Service Level Target - Availability Reliability

Description

The ARGO service collects status results and computes daily and monthly availability (A) and reliability (R) metrics of distributed services. Both status results and A/R metrics are delivered through the ARGO Web UI, with the ability for a user to drill-down from the availability of a site to individual test results that contributed to the computed figure.

Availability - Defined as the ability of a service to fulfil its intended function at a specific time or over a calendar month.

Reliability - Defined as the ability of a service to fulfil its intended function at a specific time or over a calendar month, excluding scheduled maintenance periods.

Components

ARGO is comprised of the following building blocks:

- **The consumer.** This service collects the metric results from the Message Broker Network (MBN) and delivers them to the compute engine in avro encoded format
- **The connectors.** This is a collection of python modules that periodically connect to sources of truth (such as GOCDB for topology or downtimes, or POEM services for low level metric profiles etc) and deliver the information to the compute engine in avro encoded format. The period is set to daily.
- **The prefilter.** This component is used by the ARGO compute engine in order to filter out results that may not be official (for example a non-authorative monitoring instance publishing results via the MBN)
- **The compute engine.** Using the filtered data collected the compute engine is responsible for flattening out the metric results and for computing the services availability and reliability metrics. See next section for a more detailed description on how the computations are being performed. Results (status and A/R) are passed onto a fast, reliable and distributed datastore.
- **The REST API.** This component serves all computed status and A/R results via a programmatic interface.
- **The Web UI.** This component is based on the Lavoisier software. It is used in order to present the status and A/R results graphically and gives the ability to any given user to drill down from the availability of a given resource down to the actual metric results that were recorded and contributed to the computed figures.

Definitions

Groupings of resources

The definitions of entities (resources) are the following:

- **Service Endpoint:** A service endpoint is defined as a hostname and service pair, so for example foo.example.com is a hostname, mysql is a service and a mysql database running on foo.example.com (i.e. foo.example.com:mysql) is a service endpoint.
- **Service Flavour**: A collection of same services (service endpoints). For example, multiple CREAM CEs in a site together make up the CREAM CE service flavour for the site.
- **Site:** A collection of Service Flavours. A site can be made up of one or more service flavours.
- **NGI:** A collection of Sites.

Metrics and Statuses

The following define the Metric and the Status, core building blocks of the algorithm used for A/R computations

- **Metric:** A Metric is a functional test for a given service flavour. Within a given context (i.e. ROC_CRITICAL) each service flavour has a set of service metrics that verify its functionality and performance. This correlation between service flavour functionality and Metrics is given by the POEM service. Metric results are generated when monitoring (i.e. Nagios) tests are run on a particular service endpoint.
- **Status**: Status of a metric result, service, service endpoint, service flavour or a site is the status of that entity at a given point in time. (Note here that to go from metric result onto a site hierarchy some logic is being used in the background. This is discussed more in detail below.) Possible status values are
  - OK
  - WARNING
  - CRITICAL
  - UNKNOWN
  - MISSING
  - DOWNTIME

These status values are mutually exclusive. The status of a resource can have only one value at a given point in time.

Profiles

There are three (3) types of profiles used within each A/R computation:

- **Metric profile:** A profile defines which metrics are to be considered to compute the status of a service of a particular flavour.
- **Operations profile**: An operations profile defines how to aggregate status results from the metric level onto service endpoint and service flavour status results. In principal these define how ANDing and ORing operations are performed between status values. For example:
  - OK **AND** CRITICAL => CRITICAL
  - OK **OR** CRITICAL => OK
- **Aggregation profile:** An aggregation profile defines how to aggregate service flavour statuses into site status results. As an example in the default Site A/R aggregation profiles service endpoints of the same type are ANDed to form the service flavor status (for example multiple CREAM-CE flavours are ANDed into one service flavour) while similar service flavours are ORed (for example CREAM-CE OR ARC-CE in the default profile)

- **Report**: Any given combination of one metric, one operations and one aggregation profile creates an ARGO report (see section reports below).

