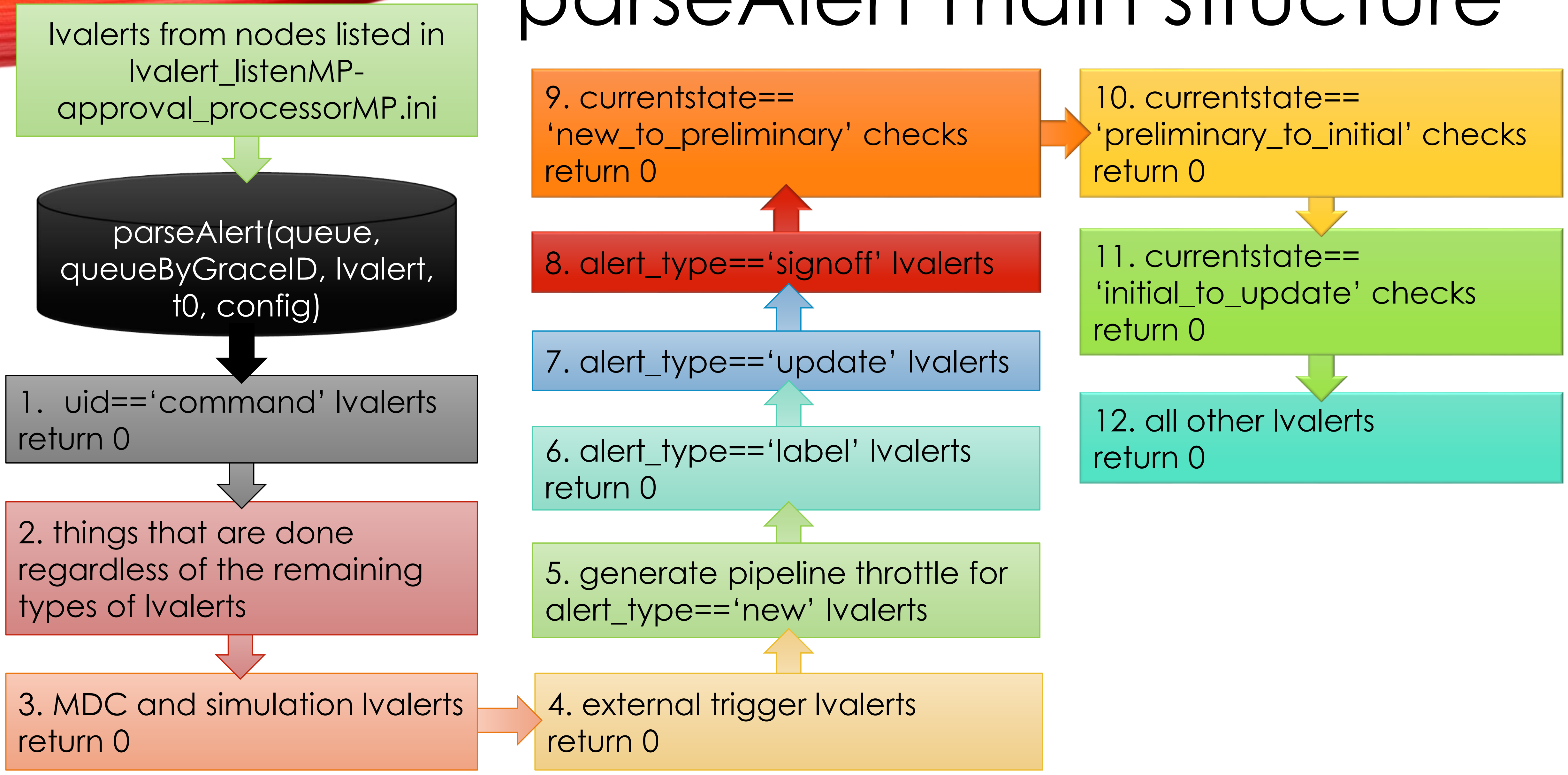
approval_processorMP Flowchart

GitHub commit: b5d981792a9c47bacab4a3efa82dae5b340533f1

parseAlert main structure



lvalerts from nodes listed in lvalert_listenMP-approval_processorMP.ini

parseAlert(queue, queueByGraceID, lvalert, t0, config)

