25/8/2021

Log Analyzer Project

1. Summary:

The Log Analyzer projects goal is to facilitate Automated Testing of communication devices infrastructure. It processes the log text files using a predefined sequence flows.

1. Input:
   1. YAML file: The sequence flows are being defined in a YAML file, for example:

global\_configuration:

start\_message:

message\_time\_stamp: '15/07/2021 15:48:05:224'

end\_message:

message\_time\_stamp: '15/07/2021 15:48:06:0'

fail\_messages:

fail\_log\_message\_g\_1:

message\_text: 'example global fail message'

sequence\_flows:

sequence\_flow\_1:

flow\_configuration:

enabled: 'True'

fail\_messages:

fail\_log\_message\_f\_1:

message\_text: 'example flow fail message'

sequence\_1:

log\_messages:

log\_message\_1.1:

message\_text: 'CP <-- DU [F1AP]: INITIAL UL RRC MESSAGE TRANSFER'

sequence\_2:

log\_messages:

log\_message\_2.1:

message\_text: 'CP --> DU [F1AP]: DL RRC MESSAGE TRANSFER'

log\_message\_2.2:

message\_text: 'CP <-- DU [F1AP]: UL RRC MESSAGE TRANSFER'

sequence\_3:

log\_messages:

log\_message\_3.1:

message\_text: 'CP --> AMF [NGAP]: INITIAL UE MESSAGE'

sequence\_4:

log\_messages:

log\_message\_4.1:

message\_text: 'CP <-- AMF [NGAP]: DL NAS TRANSPORT'

log\_message\_4.2:

message\_text: 'CP --> AMF [NGAP]: UL NAS TRANSPORT'

actions:

action\_4.2.1:

action\_name: 'verify\_repeat\_num'

action\_config:

repeat\_num: 6

fail\_messages:

fail\_log\_message\_1.1:

message\_text: 'example message fail message'

sequence\_5:

log\_messages:

log\_message\_5.1:

message\_text: 'CP <-- DU [F1AP]: CONFIGURATION UPDATE'

actions:

action\_5.1.1:

action\_name: 'verify\_message\_content'

action\_config:

nRCellIdentity: '0000000010'

#hex\_str:'00030080c1000003004e00020002003e00809f0000003d0080984000f19500000000140000f1950000000010000b0000010800f19500000083400400000020410009c12000004d002e000100000379cd2c606c8300a340410012c800000800000000c0280c0150e116219a421340600000c418d6d8d7f4a6e080400001010000800d1808509098c0840000621ab8d6d8da9280321c19aeb285200498e129c35388a7154cdc000263a010580e0020729180200059001000010058000a0000f195000000001000'

sequence\_6:

log\_messages:

log\_message\_5.1:

message\_text: 'CP <-- AMF [NGAP]: INITIAL CONTEXT SETUP REQUEST'

sequence\_7:

log\_messages:

log\_message\_6.1:

message\_text: 'CP --> AMF [NGAP]: UE INITIAL CONTEXT SETUP RESPONSE'

sequence\_8:

log\_messages:

log\_message\_7.1:

message\_text: 'CP --> UP [E1AP]: BEARER CONTEXT SETUP REQUEST'

Note that if more than one log message appears under a sequence each one of them is optional and will step forward the validation to next sequence.

For each flow these is an enable/disable flag at the flow configuration.

The messages can appear with [“POSIX Basic Regular Expressions standard”](https://www.gnu.org/software/sed/manual/html_node/Regular-Expressions.html)

* 1. Log file: The log file can be received with ANSI Control Code (As occur in all setups), these would be cleaned automatically without affecting log content.

1. Features:
   1. Truncate: The log file can be truncate by timestamp.

the following would appear in the global configuration:

start\_message:

message\_time\_stamp: '%d/%m/%Y %H:%M:%S:%f'

end\_message:

message\_time\_stamp: '%d/%m/%Y %H:%M:%S:%f'

Both are optional.

This feature can also be invoked from by calling the preprocessing function with truncate the arguments:

preprocess\_function\_name='preprocess\_with\_truncate', preprocess\_args=preprocess\_args

Should be sent to the analyze\_log function while the preprocess\_args is a dictionary:

preprocess\_args = {'start\_message': {'message\_time\_stamp': '15/07/2021 15:38:24:941'}}

* 1. Fail Messages: A fail message is a message that when received in the specific context fails the test.

A fail message can be defined in three scopes:

* + 1. A global fail message – holds for all sequence flows.
    2. A flow fail message – holds for the specific sequence flow.
    3. A message fail message – holds for specific message for messages that follows it.
  1. Actions:

There could be defined actions at the specific message scope.