Time slices

For computations of A/R results the ARGO compute engine uses 288 discrete samples on the daily timeline. The quantization of 288 values has been selected because it corresponds to a sampling frequency of 5mins. (24h * 60 = 1440 mins / 288 = 5mins).

The compute engine performs computations on a daily base timeframe (even though the computations run per hour, actually ARGO performs the same daily computation with updated metric data).

A/R Computation Algorithm

The A/R results are produced by integrating status results according to metric, operations and aggregation profiles. So the compute engine needs to handle status results from metric data in an efficient way in order to algorithmically combine and integrate upon them. When the engine creates a daily timeline for a specific service endpoint and a specific metric it initiates a 288 item array reserved for the service endpoint and metric couple.

When metric data is collected for a specific metric (for a specific service endpoint) it is roughly in the following form:

```
{ time_stamp | metric | service_flavour | hostname | status | vo | vofqan |
profile | dates }
```

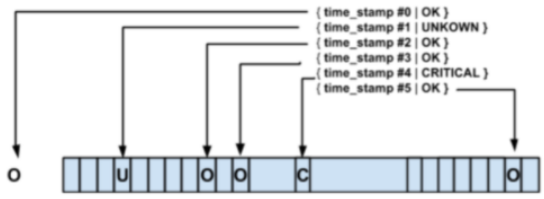
The engine then gathers all relevant daily data for the specific service endpoint and metric. For example imagine that for a given day 5 distinct metric data for the hostname foo.example.com, the service mysql. service and the metric mysql.some.metric. The data rows for that day will be of the following form:

```
{ time_stamp #1 | mysql.some.metric | mysql.service | foo.example.com |
UNKOWN | vo | vofqan | profile | dates }
{ time_stamp #2 | mysql.some.metric | mysql.service | foo.example.com | OK |
vo | vofqan | profile | dates }
{ time_stamp #3 | mysql.some.metric | mysql.service | foo.example.com | OK |
vo | vofqan | profile | dates }
{ time_stamp #4 | mysql.some.metric | mysql.service | foo.example.com |
CRITICAL | vo | vofqan | profile | dates }
{ time_stamp #5 | mysql.some.metric | mysql.service | foo.example.com | OK |
vo | vofqan | profile | dates }
```

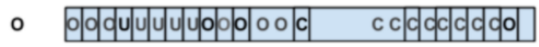
The compute engine will also grab the last metric from the previous day timeline

```
{ time_stamp #0 | mysql.some.metric | mysql.service | foo.example.com | OK |
vo | vofqan | profile | dates }
```

Based on the timestamp and status fields the compute engine will map these data points to the correct indexes of the metric array:



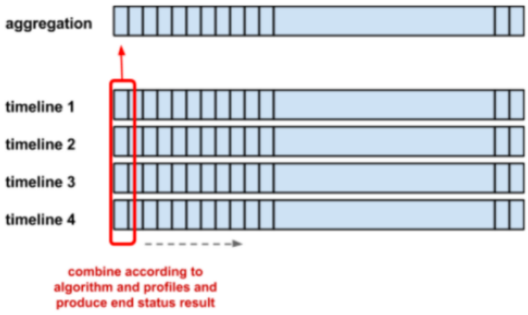
Afterwards the compute engine will fill in the gaps appropriately, like so:



When the engine needs to combine several different timelines in order to produce an aggregated timeline result (for example for a specific service flavor), it does the following:

1. Reserves a new array for the aggregation timeline
2. Aligns the relevant timeline arrays
3. Begins from index 0 and combines all array_items[0] to produce the aggregation_item[0]
4. Moves to next index

The end result is an aggregated timeline:



- Aggregation of metric timelines into service endpoint timelines is based on the given metric profile used..
- Aggregation of service endpoint timelines into service flavour timelines is based on the given aggregation profile used.
- Aggregation of service flavor timelines into group of endpoints (sites) is based also on the given aggregation profile used.

