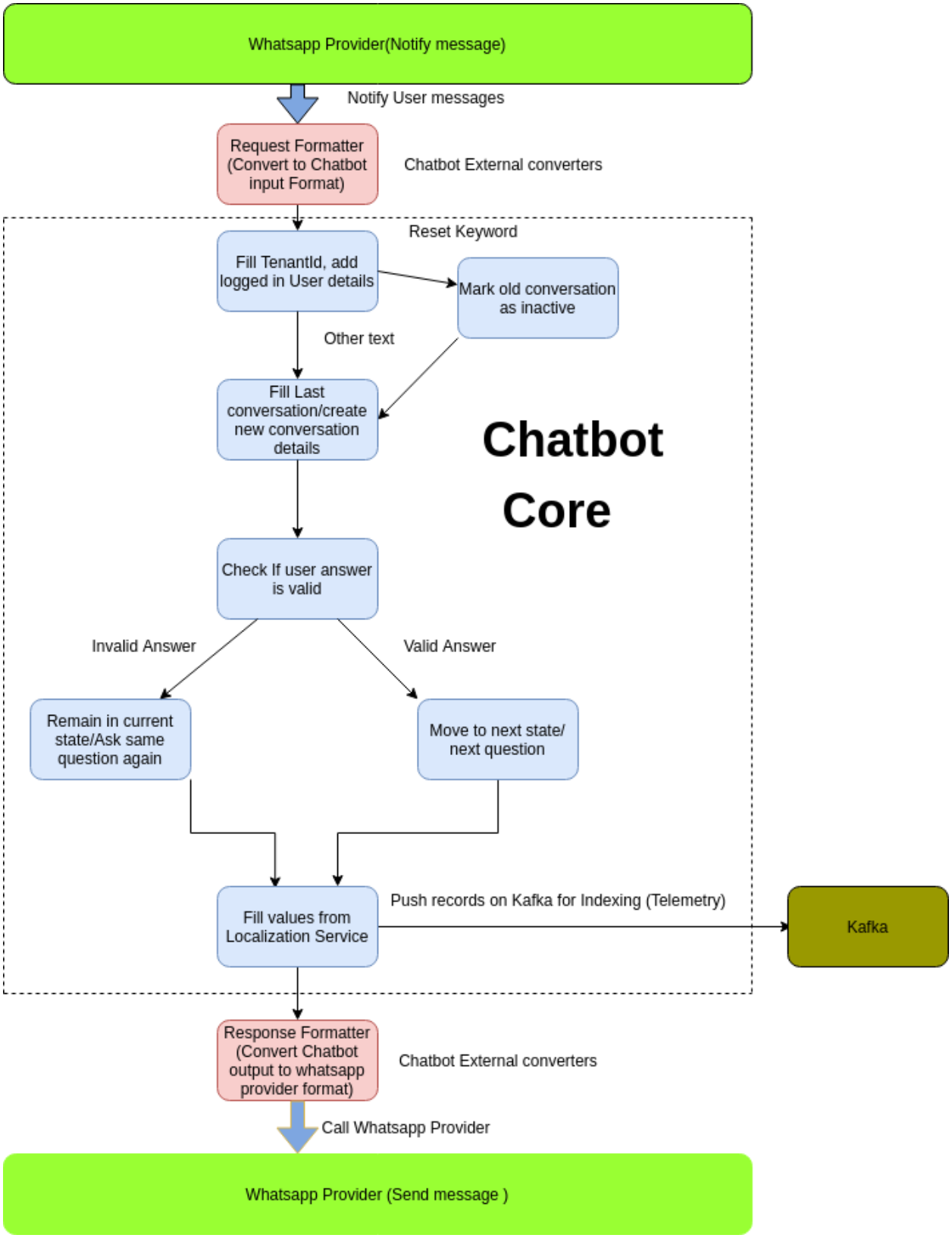
ex:-root,pgr.create.tenantId,pgr.track.end

Flow Diagram

High Level Diagram of chatbot interactions



Flow Diagram of Chatbot-User conversation


Whatsapp Provider(Notify message)
Notify User messages
Request Formatter (Convert to Chatbot input Format)
Chatbot External converters
Reset Keyword
Fill TenantId, add logged in User details
Mark old conversation as inactive
Other text
Fill Last conversation/create new conversation details
Chatbot Core
Check If user answer is valid
Invalid Answer
Valid Answer
Remain in current state/Ask same question again
Move to next state/ next question
Fill values from Localization Service
Push records on Kafka for Indexing (Telemetry)
Kafka
Response Formatter (Convert Chatbot output to whatsapp provider format)
Chatbot External converters
Call Whatsapp Provider
Whatsapp Provider (Send message)

Flow Diagram of Chatbot notifications