1. uid==‘command’ lvalerts
return 0

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

3. MDC and simulation lvalerts
return 0

4. external trigger lvalerts
return 0

5. generate pipeline throttle for alert_type==‘new’ lvalerts

6. alert_type==‘label’ lvalerts
return 0

7. alert_type==‘update’ lvalerts

8. alert_type==‘signoff’ lvalerts

9. currentstate==
‘new_to_preliminary’ checks
return 0

10. currentstate==
‘preliminary_to_initial’ checks
return 0

11. currentstate==
‘initial_to_update’ checks
return 0

12. all other lvalerts
return 0

1.uid==‘command’ lvalerts

return parseCommand(queue, queueByGraceID, alert, t0)

return 0

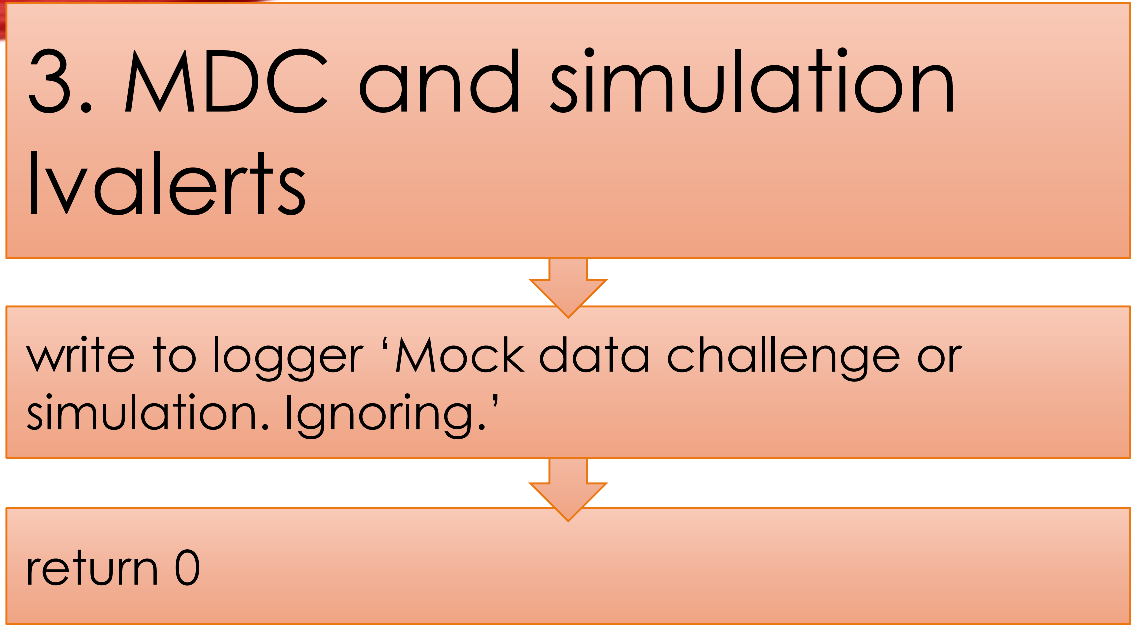
- lvalert commands will be used to unthrottle a misbehaving pipeline
- lvalert commands are sent using lvalert node ‘gdb_processor_test’

