TITLE: Web Services and AAI: PAPI in the OSIRIS platform

Ajay Daryanani Arjandas
RedIRIS / Red.es. Edificio Bronce, Plaza de Manuel Gomez Moreno, s/n. 28020 Madrid, Spain
e-mail: ajay.daryanani@rediris.es

and

Diego R. Lopez, Candido Rodriguez, Daniel Garcia
RedIRIS / Red.es. CICA. Avda. Reina Mercedes, s/n. 41012 Sevilla, Spain
e-mail: {drlopez, candido.rodriguez, daniel.garcia}@rediris.es

Abstract
As the tendency to use Service Oriented Architectures (SOA) within Internet applications is a fact, security providers cannot remain indifferent to this evolution and have to offer solutions to the new necessities. That means, in the AAI (Authentication and Authorization Infrastructure) world, to evolve web-based protocols into web services based scenarios. This paper shows the present and future of PAPI within the ITEA OSIRIS project.

Keywords
Web Services, Authentication, PAPI, Liberty Alliance, OSIRIS

1. Introduction

The use of Web Services (WS) as an interaction method has become widespread. Requirements such as adaptability, reliability, flexibility, composition and collaboration are very well met by these systems, leading the industry into the adoption of WS-based mechanisms as the core of their applications. The fact that market leading companies are involved in the development of WS-related standards (OASIS [1]) and, in the case which we're dealing with, in specifications defining identity management and federation protocols (Liberty Alliance [2]), lays the groundwork for these kind of technologies.

In this context, the OSIRIS [3] project (Open Source Infrastructure for Runtime Integration of Services), funded by ITEA[4], was launched in 2005 with the main objective of building an across-domain open source service platform that will provide support for services provisioning, aggregation, delivery, dynamic adaptation to the context and lifecycle management. Being security a key issue within the project, PAPI [5][6] is appointed to provide the basis to accomplish authentication, federation, accounting and other security requirements across the platform.

2. OSIRIS

OSIRIS aims to provide a flexible Open Source Infrastructure for Run-time Integration of Services (OSIRIS platform) that will add value by permitting smooth integration of platforms, services and devices. Its core building blocks will be a set of middleware platforms, service composition technologies and infrastructure services.
2.1. OSIRIS scope

There is an explosion in popularity of extensible systems, the functionality of which can be extended at run-time. These are often used to implement new programming approaches (e.g. service orientation, context awareness, and autonomic computing) which promote a highly dynamic infrastructure that is not only reactive, but also proactive.

In order to satisfy future requirements, service implementations from different providers will be required, each of them with a particular combination of functional and non-functional qualities. More complex services will be assembled via the set-up or orchestration of appropriate service chains. These will be able to evolve in time, by using the dynamic management capabilities offered by the service platform. The traditional static view of services development, deployment and management is thus replaced by a smooth and integrated support of an evolutionary process.

Systems will comprise implemented services that may themselves evolve and change in a continuous process throughout the overall lifecycle. The service-oriented architecture of OSIRIS will dramatically improve system integration through dynamic application composition, upgrading and enhancement.

The open source model proposes new forms of collaboration that will enhance competitiveness through the reuse of higher-level services across domains. It will provide a framework for exploring the feasibility of domain-oriented communities as the means of supporting domain-independent open source middleware, as well as for identifying infrastructural elements and common web services through a planned series of demonstrators.

The results of this project will be, amongst others, the following:

1. A reference architecture and tool framework for open service oriented solutions.
2. Highly adaptive open source middleware platforms, suited for ambient intelligence applications.
3. A service oriented platform and associated tools, including a set of basic services for software deployment, user identification, service registry and discovery, support for services composition and dynamic configuration for run-time integration of services, and dynamic configuration and administration.

2.2. OSIRIS Service Technology Platform

The OSIRIS service technology platform comprises tools, runtimes environments, and middleware elements that will support the core technical underpinning of the OSIRIS demonstrators. The OSIRIS reference architecture is meant to serve as a conceptual framework to coherently combine and integrate OSIRIS software assets. In order to obtain the reference architecture a bottom-up process has been followed, starting from the specifics of demonstrators and then trying to extract cross cutting concerns that can be promoted to a common reference architecture.

