



Open Science Grid

Document Name	Edge Services Framework for OSG
Version	OSG document 167 v1
Date last updated	June 19th, 2005
Authors	Kate Keahey, Tim Freeman (ANL)
	Abhishek Rana, Matthew Norman, Frank Würthwein (U. California San Diego)
	Rob Gardner (U. Chicago)

Abstract. We discuss the program of work towards creating an Edge Services Framework for deployment as part of the OSG 0.4 release.

Introduction.....	2
1.1 Dynamically deployed Edge Services.....	2
1.2 Initial Requirements and Specifications.....	4
1.3 Example Services to be considered for OSG 0.4.....	5
2 Schedule and Milestones.....	6
3 Assumptions.....	7

Introduction

The purpose of this document is to describe a program of work that leads to deployment of a framework for VO specific Edge Services as part of OSG 0.4.

Resources today are often configured so that all resources within a site are available only via a private network. The site as a whole can be accessed through a limited number of public addresses via which an external client can communicate with specialized services acting on the client's behalf within the site's private boundary. Such services are sometimes referred to as "gatekeepers", Compute Element (CE), Storage Element (SE) and many other less polite names. Since these services are executing at the edge of this public/private network boundary we will call them Edge Services.

Given differing requirements of different VOs, we envision a scenario in which resources will be partitioned into specific, VO-dedicated servers along side shared, open grid services used by many VOs. Each VO may want to use different software to implement any particular kind of an edge service, and each VO may put different requirements on edge service in terms of resource usage. Given these considerations, it is important that edge services have two properties:

- 1) They are easy ways to install them, preferably based on a pre-configured package. For example, Atlas and CMS may decide on different VO-specific software they may want to run and neither should be obliged to use the other's configuration. The system should also allow for easy ways of upgrading this VO-specific software: if an upgrade is needed (or needs to be rolled back) that happens seamlessly
- 2) We need a way of guaranteeing dedicated use of resources allocated to the edge services such as memory, CPU or networking. In the first iteration, we will simply operate on dedicated slots for Atlas and CMS, but will be interested in advance reservations later.

In the following we elaborate on concepts, schedule, and workplan for an Edge Services Framework to be deployed in OSG 0.4. This is in accordance with the roadmap for US LHC services on OSG.ⁱ

1.1 Dynamically deployed Edge Services

The majority of OSG clusters deploy their compute nodes on private networks. Only CE and SE straddle the public/private network boundary. For example, on a CMS site all IO operations are internal, all input to applications come from the local site (no traffic from wide area network) and all output is saved on the local site. For example, data is stored in the local SE and then registered for inter-site transfer with a transfer agent (an edge service itself).

It would provide significant flexibility to the VOs if gateway resources could be scheduled in such a way that resources like CPU, disk, port ranges, etc. are guaranteed. The Privilege projectⁱⁱ infrastructure could then be used to implement controlled deployment and upgrade of VO specific services (for

example by allowing only individuals with certain attributes/roles to deploy those services). Operationally this may be done initially via multi-month lease of a piece of hardware to ATLAS at a CMS site, and vice versa. In the future, such leases would be much more dynamic, and accessible via grid access services to many VOs in OSG.

We propose formation of an “edge services activity” in OSG with the goal of developing an edge services framework based on virtual machine technology, XENⁱⁱⁱ, that will allow dynamic deployment of VO specific workspaces (implemented via virtual machines) as VO specific edge servers. Initial example services include Condor-C^{iv} deployment at the site, including batch slot harvesting, as well as FroNTier^v DB cache based on squid technology. The layout of these services with those deployed in OSG are sketched in Figure 1. Future examples might include ATLAS and CMS specific data transfer agents, among others.

Timeline for this activity should be to have an initial set of services ready for deployment with OSG 0.4 in late 2005. We believe however that preliminary milestones along the way, demonstrating feasibility, partial deployment test, and exposing difficult aspects of the approach, should be identified and pursued early on.

At any rate, we expect both US ATLAS and US CMS will require partitioning of VO-dedicated CE and SE resources from shared, common OSG gateways until such services mature and become reliable.