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 or pull existing one

if GW trigger lvalert:
create GW trigger event_dict or pull existing one

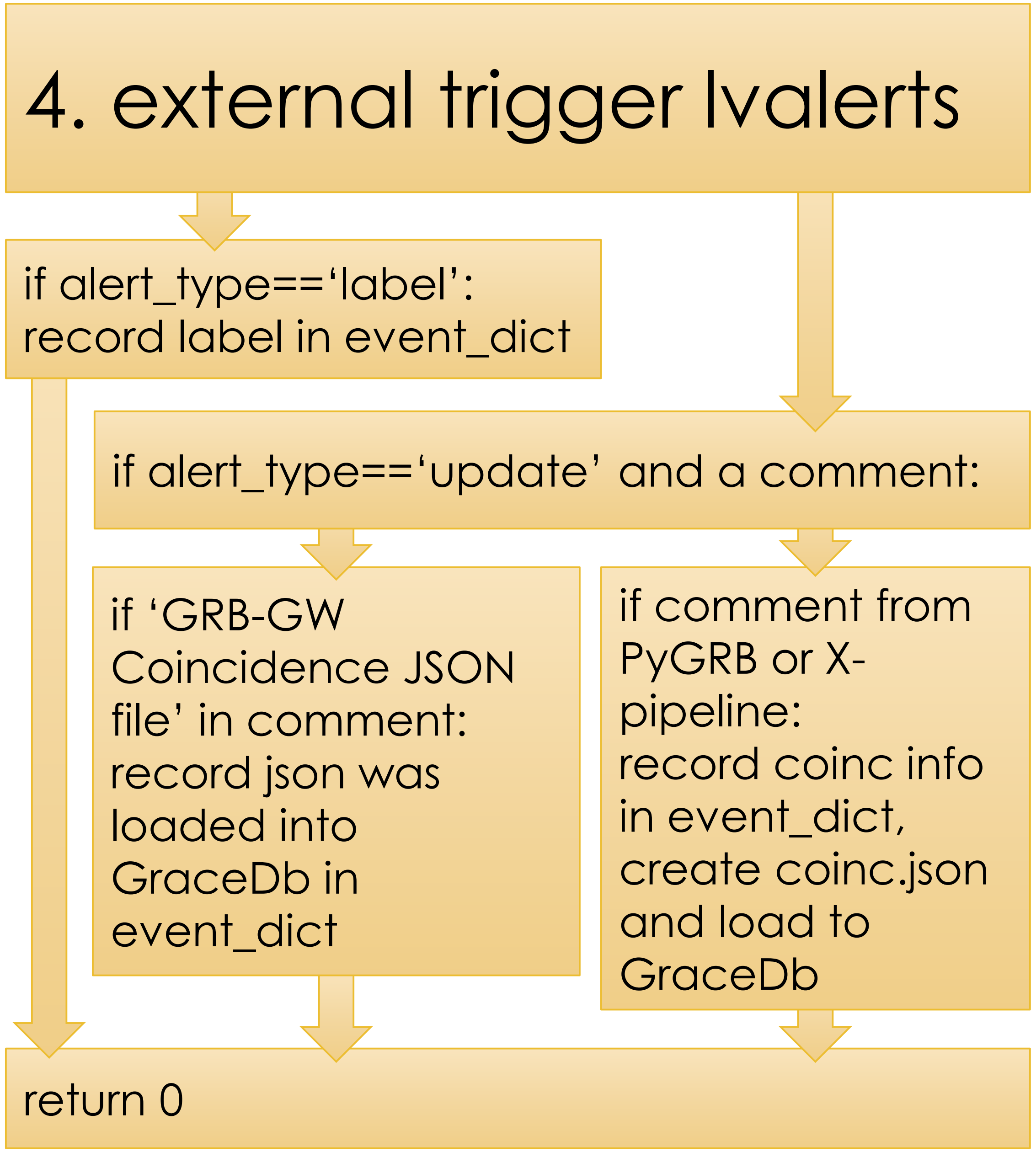


3. MDC and simulation lvalerts

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