The OSIRIS Service Technology Platform is a distributed, service-oriented middleware composed of the following elements:

- A network of OSIRIS Nodes. OSIRIS Nodes can play different roles: Service Consumer, Service Provider or Service Aggregator.
- Software Asset Repositories. Networked entities that enable deploying software packages to OSIRIS nodes. Repositories are also docking stations where software providers upload software packages.
- Service Directories. A Service Directory provides services metadata and search capabilities. Metadata can include references to Software Asset Repositories in case software packages are required before accessing a service, e.g. to enjoy VoIP service it is usually necessary to deploy software to the device.

Taking into account the heterogeneity of devices that could participate in an OSIRIS interaction (the range goes from constrained devices such as a sensor with minimum communication capabilities, to high capability servers), it has been decided to define a basic profile which all nodes should comply to, and a set of profile extensions that could optionally be integrated in the node depending on the role played in the platform.

2.2.1. OSIRIS Node

OSIRIS nodes are the core elements of the OSIRIS Service Technology Platform. An OSIRIS node is basically a device powered with a networked runtime environment. An OSIRIS node may play more than one role and thus, depending on the particular scenario, may have different needs. In order to address the diversity in which an OSIRIS node may be used, we have opted to define a basic profile (a minimum set of capabilities and technologies) that afterwards can be tailored to each scenario by means of profile extensions. Profile extensions provide specific functionalities over the basic profile.

Figure 2 shows the basic profile of an OSIRIS node. The main characteristics of an OSIRIS node are:

- It comprises an OSGi [7] framework and a set of basic extensions installed. Thus, nodes are service-oriented platforms with support for remote deployment and lifecycle management of components. This also implies that the deployment unit for OSIRIS nodes is the OSGi bundle.
- It is able to expose and access Remote Services. The Remote Services Communication component provides different remote protocols to interoperate with the OSIRIS network.
- It can be extended with Profile Extensions. A profile extension is a logical unit that adds some functionality or capability to an OSIRIS node. Profile Extensions are packaged and deployed as groups of OSGi bundles.
2.2.2. Profile extensions

OSIRIS nodes can be extended with profiles in order to provide additional system services and functionality to the runtime. Figure 3 presents some examples of profile extensions that will be needed by OSIRIS demonstrators. These profiles have been briefly described in Table 1 to clarify the concept with some examples.
<table>
<thead>
<tr>
<th>Profile Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA Security</td>
<td>Provides authentication and authorization mechanisms to the runtime environment, in both internal and external domains.</td>
</tr>
<tr>
<td>Management Agent</td>
<td>Instruments the OSIRIS node with remote management capabilities, i.e. monitoring, fault detection, configuration or operation of the platform.</td>
</tr>
<tr>
<td>Deployment Agent</td>
<td>The deployment agent is able of recursively downloading and installing those components that are necessary to extend the basic profile.</td>
</tr>
<tr>
<td>Service Discovery Agent</td>
<td>Extends the Remote Service Communication with the ability to search and discover existing services in the OSIRIS network.</td>
</tr>
<tr>
<td>Web Server</td>
<td>Deploys a web server in the node for browser-based applications. Similar extension profiles such as a Servlet container can also be considered.</td>
</tr>
<tr>
<td>Persistence Engine</td>
<td>Provides capabilities to store, search and load objects in a permanent system so they can be maintained upon reboots of the OSIRIS node.</td>
</tr>
</tbody>
</table>

Table 1. Description of some Profile Extensions

2.2.3. OSGi

The OSGi service platform is a component-based, service oriented environment initially targeted at the embedded devices market. The OSGi specification was born in the context of service provisioning to the residential domain, but its versatility has made of it a suitable platform for other environments. The OSGi Release 4 is used in a wide range of domains such as the automotive market, as the execution engine of new generation mobile phones and even into development tools (the latest Eclipse version runs over an OSGi framework). To ensure compatibility with different hardware and software configurations, the OSGi Service Platform is based on Java TM technology.

The service platform provides a computing environment suitable for the deployment and execution of applications that can be managed from anywhere in the network. OSGi components are libraries and applications which can be installed, updated and removed on the fly from the platform without disrupting its normal operation.

2.2.4. Remote services communication

OSGi provides an excellent SOA support for local services, but lacks of mechanisms to find and access remote services running on other nodes, or to expose its services to other nodes. With the purpose of providing these capabilities, another layer of communications will be added to the basic OSGi framework, obtaining OSIRIS nodes.

