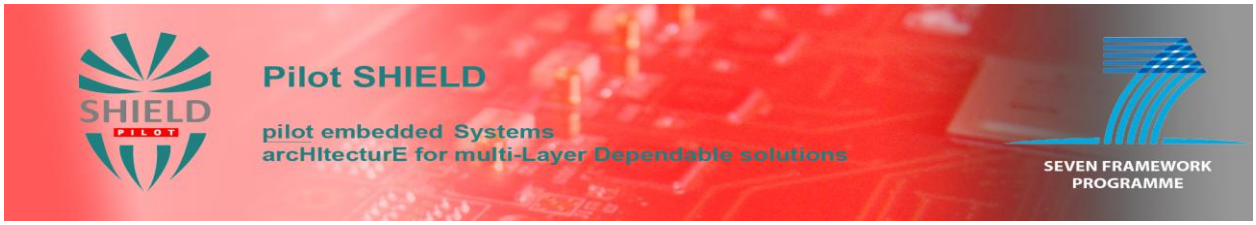


Project no: 100204

p-SHIELD

pilot embedded Systems architecture for multi-Layer Dependable solutions

Instrument type: Capability Project

Priority name: Embedded Systems (including RAILWAYS)

D5.2: SPD middleware and overlay functionalities prototype

Due date of deliverable: M15 (30th August 2011)
Actual submission date: M15 (15th September 2011)

Start date of project: 1st June 2010 Duration: 19 months

Organisation name of lead contractor for this deliverable: UNIROMA1

Revision [Version 1.0]

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2012)		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	X
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Authors and Approvals			
Authors		**Date**	**Signature**
Name	**Company**		
Andrea Fiaschetti	Univ. “La Sapienza”		
Andrea Morgagni	Elsag Datamat		
Renato Baldelli	Elsag Datamat		
Jose verissimo	CS		
Mohammad Chowdhury	CWIN		
Andrea Taglialatela	TRS		
Vincenzo Suraci	Univ. “La Sapienza”		
Andi Palo	Univ. “La Sapienza”		
Spase Drakul	Thyia		
Mohammad Chowdhury	CWIN		
Reviewed by			
Name	**Company**		
Approved by			
Name	**Company**		
Andrea Morgagni	Elsag Datamat		

Modification History		
Issue	**Date (DD/MM//YY)**	**Description**
Draft A	15/03/2011	First issue for comments
Draft B	30/06/2011	Second issue for comments
Version 1	15/09/2011	First version

Contents

Figures

Tables

Acronyms

Acronym	Meaning
ESD	Embedded System Device
ESs	Embedded Systems
L-ESD	Legacy Embedded System Device
MS	Middleware Service
MwA	Middleware Adapter
NC	Node Capability
NoA	Node Adapter
NS	Network Service
NwA	Network Adapter
pS-ESD	pSHIELD Embedded System Device
pS-MS	pSHIELD Middleware Service
pS-OS	pSHIELD Overlay Service
pS-P	pSHIELD Subsystem
pS-P	pSHIELD Proxy
pS-SPD-ESD	SPD Embedded System Device
SPD	Security Privacy Dependability
CC	Common Criteria
FUA	Faults with Unauthorized Access
HMF	Human-Made Faults
NFUA	Not Faults with Unauthorized Access
NHMF	Nonhuman-Made Faults
SoC	System on Chip

- This page intentionally left blank -

1 Executive Summary

The purpose of this document is to present the prototypes developed in WP5 with respect to:

- the pSHIELD Middleware and its core SPD services,
- the Policy Based Management
- the Overlay architecture and control algorithms.

Particular attention has been devoted to the most innovative aspect: the composability functionality performed by the Middleware and Overlay, that is the key enabling technology for the pSHIELD framework. This concept is depicted in Figure 1.1 as a closed loop problem and the WP5 activities covered by this deliverable and necessary to enable it are highlighted (Tasks 5.2, 5.3 and 5.4 as described in the Technical Annex).

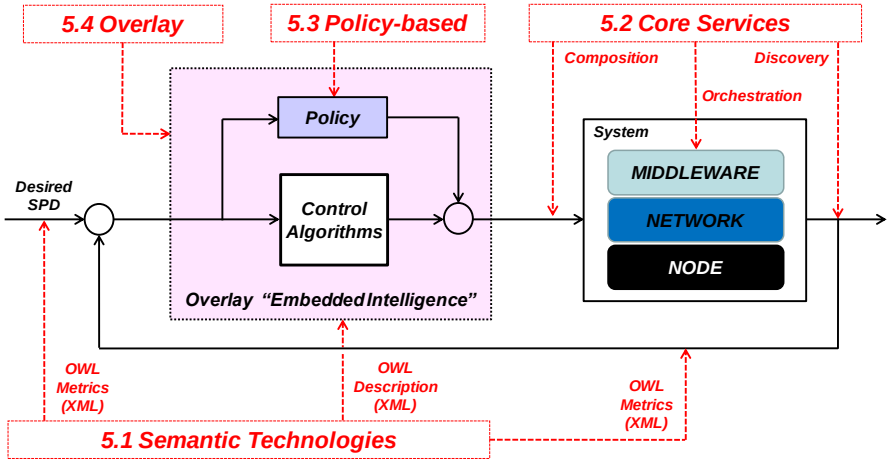


Figure 1.1 WP5 Composability Concept

Since pSHIELD is a pilot project, the solutions proposed in this document don't pretend to be exhaustive and optimal, but their purpose is to realize a proof of concept of the pSHIELD key concepts in a reduced (but significant) scope and in a simple scenario, putting the bases for further improvements and investigations.

The nature of this deliverable is "other", to indicate the heterogeneity of the output, that can be software code, architectural design, simulations, diagrams and so on. This material is provided as attachment of the document.

D5.2, enriched with background analysis, state of the art and performance considerations carried out in the D5.4 report, cover all the work carried out by WP5 partners.

The document is structured as follows: after the punctualization of terms and definitions in Section 2, in Section 3 the prototype of pSHIELD Middleware and core SPD services (based on the OSGi framework) is presented. Then in Section 4 the rationale and architectural prototype of potential implementation of Policy Based Management in pSHIELD environment is reported. Last, but not least, in Section 5 the Overlay architecture and behaviour is described, with particular attention to the underlying control theory supported by some simulations as prototype. Finally, in the Annexes the source code of the presented prototypes is included.

2 Terms and definitions

This section lists the applicable documents

Ref	Document Title	Issue/Date
TA	pSHIELD Technical Annex	1
M0.1	Formalized Conceptual Models of the Key pSHIELD Concepts	1
M0.2	Proposal for the aggregation of SPD metrics during composition	1
D2.1.1	pSHIELD Systems requirements and specifications	1
D2.2.1	pSHIELD metrics definition	1
D5.1	pSHIELD Semantic Models	1

2.1 SPD Dictionary

A comprehensive dictionary of the SPD concepts is provided by the project glossary included in Annex 1.

3 pSHIELD Middleware and core SPD Services prototype

3.1 From concept to architecture

The core SPD services are a set of mandatory basic SPD functionalities provided by a pSHIELD Middleware Adapter in terms of pSHIELD enabling middleware services. The core SPD services aim to provide a SPD middleware environment to actuate the decisions taken by the pSHIELD Overlay and to monitor the Node, Network and Middleware SPD functionalities of the Embedded System Devices under the pSHIELD Middleware Adapter control. The following core SPD services are provided:

- service discovery;
- service composition;
- service orchestration.

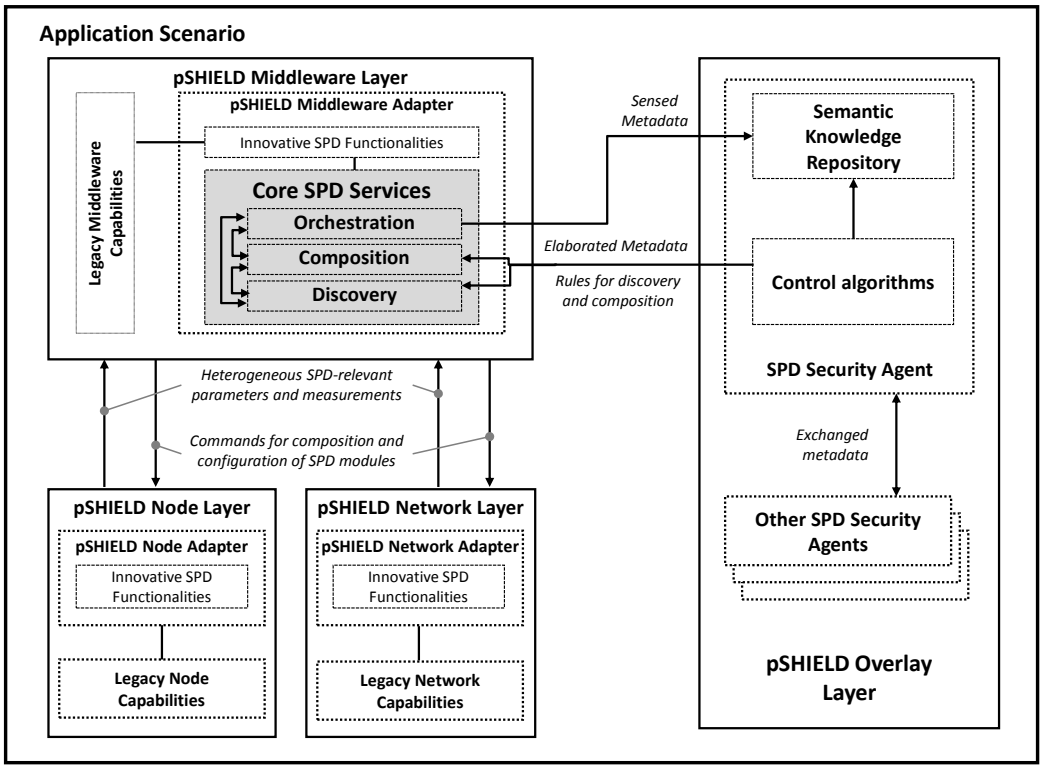


Figure 3.1 Core SPD services in the pSHIELD functional component architecture

Service discovery allows any pSHIELD Middleware Adapter to discover the available SPD functionalities and services over heterogeneous environment, networks and technologies that are achievable by the pSHIELD Embedded System Device (pS-ESD) where it is running. Indeed the pSHIELD secure service discovery uses a variety of discovery protocols (such as SLP[1], SSDP[2], NDP[3], DNS[4], SDP[5], UDDI[6]) to harvest over the interconnected Embedded System Devices (ESDs) all the available SPD services, functionalities, resources and information that can be composed to improve the SPD level of the whole system. In order to properly work, a discovery process must tackle also a secure and dependable service registration, service description and service filtering. The service registration consist in advertising in a

[1] IETF Service Location Protocol V2 - http://www.ietf.org/rfc/rfc2608.txt
[2] UPnP Simple Service Discovery Protocol - http://upnp.org/sdcps-and-certification/standards/
[3] IETF Neighbour Discovery Protocol - http://tools.ietf.org/html/rfc4861
[4] IETF Domain Name Specification - http://www.ietf.org/rfc/rfc1035.txt
[5] Bluetooth Service Discovery Protocol
[6] OASIS Universal Description Discovery and Integration - http://www.uddi.org/pubs/uddi_v3.htm

secure and trusted manner the available SPD services. The advertisement of each service is represented by its formal description and it is known in literature as service description. The registered services are discovered whenever their description matches with the query associated to the discovery process, the matching process is also known in literature as service filtering. On the light of the above a SPD services discovery framework is needed as a core SPD functionality of a pSHIELD Middleware Adapter. Once the available SPD services have been discovered, they must be prepared to be executed, assuring that the dependencies and all the services preconditions are validated. In order to manage this phase, a service composition process is needed.

Service composition is in charge to select those atomic SPD services that, once composed, provide a complex and integrated SPD functionality that is essential to guarantee the required SPD level. The service composition is a pSHIELD Middleware Adapter functionality that cooperates with the pSHIELD Overlay in order to apply the configuration strategy decided by the Control Algorithms residing in the pSHIELD Security Agent. While the Overlay works on a technology independent fashion composing the best configuration of aggregated SPD functionalities, the service composition takes into account more technology dependent SPD functionalities at Node, Network and Middleware layers. If the Overlay decides that a specific SPD configuration of the SPD services must executed, on the basis of the services' description, capabilities and requirements, the service composition process ensures that all the dependencies, configuration and pre-conditions associated to that service are validated in order to make all the atomic SPD services to work properly once composed.

Service orchestration is in charge to deploy, execute and continuously monitor those SPD services which have been discovered and composed. This is part of the pSHIELD Middleware Adapter functionality. While service composition works "off-line" triggered by an event or by the pSHIELD Overlay, service orchestration works "on-line" and is continuously operating in background to monitor the SPD status of the running services.
The Orchestration, Composition and Discovery functionalities are the enablers (i.e. the sensors and the actuators) of the decisions taken by the pSHIELD Security Agent Control Algorithms residing in the pSHIELD Overlay. The mutual interoperation between the pSHIELD Middleware Adapter and the pSHIELD Security Agent enables the pSHIELD Composability concept.
It is worth to note that not all the core SPD services must be necessarily located in each pSHIELD Embedded System Device (pS-ESD). Indeed the pSHIELD component architecture depicted in Figure 3.1 identifies the Discovery, Composition and Orchestration functionalities that must be supported by at least one pS-ESD in a network of Embedded System Devices. Moreover the core SPD services can be deployed applying centralized or distributed approaches. It is a matter of the precise application scenario to decide whether a specific functionality must be supported by each Embedded System Device (ESD). It is obvious that the more ESDs are equipped with the pSHIELD Middleware Adapter (resulting to be a pS-ESD), the more will be the coverage area and the effectiveness of the pSHIELD functionalities to guarantee a certifiable SPD level (based on common shared SPD metrics) over the whole system.

Let see more in detail the formalized conceptual model of the Core SPD services, detailing the architecture depicted in Figure 3.1 and exploding the core SPD services into their functional components, in compliance with the pSHIELD functional architecture described in deliverable D2.3.1 .

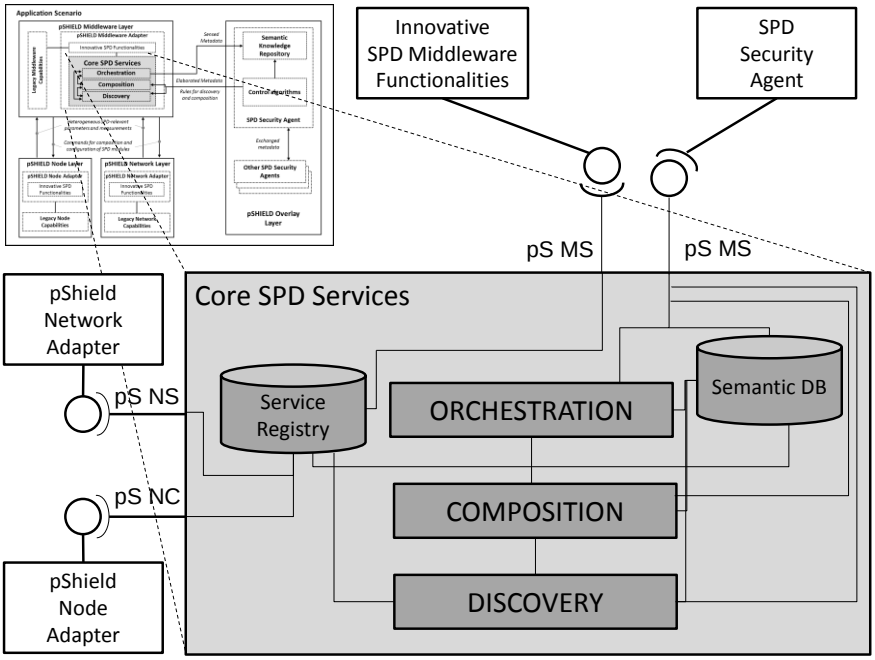


Figure 3.2 Core SPD services conceptual framework

Apart the Discovery, Composition and Orchestration components already described in the previous section, the following additional conceptual entities have been introduced:

- **Service Registry**: it acts as a database to store the service entries (e.g. the SPD components description of provided functionalities, interfaces, semantic references, etc.). Any pSHIELD Node, Network or Middleware layer component can be registered here to be discovered.
- **Semantic Database**: it holds any semantic information related to the pSHIELD components (interface, contract, SPD status, context, etc.). The use of common SPD metrics and of a shared ontology to describe the different SPD aspects involved in guaranteeing a precise level of SPD, allows to dominate the intrinsic heterogeneity of the SPD components. Any semantic data is thus technology neutral and it is used to interface with the technology independent mechanisms applied by the pSHIELD Overlay.

Focusing exclusively on the Core SPD services located in the pSHIELD Middleware Adapter, we can describe how it works when it is in an operative status. Let consider a typical situation, where the whole system is properly working at runtime. The Orchestration functionality is in charge to monitor continuously the Semantic DB with the updated status of the functionalities operating at node, network and middleware layers. The pSHIELD Adapters are in charge to update in the Semantic DB (by proper Semantic hubs that, for the sake of simplicity, have not been shown in Figure 3.2) their status.
Whenever the needed application SPD level, for any reason, due to external/internal unforeseen/ predictable events, changes and goes beyond the threshold, the Orchestrator triggers the Overlay. The Overlay try to react and to restore the SPD level back to an acceptable level identifying the best configuration rules. The Discovery and Composition are then triggered by the pSHIELD Overlay with the aim to apply the configuration rules. On the basis of the configuration rules, the Composition service make use of the Discovery service to search for all the needed and available SPD components. In particular the Composition uses the Discovery mechanism to look for the available SPD component interfaces and contracts over the network. Then the Composition, on the basis of the configuration rules provided by the Overlay, determines which SPD components are required, which should be activated and in which order to make the configuration of SPD components properly work. Thus while the Overlay operate in a technology independent fashion, the Composition service operates all the needed low-level, technology-dependent activities to actuate the Overlay decisions.
From the above description, it is clear that a key role is played by the Discovery mechanism. In pSHIELD the main focus of the Discovery is to look for any available, composable SPD component over the network. In order to cope with this important functionality the Discovery core SPD service must be decoupled into several elementary elements each deputed to a proper functionality.

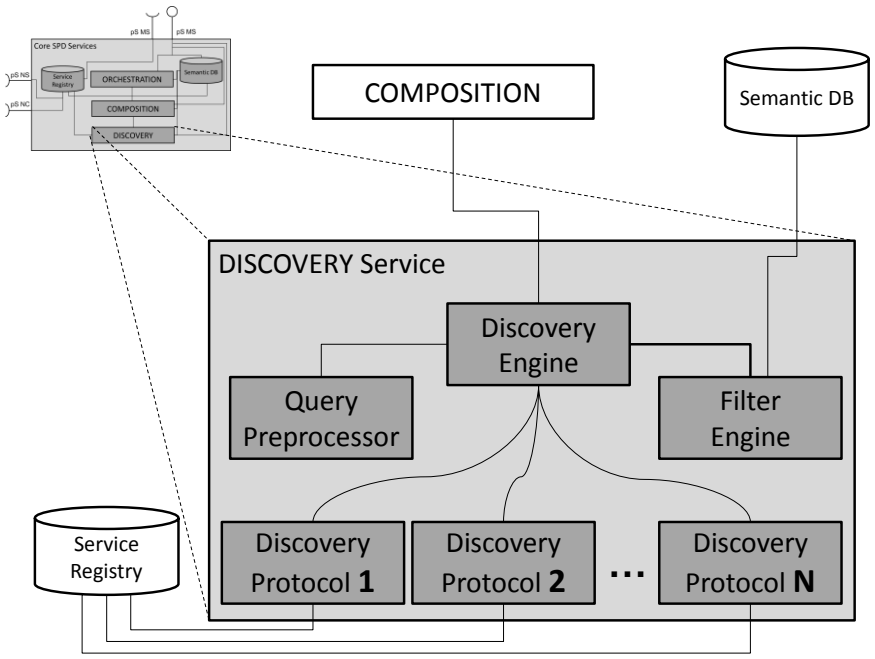


Figure 3.3 Details of the Discovery core SPD service

Zooming more in the detail the Discovery service, as shown in Figure 3.3, the following elements can be distinguished:

- **Discovery Engine**: it is in charge to handle the queries to search for available pSHIELD components sent by the Composition service. The Discovery Engine manages the whole discovery process and activates the different functionalities of the Discovery service: (i) the query pre-processor to enrich semantically and contextually the query, (ii) the different discovery protocols to harvest over the interconnected systems all the available SPD components, (iii) the Filter Engine to discard those components not matching with the enriched query;
- **Query Pre-processor:** it is in charge to enrich the query sent by the Composition service with semantic information related to the peculiar context. The query pre-processor can be configured by the Overlay to take care of the current environmental situation;
- **Discovery Protocol:** it is in charge to securely discover all the available SPD components description stored in the Service Registry, using a specific protocol (e.g. Service Location Protocol – SLP or Universal Plug and Play Simple Service Discovery Protocol – UPnP SSDP, etc.). Indeed the SPD component descriptions can be registered in different types of Service Registries, located everywhere in the network, using heterogeneous protocols to be inquired;
- **Filter Engine:** it is in charge to semantically match the query with the descriptions of the discovered SPD components. In order to perform the semantic filtering, the Filter Engine can retrieve from the Semantic DB the information associated to the SPD components, whose location is reported in the description of the SPD component.