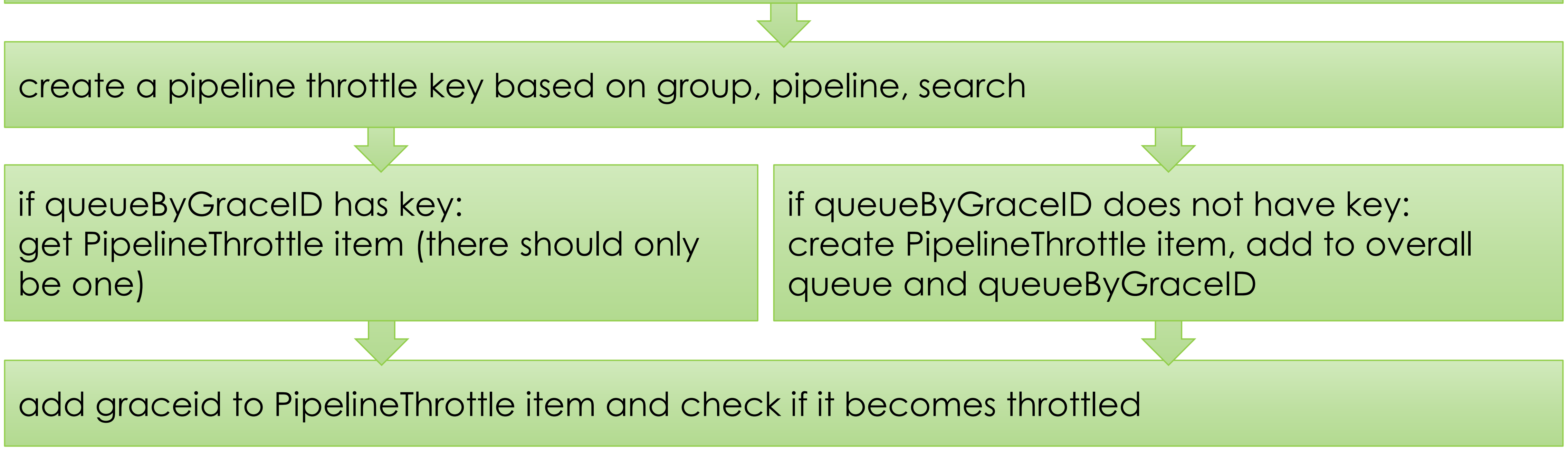
if alert_type==‘update’ and a comment:

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



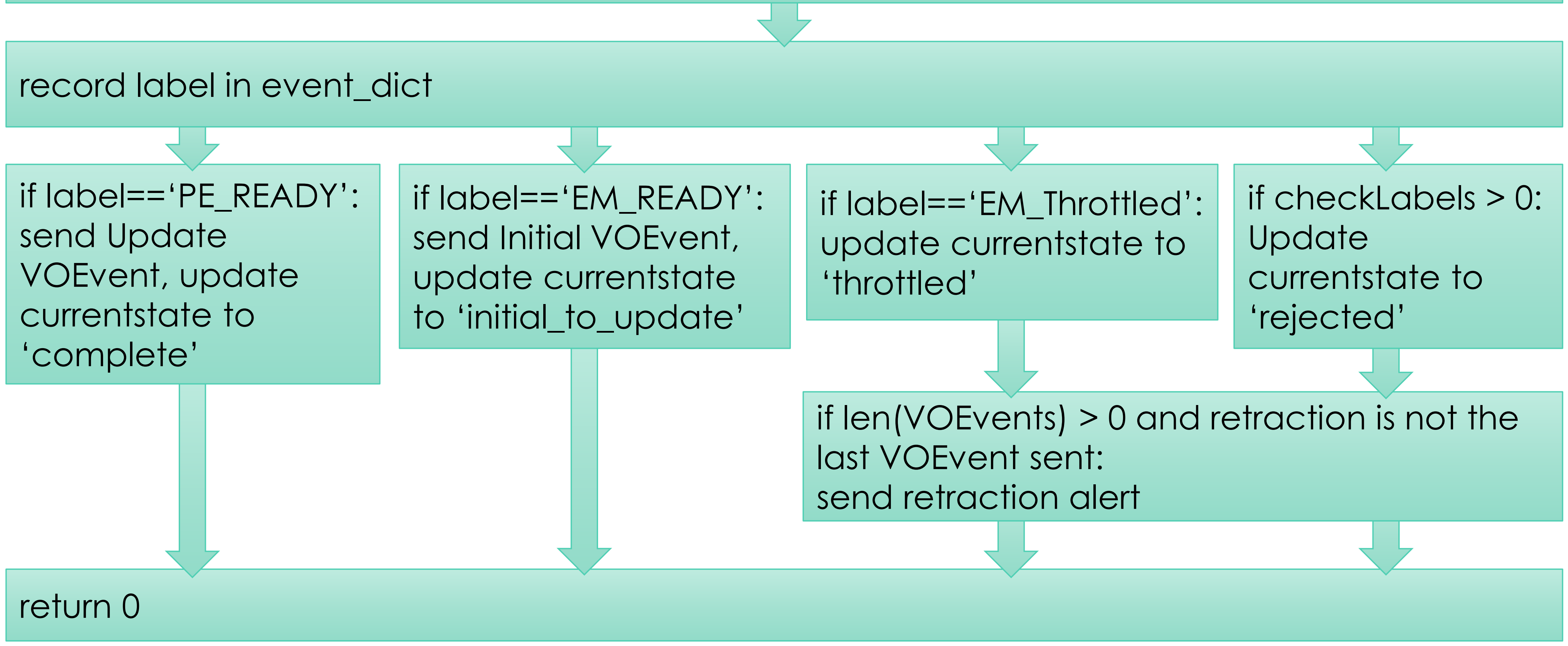
create a pipeline throttle key based on group, pipeline, search