The services running on OSIRIS nodes should be discoverable and accessible from Internet through standard protocols. But not all the OSIRIS nodes have enough execution capabilities to support a complete service container with all the needed functionalities. To solve this problem, this box is divided in two pieces aiming different domains:

- **LAN domain layer:** covers communications within a LAN (house, SOHO, etc.). This layer will be referred as “OSIRIS framework”, due to the characteristics that define it.
- **Internet domain layer:** cover communications through Internet, with firewalls or other obstacles in the channel. This layer will be referred as “Internet connector”.

The OSIRIS framework provides functionalities to communicate OSIRIS nodes in a lightweight way inside trusted environments (home or office networks for example), that allows introducing limited devices as OSIRIS nodes. The Internet connector provides mechanisms to expose and access services through different standard protocols like Web Services or CORBA, and takes the role of a proxy for those OSIRIS nodes that do not have these capabilities.

**2.3. OSIRIS node instantiation**

An OSIRIS node can access and can be accessed in Internet through standard protocols, so these mechanisms should be present in some way. An OSGi node with the Internet connector fulfils this requirement, but an OSGi node with the OSIRIS framework does not provide the needed functionality itself. To consider that an OSGi node with the OSIRIS framework supports a complete OSIRIS node, at least other OSGi node with support to the OSIRIS framework and the Internet connector should be present in the LAN.

So, there are three implementations of an OSIRIS node:

- **Single nodes.** Contains only the Internet connector box, and are nodes that manage all his communications through Internet with standard protocols, mainly Web Services.
- **Proxy nodes.** Contains both boxes, OSIRIS framework and Internet connector, and could act as intermediaries. These nodes will be present in LAN in which other nodes need their capabilities.
- **Dependent nodes.** Contains the OSIRIS framework, but depends on other nodes in the LAN supporting the Internet protocols. So, dependent nodes only can be considered as completed OSIRIS nodes if a Proxy node is present on the LAN, or at least they are connected some times to this LAN (mobile devices for example could not be present all the time).

Based on these instantiations, Figure 5 provides an overall picture of the OSIRIS scenario.
3. Security in OSIRIS

The main requirements that need to be tackled are securing user access (human or machine), component interaction in runtime and software deployment. Therefore security must not cancel the runtime integration of services, which is a core objective of the project. Seamless integration of AA features is another challenge to be achieved.

To simplify the architecture, security inside OSIRIS domains (a set of OSIRIS nodes inside the same Local Area Network) is not considered as relevant. This decision is supported on the following facts:

- An OSIRIS domain will typically belong to the same person or a small set of people such as a family. A common example of an OSIRIS domain would be a SoHo (Small Office / Home Office);
- When service consumers and providers are located in the same LAN, a trust relation between them is implicitly assumed;
- Some nodes do not have Internet communication capabilities, so they will be only directly queried from another node in the domain;

A set of OSIRIS domains will be referred as OSIRIS realm.

Regarding security software, the first results of the OSIRIS project included a state-of-the-art survey on candidate technologies for the platform. In this context, PAPI and openPMI [8] [9] where selected to cover security requirements. Further studies have appointed Liberty Alliance ID-WSF [10] as the way of communicating with non-OSIRIS services.

Due to the unavailability of open source implementations regarding Liberty Alliance ID-WSF specifications, RedIRIS has joined openLiberty [11]. Among other issues, OSIRIS will contribute with real life use cases to openLiberty, which could drive its design; on the other hand, a good knowledge of
ID-WSF profiles and source code will serve as an input to OSIRIS. Collaboration between this group and OSIRIS partners will increase visibility of both projects.

### 3.1. Security components and interactions

Taking a look at the architecture, it is clear that the main interaction component will be the proxy node, responsible of exposing domain-level services to the outer world, and also will act as an entry point to access services provided by constrained devices. Placing security services seems reasonable; with respect to figure 5, a proxy node will have the minimum set of components shown in Figure 6.