The Composition engine tries to accomplish the pSHIELD Overlay configuration rules applying the following procedure:

1. Composition service triggers the Discovery service, sending a SPD component request, looking for those SPD components defined in the configuration rules provided by the Overlay;
2. The Discovery Engine sends the request to the Query Preprocessor;
3. The Query Preprocessor enriches the service request with contextual information and sends it back to the Discovery Engine;
4. The Discovery Engine applies a global service discovery using heterogeneous Discovery Protocols, in order to collect as much available SPD functionalities as possible over the networked Embedded System Devices;
5. Each Discovery Protocol interacts with the Service Registries reachable in the network and retrieves the SPD components' descriptions and provides them back to the Discovery Engine;

6. The Discovery Engine collects the discovered descriptions and sends them to the Filter Engine;
7. The Filter Engine applies a semantic filtering, retrieving the semantic metadata from the semantic DB, accordingly with the references reported in each SPD component description. The filtered list of component is then sent back to the Discovery Engine;
8. The Discovery Engine sends the list of available, filtered SPD components to the Composition service;
9. If the Composition service, considering the available SPD components is able to provide a new configuration, these components are activated, otherwise the Composition service advise the Overlay that it is not possible to apply its decision.

It is important to note that the validity of this conceptual framework model is independent from the specific application scenario. On the basis of this conceptual framework it is possible to derive a number of possible alternative implementations, belonging to different pSHIELD compliant technology providers. If the interfaces and the operation between the different elements are respected, it is possible to setup heterogeneous systems with the enhanced pSHIELD SPD functionalities.
However being pSHIELD a pilot project, a targeted demonstrator will be described in the following. Starting from the conceptual framework depicted above, an instance of the presented framework will be derived in order to achieve the target SPD objectives defined in the pSHIELD application scenario.

3.2 The Innovative SPD Functionalities

The core SPD services introduced in the previous section are in charge to discover, compose and orchestrate those Innovative SPD Functionalities provided by any specific application scenario. An Innovative SPD Functionality can be developed from scratch or can be developed starting from an already existing legacy SPD functionality. In order to make any legacy SPD functionality to be an Innovative SPD Functionality, it must be discoverable, composable and orchestrable.
The legacy device components, i.e. the SPD functionalities already present in the legacy devices, can be classified in Node, Network and Middleware legacy device components according to whether they have been included in a legacy Node, Network or Middleware layer.

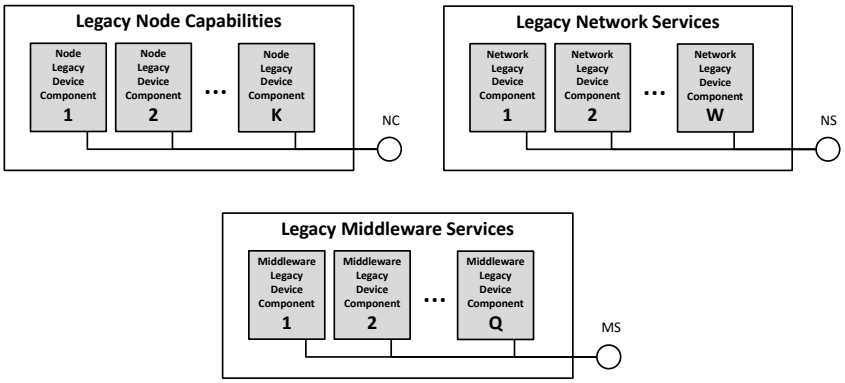


Figure 3.4 Legacy device components

The legacy functionalities interact with the surrounding world through proprietary interfaces (namely the NC, NS and MS interfaces). To cope with this heterogeneity, it is necessary to introduce an intermediate component between the legacy world and the pSHIELD framework. The role of mediator is played by the pSHIELD Adapter.
The pSHIELD Adapters hosts those Innovative SPD Functionalities that have been ad hoc developed for the pSHIELD project to let the legacy functionalities to be correctly used by the pSHIELD framework. The pSHIELD Adapter allows the legacy functionalities to be discovered, composed and orchestrated. The pSHIELD Adapters can be classified in Node, Network and Middleware pSHIELD-specific components according to whether they are included in the pSHIELD Node, Network or Middleware layer. The Adapters can be directly accessed by pSHIELD Middleware Core SPD Services through the pS-NC, pS-NS and pS-MS interfaces.

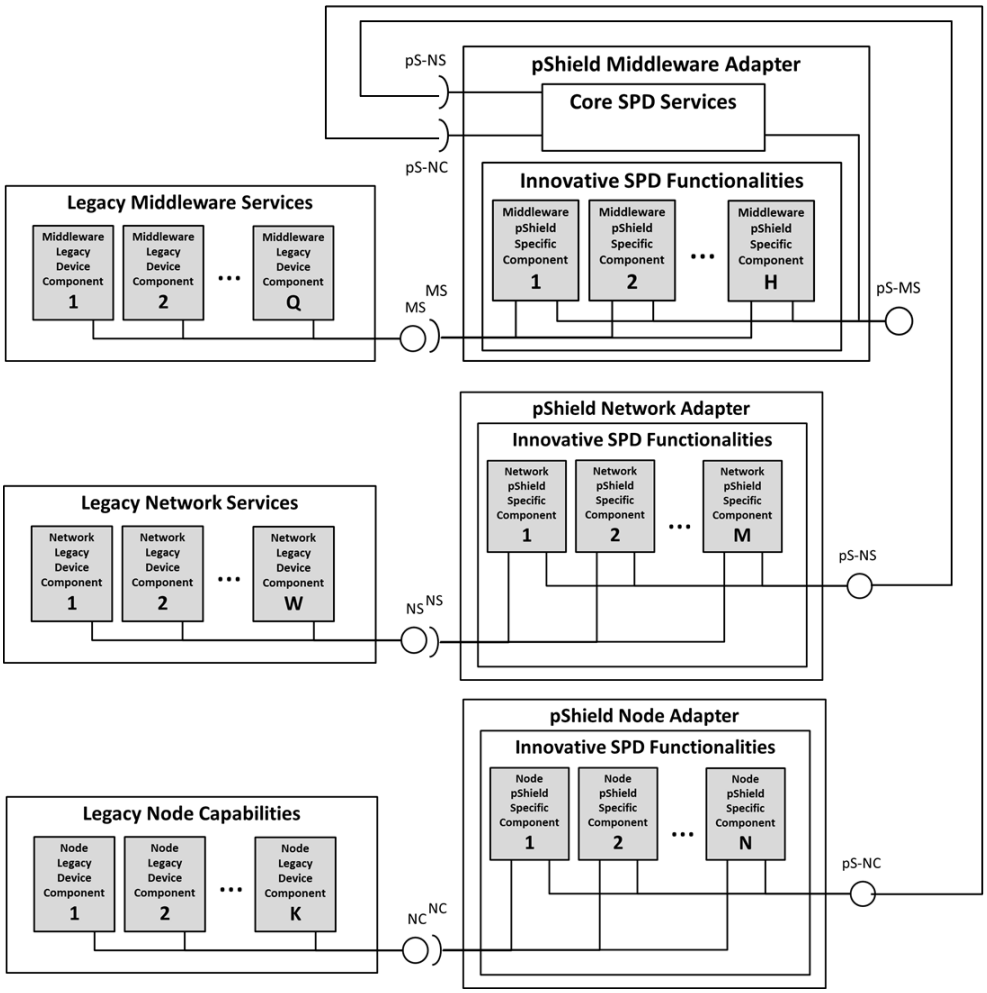


Figure 3.5 pSHIELD Components' interactions

Any Innovative SPD Functionality must be discoverable and composable. To be discoverable, an Innovative SPD Functionality need to advertise itself. To be composable, it must be described using a semantic formalism, compliant with the pSHIELD semantic model. Thus any Innovative SPD Functionality applies for the **registration** both to the Semantic Database and to the Service Registry.

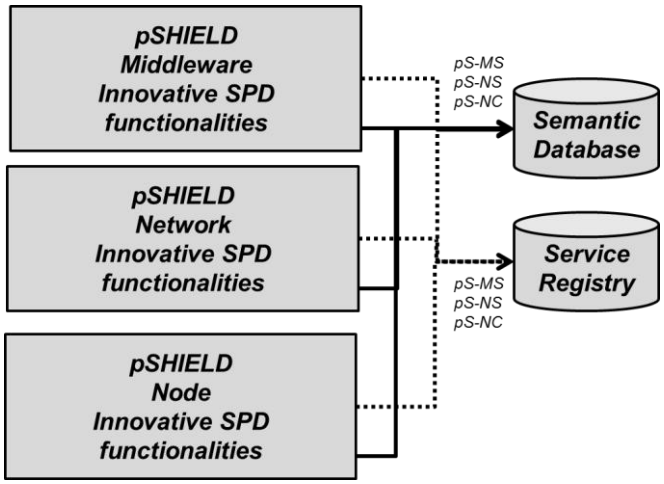


Figure 3.6 Innovative SPD Functionalities registration

3.3 Prototype objective

The pSHIELD WP5 Core SPD Services prototype aims to demonstrate the pSHIELD composability concept by means of a software implementation of the above described architecture. In order to cope with the complexity of such architecture the prototype deployment has been concentrated on the main features provided by the Core SPD Services:

- Discovery of Innovative SPD Functionalities provided by an Embedded System Device at Node, Network and Middleware discovery;
- Composition of available Innovative SPD Functionalities to guaranteed a requested level of SPD;
- Orchestration of composed Innovative SPD Functionalities to guarantee a correct operation of the whole system.

In our vision, given a proper application scenario and a desired SPD level the Core SPD Services can autonomously operate to setup a composition of available Innovative SPD Functionalities that matches the requested SPD level.
The best solution identified to develop a proof of concept demonstrator based on these requirements, and implementing the architecture described in the previous sections is to choose an open service platform based on SOA (Service Oriented Architecture). In a SOA, everything is treated as a service. Each service has its interfaces, its requirements, its dependencies, its proper dynamic and static parameters. In a SOA vision the node capabilities and the network and middleware services of an embedded system device can be described and treated as a composable service. Each sensor, each node, each protocol, each resource can be described as a service, can be discovered, composed and orchestrated to work properly. In particular we decided to exploit the potentialities of the Open Service Gateway Initiative as the reference SOA architecture to be used to develop the Core SPD Services Prototype.

3.4 The OSGi framework

Considering the possible available SOA open solutions, our decision was to select OSGi as the reference service platform to develop the proof-of-concept demonstrator. The main reasons leading to this decision are:

- OSGi is an open standard;
- OSGi has a number of open source implementation (Equinox, Oscar, Knopflerfish);
- OSGi can be executed even over lightweight nodes (Embedded Systems Devices);
- OSGi has been implemented using different programming languages (e.g. Java, C, C#);
- The Java implementations of OSGi is fast to deploy and it is much easier to learn, facilitating even an active and collaborative prototype deployment among partners;
- OSGi plugins are available for a number of IDE tools (i.e. Eclipse, Visual Studio, etc.);
- OSGi can be easily deployed in Windows (XP, 7, Mobile), Linux, MAC and Google (Android) OSes.

More in particular we decided to use the open source Knopflerfish OSGi service platform. Knopflerfish (hereafter referred as to KF) is a component-based framework for Java in which units of resources called bundles can be installed. Bundles can export services or run processes, and have their dependencies managed, such that a bundle can be expected to have its requirements managed by the container. Each bundle can also have its own internal classpath, so that it can serve as an independent unit, should that be desirable. All of this is standardized such that any valid Knopflerfish bundle can be installed in any valid OSGi container (Oscar, Equinox or any other).
Basically, running OSGi is very simple: one grabs one of the OSGi container implementations (Equinox, Felix, Knopflerfish, ProSyst, Oscar, etc.) and executes the container's boot process, much like one runs a Java EE server. Like Java EE, each container has a different startup environment and slightly different capabilities. The KF environment can be downloaded here: http://www.knopflerfish.org/
The KF start-up environment is shown below:

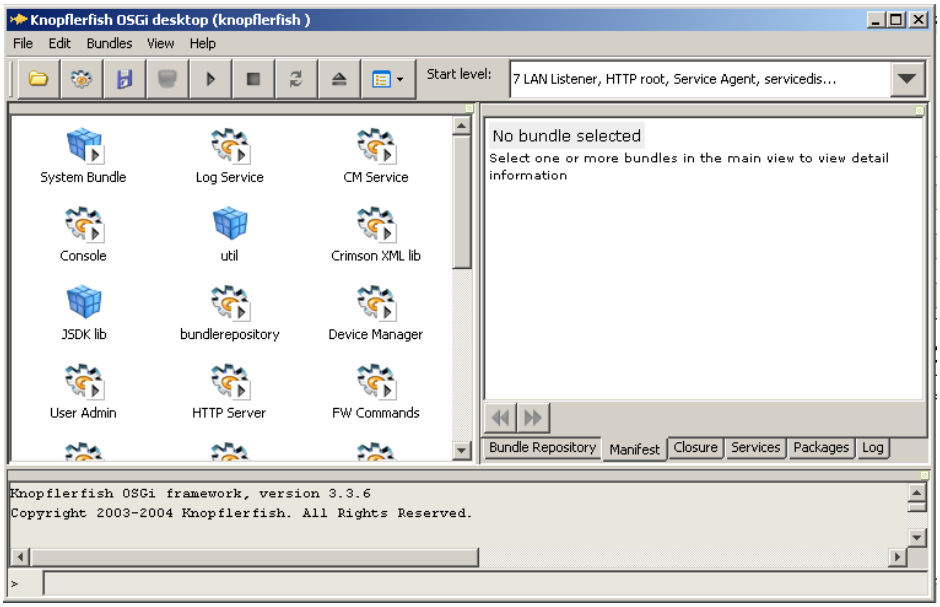


Figure 3.7 Knopflerfish start-up environment

One of the most important peculiarities of the KF OSGi is that it already offers a standard orchestration environment that, once correctly setup, can act as the pSHIELD Orchestration Core SPD Service. Thus the Orchestration functionalities comes for free when using an OSGi framework, instead of using other SOA implementations.

3.5 Prototype Architecture

The prototype architecture derives directly from the architecture described in the previous section. Each pSHIELD component is mapped into an OSGi bundle and, when needed, decoupled into a composition of interoperating bundles each providing a specific functionality. This modular approach simplify the design, development and debugging of the whole system. Even the Innovative SPD Functionalities have been implemented as OSGi bundles. Each OSGi bundle has its own dependencies, provides a set of functionalities, requires a set of functionalities and is characterized by a specific SPD level. Each bundle can be registered in the Service Registry to advertise itself, to maintain updated its status in order to be discovered. Each bundle can also store its description in the Semantic Database, to be semantically composed. Each bundle interfaces the rest of the architecture providing a set of functionalities and requiring a set of functionalities, exactly as a software component does. More in particular each bundle is decoupled into two parts: the interfacing part (API) and its implementation part (IMPL). This separation between API and IMPL ease the substitution at runtime of a specific bundle, to change from one implementation to another. This substitution can be due, as an example, to the necessity to strengthen the SPD level of a specific functionality.

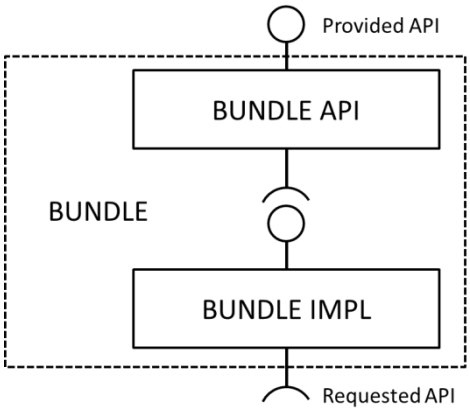


Figure 3.8 Bundle architecture

Applying for a top-down design approach, the Core SPD Services can be mapped in the following way:

- The Discovery and Composition are two separate bundles;
- The Orchestration is represented by the OSGi framework and orchestrate also the Discovery and Composition bundles.

To consider the interaction of the middleware layer with the rest of the architecture, the following additional bundles can be considered:

- The Service Registry bundle;
- The Semantic DB bundle;
- The SPD Security Agent bundle belonging to the pSHIELD Overlay layer;
- The pSHIELD Node, Network and Middleware Adapter that could be grouped into a single Adapter bundle.

The high level prototype architecture maps perfectly the Figure 3.1 Core SPD services in the pSHIELD functional component architecture, as depicted in the following figure:

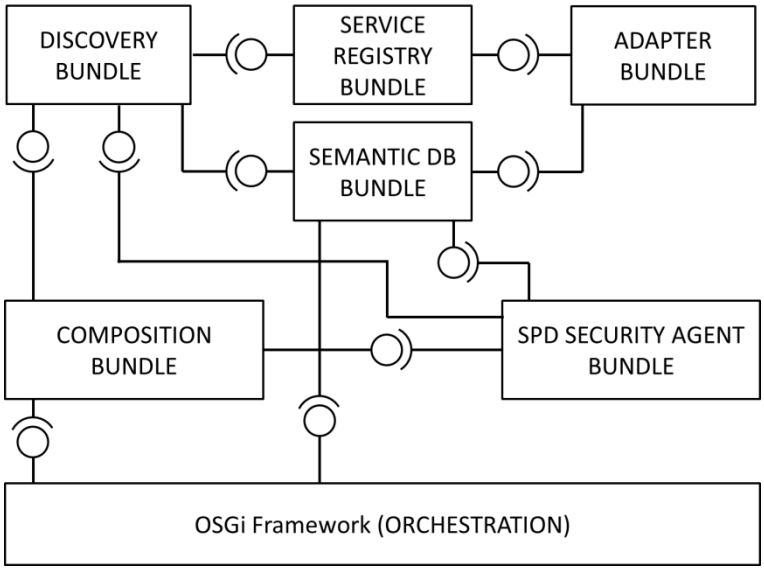


Figure 3.9 High level Core SPD Services prototype architecture

Let see in detail the structure of each bundle of the high level Core SPD Services prototype architecture.

3.5.1 Discovery Bundle

The discovery bundle structure is depicted in the following figure:

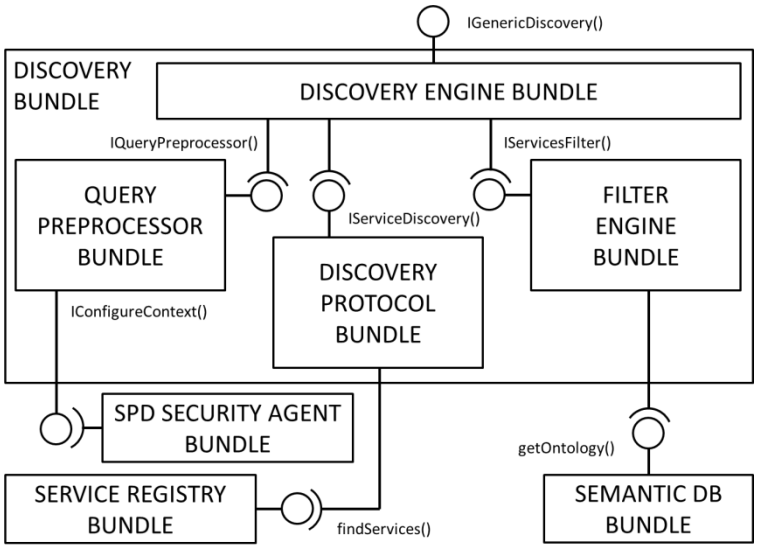


Figure 3.10 Discovery Bundle structure

As explained in the previous sections, the Discovery Bundle is composed by the following bundles:

- **Discovery Engine Bundle**: it is in charge to handle the queries coming from the *IGenericDiscovery()* interface. The Discovery Engine Bundle manages the whole discovery process and activates the different functionalities of the Discovery service. It calls the *IQueryPreprocessor()* interface to enrich semantically and contextually the query. After that the query is sent to the different underlying discovery protocols, by means of the *IServiceDiscovery()* interface, to harvest over the interconnected systems all the available SPD components. Finally the list of discovered services is sent to the Filter Engine Bundle using the *IServicesFilter()* interface to discard those components not matching with the enriched query.
- **Query Preprocessor Bundle:** it is in charge to enrich the query sent by the Discovery Engine with semantic information related to the peculiar context. The query pre-processor can be

configured by the SPD Security Agent to take care of the current environmental situation using the *IConfigureContext()* interface;

- **Discovery Protocol Bundle:** it is in charge to securely discover all the available SPD components description stored in the Service Registry Bundle, using a the *findServices()* interface;
- **Filter Engine Bundle:** it is in charge to semantically match the query with the descriptions of the discovered SPD components. In order to perform the semantic filtering, the Filter Engine can retrieve from the Semantic DB the information associated to the SPD components, by means of the *getOntology()* interface.