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’

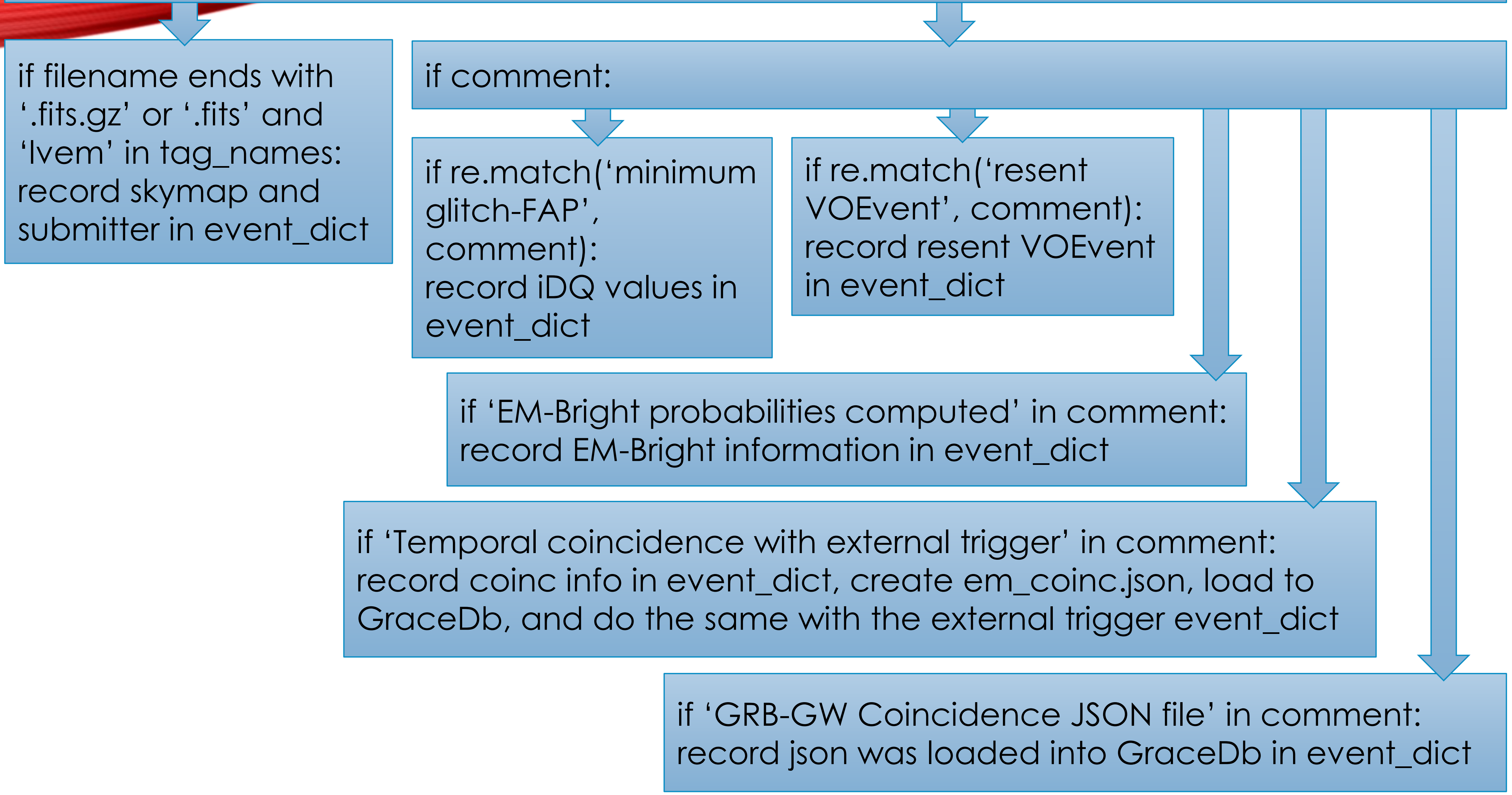
if label==‘EM_Throttled’: update currentstate to ‘throttled’

