Service level status—a new real-time status display for IT services

To cite this article: S Lopienski 2008 J. Phys.: Conf. Ser. 119 052025

View the article online for updates and enhancements.

You may also like

- Characterization and Purification of Surfactant Sodium Ligno Sulfonate (SLS) From Biomass Waste in The Application Of Enhanced Oil Recovery (EOR)
  S Priyanto, T D Kusworo, Sayyidah et al.

- Template-Assisted Electrodeposition of Fe-Ni-Co Nanowires: Effects of Electrolyte pH and Sodium Lauryl Sulfate
  D Li and E Podlaha

- Challenge sodium lignosulfonate surfactants synthesized from bagasse as an injection fluid based on hydrophil-lipophilic balance
  R Setiati, S Siregar, T Marhaendrajana et al.
Service Level Status -
a new real-time status display for IT services

Sebastian Lopienski
IT Department, FIO group
CERN, European Organization for Nuclear Research, 1211 Genève 23, Switzerland
E-mail: Sebastian.Lopienski@cern.ch

Abstract. Nowadays, IT departments provide, and people use, computing services of an increasingly heterogeneous nature. There is thus a growing need for a status display that groups these different services and reports their status and availability in a uniform way. The Service Level Status (SLS) system addresses these needs by providing a web-based display that dynamically shows availability, basic information and statistics about various IT services, as well as the dependencies between them.

This paper first introduces the requirements SLS had to meet, and the main concepts behind it, like service availability and status, Key Performance Indicators (KPIs), sub/meta-services, and service dependencies. It then describes the SLS system architecture, and some interesting implementation details, such as the usage of XML Schemas. Since clear visualization of service availability and status is one of the main goals of SLS, emphasis is put on describing the intuitive web-based user interface.

1. Introduction

Almost everyone in an organization like CERN uses one or more of the computing services provided by the IT Department on a regular basis. Being able to find out whether or not a particular service is working properly when things don’t work as expected is most important. Unfortunately, the previous IT status display at CERN (TVScreen [1]) was often misleading, as its status for a service depended on the number of alarms for the machines supporting the service, and not on the status of the service itself. This led to at least two problems:

- Many of today’s services, such as the fax gateway, run on multiple machines for redundancy and so the service can be working perfectly even if one of the machines has a problem.
- Large services, such as AFS, could be almost fully available, but with a problem on one machine affecting a certain group of users – e.g. ATLAS.

To solve these problems and improve the user visibility of the real status of CERN IT Department services, Service Level Status (SLS) system was developed. The aim was to provide a web-based tool that dynamically shows services availability, basic information and statistics about IT services as well as their dependencies and interrelationships. The SLS display, although targeted at end-users, provides a high-level view of service status that is interesting for IT Department management as well as other service providers. SLS proves to be particularly useful after incidents like power cuts or cooling problems affecting IT services – it shows which services have been restored and which require further attention.
2. Concepts
Given the diversity of computing services, SLS would not be able to measure availability of each service. Therefore, it is the manager of a given service who defines a way to calculate or estimate its availability, depending on its configuration, functionality, specific features etc. It means that SLS is not a monitoring system itself – it collects status and availability information for registered services, and displays it on the Web.

2.1. Availability and status
The service availability, as shown by SLS, is a measure that defines to what extent a given service is accessible and performing as expected. The current availability of a service is represented as a percentage – 100% means that the service is fully available, while 0% indicates that the service is completely down. Any other availability value will have different meaning for different services: 95% availability of, for instance, LXBUILD service would mean high load on one of the nodes, while for CVS service it could mean that 5% of the repositories are not accessible.

Additionally, a service is in one of the four status levels: fully available, affected, degraded and not available, which are traditionally marked with green, yellow, orange and red colours, respectively. Service status depends on the service availability and on so called availability thresholds. The latter is a series of three numbers specifying at what level a service changes its status. For example, for availability thresholds 40, 70, 90 a service would be fully available if its availability is between 90 and 100, and degraded if its availability is between 40 and 70. Service managers can modify availability threshold levels for their services.