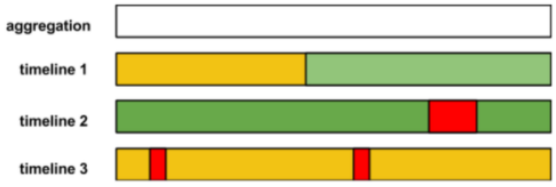
In all cases AND and OR operations are based on the Operations profile used.

It is important to note that the discrete handling of the status results as samples gives an easy and graceful way to implement aggregations.

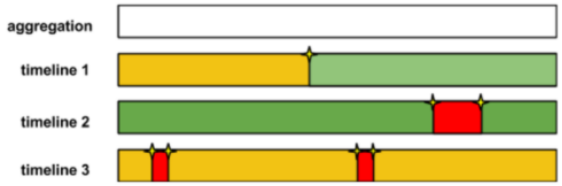
Status Aggregation Algorithm

Regarding status timelines and since there are no pre-established points in time shared by all timelines (like in sampling and A/R computations described above) the compute engine operates differently.

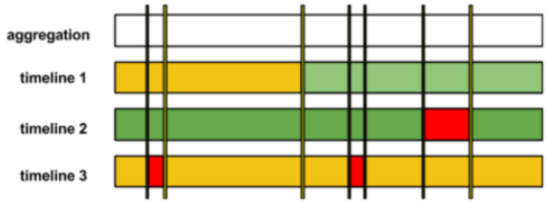
If for example the compute engine is given 3 continuous status timelines that need to be aggregated a new timeline for the aggregation is reserved.



Then the points of interest (timestamps were status changes occur) are collected

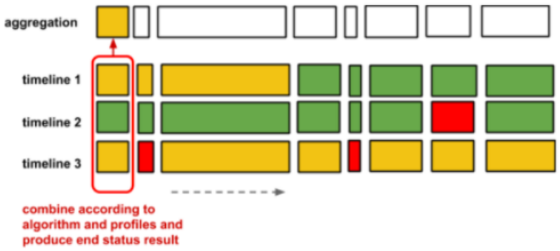


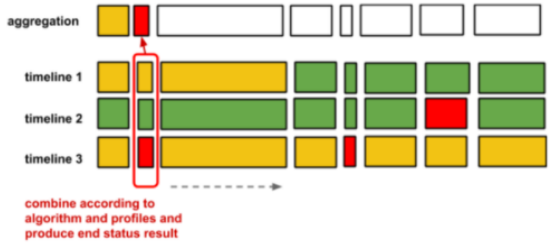
and the compute engine slices the timeline accordingly



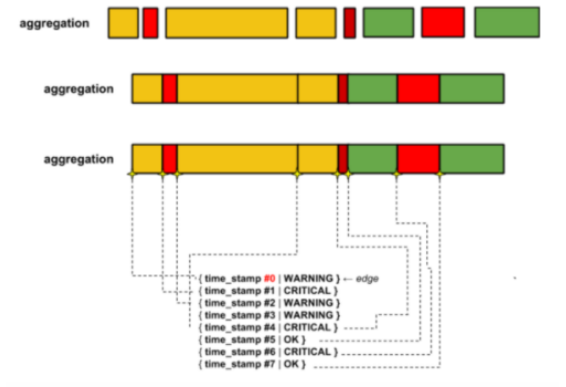
The compute engine then creates a number of chunks based on the points of interest found

And iteratively fills up the gaps progressively based on the profiles used in the given computation.





Once the filling up is completed the compute engine stitches back the complete aggregated timeline, like in the picture below:



Reports

In the following subsections the metric and aggregation profiles used for each EGI report are given.