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

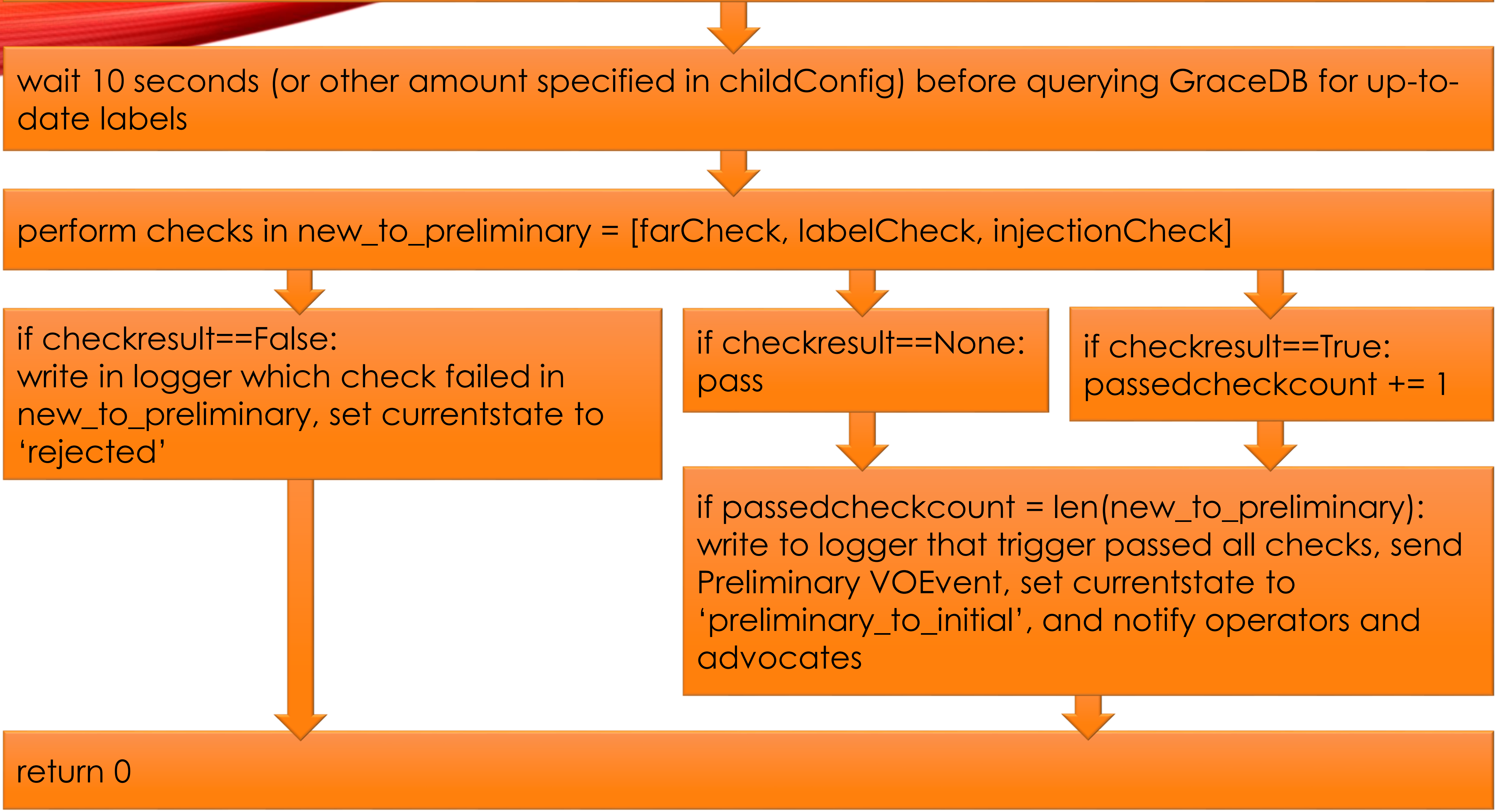
if ‘GRB-GW Coincidence JSON file’ in comment: record json was loaded into GraceDb in event_dict