From user point of view, availability value and status are independent, so one service could be marked as fully available, and another degraded, when both have availability of 85%.

| Service performance |
|---------------------|
| GRID CPU/WALL done % | ![Green dot] |
| GRID CPU/WALL done % | ![Green dot] |
| GRID CPU/WALL done % | ![Green dot] |
| GRID CPU/WALL done % | ![Green dot] |
| Other CPU/WALL done % | ![Green dot] |
| Other CPU/WALL done % | ![Green dot] |
| Other CPU/WALL done % | ![Green dot] |
| Other CPU/WALL done % | ![Green dot] |
| Other CPU/WALL done % | ![Green dot] |
| Other CPU/WALL done % | ![Green dot] |
| Other CPU/WALL done % | ![Green dot] |
| Average CPU/WALL done % | ![Green dot] |

![Figure 1. Key Performance Indicators (KPIs).](image-url)

2.2. Key Performance Indicators
Key Performance Indicators (KPIs) are metrics that indicate whether a service meets its requirements (performance or utilization level). It is an extension of the concept of availability, as KPIs describe various aspects of a service, as compared to its expected performance. A single service may declare multiple KPIs, such as CPU delivered to VO as compared to quota, or percentage of AFS volumes and servers available.

KPIs in SLS are pairs of two values: measured and target. If the measured value is worse than target value, the KPI shows that the service doesn't meet its requirements. In some cases, the measured value is expected to be higher than the corresponding target value (e.g. tape pool size in GB), and in other lower (e.g. response time in ms). A KPI is displayed in SLS as a green or red dot, depending on whether the KPI condition is met on a given day (figure 1).
2.3. Subservices and metaservices
If a given service is composed of different parts, and users might be interested in seeing details and availabilities of these parts, its service manager can declare these parts as subservices. Subservices are displayed in SLS in the same way as regular services.

Services can be grouped together into so-called metaservices (for example: Services for physics). A metaservice is not a real service as seen by users, but rather a convenient way of grouping services and reflecting their hierarchy. Availability of a metaservice can be either calculated automatically by SLS as a weighted average of availabilities of its subservices, or provided by its manager in the same way as for regular services (see section 4 for details).

2.4. Dependencies
Each service can declare which other services it uses and depends on. For example, the CVS service at CERN uses e-mail service for notifying users about commits, but if these e-mail messages are not delivered, core functionality of the CVS service is not affected. A CASTOR (CERN hierarchical mass storage system) instance, on the other hand, depends on underlying Oracle databases, and will not work if these databases are not available.

These dependencies are displayed in SLS – so users can browse between dependent services – but are not taken into account for calculating service availabilities.

2.5. Views
A view is a set of top-level services and metaservices that is displayed on the SLS entry page. Views are freely configurable, and an SLS instance may have multiple independent views set up.

The default view in the CERN instance of SLS groups services by functionality (see figure 2). Other high level views can be selected in the drop-down list at the top of the main SLS page. At present four alternative views are available, showing:

- all IT services on one page,
- the status of services provided by the different IT groups,
- the status of Grid services at sites in the World-wide LHC Computing Grid,
- the status of services for different virtual organisations (or communities); at present this shows the state of services for the different LHC experiments – showing, perhaps, that ATLAS users are affected by an AFS problem whilst services are fine for everybody else.

3. Users’ perspective
Services in SLS are grouped and organized in hierarchies. Availabilities of various types or groups of services are visible at the first glance, as shown on the SLS entry page (figure 2). One click away is a page with a list of services belonging to the chosen group (figure 3). Another click leads to a page with information about a specific service (figure 4), like its name and description, link to its web page, contact e-mail, a list of service managers, service dependencies, a list of subservices etc. From here, one can browse through service pages, following their dependencies and hierarchy. This drill-down approach makes navigating through SLS pages simple and intuitive.

