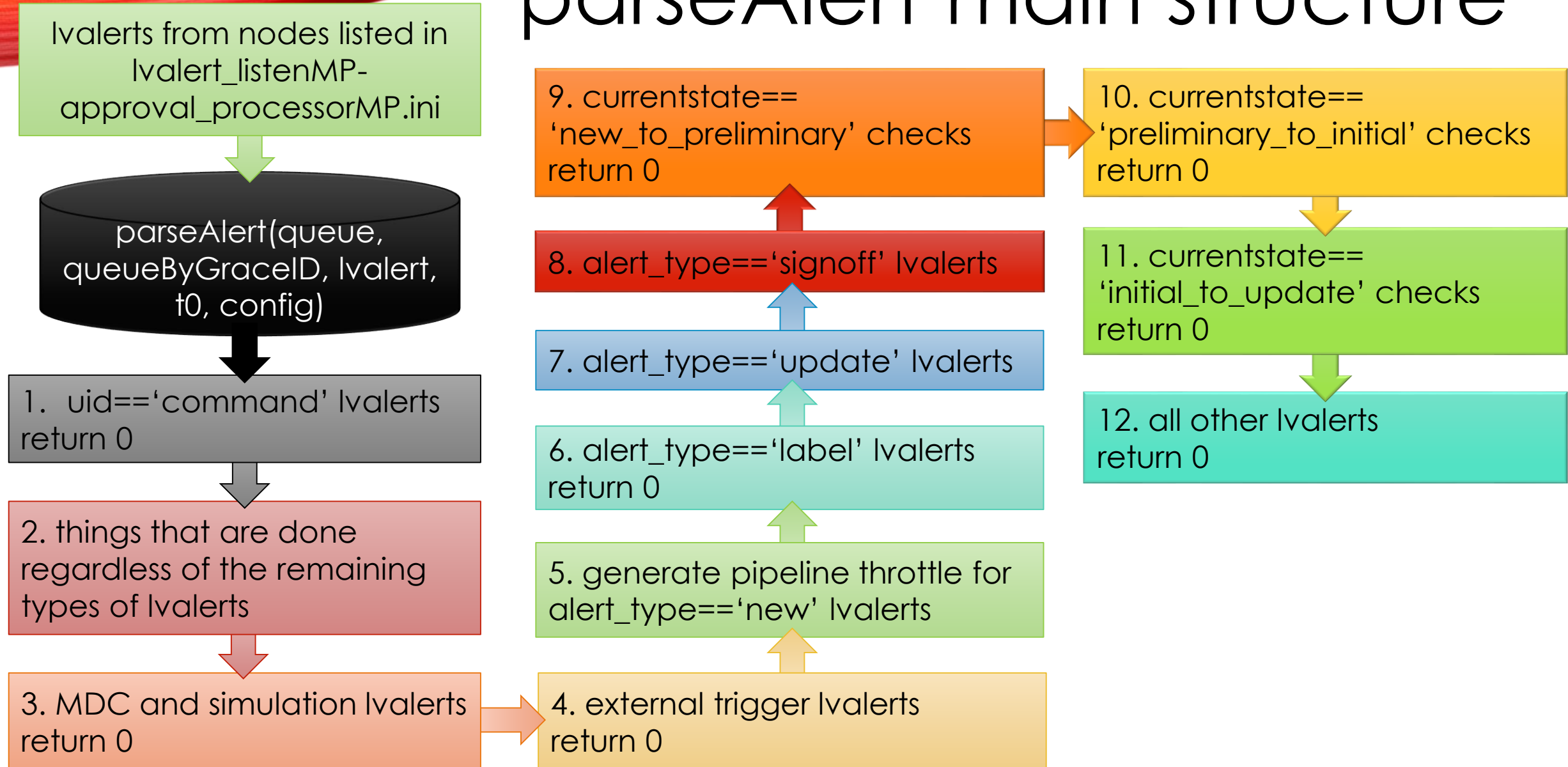




# approval\_processorMP Flowchart

GitHub commit: [b5d981792a9c47bacab4a3efa82dae5b340533f1](#)

# parseAlert main structure



1. uid == 'command'  
lvalerts

return parseCommand(queue,  
queueByGraceID, alert, t0)

return 0

- lvalert commands will be used to unthrottle a misbehaving pipeline
- lvalert commands are sent using lvalert node 'gdb\_processor\_test'

2. things that are done regardless of the remaining types of lvalerts

- instantiate GraceDB client from childConfig-approval\_processorMP.ini
- get other config settings and save in configdict
- set up logger if logger does not exist
- get alert specifics – 'uid': graceid, 'alert\_type': alert\_type, 'description': description, 'file': filename
- ensure we have an event\_dict and ForgetMeNow queue item for the graceid

if external trigger lvalert:  
create external trigger event\_dict

if GW trigger lvalert:  
create GW trigger event\_dict

### 3. MDC and simulation lvalerts

write to logger 'Mock data challenge or simulation. Ignoring.'