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’

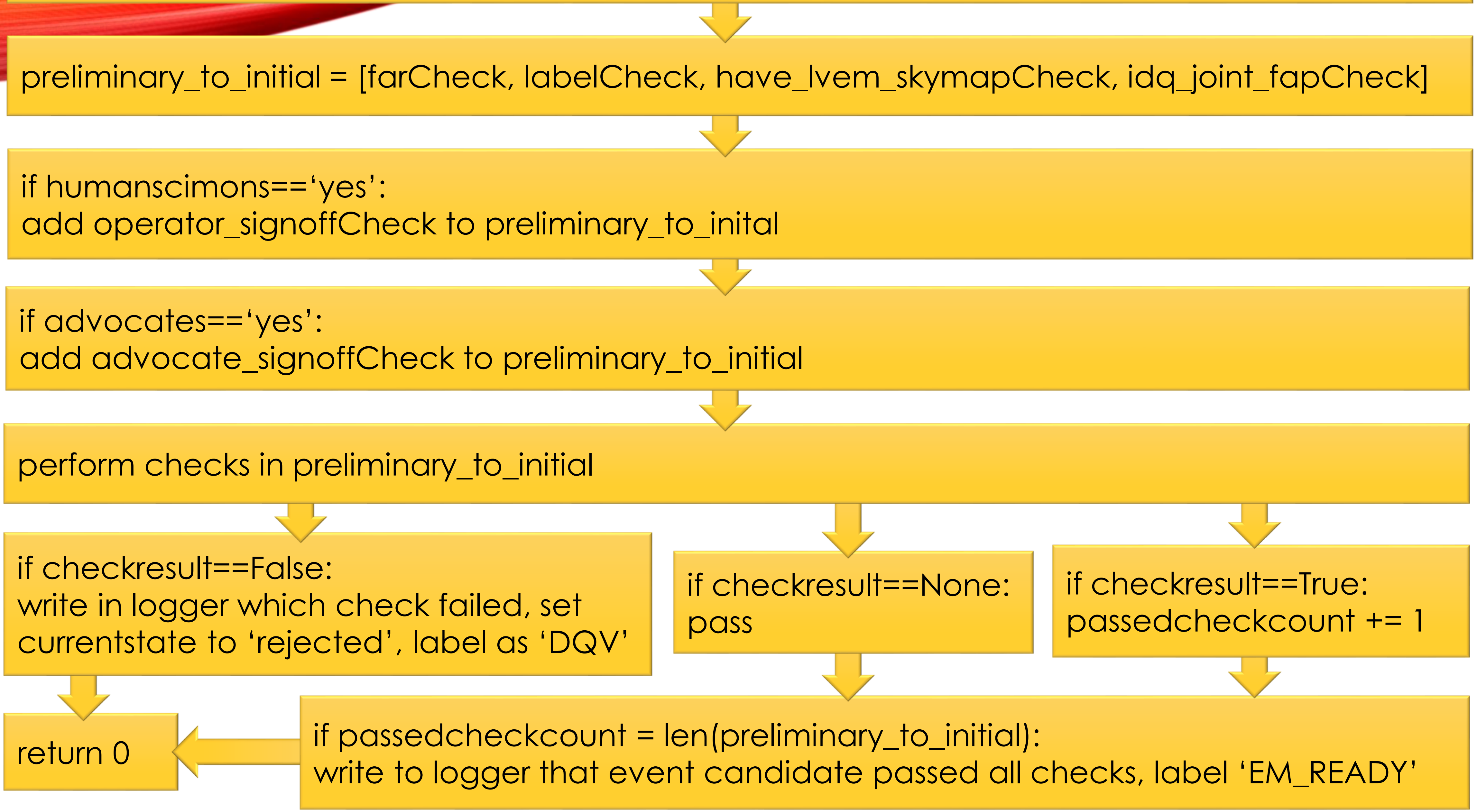
if checkresult==None:
pass

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_inital

if advocates==‘yes’:
add advocate_signoffCheck to preliminary_to_initial

perform checks in preliminary_to_initial