The operations profile used in all of the subsequent EGI reports are given in the tabulars here:

AND	OK	WARNING	UNKNOWN	MISSING	CRITICAL	DOWNTIME
OK	OK	WARNING	UNKNOWN	MISSING	CRITICAL	DOWNTIME
WARNING	WARNING	WARNING	UNKNOWN	MISSING	CRITICAL	DOWNTIME
UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	MISSING	CRITICAL	DOWNTIME
MISSING	MISSING	MISSING	MISSING	MISSING	CRITICAL	DOWNTIME
CRITICAL	CRITICAL	CRITICAL	CRITICAL	CRITICAL	CRITICAL	CRITICAL
DOWNTIME	DOWNTIME	DOWNTIME	DOWNTIME	DOWNTIME	CRITICAL	DOWNTIME

OR	OK	WARNING	UNKNOWN	MISSING	CRITICAL	DOWNTIME
OK	OK	OK	OK	OK	OK	OK
WARNING	OK	WARNING	WARNING	WARNING	WARNING	WARNING
UNKNOWN	OK	WARNING	UNKNOWN	UNKNOWN	CRITICAL	UNKNOWN
MISSING	OK	WARNING	UNKNOWN	MISSING	CRITICAL	DOWNTIME
CRITICAL	OK	WARNING	CRITICAL	CRITICAL	CRITICAL	CRITICAL
DOWNTIME	OK	WARNING	UNKNOWN	DOWNTIME	CRITICAL	DOWNTIME

Sites A/R

In the Sites A/R report the following metric profile is used:

Metric	Service Type
org.nordugrid.ARC-CE-ARIS	ARC-CE
org.nordugrid.ARC-CE-IGTF	ARC-CE
org.nordugrid.ARC-CE-result	ARC-CE
org.nordugrid.ARC-CE-srm	ARC-CE
org.nordugrid.ARC-CE-sw-csh	ARC-CE
emi.cream.CREAMCE-JobSubmit	CREAM-CE
emi.wn.WN-Bi	CREAM-CE

emi.wn.WN-Csh	CREAM-CE
emi.wn.WN-SoftVer	CREAM-CE
hr.srce.CADist-Check	CREAM-CE
hr.srce.CREAMCE-CertLifetime	CREAM-CE
hr.srce.GRAM-Auth	GRAM5
hr.srce.GRAM-CertLifetime	GRAM5
hr.srce.GRAM-Command	GRAM5
hr.srce.QCG-Computing-CertLifetime	QCG.Computing
pl.plgrid.QCG-Computing	QCG.Computing
hr.srce.SRM2-CertLifetime	SRMv2
org.sam.SRM-Del	SRMv2
org.sam.SRM-Get	SRMv2
org.sam.SRM-GetSURLs	SRMv2
org.sam.SRM-GetTURLs	SRMv2
org.sam.SRM-Ls	SRMv2
org.sam.SRM-LsDir	SRMv2
org.sam.SRM-Put	SRMv2
org.bdii.Entries	Site-BDII
org.bdii.Freshness	Site-BDII
emi.unicore.TargetSystemFactory	unicore6.TargetSystemFactory
emi.unicore.UNICORE-Job	unicore6.TargetSystemFactory
eu.egi.OCCI-IGTF	eu.egi.cloud.vm-management.occi
eu.egi.cloud.OCCI-Context	eu.egi.cloud.vm-management.occi
eu.egi.cloud.OCCI-VM	eu.egi.cloud.vm-management.occi
org.nagios.OCCI-TCP	eu.egi.cloud.vm-management.occi
eu.egi.Keystone-IGTF	org.openstack.nova
eu.egi.cloud.OpenStack-VM	org.openstack.nova
org.nagios.Keystone-TCP	org.openstack.nova

The Aggregation profile used is the following one:

Sites Aggregation Profile			
Operation	**Capability**	Operation	**Service Flavor**
AND	Compute	**OR**	CREAM-CE
			ARC-CE
			GRAM5
			unicore6.TargetSystemFactory
			QCG.Computing
	Storage	**OR**	SRMv2
			SRM
	Information	**OR**	Site-BDII
	vm-management	**OR**	eu.egi.cloud.vm-management.occi
			org.openstack.nova

NGI sites A/R

For the NGI level aggregation all A/R results for sites belonging to the NGI are collected and aggregated dynamically weighted based on the HEPSPEC factor for each site. Hence larger sites contribute more to the overall NGI A/R and smaller sites less.