return 0

- in the future, we could process MDC lvalerts to send VOEvents to GCN with mass classifier information from the EM-Bright pipeline

### 4. external trigger lvalerts

if alert\_type=='label':  
record label in event\_dict

if alert\_type=='update' and a comment:

if 'GRB-GW  
Coincidence JSON  
file' in comment:  
record json was  
loaded into  
GraceDb in  
event\_dict

if comment from  
PyGRB or X-  
pipeline:  
record coinc info  
in event\_dict,  
create coinc.json  
and load to  
GraceDb

return 0

## 5. generate pipeline throttle for alert\_type=='new' lvalerts

create a pipeline throttle key based on group, pipeline, search

if queueByGraceID has key:  
get PipelineThrottle item (there should only  
be one)

if queueByGraceID does not have key:  
create PipelineThrottle item, add to overall  
queue and queueByGraceID

add graceid to PipelineThrottle item and check if it becomes throttled



## 6. alert\_type=='label' lvalerts

record label in event\_dict

if label=='PE\_READY':  
send Update  
VOEvent, update  
currentstate to  
'complete'

if label=='EM\_READY':  
send Initial VOEvent,  
update currentstate  
to 'initial\_to\_update'

if label=='EM\_Throttled':  
update currentstate to  
'throttled'

if checkLabels > 0:  
Update  
currentstate to  
'rejected'

if len(VOEvents) > 0 and retraction is not the  
last VOEvent sent:  
send retraction alert

return 0

## 7. alert\_type=='update' lvalerts

if filename ends with  
'fits.gz' or '.fits' and  
'lvem' in tag\_names:  
record skymap and  
submitter in event\_dict

if comment:


if re.match('minimum  
glitch-FAP',  
comment):  
record iDQ values in  
event\_dict

if re.match('resent  
VOEvent', comment):  
record resent VOEvent  
in event\_dict

if 'EM-Bright probabilities computed' in comment:  
record EM-Bright information in event\_dict

if 'Temporal coincidence with external trigger' in comment:  
record coinc info in event\_dict, create em\_coinc.json, load to  
GraceDb, and do the same with the external trigger event\_dict

if 'GRB-GW Coincidence JSON file' in comment:  
record json was loaded into GraceDb in event\_dict



8. alert\_type=='signoff' lvalert



record signoff in event\_dict

- from here on out, approval\_processorMP performs checks specific to the currentstate of the event candidate as specified in the event\_dict
- all GW trigger event\_dict's start in the 'new\_to\_preliminary' currentstate
- passedcheckcount set to 0



## 9. currentstate=='new\_to\_preliminary' checks

wait 10 seconds (or other amount specified in childConfig) before querying GraceDB for up-to-date labels

perform checks in new\_to\_preliminary = [farCheck, labelCheck, injectionCheck]

if checkresult==False:  
write in logger which check failed in new\_to\_preliminary, set currentstate to 'rejected'

if checkresult==None:  
pass

if checkresult==True:  
passedcheckcount += 1

if passedcheckcount == len(new\_to\_preliminary):  
write to logger that trigger passed all checks, send Preliminary VOEvent, set currentstate to 'preliminary\_to\_initial', and notify operators and advocates

return 0

## 10. currentstate=='preliminary\_to\_initial' checks

preliminary\_to\_initial = [farCheck, labelCheck, have\_lvem\_skymapCheck, idq\_joint\_fapCheck]

if humanscimons=='yes':  
add operator\_signoffCheck to preliminary\_to\_initial

if advocates=='yes':  
add advocate\_signoffCheck to preliminary\_to\_initial

perform checks in preliminary\_to\_initial

if checkresult==False:  
write in logger which check failed, set  
currentstate to 'rejected', label as 'DQV'

return 0

if checkresult==None:  
pass

if checkresult==True:  
passedcheckcount += 1

if passedcheckcount == len(preliminary\_to\_initial):  
write to logger that event candidate passed all checks, label 'EM\_READY'

## 11. currentstate=='initial\_to\_update' checks

perform checks in initial\_to\_update = [farCheck, labelCheck, have\_lvem\_skymapCheck]

if checkresult==False:  
write in logger which check failed, set  
currentstate to 'rejected', label as 'DQV'

if checkresult==None:  
pass

if checkresult==True:  
passedcheckcount += 1

if passedcheckcount == len(initial\_to\_update):  
write to logger that event candidate passed all  
checks, label 'PE\_READY'

return 0