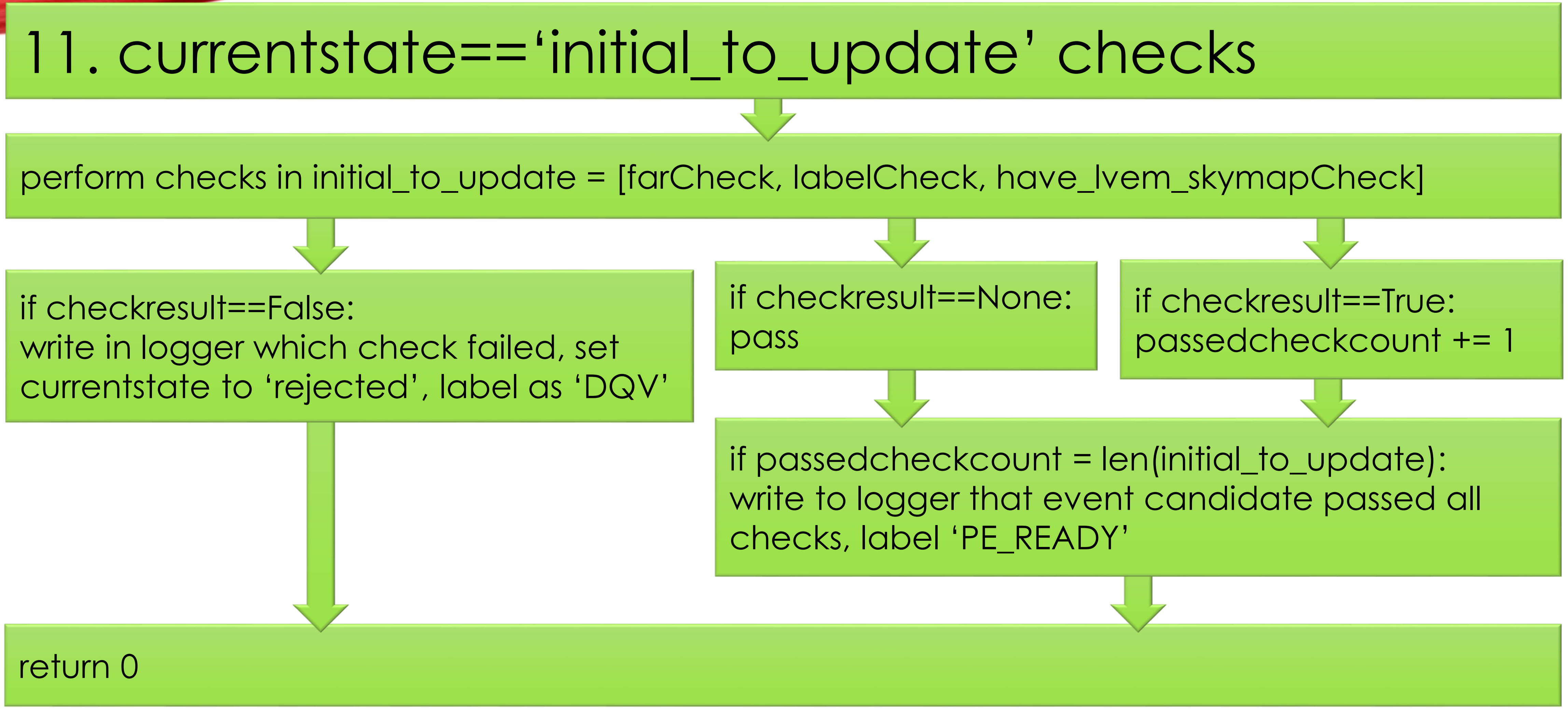
if checkresult==False:
write in logger which check failed, set currentstate to ‘rejected’, label as ‘DQV’

if checkresult==None:
pass

if checkresult==True:
passedcheckcount += 1

return 0

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