Monthly League Tables

Monthly EGI League Tables are accessible via the ARGO Web UI (Lavoisier) under the following link: **http://argo.egi.eu/lavoisier/ngi_reports?month=YYYY-MM**

To get results for a specific month one should replace YYYY and MM with the calendar year and month respectively, hence to obtain results for August 2015 the link should be formatted as follows: http://argo.egi.eu/lavoisier/ngi_reports?month=2015-08 .

Monthly Reports are also available at **Resource Centres OLA and Resource infrastructure Provider OLA reports wiki page**

Core services A/R

The Core service A/R report utilizes the following metric profile:

Metric	Service Type
org.activemq.OpenWireSSL	egi.APELRepository
org.nagiosexchange.AccountingPortal-WebCheck	egi.AccountingPortal
org.nagiosexchange.AppDB-WebCheck	egi.AppDB
org.nagiosexchange.GGUS-WebCheck	egi.GGUS
org.nagios.GOCDB-PortCheck	egi.GOCDB
org.nagiosexchange.GOCDB-PI	egi.GOCDB
org.nagiosexchange.GOCDB-WebCheck	egi.GOCDB
org.nagiosexchange.GSTAT-WebCheck	egi.GSTAT
org.activemq.Network-Topic	egi.MSGBroker
org.activemq.Network-VirtualDestination	egi.MSGBroker
org.activemq.OpenWire	egi.MSGBroker
org.activemq.OpenWireSSL	egi.MSGBroker
org.activemq.STOMP	egi.MSGBroker
org.activemq.STOMPSSL	egi.MSGBroker
org.nagiosexchange.MetricsPortal-WebCheck	egi.MetricsPortal
org.nagiosexchange.OpsPortal-WebCheck	egi.OpsPortal
eu.egi.cloud.Perun-Check	egi.Perun
org.nagiosexchange.Portal-WebCheck	egi.Portal
ch.cern.sam.SAMCentralWebAPI	egi.SAM
org.nagiosexchange.TMP-WebCheck	egi.TMP
org.nagiosexchange.OpsPortal-WebCheck	ngi.OpsPortal
org.nagiosexchange.MyEGIWebInterface	ngi.SAM
org.nagiosexchange.NagiosHostSummary	ngi.SAM
org.nagiosexchange.NagiosProcess	ngi.SAM
org.nagiosexchange.NagiosServiceSummary	ngi.SAM

org.nagiosexchange.NagiosWebInterface	ngi.SAM
org.nagiosexchange.MyEGIWebInterface	vo.SAM
org.nagiosexchange.NagiosHostSummary	vo.SAM
org.nagiosexchange.NagiosProcess	vo.SAM
org.nagiosexchange.NagiosServiceSummary	vo.SAM
org.nagiosexchange.NagiosWebInterface	vo.SAM

The Aggregation profile used is the following one:

Core Services Aggregation Profile			
Operation	**Capability**	Operation	**Service Flavor**
AND	gstat	**OR**	egi.GSTAT
	vosam	**OR**	vo.SAM
	ngisam	**OR**	ngi.SAM
	egisam	**OR**	egi.SAM
	brokering	**OR**	egi.MSGBroker
	egiportal	**OR**	egi.Portal
	egiopsportal	**OR**	egi.OpsPortal
	egimetricsportal	**OR**	egi.MetricsPortal
	registry	**OR**	egi.GOCDB
	helpdesk	**OR**	egi.GGUS
	applications	**OR**	egi.AppDB
	authentication	**OR**	egi.Perun
	tpm	**OR**	egi.TPM
	apelrepository	**OR**	egi.APELRepository
	accountingportal	**OR**	egi.AccountingPortal

Recomputation procedure

Please refer to PROC10.