Whenever a user experiences problems with a given service, SLS should be able to indicate whether these problems are detected by the service’s monitoring system (and therefore known to its managers). SLS will report the nature and details of the problem (for example: performance downgraded due to high load) if this information is provided by the service. Users can go further and see how availability and status of a given service evolved in the last 24 hours, week or month (figure 5).
Figure 2. SLS entry page at CERN with a logical structure of services.

Figure 3. List of services in the metaservice Services for physics.
Figure 4. Details, availability information, dependencies etc. for the CASTOR Tape Service.
4. Architecture and design

4.1. Architecture

As already mentioned, SLS does not monitor services – it expects service managers to provide availability information for their services. Service managers often use existing monitoring systems for calculating or estimating availability and status of their services – e.g. SiteScope [2] for monitoring administrative web applications, or Microsoft Operations Manager (MOM) [3] for Windows services. Lemon [4] and Nagios [5] are used for other, usually Linux-based services.

Every few minutes, SLS collects from each service its current availability information, in XML format (see subsection 4.2). SLS doesn’t need to know what monitoring system was used on the service side, or how its availability was calculated. Service availability information is then stored in XML files on the SLS server, and in a database, and made available on automatically refreshed web pages. SLS also provides an RSS feed for each service, which reports service status changes. (See SLS architecture diagram on figure 6).

4.2. XML files

XML format is used by SLS for transmitting and storing service information. Static service descriptions (including names of service managers, list of subservices or dependencies etc.) are stored in static XMLs on a server (see examples in appendix A). Dynamic service information (service availability and status, availability information) is generated by each service as an update XML (see examples in appendix B) and made available to SLS at the URL provided in the corresponding static XML. For each service, SLS combines information from its static XML and update XML, and stores it in a cache XML.

All XML files are validated with XML Schemas. Supported tags and their syntax are described in the SLS Manual for Service Managers [6].

4.3. Database

SLS uses an Oracle database for storing historical data like service availabilities, numeric values and KPIs. These values can be displayed as charts in the SLS web interface. When a database is not available (or not configured), SLS is still able to collect, store (in XML files) and display current status of registered services. 

Figure 5. Availability of CASTOR Service in the last 24 hours.
5. SLS installations

5.1. SLS at CERN
SLS is in production at CERN since June 2006. More than 350 different computing services provided by IT Department, or in collaboration with other departments (such as the EDMS Service), have their status reported to, and displayed by SLS. Services covered include:

- services for physics (e.g. CASTOR mass storage system, LXPLUS, Tape Service),
- LCG Tier 0 and 1 sites,
- administrative applications,
- Windows, mail and web services (e.g. Mail service, DFS, Terminal Services),
- document and conference applications (e.g. Indico, VRVS),
- networking infrastructure,
- infrastructure services (e.g. Backup, AFS, Kerberos, Linux, Remedy problem tracking, Lemon, SLS itself),
- services for developers and engineers (e.g. CVS, J2EE Public Service, TWiki),
- multiple databases.

5.2. ...and elsewhere
SLS is released under the EU DataGrid software license (a BSD-style license) and is available for downloading at http://cern.ch/project-sls [7]. There are no CERN-specific dependencies – SLS will work on Scientific Linux and requires only a few additional software packages. SLS was recently installed at ASGC (Academia Sinica) in Taiwan, and other sites have expressed interest in setting up their SLS instances.
6. Summary
The Service Level Status display provides a comprehensive and easy to interpret overview of the status of computing services, as measured in a way relevant to users. SLS is highly configurable – SLS administrator can define multiple top-level views, register services of various types and build their hierarchy and dependencies. Features like Key Performance Indicators, charts for displaying historical availability data, or RSS feed reporting status changes for a particular service give users additional information about services they use. SLS display proved to be useful also for IT management, service managers, and operators of the Computer Centre at CERN, as it provides a simple overview of which services work fine, and which need to be looked at.