The main validation loop contains a validation state with two lists, the log message list and the YAML sequence list to be validated.

The loop step forward on these lists based on various conditions.

* + 1. Actions can be performed at four positions of the validation loop (this is predefined for each action type and is technical knowledge not relevant to the definition of the action of the YAML but is provided here for reference):
       1. pre\_log\_forward\_step - before the log forward step in the validation loop.
       2. post\_log\_forward\_step - after the log forward step in the validation loop.
       3. pre\_yaml\_forward\_step - before the yaml forward step in the validation loop.
       4. post\_yaml\_forward\_step - after the yaml forward step in the validation loop.
    2. The current supported actions:
       1. get\_message\_stats:

Return the following statistics on the message:

* + - * 1. Message appearance count.
        2. List of dictionary message appearances.
        3. The time elapsed from the previous specific message.
        4. The average time between the specific message.

Example:

sequence\_1:

log\_messages:

log\_message\_1.1:

message\_text: 'CP <-- AMF [NGAP]: INITIAL CONTEXT SETUP REQUEST'

actions:

action\_1.1.1:

action\_name: 'get\_message\_stats'

* + - 1. verify\_message\_content (This feature depends on a software in Trial Mode, which is currently under negotiations):

Extract the immediate following hex string convert it to json and verify that the parameters specified under “action\_config” has the exact value in all of the appeared positions in the json.

Example:

sequence\_5:

log\_messages:

log\_message\_5.1:

message\_text: 'CP <-- DU [F1AP]: CONFIGURATION UPDATE'

actions:

action\_5.1.1:

action\_name: 'verify\_message\_content'

action\_config:

nRCellIdentity: '0000000010'

This action will take the following message hex string:

'00030080c1000003004e00020002003e00809f0000003d0080984000f19500000000140000f1950000000010000b0000010800f19500000083400400000020410009c12000004d002e000100000379cd2c606c8300a340410012c800000800000000c0280c0150e116219a421340600000c418d6d8d7f4a6e080400001010000800d1808509098c0840000621ab8d6d8da9280321c19aeb285200498e129c35388a7154cdc000263a010580e0020729180200059001000010058000a0000f195000000001000'

Parse it into json:

<F1AP-PDU>

<initiatingMessage>

<procedureCode>3</procedureCode>

<criticality>

<reject/>

</criticality>

<value>

<GNBDUConfigurationUpdate>

<protocolIEs>

<SEQUENCE>

<id>78</id>

<criticality>

<reject/>

</criticality>

<value>

<TransactionID>2</TransactionID>

</value>

</SEQUENCE>

<SEQUENCE>

<id>62</id>

<criticality>

<reject/>

</criticality>

<value>

<Served-Cells-To-Modify-List>

<SEQUENCE>

<id>61</id>

<criticality>

<reject/>

</criticality>

<value>

<Served-Cells-To-Modify-Item>

<oldNRCGI>

<pLMN-Identity>00F195</pLMN-Identity>

<nRCellIdentity>000000000000000000000000000000000001</nRCellIdentity>

</oldNRCGI>

<served-Cell-Information>

<nRCGI>

<pLMN-Identity>00F195</pLMN-Identity>

<nRCellIdentity>000000000000000000000000000000000001</nRCellIdentity>

</nRCGI>

<nRPCI>11</nRPCI>

<fiveGS-TAC>000001</fiveGS-TAC>

<servedPLMNs>

<ServedPLMNs-Item>

<pLMN-Identity>00F195</pLMN-Identity>

<iE-Extensions>

<SEQUENCE>

<id>131</id>

<criticality>

<ignore/>

</criticality>

<extensionValue>

<SliceSupportList>

<SliceSupportItem>

<sNSSAI>

<sST>01</sST>

</sNSSAI>

</SliceSupportItem>

</SliceSupportList>

</extensionValue>

</SEQUENCE>

</iE-Extensions>

</ServedPLMNs-Item>

</servedPLMNs>

<nR-Mode-Info>

<tDD>

<nRFreqInfo>

<nRARFCN>639264</nRARFCN>

<freqBandListNr>

<FreqBandNrItem>

<freqBandIndicatorNr>78</freqBandIndicatorNr>

<supportedSULBandList/>

</FreqBandNrItem>

</freqBandListNr>

</nRFreqInfo>

<transmission-Bandwidth>

<nRSCS>

<scs30/>

</nRSCS>

<nRNRB>

<nrb273/>

</nRNRB>

</transmission-Bandwidth>

</tDD>

</nR-Mode-Info>

<measurementTimingConfiguration>00</measurementTimingConfiguration>

</served-Cell-Information>

<gNB-DU-System-Information>

<mIB-message>79CD2C</mIB-message>

<sIB1-message>6C8300A340410012C800000800000000C0280C0150E116219A421340600000C418D6D8D7F4A6E080400001010000800D1808509098C0840000621AB8D6D8DA9280321C19AEB285200498E129C35388A7154CDC000263A010580E002072918020</sIB1-message>