![Figure 6. Proxy node](image)

Also the overall scenario changes when referring to security interactions. Figure 7 shows these interactions in the new scenario.
Most of the components in Figure 7 are the same as in Figure 5, except the proxy node (yellow triangle) which is explained in Figure 6; new items appear in the legend of Figure 7, and have the following description:

- **WS-PAPI protocol**: For intra-realm security services, a WS-based PAPI protocol will be used. Although PAPI has its own set of operations, there will be an effort to map it to a Liberty ID-WSF profile if possible; otherwise, a new Liberty profile can be defined and forwarded to the Liberty Alliance as an extension to the standard;

- **GPoA – Liberty WSC/WSP**: It is a component of the PAPI protocol. It will act as:
  - A trust aggregator for inter-realm communication. When authentication/authorization is not valid at the proxy nodes, they will forward the request to the trust agent, which will validate the user or redirect the user to his/her Identity Provider;
  - When the consumer/provider of the service belongs to another realm, this component will take the role of a Liberty WSC (Web Service Client) or a WSP (Web Service Provider)

- **Liberty protocol**: Communications with non-OSIRIS realms will be based on standard Liberty ID-WSF profiles;

This diagram has been designed taking into account: the variety of components participating in the platform; the need for lightweight procedures, small footprints and open and seamless management (that support the use of PAPI); and, as a result of the heterogeneity (in service/identity consumers/providers) and distributed functionality of the platform, the election of the Liberty Alliance profiles as a means to communicate and federate.
3.2. Deployment of security

Security will be provided by means of profile extensions, that will eventually be deployed in the OSIRIS proxy node.

In order to protect services exported by OSIRIS proxy nodes, security extensions will interact with the Internet Connector component, which is the one in charge of exposing domain services. The security extension intercepts service requests to the Internet Connector. Service requests will not be forwarded to the Internet connector unless service consumers are appropriately authorized. We envision two feasible alternatives for enforcing this mechanism:

- Use AOP (Aspect Oriented Programming) to weave the Internet Connector code with security code. For this purpose, candidate technologies could be Spring-OSGi framework or AOSGi (Aspect OSGi, a project hosted by the Eclipse Equinox project);
- Use Servlet filters. PAPI already has an application filter framework that can be integrated with the Internet Connector component;

This approach is transparent for service providers, but we are still exploring how to provide something equivalent for service consumers.

4. Expected results

The main results expected from our work in the OSIRIS project are the following:

- A web service based implementation of the PAPI protocol, applicable to non-web profiles of eduGAIN [12].
- Awareness of PAPI, eduGAIN and other GÉANT2 projects will increase in OSIRIS partners, other ITEA2 projects and, in general, the industrial environment.
- An open source implementation of some of the Liberty Alliance ID-WSF profiles. RedIRIS intends to contribute with these implementations to the OpenLiberty project.
- Bundled security services for OSGi environments, together with a better understanding of the OSGi standards applicability to the research and academic networks environment.

5. Author Biographies

Ajay Daryanani Arjandas works as a Middleware Engineer at RedIRIS since January 2005, after getting his degree in Computer Science at Universidad Complutense de Madrid. Besides, at the same school where he graduated he's carrying out his PhD, mainly focused on AAIs.

Diego R. Lopez is the responsible for the Middleware Area of RedIRIS. He graduated in Physics at the University of Granada in 1985, and earned his PhD degree at the University of Seville in 2001. In 1985, he joined the Conformance Testing Division of Telefonica I+D, working in several projects related to e-mail and directory services. From 1992 to 2000 was member of the technical staff of CICA, acting as the responsible for network services. In January 2000 he joined RedIRIS. Among his responsibilities, he is the leader of the AAI work item within GN2 Roaming and Authorization activity (JRA5).

Candido Rodriguez Montes is doing a PhD. in Computer Science at the University of Seville, Spain. In 2004 he got a university degree in Computer Science at the University of Seville. From 2001 to 2005 he worked for Optima Technologies, acting as the responsible for R& D Department, involved in the development of security products. In 2004, he started to work with RedIRIS, having a master thesis about AAIs.

Daniel Garcia Franco is a Computer Science student at University of Seville, where he got a BSc in Computer Science in 2004. From 2002 to 2004 he worked for Sadiel making system deployment for its customers. In 2004 he joined RedIRIS for the pkIRISGrid.
6. References


http://www.projectliberty.org/index.php/liberty/about

http://www.itea-office.org/public/project_leaflets/OSIRIS_profile_oct-05.pdf

[4] ITEA2 Office, Information Technologies for European Advancement,
http://www.itea-office.org/


http://www.rediris.es/jt/jt2006/archivo/16Jueves/1130-1400/A/OpenPMI.ppt

http://openpmi.sourceforge.net/

http://www.projectliberty.org/liberty/content/download/889/6243/file/liberty-idwsf-overview-v2.0.pdf

[11] openLiberty project
http://www.openliberty.org