Acknowledgements
The author wishes to thank German Cancio Melia, Tony Cass and Bill Tomlin for their input and ideas.

Appendix A. Sample static XMLs
Static XML for CASTOR service:

```xml
<?xml version="1.0" encoding="utf-8"?>
<service xmlns="http://cern.ch/SLS/XML/static">
  <id>CASTOR</id>
  <type>metaservice</type>
  <fullname>CASTOR Service</fullname>
  <servicemanagers>
    <servicemanager main="true">John Smith</servicemanager>
    <servicemanager>James Bond</servicemanager>
  </servicemanagers>
  <datasource>
    <url downifnoupdate="false">http://cern.ch/castor/updateXML</url>
  </datasource>
  <site>CERN</site>
  <group>IT/FIO</group>
  <email>Castor.Support@cern.ch</email>
  <webpage>http://cern.ch/castor</webpage>
  <subservices>
    <subservice weight="2">CASTORALICE</subservice>
    <subservice weight="2">CASTORATLAS</subservice>
    <subservice weight="2">CASTORCMS</subservice>
    <subservice weight="2">CASTORLHCB</subservice>
    <subservice weight="1">CASTORPUBLIC</subservice>
    <subservice weight="0.3">CASTORITDC</subservice>
  </subservices>
  <availabilitydesc>
    Our service availability is determined by an algorithm that tries to reflect the current response experienced by a user.
  </availabilitydesc>
</service>
```
Static XML for CoffeeMachine service:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<service xmlns="http://sls.cern.ch/SLS/XML/static">
  <id>CoffeeMachine</id>
  <fullname>Coffee Machine in the main building</fullname>
  <shortname>Coffee Machine</shortname>
  <email>help@coffee.com</email>
  <webpage>http://www.coffee.com</webpage>
  <alarmpage>http://www.coffee.com/status.php?machine=4351</alarmpage>
  <dependencies>
    <dependency level="dependson">water</dependency>
    <dependency level="dependson">electricity</dependency>
    <dependency level="dependson">coffee</dependency>
    <dependency level="uses">sugar</dependency>
  </dependencies>
  <datasource>
    <url>http://www.coffee.com/UpdateXML.php?machine=4351</url>
  </datasource>
</service>
```

Appendix B. Sample update XMLs

Update XML for CASTOR service:

```xml
<?xml version="1.0" encoding="utf-8"?>
<serviceupdate xmlns="http://sls.cern.ch/SLS/XML/update">
  <id>CASTOR</id>
  <availability>100</availability>
  <data>
    <numericvalue name="Active transfers">17</numericvalue>
    <numericvalue name="Queued transfers">0</numericvalue>
    <numericvalue name="Files to be migrated">676</numericvalue>
    <numericvalue name="Files to be recalled">0</numericvalue>
    <textvalue>Storage Class: disk0tape1</textvalue>
  </data>
  <timestamp>2007-09-01T14:31:41+02:00</timestamp>
</serviceupdate>
```

Update XML for CoffeeMachine service:

```xml
<?xml version="1.0" encoding="utf-8"?>
<serviceupdate xmlns="http://sls.cern.ch/SLS/XML/update">
  <id>CoffeeMachine</id>
  <availability>80</availability>
  <availabilityinfo>Cappuchino not available</availabilityinfo>
  <timestamp>2007-08-05T18:15:43+02:00</timestamp>
</serviceupdate>
```

References
[1] TVScreen service description (http://tvscreen.cern.ch/service.html)
[2] Mercury SiteScope (http://www.mercury.com/us/products/business-availability-center/sitescope)
[3] Microsoft Operations Manager (http://www.microsoft.com/mom)
[4] Lemon – CERN monitoring system (http://cern.ch/lemon)
[5] Nagios – service and network monitoring program (http://www.nagios.org)
[6] SLS Manual for Service Managers (https://twiki.cern.ch/twiki/bin/view/FIOgroup/SLSManualForSM)
[7] SLS project webpage (http://cern.ch/project-sls)