</gNB-DU-System-Information>

</Served-Cells-To-Modify-Item>

</value>

</SEQUENCE>

</Served-Cells-To-Modify-List>

</value>

</SEQUENCE>

<SEQUENCE>

<id>89</id>

<criticality>

<reject/>

</criticality>

<value>

<Cells-Status-List>

<SEQUENCE>

<id>88</id>

<criticality>

<reject/>

</criticality>

<value>

<Cells-Status-Item>

<nRCGI>

<pLMN-Identity>00F195</pLMN-Identity>

<nRCellIdentity>000000000000000000000000000000000001</nRCellIdentity>

</nRCGI>

<service-status>

<service-state>

<in-service/>

</service-state>

</service-status>

</Cells-Status-Item>

</value>

</SEQUENCE>

</Cells-Status-List>

</value>

</SEQUENCE>

</protocolIEs>

</GNBDUConfigurationUpdate>

</value>

</initiatingMessage>

</F1AP-PDU>

And verify in all appearances of nRCellIdentity that its value is '0000000010'.

In case it is not, the test will return FAIL with explanation of the failing reason.

* + - 1. verify\_max\_repeat:

Verify that the specific message does not repeat more than max\_repeat number of times.

Example:

sequence\_4:

log\_message\_4.2:

message\_text: 'CP --> AMF [NGAP]: UL NAS TRANSPORT'

actions:

action\_4.2.1:

action\_name: 'verify\_max\_repeat'

action\_config:

max\_repeat: 6

* + - 1. verify\_repeat:

Verify that the specific message repeats exactly repeat\_num number of times.

Example:

sequence\_4:

log\_message\_4.2:

message\_text: 'CP --> AMF [NGAP]: UL NAS TRANSPORT'

actions:

action\_4.2.1:

action\_name: 'verify\_repeat'

action\_config:

repeat\_num: 6

* + - 1. verify\_min\_repeat:

Verify that the specific message does not repeat less than min\_repeat number of times.

Example:

sequence\_4:

log\_message\_4.2:

message\_text: 'CP --> AMF [NGAP]: UL NAS TRANSPORT'

actions:

action\_4.2.1:

action\_name: 'verify\_min\_repeat'

action\_config:

min\_repeat: 6

* + - 1. log\_message\_before:

Gets the immediate previous message.

Example:

sequence\_4:

log\_message\_4.2:

message\_text: 'CP --> AMF [NGAP]: UL NAS TRANSPORT'

actions:

action\_4.2.1:

action\_name: 'log\_message\_before'

* + - 1. log\_message\_after:

Gets the immediate following message.

Example:

sequence\_4:

log\_message\_4.2:

message\_text: 'CP --> AMF [NGAP]: UL NAS TRANSPORT'

actions:

action\_4.2.1:

action\_name: 'log\_message\_after'

* + 1. Actions to be implemented:
       1. sequence\_before

Gets the immediate previous sequence.

Example:

sequence\_4:

log\_message\_4.2:

message\_text: 'CP --> AMF [NGAP]: UL NAS TRANSPORT'

actions:

action\_4.2.1:

action\_name: ' sequence\_before'

* + - 1. sequence\_after

Gets the immediate following sequence.

Example:

sequence\_4:

log\_message\_4.2:

message\_text: 'CP --> AMF [NGAP]: UL NAS TRANSPORT'

actions:

action\_4.2.1:

action\_name: ' sequence\_after'

* + - 1. find\_messages\_time\_delta\_first\_to\_first:

Finds the time delta that passed between the first appearance of the first message to the first appearance of the second message.

* + - 1. find\_messages\_time\_delta\_first\_to\_last:

Finds the time delta that passed between the first appearance of the first message to the last appearance of the second message.

* + - 1. find\_messages\_time\_delta\_last\_to\_first:

Finds the time delta that passed between the last appearance of the first message to the first appearance of the second message.

* + - 1. find\_messages\_time\_delta\_last\_to\_last:

Finds the time delta that passed between the last appearance of the last message to the first appearance of the second message.

* + - 1. find\_sequences\_time\_delta\_first\_to\_first:

Finds the time delta that passed between the first appearance of the first sequences to the first appearance of the second sequences.