3.5.2 Service Registry Bundle

The Service Registry Bundle structure is depicted in the following figure:

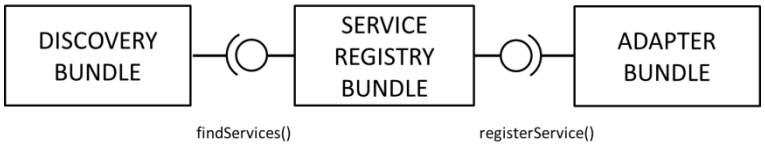


Figure 3.11 Service Registry Bundle

- **Service Registry Bundle:** it is in charge to store the bundle (i.e. SPD component) description in terms of provided functionalities, interfaces, semantic references, etc.. Any pSHIELD Node, Network or Middleware layer component can be registered here to be discovered by its own proper pSHIELD Adapter. The Adpater registers each bundle as a service, using the *registerService()* interface. The Service Registry provides the services entries information to the Discovery Bundle by means of the *findServices()* interface.

3.5.3 Adapter Bundle

The Adapter Bundle structure is depicted in the following figure:

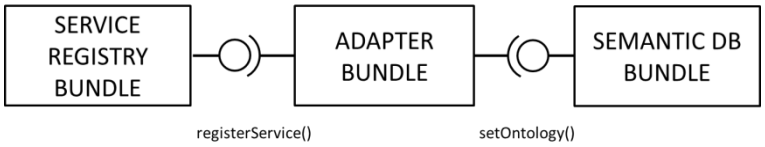


Figure 3.12 Adapter Bundle

- **Adapter Bundle**: it represents a generic (Node, Network or Middleware) pSHIELD Adapter for any type of legacy SPD functionality. The Adapter Bundle is in charge to:
  1. Provide an Innovative SPD functionality interacting with the underlying legacy services, capabilities and resources;
  2. register the provided Innovative SPD Functionality in the Service Registry using the *registerService()* interface;
  3. publish the semantic description of the Innovative SPD Functionality in the Semantic DB using the *setOntology()* interface;

3.5.4 Semantic DB Bundle

The Semantic DB Bundle structure is depicted in the following figure:

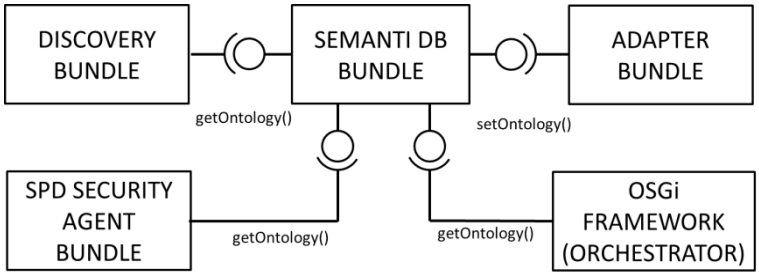


Figure 3.13 Semantic DB Bundle

- **Semantic DB Bundle**: it is in charge to store properly the semantic set by each Adapter Bundle through the *setOntology()* interface. The stored ontologies contains all the information to compose the available Innovative SPD functionalities. The Semantic DB Bundle provide access to the ontologies through the *getOntology()* interface.

3.5.5 Composition Bundle

The Composition Bundle structure is depicted in the following figure:

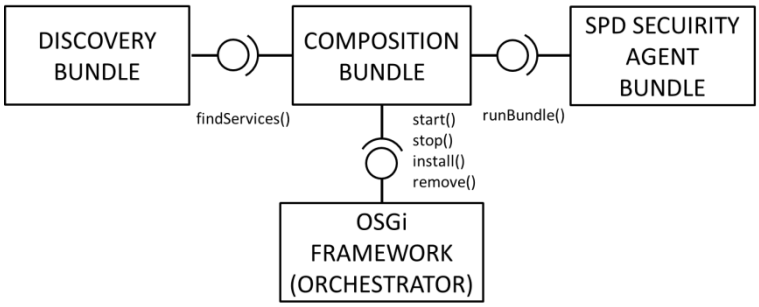


Figure 3.14 Composition Bundle

- **Composition Bundle**: it is in charge to compose the discovered bundles accordingly with the composition rules determined by the SPD Security Agent. Once the SPD Security Agent communicates through the *runBundle()* interface the necessity to run a composed functionality, the Composition Bundle use the *findServices()* interface to discover any suitable SPD component to be composed. Then the Composition Bundle compose the available bundles (taking care of the inter-bundle dependencies and the API-IMPL relationships) and uses the *start()*, *stop()*, *install()* and *remove()* interfaces provides by the Orchestrator (that is the OSGi framework itself).

3.5.6 SPD Security Agent Bundle

The SPD Security Agent Bundle structure is depicted in the following figure:

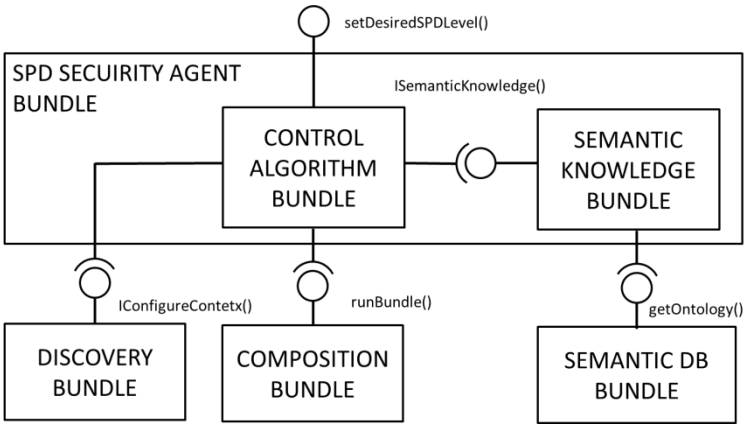


Figure 3.15 SPD Security Agent Bundle

As explained in the previous sections, the SPD Security Agent is composed by the following bundles:

- **Semantic Knowledge Bundle:** it is in charge to get the semantic description of the available services using the *getOntology()* interface and to make inference on their semantic model to extract the SPD level of their composition;
- **Control Algorithm Bundle**: it is in charge to evaluate the best control strategy for the whole system in terms of proper configuration rules both for the Discovery and the Composition Bundle, respectively through the *IConfigureContext()* and *runBundle()* interfaces. The Control Algorithm can influence which services can be discovered configuring the query preprocessor and can influence the composition process limiting the composition only to the best SPD functionalities that can assure the desired SPD level.

3.6 Deployment details

The prototype infrastructure has been deployed into a real OSGi framework. A screenshot of the OSGi control panel is reported below:

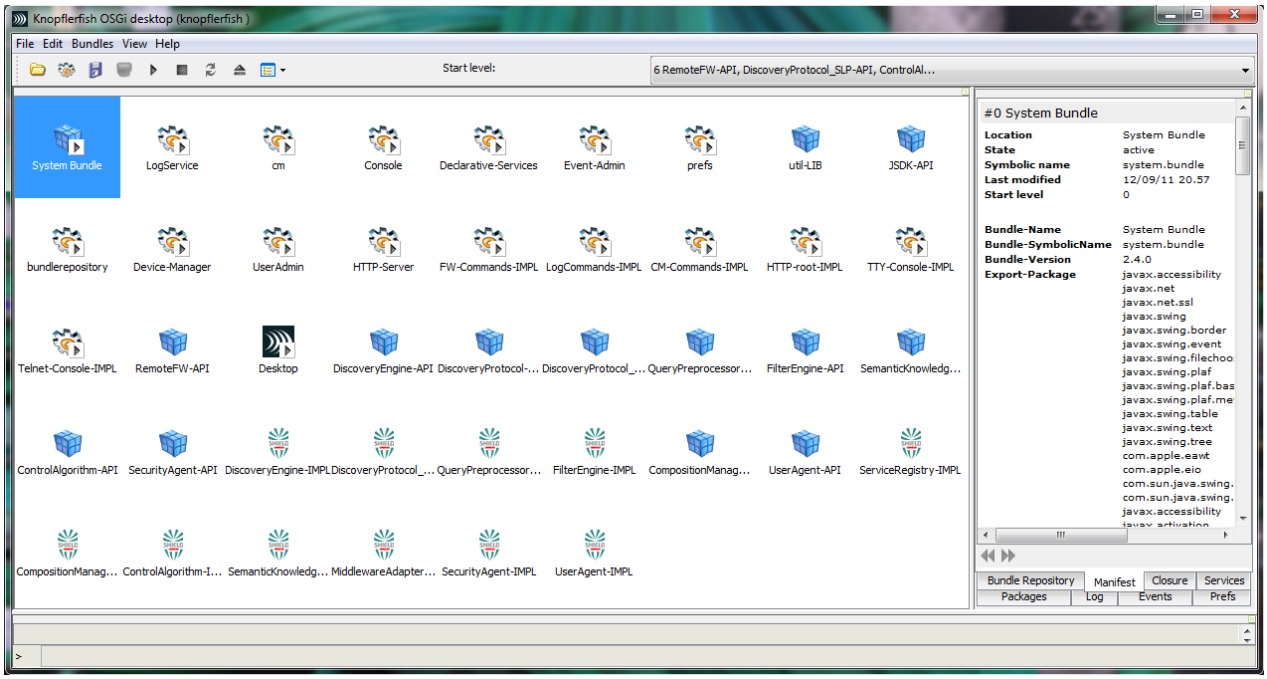


Figure 3.16 OSGI Environment

In this screenshot all the above introduced bundles are shown correctly running in a OSGi environment. The Core SPD Services prototype will be used to setup the pSHIELD pilot Demonstrator by adding proper pSHIELD Adapters representing meaningful components of the Railway Application Scenario.

In the Annex 2 the source code of the main OSGi bundles implemented for WP5 prototypes is listed.

4 Policy based management: rationale and prototype

4.1 Introduction

This section focuses on the paradigm of policy-based management (PBM) by providing an overview of the state-of-the-art and an elaboration on the mapping to pSHIELD especially to its scenario. The motivations behind the interest in PBM are clarified besides a description of what is meant by policy (its types) and PBM. The typical architecture followed in PBM will be detailed with an emphasis on policy specification means, their discussion and affiliated protocols.

4.1.1 Policy

A policy can be abstractly seen as a mapping from a set of conditions to a set of actions. Also, policies are meant to serve as the governing reference for any required adaptation a system may require. Others see a policy as in: “information which can be used to modify the behaviour of a system”(Lupu, et al., 1999). While in (Verlaenen, et al., 2007) policies are seen as a means to “configure the behaviour of services” and are described in a declarative high-level manner that identifies what should be done in a specific situation but not how it should be done. All of the above descriptions converge to the same essence that a policy is intuitively a set of rules drawn in a high-level manner where under a given evaluated situation; certain actions are dispatched for execution.

4.1.2 Policy-Based Management

Management comes as an intrinsic requirement for virtually all systems and indeed varies in type. Essentially, Policy-Based Management (PBM) can be seen as a type of system management that allows for minimal manual intervention in controlling, (re)configuring and monitoring system’s constituents and its overall behaviour through predefined governing policies. In (Lymberopoulos, et al., 2003), the objective of PBM is identified as to “allow flexible and adaptive management where the policies define the adaptation choices or strategy which can be modified without recoding or even shutting down the system.” This objective stresses on that PBM is meant mainly to provide for and regulate the adaptation a system may require at certain points throughout its lifetime. However, PBM is not meant solely for adaptation but also to aid in configuring, controlling and monitoring different system constituents (or components). Moreover, PBM allows for system adaptation without the need for changing its implementation (Verlaenen, et al., 2007). A specialisation of PBM that emphasise on managing networks and their services from a business perspective through the usage of policies is referred to as Policy-Based Network Management (PBNM) (Strassner, 2003). We mention the latter for disambiguation reasons but we continue to concentrate on PBM in this section as it encapsulates the essentials of PBNM.

4.1.3 Motivations

The rationale behind choosing PBM comes as a natural consequence of the advancement of computerised systems in general. Such systems tend to increasingly be distributed, complex, and heterogeneous operating in a dynamic surrounding environment. Assuming the efficiency of human intervention and continuous management is not realistic especially if the system is bound to alter its behaviour frequently and adapt to emerging situations. Consequently, PBM comes -and increasingly so- as an adopted simplifying approach to handling various operational, administrative and configuration matters in different computerised systems (Verma, 2002). Essentially, allowing designers to specify in a high-level descriptive manner a system’s policy is advantageous especially as they are consequently relieved from defining various and often heterogeneous lower level strategies. These strategies are inferred by the PBM which is responsible for interpreting the required steps and actions to be distributed and enforced system-wide. More importantly, it will minimize human intervention in case of system adaptation and will ensure the avoidance of conflicts among the policies to enforce. However, the use of PBM does rely heavily on the quality of identified policies and hence they need to be carefully drawn.

4.2 Typical Architecture

A typical PBM architecture is defined by the IETF policy framework (IETF, 2000). The architecture constitutes several points and elements, i.e., Policy Management Tool (including the required tools and a policy repository), Policy Decision Point and a Policy Enforcement Point. Policy specification is also considered as a main element and is discussed separately from the architecture onwards, see Section Policy Specification.

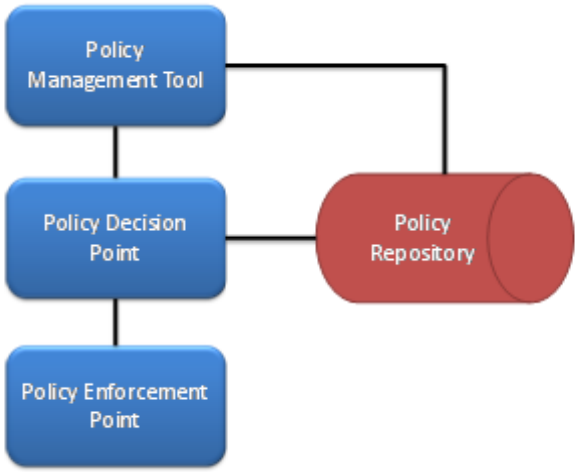


Figure 4.1 Typical IETF PBM Architecture

Figure 4.1 presents an illustration of the main points in a typical PBM.

4.2.1 Policy Management Tool

Different terminologies are used to refer to PMT such as Policy Administration Point (PAP) for instance. PMT is mainly used by the administrator(s) in order to specify business-level (high-level) abstractions that constitute polices. A number of elements are needed at this point typically (Verma, 2002; IETF, 2000):

1. A user interface that could be graphical with command-line support. Used as a policy editor with simple validation
2. A resource discovery element that determines the network topology, its capabilities, constituents, users and running applications
3. A policy translation (or transformation) element that transforms high-level policies into a lower-level constituents-specific policies. It also ensures policies' consistency, correctness and distribution feasibility through a validation process
4. A policy distributer/storage-retrieval element; as the name suggests, it interacts with the policy repository (explained onwards) to store low-level policies and allow for their retrieval

An example of policy translation as described in (IETF, 2000) would be; assume a high-level policy segment that defines *"Premium Traffic between Point A and Point B"*. This can be translated into a low-level policy rule: *"source = 10.24.195.x, dest = 10.101.227.x, any protocol, perform Premium Service action".* Indeed, a validation check can only be carried out in an offline manner here where the syntactic and semantic integrity of each policy must be preserved. Some semantic validation checks are defined by (Verma, 2002) as in:

1. Bounds checks: ensure that a given attribute value is within a predefined range
2. Relation checks: ensure that any two values assigned to interrelated policy parameters are satisfactory to their relationship
3. Consistency checks: ensure a conflict-free set of policies
4. Dominance checks: ensure that all specified policies are reachable and are to be active at some point during the system's lifetime

5. Feasibility checks: these are domain dependent checks that need to ensure that the underlying environment can support the specified policies

Moreover, while consistency checks need to ensure that conflicts among policy rules are avoided and given that this is an offline stage, other checks should be carried out at runtime to avoid potentially triggered conflicts.

4.2.1.1 Policy Repository

A policy repository is mainly concerned with managing translated policies, e.g., Directory Server, Database. It should allow for the storage, search and retrieval of policies and interface with other elements using for instance, a Lightweight Directory Access Protocol (LDAP) protocol (see Section LDAP).

4.2.2 Policy Decision Point

PDP is mainly a set of modules that are capable of examining applicable policies and consequently determine the decisions required for the system to comply with that policy. PDP is responsible for communicating policy-inferred decisions/actions to the Policy Enforcement Point (PEP) that could reside on several physical devices. That channel of communication is governed by a protocol such as SNMP (see Section SNMP). PDP also needs to interact, (i.e., fetch policies) with the policy repository using a protocol such as LDAP.

4.2.3 Policy Enforcement Point

PEP is the final point in a typical IETF policy framework architecture. PEPs act as logical entities that interface between systems' devices/resources such as sensors, where they are likely to reside, and the PDP by processing exchanged requests and responses. As the name suggests, PEP is responsible for enforcing actions communicated from the PDP at the device-level. Those actions reflect the policy or policies to be deployed at the local level.

4.2.4 Policy Types

Policies can naturally be divided into two main types, i.e., functional and on-functional policies. Functional policies are intrinsic to the system's operational tasks which is more of an application-specific issue. On the other hand, non-functional policies for instance, those meant for maintaining a certain level of fault tolerance, security and Quality of Service (QoS), tend to be less dependent on the application nature but not completely (Robben, et al., 1999).
An adapted example inspired from (Matthys, et al., 2008) illustrates how a high-level security policy is mapped to a more detailed policy for door monitoring data, see Figure 4.2. Detailed or lower-level policies may sometimes comprise functional policies such as enabling/disabling access control on a gateway.

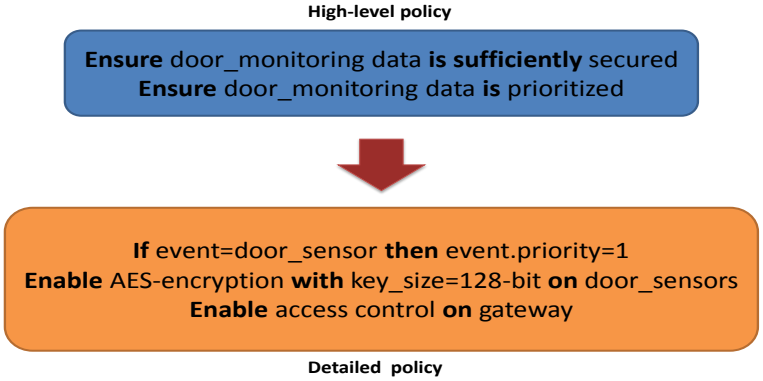


Figure 4.2 Non-Functional Policy Transformation Example (Matthys, et al., 2008)

In the aforementioned example, the need to prioritize door monitoring data results in assigning a high priority to the data originating from that door (by dedicated sensors) and that is encapsulated in a

door_sensor event. Moreover, in order to ensure sufficient security is applied on the exchanged door_monitoring data, the translation to lower-level policy specifies the encryption algorithm to be used with the adequate key size proportional to the security level requested in the high-level policy.
From a network perspective, policy-based QoS helps manage network traffic through regulating and controlling bandwidth cost (with possible negotiation with bandwidth providers) which results in a better end-user experience (Microsoft, 2009). For example, QoS policies can specify Differentiated Services Code Point (DSCP) (Cisco, 2005) values and throttling rates on routers and switches in order to balance the cost of service and the network performance. DSCP values are used to classify traffic into different levels of importance by assigning priority values to different packets, (e.g., high priority, best effort, low priority) where throttling can be used to set the rate of outbound traffic.

