ESI - Flow(Failure) detection Tool

[**Introduction**](#_tszo41ewnkvx) **2**

[Goal](#_27jrwlyudvua) **2**

[**Solution**](#_5gtst1nzygbk) **2**

[Transition-Meta](#_e6fi5sxtjaha) 3

[Rule-Meta](#_28bvq7e5q2yi) 3

[Labels](#_bz6vof7ao87) 3

[Rules](#_jxscyp1dk2x1) 3

[Runtime-Input](#_3h2l4yjniihn) 3

[Sample Rule-Meta](#_sxr6ph3ouvi7) 4

[Sample Report](#_ninby04w6otg) 5

[Approach](#_sqtarnbr8h0t) 5

[Maintenance](#_lonxnn5yg61j) **6**

[**Limitations**](#_8zpzfs736cjg) **6**

[Other considerations](#_wo62mts438i9) **6**

# 

## Introduction

There are some cases identified where some of the calls are being transferred to agents or dropped due to some or other failure scenarios. Few other possibilities are they may be taking the user to a different flow instead of the actual flow and most of the time these incidents are being identified by the business team after listening to a few call recordings and it has become a tedious process if the volume of calls is very high and at times it is taking a lot of manual efforts and time to identify that there is a problem in the flow. Quantifying the number of failures is also a challenging metric to fetch.

## Goal

The idea is to identify the failed calls without any manual intervention of listening and quantify them. For this, we are planning to build a tool that can inspect the flow transitions and detect the failed calls based on predefined criteria. The criteria would be a set of rules that would be defined by the Business /Development Teams based on the flow transitions of the Bot. Upon successful execution of the tool, a summary with the following details will be generated.

* Total number of calls inspected
* Number of calls matching the rule criteria - Successful calls
* Number of calls that don’t match the rule criteria - Failed calls along with the details,
  + The flow - List of nodes traversed
  + Failed rule(s) details

The tool can either be run on-demand or scheduled to run periodically(every hour) with the help of a crontab.

## Solution



### 

### Transition-Meta

The transition data is stored in the database which is available in “transitionrecords” collection for a limited period of time.

### Rule-Meta

A file containing a set of rules in the pre-defined structure. It contains two sections.

* labels
* rules

#### Labels

Each conversation can be tagged with one or more labels based on the list of nodes traversed. A set of named labels can be defined by defining label criteria. The label criteria have two parts: qualification and fulfillment. Qualification is used as a pre-condition and Fulfillment is used to evaluate if a label can be applied to the qualified conversation. Each of these can be defined by mentioning ***all-of-nodes***(the list-of-nodes that must have been traversed) and ***none-of-nodes*** (the list-of-nodes that none of them must have traversed)

#### Rules

Once conversations are marked with labels, these labels can be used to define a set of rules. Rule criteria have two parts: qualification and fulfillment. Qualification is used as a pre-condition and Fulfillment is used to mark a qualified conversation as fulfilled/unfulfilled. Each of these can be defined by mentioning ***all-of-labels***(the list-of-labels that must have tagged for a conversation) and ***none-of-labels*** (the list-of-labels that none of them must have been tagged for the conversation)

### Runtime-Input

The tool accepts time range as input (startTime, endTime). In case the time range is not given, it takes the last hour as the time range and does the inspection.

#### 

### Sample Rule-Meta

{

"\_v": "1.0",

"labels":{

"label-1": {

"desc": "some description for reporting purpose",

"qualififcation":{

"all-of-nodes":["n1", "n2"],

"none-of-nodes":["n3"]

},

"fulfillment": {

"all-of-nodes":["n5", "n6"],

"none-of-nodes":["n7", "n8", "n9"]

}

},

"label-2": {

"desc": "some description for reporting purpose",

"qualififcation":{

"all-of-nodes":["n1"],

"none-of-nodes":[]

},

"fulfillment": {

"all-of-nodes":["n5"],

"none-of-nodes":["n10", "n20"]

}

}

},

"rules":{

"rule-1": {

"desc": "some description for reporting purpose",

"qualififcation":{

"all-of-labels":["label-1"],

"none-of-labels":[]

},

"fulfillment": {

"all-of-labels":[],

"none-of-labels":["lable-2"]

}

},

"rule-2": {

"desc": "some description for reporting purpose",

"qualififcation":{

"all-of-labels":[],

"none-of-labels":[]

},

"fulfillment": {

"all-of-labels":[],

"none-of-labels":[]

}

}

}

}

### Sample Report

* Excel/CSV with call/conversation meta. Each entry must have
  + userId, identity
  + Nodes traversed ( in the order )
  + Labels applicable based on label criteria defined in the Rule-Meta
* Rule processing summary. For each rule:
  + Count of calls/conversations inspected
  + Count of qualified
  + Count of fulfilled
  + Count of unfulfilled
    - Details of the unfulfillment
* Overall Summary (Across all the rules)
  + The total count of conversations inspected
  + The total count of conversations qualified
  + The total count of conversations fulfilled
  + The total count of conversations unfulfilled

### Approach



* Based on the Runtime-Input(time range), query the database and identify the list of unique calls
* Fetch the transition data for each call
* **Node processor:** Identify the list of nodes traversed in the order execution
* **Label processor:** Loop through the list of labels defined in the Rule-Meta, evaluate the criteria against each call transition-data.
  + The output is made available in an excel/CSV file, which will enable the customer to derive any custom metrics apart from success failure criteria.
* **Rule processor:** Process rules based on the labels applicable for each call. It can also take previously generated label summaries and apply rules against them.
  + Capture the results for each call inspection
* Generate the summary
* Trigger an email notification with the details to the list of pre-configured emails

## Maintenance

The criteria being used must be in line with the regular changes being done on the bot development. This means, if there is a change in the flow or in the node names, the criteria must be changed accordingly. Otherwise, it may either trigger false alarms or may not detect the failures.

## Limitations

* The order of the node traversal can’t be defined as a criterion for now. Can be supported future?
* Complex logical operation with the combination of AND/OR logic is not supported
* There may be calls that are initiated in the given time range but are yet to complete. Need to find out a way to ignore such calls during the inspection.

## Other considerations

* The tool must use the secondary database as the data source
* If the time range given as runtime input is huge, it may have an impact on the database. Should we limit the max time range that can be given as runtime input?