* + - 1. find\_ sequences\_time\_delta\_first\_to\_last:

Finds the time delta that passed between the first appearance of the first sequences to the last appearance of the second sequences

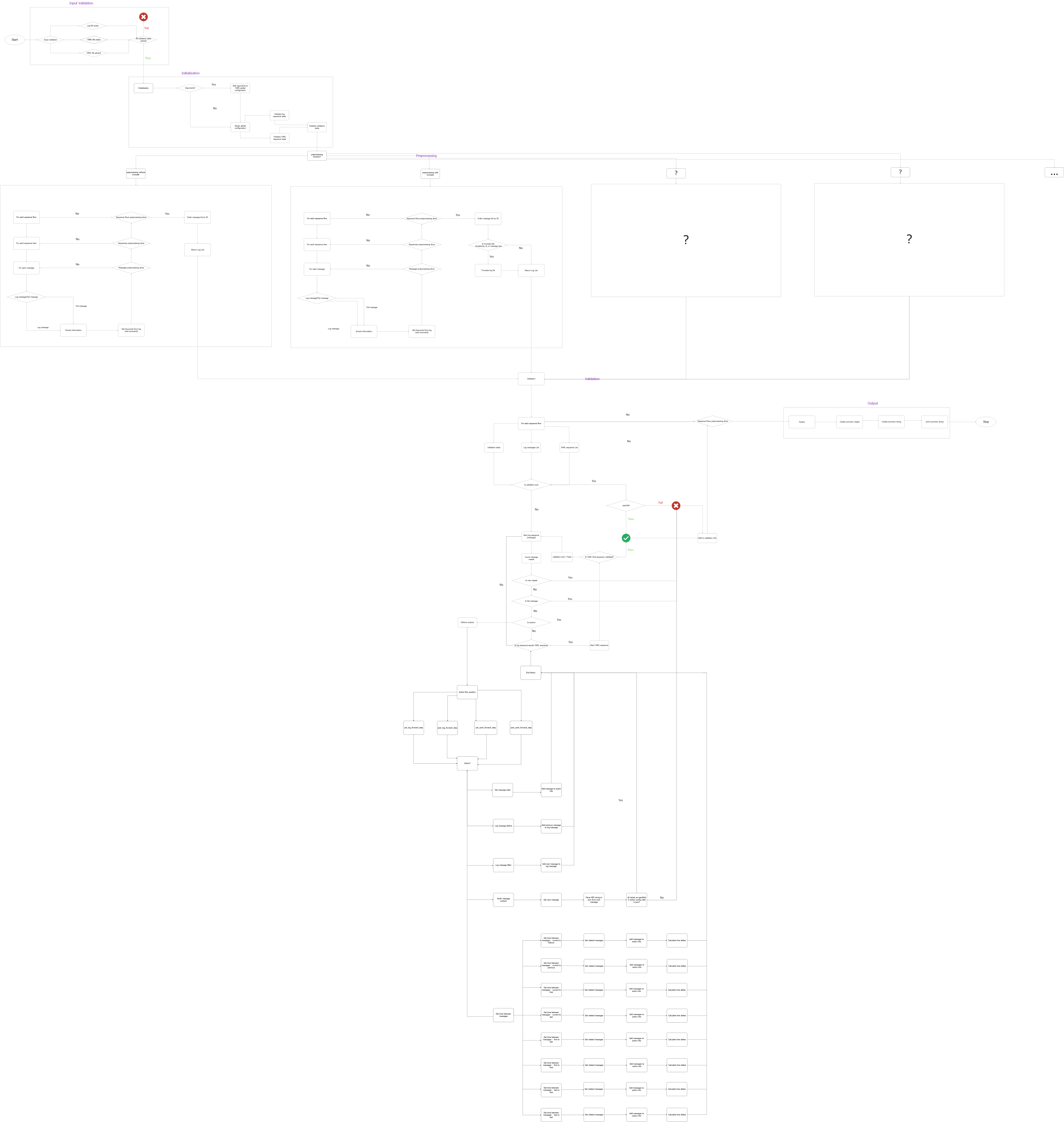
* + - 1. find\_ sequences\_time\_delta\_last\_to\_first:

Finds the time delta that passed between the last appearance of the first sequences to the first appearance of the second sequences.

* + - 1. find\_ sequences\_time\_delta\_last\_to\_last:

Finds the time delta that passed between the last appearance of the last sequences to the first appearance of the second sequences.

1. Design flow (can be zoomed in [Ctrl + mouse wheel]):



1. Questions:

Questions can be directed to Rotem Blumberg:

**Rotem Blumberg**

Python Developer Engineer

Airspan Networks

Mobile: +972-52-3950-387

Email: [rblumberg@airspan.com](mailto:rblumberg@airspan.com)