4.3 Policy Specification

A representative group of state-of-the-art policy specification languages/models are presented here. Policy specification is essentially the initial phase where researched policies are concretized using a model or language of choice. The latter can be regarded as an attempt to formalize administrators' intents in a machine understandable form.

4.3.1 XACML

The eXtensible Access Control Markup Language (XACML) is an XML-based declarative language designed for specifying access control policies and follows a specific processing model. Up to date, the most recent version is XACML 3.0 that was ratified by OASIS (OASIS) and includes several additional features such as delegation.

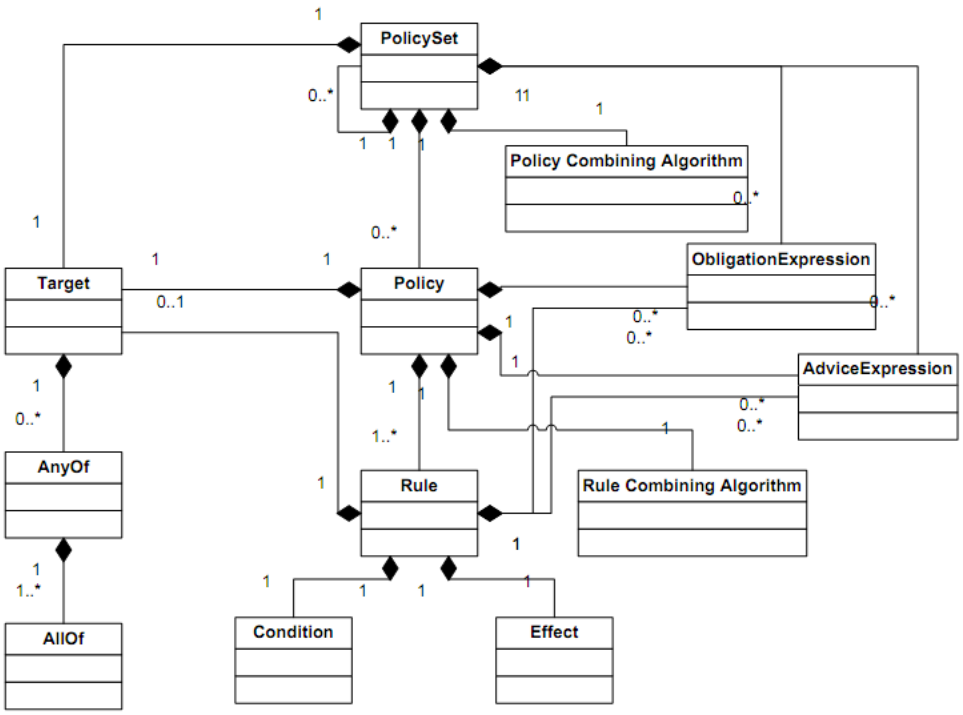


Figure 4.3 XACML Policy Language Model (OASIS, 2009)

XACML comprises a large set of abstracted elements. A comprehensive list of those elements is presented in Section 5 of (OASIS, 2009). The main policy language model used in XACML is presented in Figure 4.3 where it exhibits three essential elements/components, i.e., the *policy*, the *policySet* and the *rule*. The *rule* is the most basic constituent of the *policy* that usually encapsulates a number of rules. A given *rule* is evaluated based on its composing elements that are mainly: *target*, *effect,* (i.e., Permit/Deny), *condition,* (i.e., Boolean expression - optional), *obligation* and *advice*. Besides holding a set of rules, a policy comprises other elements such as a *target*, *rule-combining algorithm identifier*, *obligations* and *advice*. This is similar to the *policySet* in terms of constituents but with the differentiation that a *policySet* holds a set of policies and *policy-combining algorithm identifier* whereas a *policy* holds a set of *rules* and *rule-combining algorithm identifier*. Therefore, *target*, *obligations* and *advices* are relevant to policies in a given *policySet* while they are relevant to *rules* under a given *policy*. Essentially, an XACML *policySet*, *policy* or *rule* element comprises a *target* that specifies the set of requests to which it applies. For example, a *rule target* specifies the set of requests in the form of a logical expression on some or all attributes in the request. Also, a rule/policy-combining algorithm specifies the procedure through which an authorization decision can be reached based on the evaluation results of a set of rules/policies.

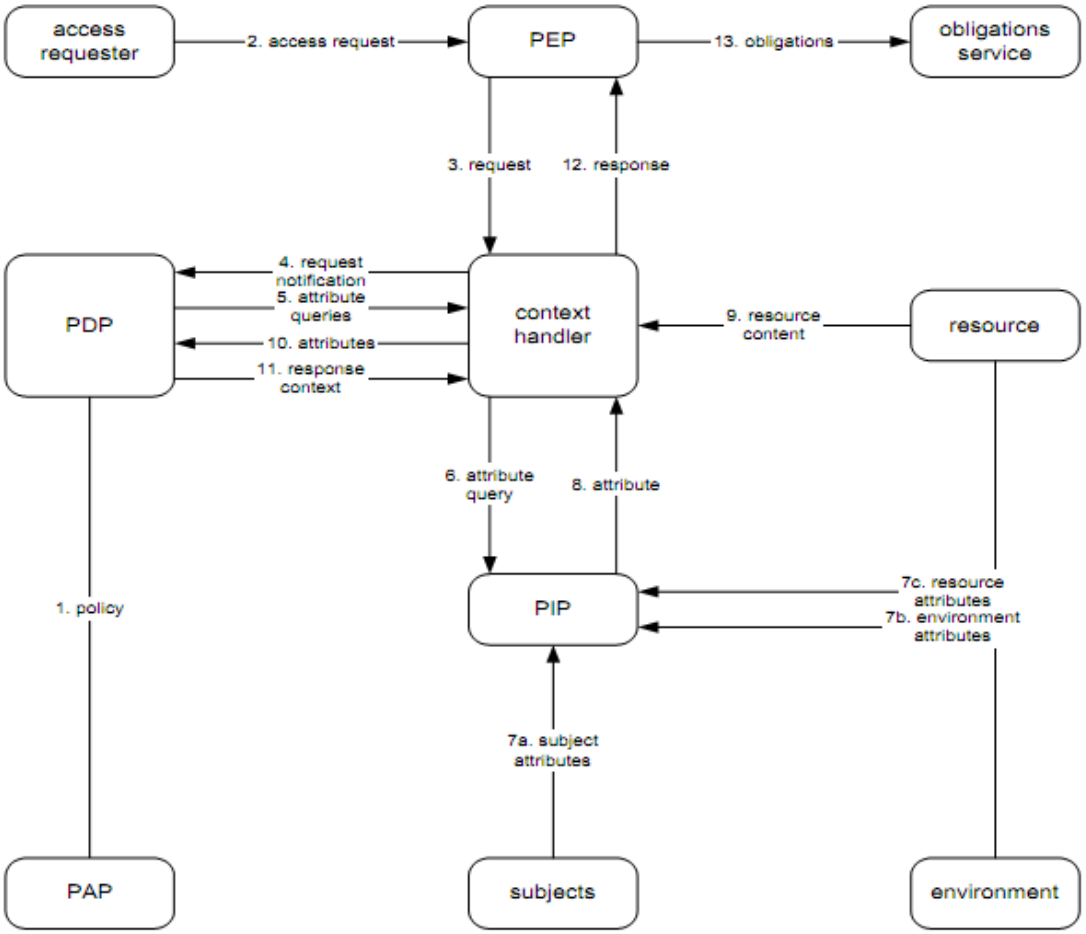


Figure 4.4 XACML Data-Flow Diagram (OASIS, 2009)

Referring to the XACML data-flow diagram (see Figure 4.4) clarifies more the connections among the different policy model elements. The architecture can also be seen to share some of the PBM typical architecture defined by the IETF that was discussed earlier. The following is a description based on (OASIS, 2009) of the entities mentioned in the XACML data-flow diagram:

- Policy Enforcement Point (PEP): perform access control by issuing decision requests and enforcing authorization decisions
- Policy Information Point (PIP): act as a source of attribute values
- Policy Administration Point (PAP): responsible for creating a policies or policy sets
- Policy Decision Point (PDP): evaluate the applicable policy and renders an authorization decision
- Context Handler:
    - Convert decision requests in the native request format to the XACML form
    - Convert XACML authorization decisions to the native response format
- Environment: “The set of attributes that are relevant to an authorization decision and are independent of a particular subject, resource or action”
- Resource: “Data, service or system component”
- Subject: “An actor whose attributes may be referenced by a predicate”
- Obligation: “An operation specified in a rule, policy or policy set that should be performed by the PEP in conjunction with the enforcement of an authorization decision”

PAP composes policies and policy sets which are made available to the PDP. The communication between the PAP and the PDP can be facilitated by a repository that XACML does not impose restrictions on its location. This can also apply to the communication between the context handler and the PIP. In a typical scenario, the access requester sends its request to the PEP which forwards it (possibly including attributes of the subjects, resource, action, environment, etc.) to the context handler. The latter converts the native request to an XACML format and sends it to the PDP that could request additional attributes

from the context handler itself. The context handler retrieves the needed attributes through the PIP and returns them to the PDP that evaluates the policy. Consequently, the PDP returns the authorization decision to the context handler which converts it to the native response format and sends it to the PEP. Finally, the PEP fulfils any obligations if present and permits or denies access to the resource.
An XACML implementation (up to version 2.0) made by SUN is available online under a reasonably flexible license (See (Sun, 2004)). Moreover, a comprehensive presentation detailing many aspects of XACML with elaborating examples is available from *Axiomatics AB* in (Gebel, 2010).

4.3.2 IBM EPAL

Enterprise Privacy Authorization Language (EPAL) (IBM, 2003) is an IBM developed (proprietary) policy specification language that is a subset of XACML and mainly aimed at the support of enterprise privacy policies. Where EPAL is close to XACML's structure and concept it however falls short in many aspects against XACML. In (Anderson, 2006), a detailed comparison between EPAL v1.2 and XACML v2.0 is presented which clearly shows how XACML is a more comprehensive and standardized access control and privacy policy specification language. For example, EPAL does not support nested and distributed policies besides the lack of digitally signed policies. Moreover, EPAL allows only one subject per access request and evaluates only first applicable rule. All of this is added to EPAL's inconsistent treatment of attributes and limitations when it comes to (hierarchical) roles. Indeed, XACML especially v3.0 provides a more complete privacy and access control policy specification solution than EPAL (Anderson, 2006).

4.3.3 WSPL

Web Services Policy Language (WSPL) (Anderson, 2004) is an XACML-based policy specification language that focuses on the web services domain. WSPL is driven by the intention of standardization. It is motivated by the several aspects the web services domain comprises that can be controlled and described using policy rules. Those aspects could involve authorization, authentication, quality of protection and service, messaging reliability, privacy and other service-specific options. A main feature in WSPL is policy negotiation and its ability to merge different policies which represent the intersection of such policies if such an intersection resulting policy is valid (see Figure 4.5 for an elaboration). Policies in WSPL can be also based on parameter (standard data-types and functions) comparison as opposed to pure equality matching. This allows for a more "fine-grained" choice of parameters (Anderson, 2004).

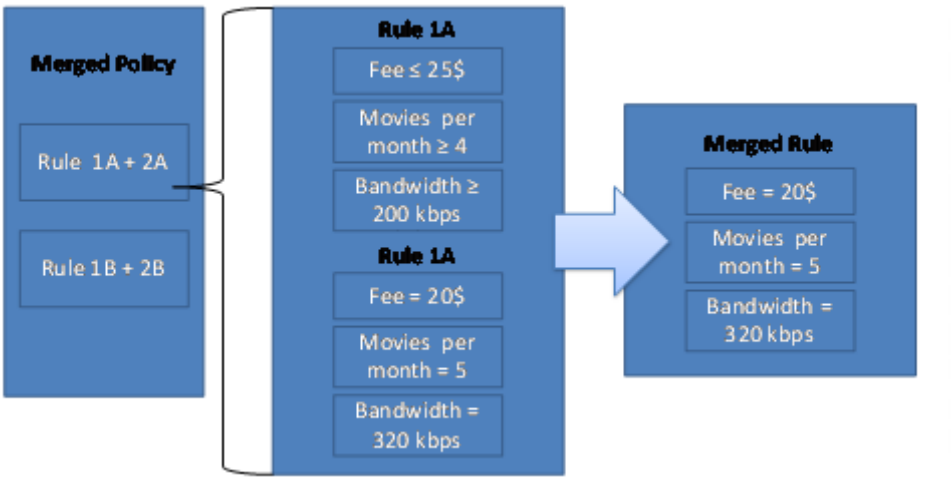


Figure 4.5 An Example of Policy Merging in WSPL (Anderson, 2004)

WSPL was mainly developed based on a number of use cases that were analysed and reviewed for that purpose in a public forum. Indeed, it was henceforth formally analysed and is seen to be "good" basis for a web services policy standard.

4.3.4 Ponder (2)

Ponder (Damianou, et al., 2001) is an Object-Oriented declarative policy specification language aimed at defining management and security policies for distributed systems. The language was developed at the Imperial College London over several years. Ponder includes essential interrelated concepts in its core, i.e., domains, roles, relationships and management structures. Domains are used to group objects of similarly applicable policies where roles are used to group policies for identified positions in a given organisation. Relationships are used to specify interactions between roles where a management structure is used to capture the configuration of roles and relationships for an organisational entity such a business unit.
The base-class diagram used in Ponder is presented in Figure 4.6. The detailed explanation of the diagram is out of scope but a comprehensive description can be found in (Damianou, et al., 2000).

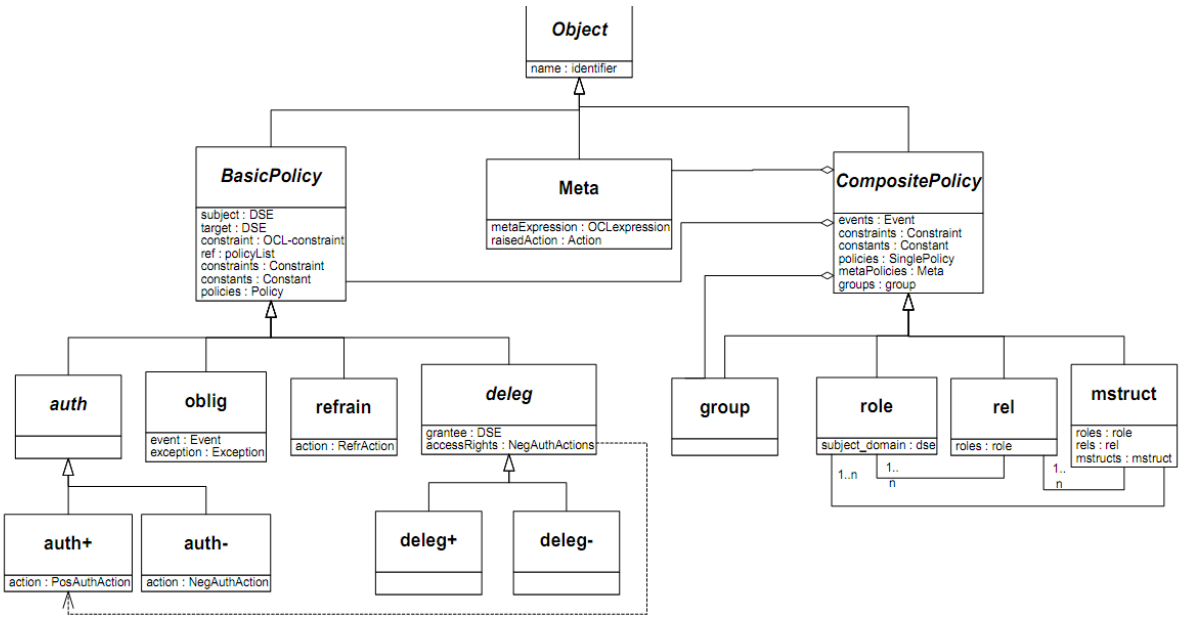


Figure 4.6 Ponder Base-Class Diagram (Damianou, et al., 2000)

Moreover, authorisation polices in Ponder can be implemented using available control methods such as those in firewalls and operating systems. Also, obligation policies (event-based condition-action rules) are supported. In addition, Ponder can be used for security management in distributed systems thus allowing for user registration and activity logging.
Onwards, two simple examples of authorization and role polices are presented (please refer to (Damianou, et al., 2000) for more examples). Figure 4.7**Errore. L'origine riferimento non è stata trovata.** presents the authorisation policy example where members of the *NetworkAdmin* domain are allowed to load, remove, enable or disable objects of type *PolicyT* in the *Nregion/switches* domain.

```
inst auth+ switchPolicyOps {
        subject              /NetworkAdmin;
        target     <PolicyT> /Nregion/switches;
        action     load(),    remove(),    enable(),
disable() ;
}
```

Figure 4.7 Ponder Authorisation Policy Example (Damianou, et al., 2001)

In Figure 4.8, a Ponder role policy example is presented. A role type is modelled for a telecommunication service engineer. In this role, the service engineer is responsible for handling customer complaints in addition to service requests. The role takes as input the calls database that provides information about subscribers' calls. In the case of a *customerComplaint* event, the *serviceComplaint* obligation is triggered where the subject of the role follow a sequence of actions on the calls database. All policies within this role have the same implicit subject so the obligation policy does not need to specify a subject explicitly.

```
type role ServiceEngineer (CallsDB callsDb) {
        inst oblig serviceComplaint {
                on          customerComplaint(mobileNo) ;
                   do       t.checkSubscriberInfo(mobileNo, userid) ->
                            t.checkPhoneCallList(mobileNo) ->
                            investigate_complaint(userId);
                target      t = callsDb ; // calls register }
        inst oblig deactivateAccount {...}
        inst auth+ serviceActionsAuth { . . . }
        // other policies
}
```

Figure 4.8 Ponder Role Policy Example (Damianou, et al., 2001)

While having been successfully used in several applications, Ponder suffered from different disadvantages which triggered a more recent and revised Ponder2 (Twidle, et al., 2009). Ponder2 aims at pervasive systems where Ponder's limitations such as the centralised infrastructure design, lack of support for collaboration between different polices' elements and the off-line nature of its policy specification task, are all dealt with. Hence, Ponder2 was implemented as a self-managed cell: "A self-managed cell is defined as a set of hardware and software components forming an administrative domain that is able to function autonomously and is capable of self-management" (Twidle, et al., 2009). Ponder2 allows for user-written Model Objects using Java with simple annotations. A smalltalk-like language, namely, PonderTalk is used for message communication among different Message Objects. Ponder2 is available for download under the GNU Lesser General Public License as published by the Free Software Foundation.

4.3.5 IBM TPL

The Trust Policy Language (TPL) (Herzberg, et al., 2000) was developed by IBM with an XML-based syntax. The language was only meant to provide for access control to resources. Its policy rule includes a set of certificates that are required for a given user to belong to a given group and an evaluation function to decide on that. That function uses fields from the included certificates in the evaluation process. TPL is limited only to defining access control for security policies. However, TPL allows policy authors to express constraints on several levels such as on (combinations) of credentials/inter-credentials and authentication. TPL has a definite version referred to as DTL where negative rules are not allowed. This allows for DTL to be easily mapped to a logic programming language. Based on (Seamons, et al., 2002), which provided a set of requirements to evaluate different policy languages for trust negotiation, TPL was found to be fulfilling some of the requirements. However, Portfolio and Service Protection *Language* (PSPL) was found to be a better policy language than TPL for trust negotiation.

4.3.6 PeerTrust

PeerTrust (Nejdl, et al., 2004) is a policy management framework based on the first order logic of Horn rules. It is mainly aimed at specifying requirements for security and trust on the sematic web. An elaborating example is provided in Figure 4.9 which forms a part of a more detailed informative scenario in (Nejdl, et al., 2004).

```
E-Learn:
discountEnroll(Course, Party) $ Requester = Party←
        discountEnroll(Course, Party).
discountEnroll(Course, Party)←
        eligibleForDiscount(Party, Course).
eligibleForDiscount(X, Course) ←preferred(X) @ "ELENA".

preferred(X) @ "ELENA"←
        signedBy ["ELENA"]
 student(X) @ "UIUC".
```

Figure 4.9 PeerTrust Policy Example (Nejdl, et al., 2004)

In Figure 4.9, an institute called E-Learn sells learning resources and provides discounts to some users. Discount eligible users are classified as the preferred customers in the ELENA group. Given that E-Learn was given a singed rule by ELENA specifying how to check whether a user is a preferred one or not, E-Learn does not need to communicate with ELENA each time it needs to check for discount eligibility.
Update of PeerTrust was halted 2004-2005 and was followed by a different project called Protune which is covered in the section to follow.

