Table of Contents

SAMJob results in SAM-Nagios

⚠ still under development (but should already work in the certification/development infrastructure)

Summary

The general job state and some debug information gathered during a SAM Job will be published to the SAM-Nagios Framework via the ActiveMQ-messaging infrastructure. These are then considered for high-level availability/reliability reports and the dashboard display:

http://dashb-lhcb-sum.cern.ch http://grid-monitoring.cern.ch/mywlcg/ss/

Detailed information can be accessed on the SAM-Nagios Web interface (click Services and look for "org.lhcb.DiracTest-lhcb". The full debug information string should be displayed when clicking on the service name.)

(certificate login required)

https://sam-lhcb-prod.cern.ch/nagios/

https://sam-lhcb-preprod.cern.ch/nagios/ (Preproduction)

https://sam-lhcb-dev.cern.ch/nagios/ (Development)

Also in these web portals, the results of the functional tests (documented under https://tomtools.cern.ch/confluence/display/SAMDOC/Probes) can be accessed.

Technical Details

Configuration of SAM-Nagios

The passive service check feature of Nagios (https://twiki.cern.ch/twiki/bin/view/EGEE/GridMonitoringNcgOverview) allows the seamless integration of external service checks. A step by step (but not verbatim) documentation page for configuring is accessible under https://tomtools.cern.ch/confluence/display/SAM/Support+for+external+publishers

This page describes in detail

1. the configuration of the profile manager POEM (https://tomtools.cern.ch/confluence/display/SAMDOC/POEM+User%27s+Guide) to include the new metric (in this case "org.lhcb.DiracTest-lhcb")

2. the changes needed in the configuration files for the Nagios Configuration Generator (ncg).

This procedure should result in a SAM-Nagios instance ready to accept SAM-Job results. The next section describes the code for the publication of those results in LHCbDIRAC.

in LHCbDIRAC

One of the goals of this implementation is to give not only the global job result, but in case of failure give detailed information on which step encountered what problem. To this end, the `workflow_commons` dictionary used in sharing information between the different steps of one jobs has to be updated to include values for the

"SAMResults" and "SAMDetails" keys. This currently takes place both in `Workflow/Modules/GaudiApplication.py` and `SAMSystem/Modules/CVMFSCheck.py` of LHCbDIRAC.

```
#Example from CVMFSCheck.py
    except Exception, e:
      self.workflow_commons.setdefault( 'SAMResults', {})[ 'CVMFS' ] = 'CRITICAL'
      self.workflow_commons.setdefault( 'SAMDetails', {})[ 'CVMFS' ] = e
```

The last step before completion of the Job is the Module `SAMSystem/Modules/UploadSAMLogs.py` which was be expanded to not only log the results collected during the SAMJob, but also publish it to Nagios.

```
#Example from UploadSAMLogs.py
   # publish information to SAM-Nagios
      self.nagiosConnector.readConfig()
      self.nagiosConnector.initializeConnection()
      self.nagiosConnector.assembleMessage(
                            serviceURI=self.workflow_commons[ 'GridRequiredCEs' ],
                            # if one step fails, set the status to 'CRITICAL'
                            # in a SAMJob only OK (False) and CRITICAL (True)  are used
                            # this function will also accept appropriate strings or numbers 0-3
                            status='CRITICAL' in self.workflow_commons[ 'SAMResults' ].values(),
                            details ="Job_ID: %s. Logfile: %s Details: %s " % (
                                        self.workflow_commons[ 'logURL' ].split('/')[-1],
                                        self.workflow_commons[ 'logURL' ],
                                        ".  ".join(self.workflow_commons['SAMDetails'].values())
                                        ),
                            nagiosName='org.lhcb.DiracTest-lhcb'
                            )
      self.nagiosConnector.sendMessage()
      self.nagiosConnector.endConnection()
```

The messaging infrastructure (http://mig.web.cern.ch/mig/) currently consists of ActiveMQ - brokers the text-based STOMP protocol using the text-based STOMP protocol. Sophisticated solutions for clients exist that are independent of the messaging technology used (https://tomtools.cern.ch/confluence/display/SAM/MEG, https://messaging.readthedocs.org). To reduce complexity and improve maintainabilty, in this project the recommended python messaging client library stomp.py (https://code.google.com/p/stomppy/) is used without an additional directory queue.

The required methods of stomp.py are bundled in the small utility `Core/Utilities/NagiosConnector.py` together with a function to assemble the information gathered and stored in `workflow_commons` into a message of the correct format and are executed at the end `UploadSAMLogs`. The broker address and port, queue name and have to be stored in the DIRAC configuration according to

```
Operations
{
  <setup>
  {
    NagiosConnector
    {
      MsgBroker = sam-validation.msg.cern.ch
      MsgPort = 6163
      MsgQueue = /queue/grid.probe.metricOutput.EGEE.sam-lhcb-dev_cern_ch

    }
  }
}
```

where it is possible to address different Nagios instances according to which `setup` (Certificaton, Production, ...) is used.

The direct publication to the Message Result Store (MRS) without the detour to Nagios works exactly the same way, where the queue name can be taken from

https://tomtools.cern.ch/confluence/display/SAM/MRS

Note that for the MRS some fields of the message not parsed in Nagios actually do have meaning (there is for example need to specify the experiment, whereas the nagios instance is setup only for the vo and does not need this info). These are not send by NagiosConnector at the moment. The assembleMessage function would have to be modified according to

https://tomtools.cern.ch/confluence/display/SAM/Support+for+external+publishers

---

This topic: LHCb > LHCbDIRACNagiosConnector
Topic revision: r2 - 2013-09-20 - ValentinVolkl