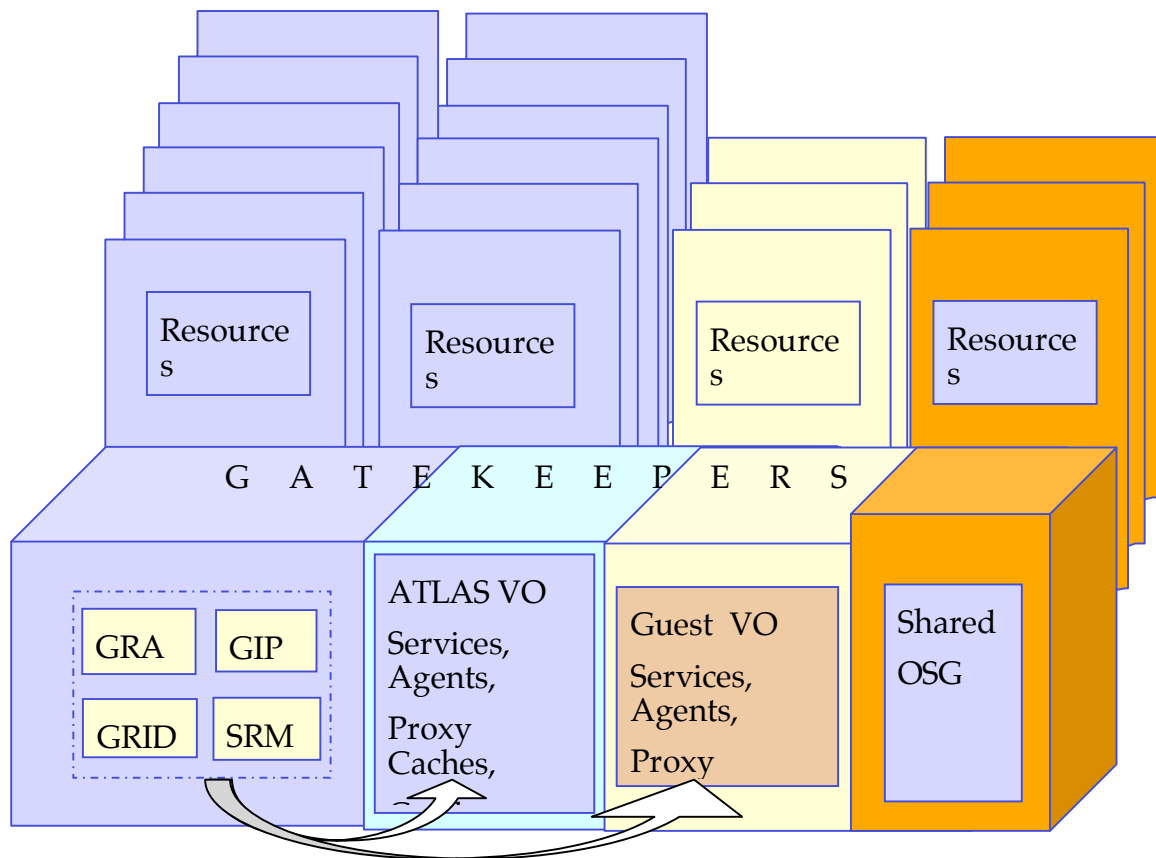


Figure 1 Site Architecture illustrating deployment of "Edge Services" for persistent, VO specific services and agents.

1.2 Initial Requirements and Specifications

In the following, we summarize the basic features of the Edge Services Framework.

- Service has Internal and External network connectivity with iptables arrangements that are determined by the site (or software controlled by the site), rather than the service itself.
- Service has guaranteed hardware resources, including well defined amount of disk space, memory, CPU power, and port ranges both on the internal and external networks. All of these characteristics are defined by the hosting site. At least initially, these will be determined by configuration and prior offline agreement rather than systems such as advance reservation. If this pilot project is successful we may invest in advance reservations next year.
- Service is dynamically deployed using the Virtual Workspace (VW) Manager^{vi} which supports all the standard GT4^{vii} authentication and

- authorization methods. In addition, the VW Manager also offers support for VOMS credentials which is important as the deployment is based on specialized roles. Initially, we will define a single “ESdeployer” role which will suffice for all edge services a VO may want. Later, a specific role might mean the right to deploy only specific workspaces configured to run edge services of choice. Roles are also tied to information about how much resource can be allocated to a specific VM (this will be treated more extensively in future work). It is also important that we support an authorization callout to SAML^{viii}.
- (d) A deployed service can be torn down dynamically, and another deployed in its place via the same mechanisms as dynamic deployment of the service in first place.
 - (e) The existing Edge Service “slot” is advertised via a standard OSG mechanism. This includes advertisement of the hardware characteristics mentioned in (b) above. Initially, there shall be no time-out of the lease of the Edge Service slot. Termination of the lease agreement thus requires human intervention, at least initially.
 - (f) Deployment goal for OSG 0.4 are one Edge Service slot per site for ATLAS and CMS each, at each of the ATLAS and CMS Tier-2 sites. Both VOs are free to pick the edge service (CE, SE, NFS,...) they want to deploy.
 - (g) ATLAS and CMS can change both the configuration as well as type of their Edge Service without consultation of the hosting site. This would be done simply by shutting down one workspace and deploying another one with support for another edge service.
 - (h) Service is able to discover its hardware characteristics after deployment, and configure itself appropriately. Initially this probably refers only to things like port ranges, mount points, and whatever else is essential for operations of the service.
 - (i) Service registers itself with Discovery Service after it has successfully been deployed and configured. Part of the configuration is a self-test of some sort.
 - (j) All Edge Services are user level installs. No root access is granted via the Edge Services framework, i.e., in a workspace no root privilege is given to clients.