4.3.7 Protune

The PRovisional TrUst NEgotiation (Protune) (Bonatti, et al., 2005) is a framework for policy specification and deployment that merges between distributed trust management policies, business rules and actions required for access control. Its policy language is based on two existing languages, namely, PAPL (concerned with trust negotiation) (Bonatti, et al., 2000) and PeerTrust. Protune also provides a meta-language for handling critical negotiation decisions as well as integrity constraints that are meant for monitoring the disclosure of credentials. Protune follows an ontology-based approach for easier importing and exporting of metapolicies and also to allow for further language extension. The Protune project is maintained online at (Pro11) where an implementation is available for download including a language editor.

4.3.8 KAoS

KAoS (Uszok, et al., 2004) is a framework for policy and domain services that is organised in a set of components compatible with several agent platforms such as DARPA's CoABS Grid and Cougaar agent framework and CORBA. KAoS makes essential use of OWL which is an ontology authoring standard based on descriptive logic. OWL is used in KAoS to represent policies and domains besides other managed entities and affiliated elements. Also, OWL allows for the framework to be easily extended while being consistent with advanced semantic requirements.
KAoS domain services allow for semantically describing and structuring of software components, people and resources into groups of (sub)-domains which is believed to ease external policy administration and collaboration. Moreover, KAoS policy services are used to specify, manage and enforce policies as well as resolve conflicts among policies.

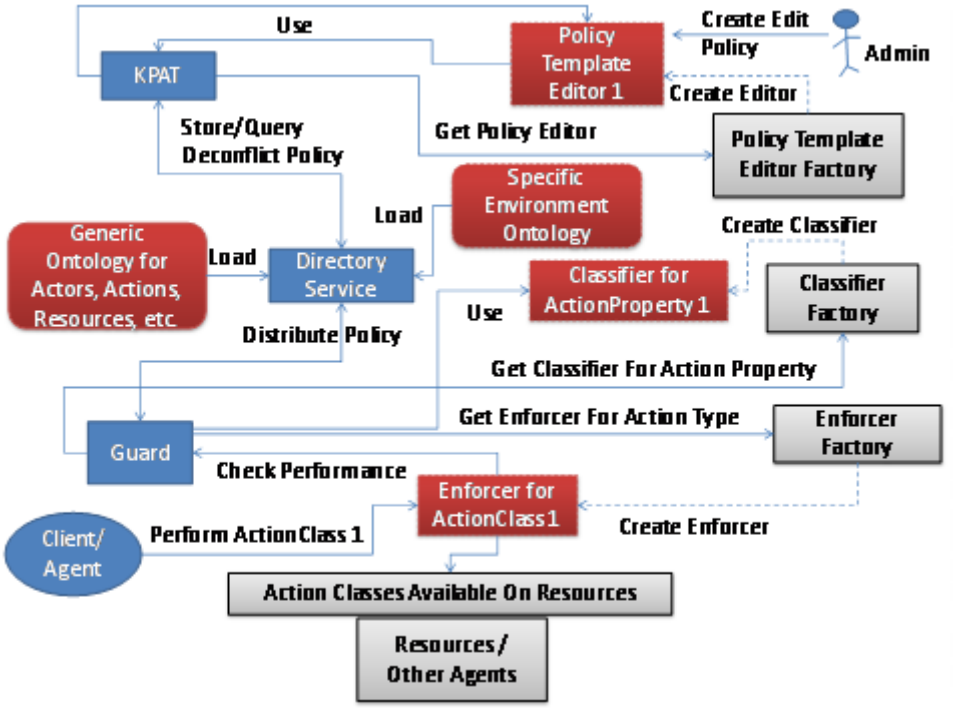


Figure 4.10 Basic KAoS Framework Elements (Uszok, et al., 2004)

Three types of conflicts can be detected in KAoS that are based on the positive or negative nature of an authorisation or obligation:

- positive vs. negative authorisation
- positive vs. negative obligation
- positive obligation vs. negative authorisation

KAoS resolves these conflicts through “harmonization” which involves assigning different policies different priorities (Bonatti, et al., 2007).
In Figure 4.10, the basic elements of the KAoS framework are presented. Two groups of functionalities are naturally found in the framework, i.e., generic and application-specific. Generic functionalities include ontologies creation and management as well as policies storage, querying, distribution, enforcement, disclosing and conflict resolving. Moreover, application-specific functionalities may include defining new ontologies and creating extension plug-ins such as policy templates, custom action property editors and subclasses action enforcers and classifiers.
More details can be found on the project’s website at (KAoS) where the policy ontology used is also presented in details.

4.3.9 REI

REI (Kagal, 2002) is a policy language similar to what is used in the KAoS framework in many forms. REI was designed to deal with security issues where heterogeneous entities exist in a dynamic environment through policies. REI uses OWL-Lite and logic-like variables which allow for easier application-specific extensions and different relations specification. Polices here are defined based on what entities can/cannot do or should or should not do. REI policies follow an ontology that includes different concepts such as permissions, obligations, actions etc. as well as the ability to import domain/application-dependent ontologies. Analogous to KAoS’s positive/negative authorisations and obligations, REI policies include “deontic concepts” such permissions/prohibitions and obligations/dispensations. Moreover, REI allows for rights and obligations to be dynamically exchanged among entities through right delegation and revocation as well as action or delegation request and cancelation. Policy conflict detection and resolution is handled through overriding policies which is a similar technique as in prioritisation in KAoS.
The project ended in May 2005 but its website is still alive at (REI, 2005) where more details can be sought if needed.

4.3.10 Discussion

Several reviews have already discussed and compared the aforementioned policy specification languages and models. Based on the following studies (Olmedilla, 2008)(Duma, et al., 2007), a set of metrics is selected to form a representative criteria for evaluating these policy languages/models in light of pSHIELD’s characteristics. Following is a description per each:

- **Well-Defined Semantics:** the meaning of the policy is independent from the implementation of the language it is written in
- **Access Control:** requesters with valid credentials are given access to certain data/attributes
- **Minimal Information Disclosure:** credentials or attributes sensitivity specification
- **Mutual Exclusiveness:** control of simultaneous release of data that could be sensitive collectively
- **Sensitive Policies Protection:** control the disclosure of policies and assigning them sensitivities
- **Usage Control:** ability to impose restrictions and obligations on released data consumers
- **Action Execution:** allowing policy writers to specify actions in policies
- **Delegation:** passing on or transfer rights such as access rights
- **Evaluation Type:** distributed (able to gather policies or policy elements spread on the net for evaluation) or local policy evaluation
- **Evidences:** support for the concept of user credentials
- **Extensibility:** ability to adapt to user needs
- **Standardization:** the extent to which the approach has been standardized

Criterion	XACML v.3	EPAL	WSPL	Ponder	TPL	PeerTrust	Protune	KAoS	REI
Well-Defined Semantics	N	Y	N	N	N	Y	Y	Y	Y
Access Control	Y	Y	Y	Y	Y	Y	Y	Y	Y
Minimal Information Disclosure	Y	Y	-	Y	-	-	Y	-	N
Mutual Exclusiveness	-	-	-	Y	-	-	Y	-	Y
Sensitive Policies Protection	Y	-	-	Y	-	-	Y	-	Y
Usage Control	E	-	-	Y	-	-	N	-	Y
Action Execution	Y	Y	Y	Y	N	Y	Y	N	N
Delegation	Y	N	N	Y	N	Y	Y	N	Y
Evaluation Type	Dist.	L	Dist.	L	L	Dist.	Dist.	L	Dist.
Evidences	E	N	N	-	Y	Y	Y	N	-
Extensibility	Y	Y	N	Y	N	Y	Y	Y	Y
Standardization	A	U	-	-	-	-	-	-	-
A: Available, U:Unmaintained, Dist.: Distributed, L: Local, E: Extendable, -: Not Available/Applicable									

Table 1 Policy Language/Model Evaluation Table

Table 1 presents a comparison among the described policy languages/models based on the selected criteria. Support of privacy-centric metrics such as minimal information disclosure or usage control could not be verified for policy languages such as KAoS, EPAL, PeerTrust, etc. It can be noticed that all described policy languages support access control while they vary in their support to other metrics. XACML, Ponder, Protune and Rei seem to satisfy most of the metrics with the exception of well-defined semantics for XACML. However, XACML is the only model that is well standardised and maintained which makes it a viable model for adoption. EPAL had some standardization efforts by W3C in 2003 but it has not been maintained where XACML has been identified as better option in an evaluation against EPAL (Anderson, 2006). Usage control and Evidences in XACML are supported by extension as in (Colombo, et al., 2010) and (Ardagna, et al., 2009) respectively. Essentially, XACML is seen as a viable choice for PBM in pSHEILD given its concrete standardization and its support for access control policies besides other relevant features.

4.4 Affiliated protocols

A set of protocols that are usually affiliated with the management and propagation of policy interpretation to the lower level of resources are presented here.

4.4.1 COPS

Common Open Policy Service (COPS) protocol governs a client/server form of communication for policy control on QoS signalling protocols, especially between a Policy Decision Point (PDP)/server and a Policy Enforcement Point (PEP)/client. COPS is defined by IETF's RFC 2748 (IETF, 2000) and it supports two modes of operation; COPS Provisional Model (COPS-PR) and COPS Outsourcing Model. The latter is COPS' simple form of operation where a PEP sends requests for policy decisions to a dedicated PDP. The PDP then evaluates the request against a given policy and responds to the sending PEP with the applicable policy decision. In the COPS-PR mode, PEPs will inform the PDP about their ability to take policy decisions including local specifications. The PDP hence uploads the provisioned policies onto the PEP which stores them in a local Policy Information Point/Base (PIP/B) and carries out requests for policy decisions locally. While online, the PDP can update or remove policies on the PEP if it found that necessary due to some external changes. Also, PEP is expected to send a request for update to the designated PDP if its local configurations have changed where the PDP will upload any additional provisional policies on the PEP.
COPS uses TCP in order to exchange messages between clients and servers. It can also use existing security protocols such as IPSEC and TLS to support an authenticated and secure channel between PEP and PDP. Moreover, COPS is stateful as a request/decision state is shared between the PDP and the PEP as well as that pairs of requests and decisions (also referred to as events) can be interrelated.

4.4.2 SNMP

Simple Network Management Protocol (SNMP) (IETF, 1990) is a well-established protocol (approved in 1990) for monitoring entities or devices in a TCP/IP network. SNMP allows for different network management tasks such as auditing, performance monitoring, faults detection and remote configuration. It is designed to be simple where its deployment on several managed devices will not be resource demanding while still being robust. Latest versions of SNMP improved security and data integrity by using MD5 hashing and DES encryption in addition to protection against reply attacks through authentication.
As an alternative to COPS, SNMP can be used for governing the communication channel between the PDP and PEP. For instance, SNMP was recommended in the early stages of PBM to be used in network management as in (Boros, 2000). Also, in (Rana, et al., 2009) SNMP and COPS were recommended in the management of home area networks through PBM.

4.4.3 LDAP

Lightweight Directory Access Protocol (LDAP) (IETF, 2006) is an application protocol that allows for accessing distributed directory services that follow the X.500 model. LDAP is a variation of X.500 customized to provide a lightweight implementation with TCP/IP support. It is considered to be cross-platform suitable for read-intensive operations hence allowing for directory access from different platforms and locations with infrequent update requirements. From a PBM perspective, the repository used to manage stored policies can serve as an LDAP server where PDP and PAP are seen as LDAP clients. In different research concerning PBM in networks management on different aspects, LDAP is recommended for accessing policy repository (Verma, 2002)(Sloman, et al., 2002)(Flegkas, et al., 2002). An open source implementation, namely, OpenLDAP is available for free at (OpenLDAP, 2011).

4.5 Reflection on pSHIELD

In this section, a typical PBM architecture is mapped to pSHIELD's general architecture. The latter includes two types of nodes at the button node layer that are categorized based on their capabilities in terms of processing power and capacity, i.e., power nodes and sensor nodes. Power nodes are described to be more resourceful while sensor nodes are typically seen as resource constrained devices. Upper supporting layers constitute network, middleware and application layers while agents in a vertical overlay monitor/tune those layers. Given the aforementioned architecture, a PDPs and PEPs from a typical PBM architecture can be mapped naturally to power and sensor nodes respectively. Figure 4.11 presents the proposed PBM mapping.
On the lower layer, sensor nodes being the managed resources are considered as policy enforcers, i.e., PEPs. The latter, based on the XACML model, should enforce authorisation decisions and handle affiliated obligations specified by applicable rules. PEPs can support local policy storage in order to comply with COPS-PR mode of operation hence the provision of a local PIP although not compulsory. However, this depends on the capabilities of deployed sensor nodes whether they can afford a form of local policy storage and decision making. Moreover, power nodes are those nodes that are more resourceful than the sensor nodes which make them natural decision making points able to process/translate policies and deduce rules to be enforced by affiliated PEPs. The COPS protocol can govern the communication between PDPs and affiliated PEPs but not exclusively as SNMP is an option as well (where an LPIP is no more required).

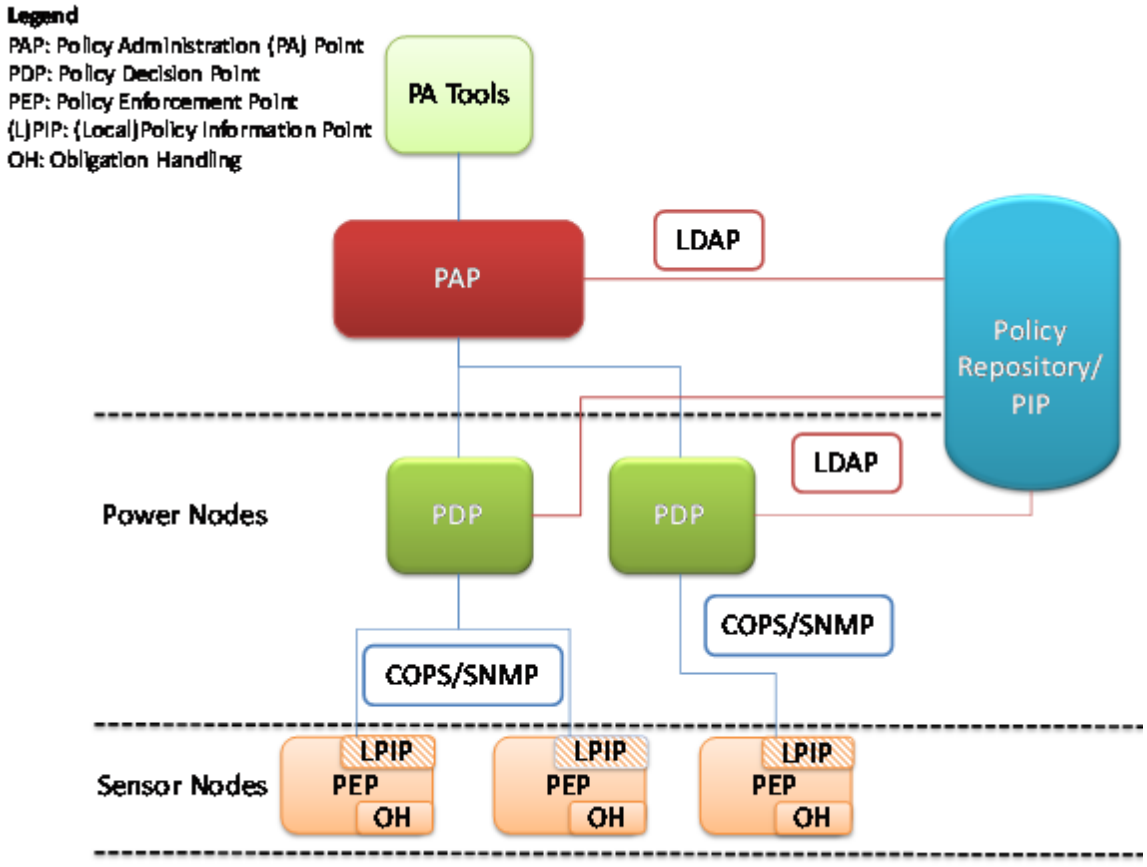


Figure 4.11 PBM Mapping

A group of PDPs can access the repository of policies, (i.e., PIP) in order to retrieve needed polices for evaluation. This is done through LDAP that is a protocol suited for lightweight read-intensive operations allowing for directory access from different platforms and locations. The policy repository is managed solely by the policy administrator point (PAP). Also, PAP is responsible for providing policy authoring tools besides management and control capabilities. These could include creation, termination, activation, listing, amending and synchronizing policies. Commercially for instance, Cisco's PAP (Cisco) manages,

administer and monitor policies in a central manner compatible with LDAP. It also provides several features such as web-based editing and composing of policies in a drag and drop manner, policy scope definition, delegation management and aiding functionalities for understanding the implications of policy changes. Concerning, LDAP implementations, an open source version is available as mentioned earlier at (OpenLDAP, 2011). However, COPS does not seem to have an open source implementation where some commercial ones could be available. If SNMP is considered as an alternative for COPS, some open source and free implementations are available such as in (NET11). XACML is indeed the policy specification language to be used as argued in earlier discussion.
Concerning pSHIELD's main scenario where a monitoring and access control system is put in place to oversee rail-transported hazardous materials, the above PBM is considered suitable. Locking and access control mechanism in addition to installed sensors can be seen as PEPs where the central control unit in the train carriage can be seen as a PDP with local access to PIP. Moreover, the central command centre overseeing the operation of the monitoring system is seen as a PAP with policy administration tools and repository support. The PIP is expected to be distributed which allows a given PDP to access it locally where a PAP can manage such a distributed PIP through LDAP.

4.5.1 Performance evaluation

As a final step, a performance evaluation of the policy execution (as PDP) is reported, with specific focus on the computation time during policy execution.

The processing time depends on:

- Policy specification language (high level or low level)
- Type of policy execution engine
- Underlying formalisms used to describe attributes (e.g. Knowledgebase in pSHIELD)
- No. of simulteneous execution

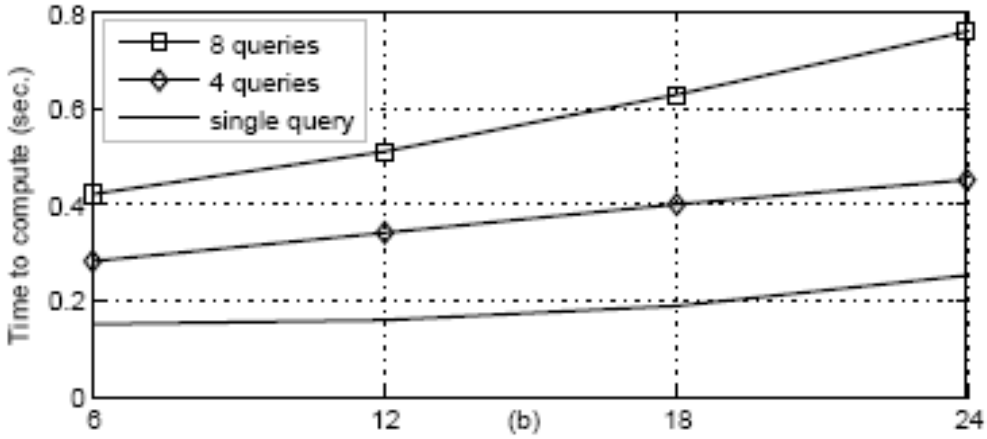


Figure 4.12 N° of instances/class in Knowledge Base

Figure 4.12 shows the computation time for simultaneous policy execution with the following simulation parameters:

- Rule-based semantic policies (SWRL-SQWRL)
  - *A high level language*
- Underlying formalism: OWL
  - *High level comapare to simple XML*
- Simulteneous queries from Application requir-es execution of simulteneous policies

- Performance measure in P4 2.0 GHz,1GB RAM windows machine

Implications for pSHIELD: Latency is one of the QoS requirements for Web services at Middleware level; latency includes request processing time. Figure shows that even with small no. of instances simulteneous policy execution takes increasing amount of time.

For that reason run-time decision support with simultaneous query processing may not be possible with such settings.

4.6 Conclusions

This section presented and discussed technologies related to Policy-Based Management (PBM). This included policy specification languages and models in addition to affiliated protocols. Based on the discussion of available PBM technologies, a suggested PBM mapping was presented on pSHIELD's general architecture. XACML is recommended for policy specification given its standardized nature with access control support. Moreover, LDAP is found to be widely used for governing communication between PDP and PAP from one side and the repository of polices on the other side. Also, COPS or SNMP are both widely used as protocols for communication between PEPs and PDPs.

5 pSHIELD Overlay and control algorithms prototypes

5.1 Introduction

With reference to the pSHIELD functional view (see deliverable M0.1) the following figure highlights the key functionalities and interfaces involving the Overlay layer.

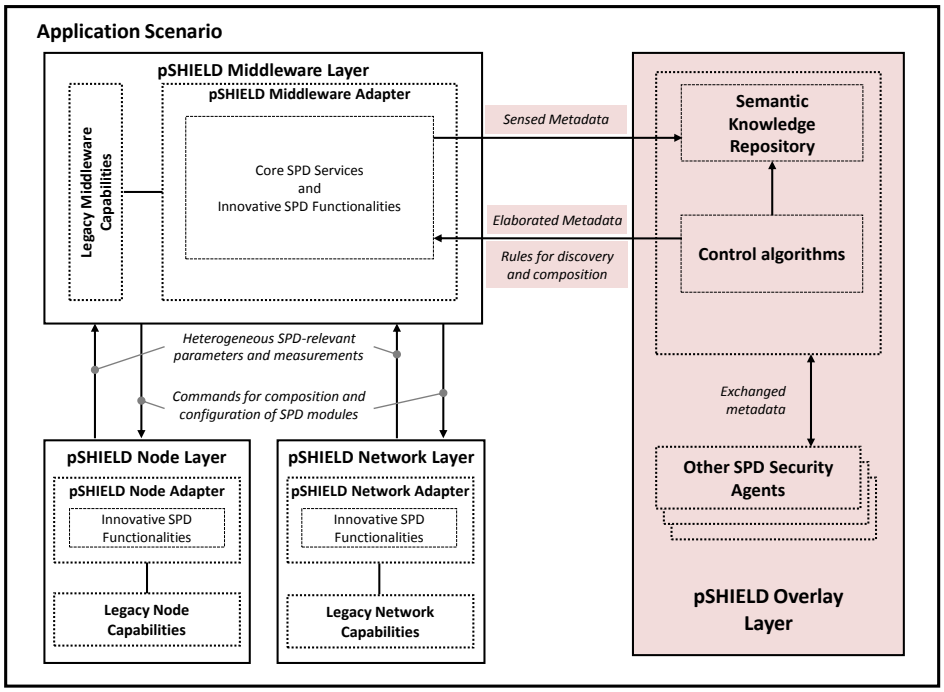


Figure 5.1 pSHIELD overlay: a functional view

The Overlay consists of a set of SPD Security Agents, each one controlling a given pSHIELD subsystem. Subsystem identification has to be carefully performed scenario by scenario. Expandability of such framework is obtained by enabling communication between SPD Security Agents controlling different sub-systems. As a matter of fact, the presence of more than one SPD Security Agent is justified by the need of solving scalability issues in the scope of system-of-systems: exponential growth of complexity can be overcome only by adopting the policy of *divide et impera* (divide and rules).

Each SPD Security Agent, in order to perform its work, exchange carefully selected information with the other SPD Security Agents, as well as with the three horizontal layers (node, network and middleware) of the controlled pSHIELD subsystem. However, in the scope of the pilot project the interaction between different security agent is not taken into account, since the controlled subsystem is reduced and only the "single security agent" case is considered.

Each SPD Security Agent collects properly selected heterogeneous SPD-relevant and context measurements and parameters coming from node, network and middleware layers of the controlled pSHIELD subsystem; this information is used as valuable input for the Control Algorithms (see next paragraph). Since the SPD Security Agent is mainly a software functionality and not a physical system itself, it needs the mediation of the pSHIELD Middleware Core SPD Services as it has been explained in the previous section.

Summarizing, each SPD Security Agent consists of two key elements:

(i) the Semantic Knowledge Repository (i.e. a database) storing the dynamic, semantic, enriched, ontological aggregated representation of the SPD functionalities of the pSHIELD subsystem controlled by the SPD Security Agent;

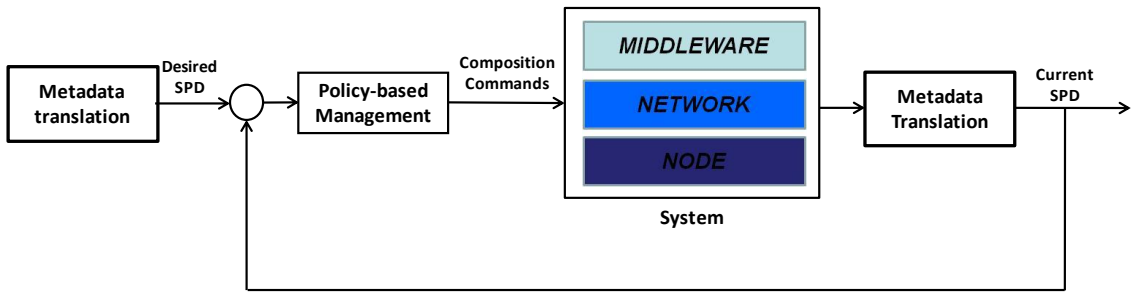
(ii) the Control Algorithms generating, on the grounds of the above representation, key SPD-relevant decisions (consisting, as far as the Composability feature is concerned, in a set of discovery, configuration and composition rules).

In deliverable D5.1 as well as in the SPD Core Services section of the current document, the first point has been widely exposed by means of prototypes. The purpose of this section is to formalize the behaviour of the Overlay and the SPD Security Agent with respect to the control algorithms (often referred to as “embedded intelligence”).

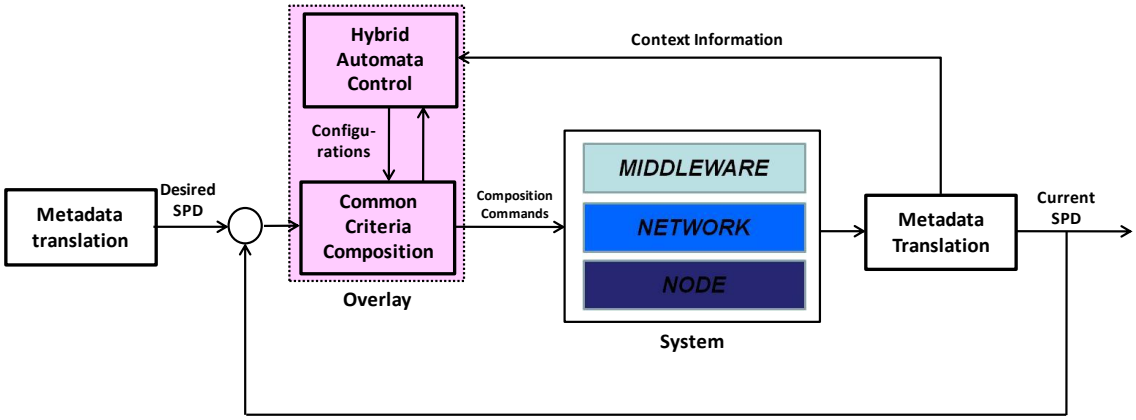
5.2 Overlay Behaviour

The behaviour of the Overlay is provided in

Figure 5.2: thanks to the pSHIELD metrics identified in WP2, the desired SPD level can be quantified and translated into a reference signal for a closed loop control scheme. Two potential paths can be followed to achieve this desired level: (i) the first one is by means of a **policy based management**, with a set of pre-determined actions and rules (policies); (ii) the second one is by means of the **common criteria approach** enriched with **Hybrid Automata control algorithms**.



Policy-based Management approach



Overlay control algorithms approach

Figure 5.2 Overlay behaviour

The first way (the Overlay acts as policy based manager) has been analyzed in the previous section with the performances consideration already carried out.

The second way will be shown to be more efficient and to provide the best performance, but at the same time requires a biggest effort to be developed in the real system. For that reason the implementation of this solution, in the pilot project, will be limited to a simple (static) case and a set of simulations, while the theory and the rationale behind it will be addressed in this section. This procedure can be summarized in the following steps:

1) The *semantic representation* of the system (including the functional dependencies between the components as well as the atomic values of SPD for each element) *is discovered* (measured) to provide the metadata necessary to the Overlay.

2) This knowledge, thanks to the *inferential engine* described in D5.1 and in D2.2.1 as Common Criteria approach, is used to find a list of *configurations* of pSHIELD components that *satisfies the*

user requirements in terms of SPD. Since the ontology models the functional dependencies of atomic components, all the *identified configurations are feasible*.

3) Then the semantic knowledge is used to create a model *of the system and its evolution* by means of *Hybrid Automata Theory* (see next paragraph) and this model, enriched with *optimization* control algorithms, is used to decide which, among the suitable configuration, is the *best*, with respect to context information.

These logical flow is depicted in Figure 5.3.

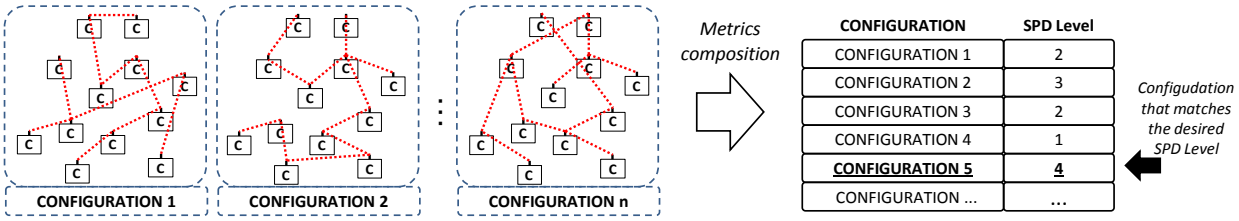


Figure 5.3 Overlay approach for configuration identification

The semantic representation (step 1) has been widely addressed in deliverable D5.1.

The rules and mechanism to compose metrics to measure the configuration SPD level (step 2) are available in deliverable D2.2.2 as well as in the reasoning section of D5.1.

The configuration choice basing on context information is the objective of the following paragraph.

5.3 Hybrid Automata approach to model and control complexity in ESs (by means of context information)

Modelling the **composability** is a challenging issue faced by different research activities both in academic and in European Project environments. This is the case, for example, of the European project *Connect* (http://connect-forever.eu) or *EternalS* (http://www.eternals.eu) whose aim is to develop an abstraction framework for the "eternal" connection of evolving networked devices and/or software functionalities.

In these context the most common methodologies to model the composition of elements (on a theoretical point of view) are widely used: Petri-Nets, Bayesian Networks, Graphs, Markov Chains, and so on.

These methodologies have been the bases of the technological scouting performed by pSHIELD in this field as well, to check the feasibility of these solution. However all the classical mathematical and logical methodologies have shown not to be adequate to model the pSHIELD contex due to two peculiarities:

1) pSHIELD is an **heterogeneous** system, so it is necessary to find a theory able to model "heterogeneous elements" in a single entity at a time.

2) pSHIELD is a **dynamically composable** system, so the modelling tool should be scalable and composable as well (no unique information/energy flow among components)

Furthermore, this model should be ready to be integrated and implemented in analog/digital closed loop control schemes, meaning that the resulting composition should **model the evolution** of the system (in nominal condition or in presence of faults).

For all these reasons, the adopted approach will be based on Hybrid Automata: a mathematical framework by which it is possible to model Hybrid Systems (continuous/discrete/heterogeneous domains like Embedded Systems) compose them (parallel composition) and make them evolve (automaton).

Different models have been tailored for the pSHIELD purposes and in the following paragraphs two of them will be presented: one static, simpler but less scalable, and one dynamic, more complex but scalable and more expressive.

Before continuing, a short recall of what is an Hybrid Automata is provided (see [1][2])

5.3.1 Hybrid Automata

An *Hybrid Automaton (HA)* $A = (W, X, Q, \Theta, E, H, D, T)$ is an element composed by:

- A set of external variables *W* as well as internal variables *X*, disjoint $V \triangleq W \cup X$
- A set of finite states $Q \subseteq val(X)$.
- A set, non empty, of initial states $\Theta \subseteq Q$
- A set *E* of external actions and a set *H* of internal actions disjoint $A \triangleq E \cup H$
- A set of discrete transitions *D* and a set of trajectories *T*

To describe the behaviour (evolution) of an automaton, it is sufficient to indicate the starting state, the transitions (with or without resets) and the next state. Depending of the time horizon, the starting states and the external actions, this description is done by means of: *execution fragments, executions, trace fragments* e *traces*.

The most powerfull feature of Hybrid Automata is that they can evolve independently, but can be composed together to exchange Input/output and to drive the evolution of each other transitions. This is known as *parallel composition*.

In the scope of the pSHIELD project, some Hybrid Automata have been designed to model the pSHIELD generic component, composed together and controlled.

5.3.2 Prototype a – Static Approach with Simple Optimization

The first, simple approach, is based on the following steps.

At first we need to identify the system "state", i.e. the set of active components (node, protocols or applications). This will be the system identifier in a specific circumstance. A state is a screenshot of the system in a specific condition (for example with the node E switched on) and with the dynamics associated to this condition (for example the evolution of the node's power consumption).

The selected dynamics considered for this model constitutes the so-called context information: since the SPD is controlled via the common criteria approach, we need to insert into the model variables that could be significant to control (optimize) the evolution of the system. They could be, for example, the power consumption, the computational resources utilization, the bandwidth utilization, and so on.

The state identified in this step is depicted in Figure 5.4.

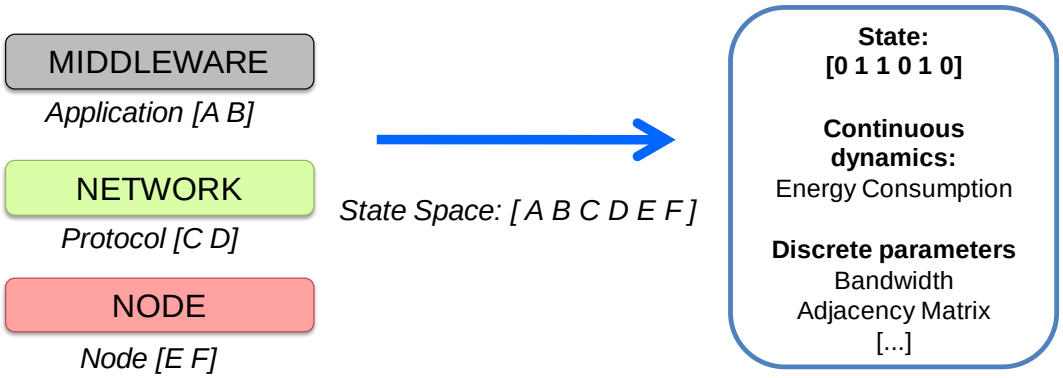


Figure 5.4 Single State

The second step is to concatenate the different states to obtain the universe of all the possible condition of the system: this is an enumeration of configurations. For example in a system with two nodes, two network protocols and two middleware services we 8 states (at least one component must be active).

Q = {[101010], [101001], [100110], [100101], [011010], [011001],[010110], [010101]}.

The result is depicted in Figure 5.5

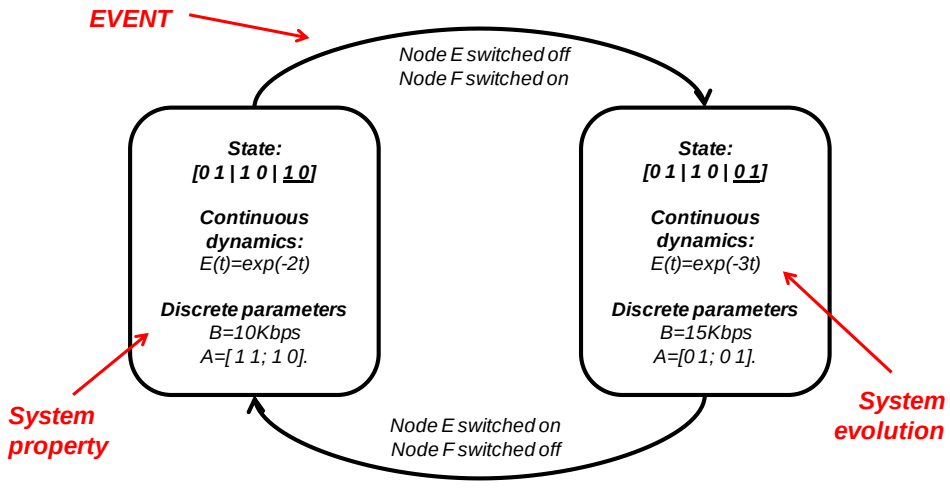


Figure 5.5 Hybrid Automata to describe all the possible configurations

The transition can be voluntary and expected (control action) or not (due to fault) but in any case each event is captured and in evry moment it is possible to check the status (and evolution) of the system:

D = {switch configuration$_1$, fault$_1$, …, switch configuration$_n$, fault$_n$}.

The third step is the indentification of the internal variables (and dynamics) to control. For the pilot project a simple case is considered where the relevant dynamic is the power consumption of the system in a specific configuration and the amount of bandwidth provided by the network layer. These variable have opposite behaviours (higher bandwidth, higher power consumption) so the purpose of the control algorithm is to choose the configuration that optimizes one of them.

This scenario has been implemented in Matlab-Simulink and is composed by two nodes with two different dynamics for the power consumption and for bandwidth utilization. It is important to notice that both these configurations should be valid SPD configurations (see CC approach).

A simple controller is in charge to switch to the configuration that guarantees the longest duration of the node batteries

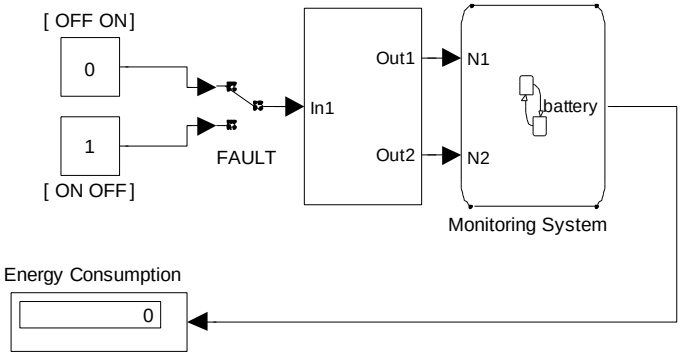


Figure 5.6 Simulink model

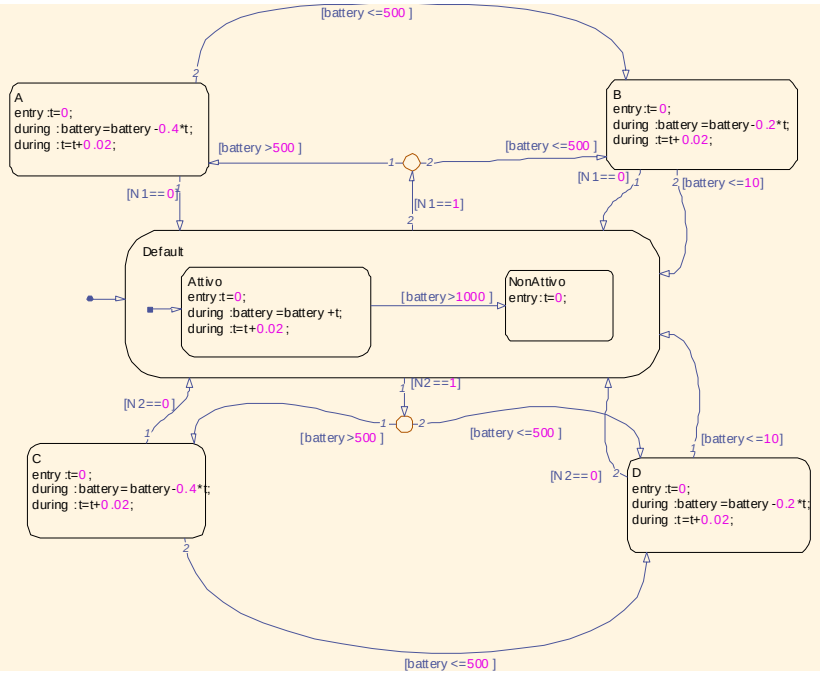


Figure 5.7 Hybrid automata with four states

In Figure 5.8 the result of the simulation is carried out: when the power consumption is not a problem, the system remains in the configuration ‘A’ that allows him to use all its resources: when the dotted line (energy consumption reference) is lowered, the node switch in a saving configuration ‘B’ and starts wasting less power, thus lasting more.