In addition to these requirements and specifications, we assume that ATLAS and CMS sites will prepare a hardware resource that is dedicated to the Edge Services Framework, and is pre-configured with a hypervisor installation.

1.3 Example Services to be considered for OSG 0.4

In the following we describe example services to be developed and deployed. An example workspace will be provided as a test followed by the development of other workspaces.

- (a) An example edge service, supporting the deployment of GT4 GRAM will be provided as a test case.
- (b) VO specific CE based on Condor-C. This implies outgoing network connectivity via some port (range? Can this be set arbitrarily in Condor-C?), as well as ingoing connectivity. The service instantiates a schedd, negotiator, collector, and glide-in manager. The glide-in manager submits glide-ins via the site's GT4 GRAM to the sites batch system, thus requiring a "service certificate" of some sort. The glide-in announce themselves to the collector after starting up on the site's batch system, and are fed with jobs from the schedd based on the negotiator's matchmaking. All of the above is done within the same X509 certificate context.
- (c) Deployment of a FroNTier based squid cache. This may not go beyond osg-itb^{ix} deployment.

2 Schedule and Milestones

In the present section we provide an initial set of milestones. We consider this very much a work in progress. It is meant to seed the discussion.

- (a) Dynamic deployment of a virtual machine (VM) into a site's edge slot such that it can discover its port ranges, start up a "Hello World" server, advertise this service to the Discovery Service after going through a selftest. Verify the service by connecting to it both from the inside and outside of the site. You'd get back something like "Hello - you have reached my outside" or the inside equivalent. This will include the role based deployment as well as a service principal that allows submission of a job to the site in order to do the inside and outside testing. In other words, at this point the service would be able to perform a self-test and advertise to via the OSG Discovery Service.
- (b) Preparation of Condor-C infrastructure excluding the glide-in manager. Just bring up a couple of static batch slots (vm) with the Condor-C infrastructure inside a virtual machine VM, i.e. a few vm's inside a VM. Don't sweat the dynamic deployment. Just make it work at all such that it can receive and run a job via condor-c on a vm inside the VM.
- (c) Figure out how to connect standard condor vm's that are statically deployed to the Condor-C infrastructure deployed inside the VM in (b).
- (d) Preparation of a glide-in manager that works. This is outside the scope of the Edge Services framework, and will be done by CMS as part of the late binding work. We can exercise this without GT4 GRAM initially.
- (e) Test submission from the VM into the compute nodes.
- (f) Put (a) to (d) together as one fully functional service, excluding GT4 GRAM for convenience. This does not yet address the late binding security issues on the compute node. I.e. this is still a toy.

- (g) Resolve the late binding security issues. This is outside the scope of the Edge Services work and will be pursued within OSG as part of the late binding activity.
- (h) Get the whole system ready for osg-int deployment, with readiness plan and the whole works.
- (i) Role it out in time for OSG 0.4 deployment: the level of code readiness will be “as much as we can do in the time available”. In other words, it may be a tech preview or alpha level infrastructure. Though the goal is to have reliability and scalability features at least as good as the existing pre-webservices GRAM.

3 Acknowledgements

We thank the organizers and participants of the two OSG workshops “US LHC Technology Roadmap Meeting”^x in Madison and “OSG Application workshop”^{xi} at SLAC. It is fair to say that this document would not exist in its present form without the discussions at these two workshops.

In addition, we’d like to thank Ian Fisk for his suggestions upon carefully reading the initial draft.

ⁱ See OSG document 137 for details.

ⁱⁱ See OSG document 134 “OSG Authorization Infrastructure” for more details.

ⁱⁱⁱ <http://www.cl.cam.ac.uk/Research/SRG/netos/xen/>

^{iv} For details on Condor-C see the Condor manual , e.g. Section 5.4 in release 6.7.8, or OSG document 145.

^v S. Kosyakov, et al. “FroNTier: High Performance Database Access Using Standard Web Components in a Scalable Multi-Tier Architecture”, in Proceedings of Computing in High Energy Physics (CHEP) 2004, Interlakken, Switzerland.

^{vi} For additional details see OSG document 151.

^{vii} For details on GT4, and the globus toolkit in general, see: <http://www.globus.org/toolkit/>

^{viii} P. Hallam-Baker, E. Maler, et al, “Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML), Oasis Standard, November 5th, 2002

^{ix} <http://osg.ivdgl.org/twiki/bin/view/Integration/WebHome>

^x <https://www.opensciencegrid.org/events/meetings/osgcouncil0505/>

^{xi} <https://www.opensciencegrid.org/events/meetings/slac0605/>