Since many parameters can be included in the continuous dynamic, a multi-objective optimization can be performed to obtain optimal performance driven by context information.

On a deployment point of view, this model could be implemented in an intelligent node that can simulate the system behaviour and adequately react to faults and evolutions.

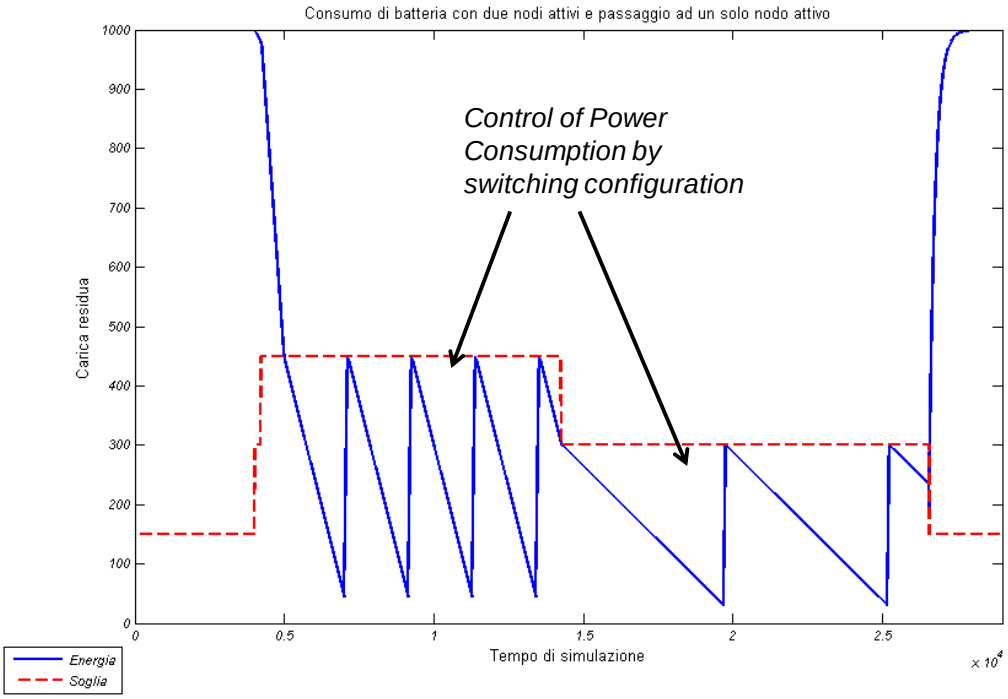


Figure 5.8 Configuration switching to optimize piower consumption

Due to its simplicity, this approach suffer for a scalability problem, since when the number of component rises, the number of possible configurations grows exponentially. A more intelligent and efficien approach is needed.

5.3.3 Prototype b – Operating conditions approach with MPC Control

A better solution has been found starting from the analysis, in literature, of the work carried out by Balduzzi (see [4][5]) to model manufacturing environment by means of Hybrid Automata. His idea is very simple: in a complex production system ,where all the machines (elementary services) are connected to produce goods, only the operating conditions of the machines influences the system behavior. Moreover the dynamic of the machine changes only when the operating conditions change. For example a machine could be broken, operative in linear conditions or operative and saturated. By using these conditions as “states” of the hybrid automata, it is possible to describe in a simple but expressive way, all the relevant dynamic of the system and the control it. A sample screenshot taken from [4] is depicted in Figure 5.9.

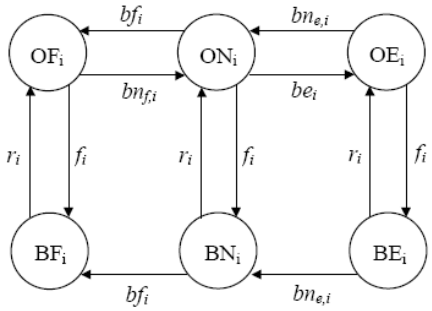


Figure 5.9 Operating condition of a manufacturing system as modelled in [4]

Given an Embedded System (pSHIELD Node) it is possible to identify a set con elements (battery, buffers, CPU) that can be associated to an operating conditions: a buffer can be saturated, full or empty; a CPU can be idle, working or overloaded; a battery can be full or empty. All these components can also be broken. The combination and aggregation of these conditions allows to create an exhaustive model of a pSHIELD node, as depicted in Figure 5.10. The aggregation is possible, since some behaviours of the components have the same effect of the system (if the CPU or the Buffer is full, the result is always the impossibility of processing data).

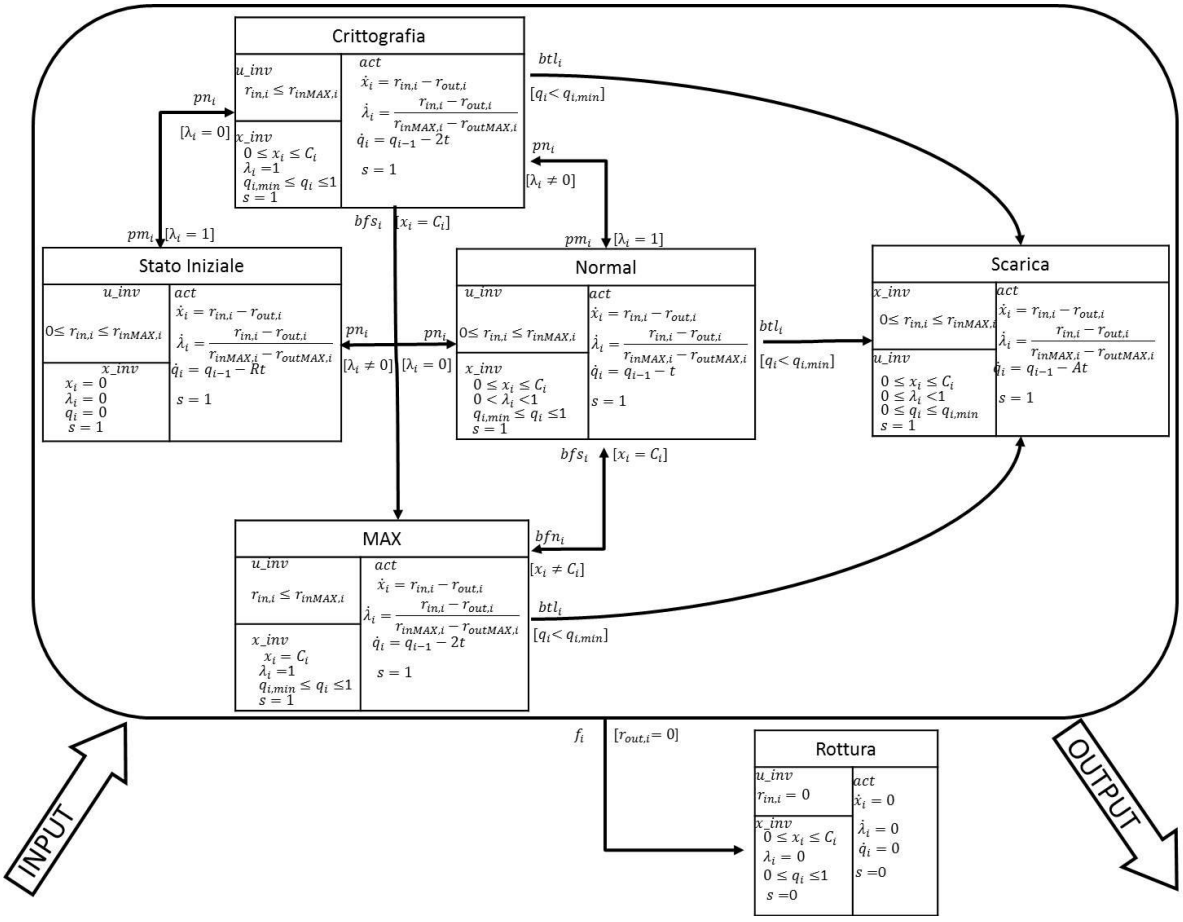


Figure 5.10 Hybrid Automata representing the pSHIELD node

The same process can be done for the Network Layer, since the only relevant operating conditions for a network (at this stage) are two: its *congestion status* and its *connectivity* (i.e. the possibility of reaching all the nodes). A four state model is obtained (congestioned and connected, congestioned and not connected, connected and not congestioned, not connected and not congestioned). See Figure 5.11 for the model.

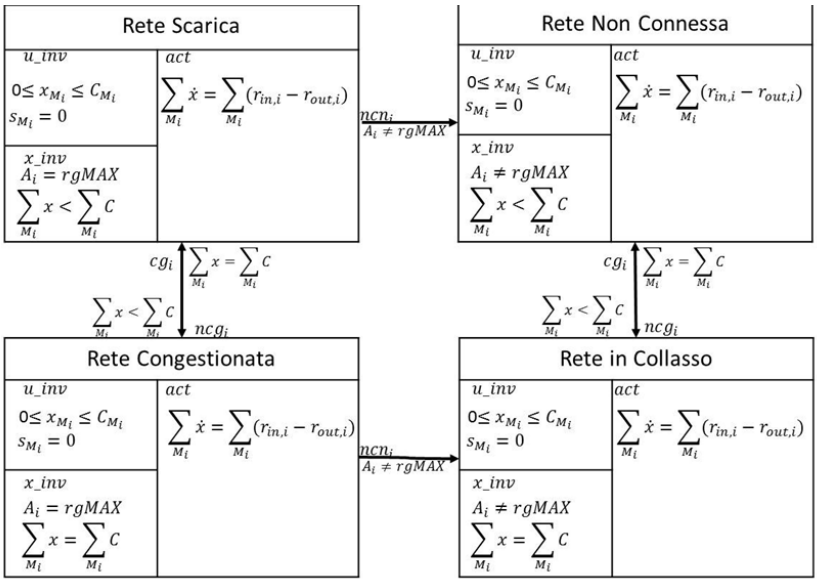


Figure 5.11 Hybrid Automata representing the pSHIELD network

At this point the composition problem is solved, since the introduction of a new node in the system doesn't imply an exponential increase in the model size, but a linear growth (6 states for each additional node and 4 states for each additional network layer).

Last, but not least, interesting control algorithms can be applied to the system model due to its formulation by means of these operating conditions (see for example the work of Bemporad [6] and [7]). In particular for the pSHIELD purposes the framework developed in [7], based on Model Predictive Control (MPC), has been considered to verify the effectiveness of the Hybrid Automata approach.

For the complete formulation of the MPC problem please refer to forthcoming deliverable D5.4 or [7].

For the simulations it has been used the Matlab Toolbox for Hybrid System with the default configuration (standard MPC problem). The Objective of the control algorithm has been to maximize the amount of data data processed by the node while preserving the battery and leaving a certain amount of "reserved" resources for potential emergency tasks. This objective is "ambitious" but the power of MPC control is the multi-objective optimization over a temporal horizon, and the results of Figure 5.12 demonstrate that the system follows the desired behaviour and all the objectives are met, On a practical point of view this is translated into a set of control commands from the overlay to the system, that decide time by time which component should be activated to satisfy the requirement.

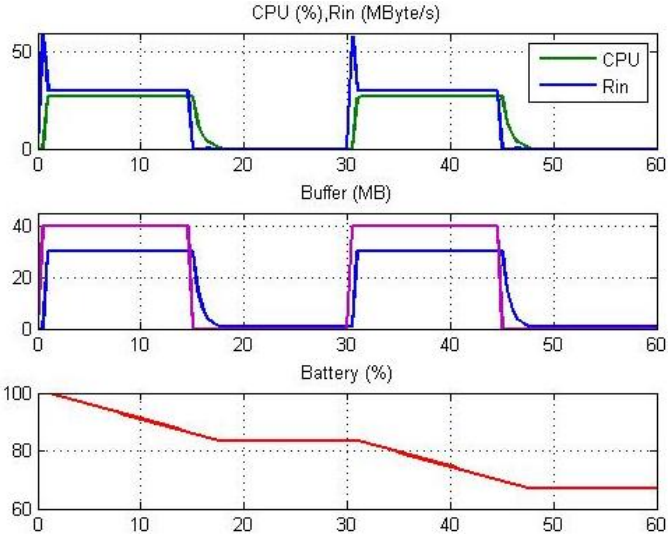


Figure 5.12 System behaviour with MPC control

Also in this case it is important to notice that all the behaviour and the configurations are considered to be “feasible”, so the objective of the controller is to tailor parameters or switch component, according to context information, to perform context optimization (even not directly connected to SPD issues).

5.4 Conclusions

In this section an innovative formulation of control problems in Embedded Systems environment have been presented. In literature there is no model for a generic ES as well as a composition of ESs, so this analysis constitute the first step towards the formalization of a theory useful to control composability of heterogeneous devices.

Two approached have been presented: the first one is simple but not scalable, while the second one requires a biggest effort (mainly in the implementation) but assures a major scalability.

Both the approaches are used to model contex information of the pSHIELD system and to “simulate” its evolution over the time, even in response to fault or unexpected events. The objective of this formulation is to apply a control algorithm to control the value of the above mentioned context parameters, thus optimizing the choice of the system configuration.

On a deployment point of view (that will be addressed in the nSHIELD project) these models could be implemented in real time emulators running into the most powerfull node and could provide support to the overlay decisions.

6 References

[1] Henzinger T. A., “The Theory of Hybrid Automata”, *Proceedings of 11th Annual IEEE Symposium on Logic in Computer Science (LICS96)*, pp. 278-292, New Brunswick, New Jersey, 27-30 July, 1996

[2] Lunze J. and Lamnabhi-Lagarrigue F., “Handbook of Hybrid Systems Control – Theory, Tools, Applications”, *Cambridge University Press*, 2009

[3] Yang H., Jiang B. and Cocquempot V., “Fault Tolerant Control Design for Hybrid Systems”, *Lecturer Notes in Control and Inofrmation Sciences*, Springer, 2010

[4] Balduzzi F. and Kumar R., “Hybrid Automata Model of Manufacturing Systems and its Optimal Control Subject to Logical Constraints”, *International Journal of Hybrid Systems*, pp 61-80, volume 3, number 1, 2003.

[5] Balduzzi F., Giua A. and Seatzu C., “Modelling Automated Manufacturing Systems with Hybrid Automata”, *Proceedings of Workshop on Formal Methods and Manufacturing (WFMM99)*, Zaragoza, Spain, pp. 33-48, 6 September, 1999.

[6] Bemporad A. and Di Cairano, S., “Optimal Control of Discrete Hybrid Stochastic Automata”, *Proceedigns of ACM International Conference on Hybrid Systems: Computation and Control (HSCC05)*, pp. 151-167, Zurich, Switzerland, 9-11 March, 2005

[7] Bemporad A. Di Cairano S. and Giorgetti N., “Model Predictive Control of Hybrid Systems with Applications to Supply Chain Management”, *Proceedings of 49th Convegno Nazionale ANIPLA*, Naples, Italy, Nov, 2005

Annex 1 – pSHIELD Glossary

Concept	Description	Source
Application Processor (AP)	Main processing unit	5.1 pSHIELD Node –M01
Atomic pSHIELD SPD Component	Is a generic atomic SPD Functionality (innovative or legacy) provided by a pSHIELD device at node, network or middleware level	4.4.2.2 pSHIELD SPD components composition
Audit	Involves recognizing, recording, storing, and analyzing information related to SPD relevant activities. The resulting audit records can be examined to determine which SPD relevant activities took place	3.2 - The pSHIELD System: Application-Oriented Definitions- D 2.1.1
Automatic Web Service Composition and Interoperation	This task involves the automatic selection, composition, and interoperation of Web services to perform some complex task, given a high-level description of an objective.	http://www.w3.org/Submission/OWL-S/
Automatic Web Service Discovery	Is an automated process for location of Web services that can provide a particular class of service capabilities, while adhering to some client-specified constraints	http://www.w3.org/Submission/OWL-S/
Automatic Web Service Invocation	Is the automatic invocation of an Web service by a computer program or agent, given only a declarative description of that service, as opposed to when the agent has been pre-programmed to be able to call that particular service.	http://www.w3.org/Submission/OWL-S/
Availability	Refers to a system's readiness to provide correct service, whilst reliability refers to continuity of correct service, but these two attributes can considered as one because both guarantee the correct service with an error $e(t) < \varepsilon$.	4.6.2 Formalized conceptual model - M01-
Awareness	Capability of the Cognitive Radio to understand, learn, and predict what is happening in the radio spectrum, e.g., to identify the transmitted waveform, to localize the radio sources, etc.	5.3.2 Formal conceptual Model –M01-
Bridge	Is a network device that is at the link layer of the ISO / OSI model and translates from one physical media to another within the same local network.	-
Cognitive Radio	Is an intelligent wireless communication system that is aware of its surrounding environment (i.e., outside world), and uses the methodology of understanding-by-building to learn from the environment and to adapt its internal states to statistical variations in the incoming Radio-Frequency (RF) stimuli by making corresponding changes in certain operating parameters (e.g., transmit-power, carrier-frequency, and modulation strategy) in real-time, with two primary objectives in mind: (i) highly Reliable and Dependable communications whenever and wherever needed and (ii) efficient utilization of the radio spectrum.	5.3.2 Formal conceptual model- M01-

Concept	Description	Source
Common Criteria Approach	Is approach based in three fondamental part: • Assets to protect and in particular SPD attribute of these assets definition • Threats identifications (Fault Errors Failures); in our approach faults are grouped in HMF (FUA, NFUA) and NHMF. SPD functionalities (whose purpose is to mitigate threats) are implemented to meet pSHIELD SPD objectives.	5.7 Common Criteria Approach –M01-
Composability	Is the possibility to compose different (possibly heterogeneous) SPD functionalities (also referred to as SPD components) aiming at achieving in the considered system of Embedded System Devices a target SPD level which satisfies the requirements of the considered scenario.	4.4 Composability -M01-
Confidentiality	Refers to the property that information or data are not available to unauthorized persons or processes, or that unauthorized access to a system's output will be blocked by the system's filter. Confidentiality faults are mainly caused by access control problems originating in cryptographic faults, security policy faults, hardware faults, and software faults.	4.6.2 Formalized conceptual modelM01-
Contiki Operating System	Contiki is also an open source, highly portable, multi-tasking operating system for memory-efficient networked ESDs and WSNs	5.1.3.2 Nano, Micro and Personal Node operating systems -M01-
Control Algorithms	Retrieves the aggregated information on the current SPD status of the subsystem, as well as of the other interconnected subsystems, by the pS-CA interface connected to the Semantic Knowledge Representation; such retrieved information is used as input for the Control Algorithms. The outputs of the Control Algorithms consist in decisions to be enforced in the various ESDs included in the pSHIELD subsystem controlled by the Security Agent in question; these decisions are sent back via the pS-MS interface, as well as communicated to the other Security Agents on the Overlay, through the pS-OS interface.	3.1.1 pShield Functional Architecture –M01-

Concept	Description	Source
Core SPD Services	The core SPD services are a set of mandatory basic SPD functionalities provided by a pSHIELD Middleware Adapter in terms of pSHIELD enabling middleware services. The core SPD services aim to provide a SPD middleware environment to actuate the decisions taken by the pSHIELD Overlay and to monitor the Node, Network and Middleware SPD functionalities of the Embedded System Devices under the pSHIELD Middleware Adapter control.	5.5 -M01-Core SPD services
Cryptographic Algorithms	Algorithms to hiding the information, to provide security and information protection against different forms of attacks	5.2 Cryptographic algorithms –M01-
Dependability	Is a composite concept that encompasses the following attributes: Availability, Reliability, Safety Integrity, Maintainability	5 Reference SPD Taxonomy -pShield System requirement Specification-D 2.1.1-
Discovery	Provide to the pSHIELD Middleware Adapter the information, raw data, description of available hardware resources and services in order to allow the system composability	5.1.1.3 pSHIELD Node SPD components –M01-
Discovery Engine	it is in charge to handle the queries to search for available pSHIELD components sent by the Composition service.	5.5.2 formalized conceptual model -M01-
Discovery Protocol	it is in charge to securely discover all the available SPD components description stored in the Service Registry, using a specific protocol	5.5.2 formalized conceptual model -M01-
Error	Is defined as the part of a system's total state that may lead to a failure.	5.2-Fault Errors Failure - System Requirement Specification D 2.1.1-
Failure	Occurs when an error causes the delivered service to deviate from correct service.	5.2-Fault Errors Failure - System Requirement Specification D 2.1.1-
Fault	Is defined as a cause of an error	5.2-Fault Errors Failure - System Requirement Specification D 2.1.1-
Fault injection	This block has the responsibility to inject a fault into Demodulator	2 SPD Node Internal Demonstrator Structural Description SPDDemosv7-EB
Faults with Unauthorized Access	The class of Faults with unauthorized access (FUA) attempts to cover traditional security issues caused by malicious attempt faults. Malicious attempt fault has the objective of damaging a system. A fault is produced when this attempt is combined with other system faults.	3- Term and definition-M0.2: Proposal for the aggregation of SPD metrics during composition
Filter Engine	it is in charge to semantically match the query with the descriptions of the discovered SPD components	5.5.2 formalized conceptual model -M01-
Flash Memory	It stores the bit-stream and system status information	2 SPD Node Internal Demonstrator Structural Description SPDDemosv7-EB
Forecasting (Fault)	Mechanism that predicts faults so that they can be removed or their effects can be circumvented	-

Concept	Description	Source
Functional Component	Describes a functional entity that, in general, does not have necessarily a physical counterpart (e.g. a software functionality, a middleware service, an abstract object, etc.).	3.1.1 pSHIELD functional architecture –M01-
Gateway	Is a network device that operates at the network layer and above the ISO / OSI model. Its main purpose is to transmit network packets outside a local area network (LAN).Functional Component	-
Grounding	Provides details on how to interoperate with a service, via messages.	http://www.w3.org/Submission/OWL-S/
Health Status Monitoring (HSM)	Monitoring for checking the status of each individual component.	5.1 Pshield Node –M01
Hub	Is a concentrator, a network device that acts as a routing node of a data communications network	
Human-Made Faults	Human-made faults result from human actions. They include absence of actions when actions should be performed (i.e., omission faults). Performing wrong actions leads to commission faults. HMF are categorized into two basic classes: faults with unauthorized access (FUA), and other faults (NFUA).	3- Term and definition-M0.2: Proposal for the aggregation of SPD metrics during composition
Hybrid Automata	Is composed by automaton formulation hybrid formulation that Permit to choose the most suitable configuration rules for components that must be composed on the basis of the Overlay control algorithms.	5.6 Hybrid Automata –M01-
HYDRA Middleware	Middleware for Heterogeneous Physical Devices	5.1.3.2.3 Hydra Middleware -M01-
I/O Interface	(I/O) to connect to any peripheral and to the rest of the pSHIELD embedded functionalities.	5.1 Pshield Node –M01
Innovative SPD Functionalities	Reside in the pSHIELD Middleware, Network and Node Adapters. They are constituted by a variety of pSHIELD-specific components. Each pSHIELD-specific component. represents an innovative SPD functionality ad hoc developed for the pSHIELD project which is included in the pSHIELD Node, Network or Middleware Adapter.	4.5 Innovative SPD functionalities –M01-
Integrity	Refers to the absence of improper alteration of information. An integrity problem can arise if, for instance, internal data are tampered with and the produced output relies on the correctness of the data.	4.6.2 Formalized conceptual model -M01-
Legacy Device Component	i.e. the SPD functionalities already present in the legacy devices which can be accessed through the pSHIELD Adapters; they can be classified in Node, Network and Middleware Component according to whether they are included in a legacy Node/Network/Middleware which can be accessed through the corresponding pSHIELD Adapter.	4.4.2 Formalized Conceptual model -M01-

Concept	Description	Source
Legacy Embedded System Device (L-ESD)	It represents an individual, atomic physical Embedded System device characterized by legacy Node, Network and Middleware functionalities.	3.1.1 pShield Functional Architecture –M01-
Legacy Functionalities of L-ESD	Is a functionality partitioned into three subsets: - Node layer functionalities: hardware functionalities such as processors, memory, battery, I/O interfaces, peripherals, etc. - Network layer functionalities: communication functionalities such as connectivity, protocols stack, etc. - Middleware layer functionalities: firmware and software functionalities such as services, functionalities, interfaces, protocols, etc.	3.1.1 pShield Functional Architecture –M01-
Legacy Middleware Services	Includes all the legacy middleware services (i.e. messaging, remote procedure calls, objects/content requests, etc.) provided by the Legacy Embedded System Device which are not pSHIELD-compliant.	Par 3.1 pShield functional architecture -M01-
Legacy Network Services	Includes all the legacy network services (protocol stacks, routing, scheduling, Quality of Service, admission control, traffic shaping, etc.) provided by the Legacy Embedded System Device which are not pSHIELD-compliant.	Par 3.1 pShield functional architecture – M01-
Legacy Node Capabilities	Component includes all the legacy node capabilities (i.e. battery, CPU, memory, I/O ports, IRQ, etc.) provided by the Legacy Embedded System Device which are not pSHIELD compliant.	3.1.1 pShield Functional Architecture –M01-
Maintainability:	Ability to undergo modifications and repairs.	4.6.2 Formalized conceptual model -M01-
Mean	All the mechanisms that break the chains of errors and thereby increase the dependability of a system	-
Memory	Memory RAM, SRAM, DRAM,	5.1 Pshield Node –M01
Metadata	Information that describe set of data	-
Micro/Personal nodes	Are richer of nanonode in terms of hardware and software resources, network access capabilities, mobility, interfaces, sensing capabilities, etc.	5.1 Pshield Node –M01
Middleware Layer	Includes the software functionalities that enable the discovery, composition and execution of the basic services necessary to guarantee SPD as well as to perform the tasks assigned to the system (for example, in the railway scenario, the monitoring functionality)	3.1 The pSHIELD System: General Definitions - System Requirement Specification D 2.1.1-
Nano Nodes	Are typically small ESD with limited hardware and software resources, such as wireless sensors.	5.1 Pshield Node –M01-
Net Device	Components used to connect computers or other electronic devices	-
Network CAN	Control Area Network	-
Network LAN	Local Area Network	-

Concept	Description	Source
Network Layer	Includes the communication technologies (specific for the rail transportation scenarios) that allow the data exchange among pSHIELD components, as well as the external world. These communication technologies, as well as the networks to which pSHIELD is interconnected can be (and usually are) heterogeneous.	3.1 The pSHIELD System: General Definitions - System Requirement Specification D 2.1.1-
Network MAN	Metropolitan Area Network	-
Network VPN	Virtual Private Network	-
Network WAN	Wide Area Network	-
Node Layer	Includes the hardware components that constitute the physical part of the system.	3.1 The pSHIELD System: General Definitions - System Requirement Specification D 2.1.1-
Node Metrics / Health Status	It receives periodic health messages and metrics from the other blocks. This block contains the information about the full node configuration, metrics and health status. If e.g. the “Assertions” block detects some erroneous output, “Node Metrics / Health Status” block receives this information and must act accordingly.	2 SPD Node Internal Demonstrator Structural Description SPDDemosv7-EB
NonHuman-Made Faults	NHMF refers to faults caused by natural phenomena without human participation. These are physical faults caused by a system’s internal natural processes (e.g., physical deterioration of cables or circuitry), or by external natural processes. They can also be caused by natural phenomena	3- Term and definition-M0.2: Proposal for the aggregation of SPD metrics during composition
Non-Volatile Memory (NVM)	Memory ROM, EEPROM, FLASH, Hard Disk	5.1 Pshield Node –M01
Not Faults with Unauthorized Access	Human-made faults that do not belong to FUA. Most of such faults are introduced by error, such as configuration problems, incompetence issues, accidents, and so on.	3- Term and definition- M02--M0.2: Proposal for the aggregation of SPD metrics during composition
Overlay	Consists of a set of SPD Security Agents, each one controlling a given pSHIELD subsystem.	4.2 Overlay –M01-
Overlay Layer	The “embedded intelligence” that drives the composition of the pSHIELD components in order to meet the desired level of SPD. This is a software layer as well.	3.1 The pSHIELD System: General Definitions - System Requirement Specification D 2.1.1-
Physical Component	Describes an entity that can be mapped into a physical object (e.g. a hardware component).	3.1.1 pSHIELD functional architecture –M01-
Power Management (PM)	Module for managing power sources, monitoring power consumption, etc.	5.1 Pshield Node –M01
Power nodes	Is a node that Offer high performance computing in one self-contained board offering data storage, networking, memory and (multi-)processing.	5.1 Pshield Node –M01
Prevention (Fault)	Mechanism that deals with preventing faults incorporated into a system	
Privacy	It is a information that must be accessed only by authorized users, for confidentiality reasons.	-

Concept	Description	Source
pSHIELD Adapter	Is a technology dependent component in charge of interfacing with the legacy Node, Network and Middleware functionalities (through the MS, NS and NC interfaces). The legacy functionalities can be enhanced by the pSHIELD Adapter in order to make them pSHIELD-compliant, i.e. they become SDP legacy device components. In addition, the pSHIELD Adapter includes Innovative SPD functionalities which are SPD pSHIELD-specific components, which can be composed with other SPD components. The pSHIELD Adapter exposes the technology independent pSHIELD Middleware layer functionalities that are used by the Security Agent component.	3.1.1 pShield Functional Architecture –M01-
pSHIELD Communication	This block interfaces SPD Node to pShield Network. It is composed by: Ethernet interface: it allows a communication data interface based on a TCP/IP protocol Message encoding/decoding: it receives the commands from pShield network, decodes them, and acts accordingly. It also sends messages to the network.	2 SPD Node Internal Demonstrator Structural Description SPDDemosv7-EB
pSHIELD Demonstrator	Demonstrator for the project that Have the task To monitor on-carriage environment; To integrate the sensors at the wagon with the M2M platform; To allow secure interoperability of sensor information (between railway infrastructure and third party service provider).	5.8 pShield Demonstrator -M01-
pSHIELD Embedded System Device (pS-ESD)	It is a L-ESD equipped at least with the minimal set of pSHIELD functionalities at Middleware Layer. The pS-ESD exposes the same functionalities as the L-ESD plus an additional interface: the pSHIELD Middleware layer services.	3.1.1 pShield Functional Architecture –M01-
pSHIELD Middleware Adapter	Is a component partitioned in the Core SPD services and in the Innovative SPD functionalities. These two components are linked through the pS-MS interface. The pSHIELD Middleware Adapter should also carry into operation the decisions taken by the Overlay and communicated via the pS-MS interface by actually composing the discovered SPD functionalities. The pSHIELD Middleware Adapter includes a set of Innovative SPD functionalities interoperating with the legacy ESD middleware services (through the MS interface) in order to make them discoverable and composable SPD functionalities.	3.1.1 pShield Functional Architecture –M01-

Concept	Description	Source
pSHIELD Multi-Layered Approach	The pSHIELD multi-layered approach considers the partition of a given Embedded System into three technology-dependent horizontal layers: the node layer (meaning the hardware functionalities), the network layer (meaning the communication functionalities) and the middleware layer (meaning the software functionalities).	4.1 pSHIELD multi-layered approach – M01-
pSHIELD Network Adapter	Includes a set of Innovative SPD functionalities interoperating with the legacy ESD network services (through the NS interface) and the pSHIELD Node Adapter (through the pS-NC interface) in order to enhance them with the pSHIELD Network layer SPD enabling technologies (such as Smart Transmission). This adapter is also in charge to provide (through the pS-NS interface) all the needed information to the pSHIELD Middleware adapter to enable the SPD composability of the Network layer legacy and Network pSHIELD-specific functionalities. Moreover, the pSHIELD Network Adapter translates the technology independent commands, configurations and decisions coming from the pS-NS interface into technology dependent ones and enforce them also to the legacy Network functionalities through the NS interface.	3.1.1 pShield Functional Architecture – M01-
pSHIELD Node	Is an Embedded System Device (ESD) equipped with several legacy Node Capabilities and with a pSHIELD Node Adapter. A pSHIELD Node is deployed as a hardware/software platform, encompassing intrinsic, Innovative SPD functionalities, providing proper services to the other pSHIELD Network and Middleware Adapters to enable the pSHIELD Composability and consequently the desired system SPD	5.1 pshied Node –M01-

Concept	Description	Source
pSHIELD Node Adapter	Includes a set of Innovative SPD functionalities interoperating with the legacy ESD node capabilities (using the NC interface) in order to enhance them with the pSHIELD Node layer SPD enabling technologies (such as FPGA Firmware and Lightweight Asymmetric Cryptography). This adapter is in charge to provide (through the pS-NC interface) all the needed information to the pSHIELD Middleware adapter to enable the SPD composability of the Node layer legacy and Node pSHIELD-specific functionalities. Moreover, the pSHIELD Node Adapter translates the technology independent commands, configurations and decisions coming from the pS-NC interface into technology dependent ones and enforce them also to the legacy Node functionalities through the NC interface.	3.1.1 pShield Functional Architecture –M01-
pSHIELD Proxy (pS-P)	Is a technology dependent component of a pS-SPD-ESD that, interacting with the available legacy Node, Network and Middleware capabilities and functionalities (through the NC, NS and MS interfaces, respectively), provides all the needed pSHIELD enhanced SPD functionalities.	3.1.1 pShield Functional Architecture –M01-
pSHIELD SPD Embedded System Device (pS-SPD-ESD):	It is a pS-ESD equipped at least with the minimal set of pSHIELD Overlay functionalities. The pS-SPD-ESD exposes the same functionalities as the pS-ESD plus an additional interface: the pSHIELD Overlay layer SPD services provided by a so-called Service Agent operating in that ESD.	3.1.1 pShield Functional Architecture –M01-
pSHIELD Subsystem (pS-S)	Is an architecture of a set of Embedded System Devices including several L-ESD, connected to several pS-ESD and one and only one pS-SPD-ESD. Connections between two generic ESDs (L-ESD, pS-ESD or pS-SPD-ESD) can be performed, by means of legacy functionalities at Node, Network and/or Middleware layer, through the so-called NC, NS and MS functionalities, respectively.	3.1.1 pShield Functional Architecture –M01-

Concept	Description	Source
pSHIELD-Specific Components	It is i.e. the innovative SPD functionalities ad hoc developed for the pSHIELD project which are included in the pSHIELD Adapters. They can be classified in Node Network and Middleware pSHIELD-specific components according to whether they are included in the pSHIELD Node, Network or Middleware Adapter. They can be directly accessed by pSHIELD Middleware Core SPD Services through the pSNC, pS-NS and pS-MS interfaces.	4.4.2 Formalized Conceptual model -M01-
Query Preprocessor	it is in charge to enrich the query sent by the Composition service with semantic information related to the peculiar context.	5.5.2 formalized conceptual model -M01-
Reconfigurability	Provide self-configuration of some internal parameters according to the observed radio spectrum.	5.3.2 Formal Conceptual Model -M01-
Reconfiguration / Recovery	This block runs at the PPC static core. It must receive periodically health status information, otherwise it restarts the system	2 SPD Node Internal Demonstrator Structural Description SPDDemosv7-EB
Reconfiguration/Recovery Controller	This is a hard processor or microcontroller, responsible for either reconfiguring the node or recovering in case of an error. It	5.1.1.2 –Detailded Module description –M01-
Recovery Watchdog Timer (RWDT)	Timer for restarting recovery if no activity is detected from the SHSM.	5.1 Pshield Node –M01
Reliability	Continuity of correct service.	4.6.2 Formalized conceptual model-M01-
Removal (Fault)	mechanism that permits to the system to record failures and remove them via a maintenance cycle	
Repeater	A digital device that amplifies, reshapes, retimes, or performs a combination of any of these functions on a digital input signal for retransmission	-
Router	Is a device that forwards data packets between telecommunications networks, creating an overlay internetwork. A router is connected to two or more data lines from different networks.	-
Rules for Discovery, Configuration and Composition of the SPD Components.	Design and implementation of the Control Algorithms which, on the basis of the sensed metadata (i.e) on the basis of the ontological description (possibly semantically enriched) of the SPD Components provide Rules for discovery, configuration and composition of the SPD components.	4.4 Composability

Concept	Description	Source
Safety	Refers to absence of catastrophic consequence on System users end environment. A safety fault can arise if, for instance, an unauthorized system access can cause the possibility of human lives being endangered.	4.6.2 Formalized conceptual modelM01-
SDP Network	Is a Network implementable and interoperable with standard networks to comply the main business cases of the application scenarios.	5.1.3 Nano, Micro and Personal nodes –M01-
Seamless Approach	Common approach which leaves out of consideration the specificity of the underlying technologies providing enriched SPD functionalities to heterogeneous Embedded Systems	4.3 Seamless Approach -M01-
Secure Service Discovery	Allows any pSHIELD Middleware Adapter to discover in a secure manner the available SPD functionalities and services over heterogeneous environment, networks and technologies that are achievable by the pSHIELD Embedded System Device (pS-ESD) where it is running.	5.5 Core SPD Service –M01-
Secure/Privacy (SP)	Module to perform security and privacy actions, such as encryption, decryption, key generation, etc.	5.1 Pshield Node –M01
Security	Is a composite of the attributes of confidentiality, integrity (in the security context, “improper” means “unauthorized"), and availability (for authorized actions only),	5 Reference SPD Taxonomy -pShield System requirement Specification-D 2.1.1-
Security Agent	Is a technology-independent component in charge of aggregating the information coming from the pSHIELD Middleware Services provided by the internal pSHIELD Adapter or by other pSHIELD Proxies located in the same subsystem. The Security Agent is also in charge of gathering the information coming from other Security Agents connected on the same Overlay (through the pS-OS interface). The Security Agent includes proper control algorithms working on the basis of the available information; the decisions taken by these Control Algorithms are enforced through the pS-MS and the pS--OS interfaces.	3.1.1 pShield Functional Architecture –M01-
Semantic Database	It holds any semantic information related to the pSHIELD components (interface, contract, SPD status, context, etc.).	5.5.2 formalized conceptual model -M01-
Semantic Engine (Reasoner)	Enable interoperability within Middleware Layer and rule based discovery and composition within Overlay Agents.	5.4 Semantic Model –M01-

Concept	Description	Source
Semantic Knowledge Repository	Is (i.e. a database) that storing the dynamic, semantic, enriched, ontological aggregated representation of the SPD functionalities of the pSHIELD subsystem controlled by the SPD Security Agent;	4.2 Overlay -M01-
Semantic Knowledge Representation	Is in charge of bi-directionally exchanging technology independent (and semantic enriched) information from the pS-MS and the pS-OS interfaces. It is also in charge to provide such information via the pS-SKR interface to the Control Algorithms component.	3.1.1 pShield Functional Architecture –M01-
Semantic Model	It is a conceptual model in which semantic information is included.	-
Sensor/Actuator	Are represented by the Core SPD Services lying at the pSHIELD Middleware layer.	4.2.2 Formalized concept Model –M01-
Sensors/Actuators	Represent the Core SPD Services lying at the pSHIELD Middleware layer.	4.2.2 Formalized conceptual Model-M01-
Service Composition	Is in charge to select atomic SPD services that, once composed, provide a complex and integrated SPD functionality that is essential to guarantee the required SPD level. The service composition is a pSHIELD Middleware Adapter functionality that cooperates with the pSHIELD Overlay in order to apply the configuration strategy decided by the Control Algorithms residing in the pSHIELD Security Agent.	5.5 Core SPD Services –M01-
Service Grounding	Specifies the details of how an agent can access a service-details having mainly to do with protocol and message formats, serialization, transport, and addressing	http://www.w3.org/Submission/OWL-S/
Service Orchestration	Deploy, execute and monitoring SPD services.	5.5 Core SPD Services –M01-
Service Profile	Tells "what the service does", in a way that is suitable for a service-seeking agent (or matchmaking agent acting on behalf of a service-seeking agent) to determine whether the service meets its needs.	http://www.w3.org/Submission/OWL-S/
Services Model	Tells a client how to use the service, by detailing the semantic content of requests, the conditions under which particular outcomes will occur, and, where necessary, the step by step processes leading to those outcomes.	http://www.w3.org/Submission/OWL-S/
Services Registry	It acts as a database to store the service entries	5.5.2 formalized conceptual model -M01-
Smart SPD Driven Transmission	New advances signal processing techniques.	5.3 Smart SPD Driven Transmission –M01-
SOA	Service-oriented architecture	-
Software Agent	Permit a computer-interpretable description of the service.	http://www.w3.org/Submission/OWL-S/
Software Defined Radio (SDR)	Software programmable Components	5.3 Smart SPD driven transmission-M01-

Concept	Description	Source
Software Wireless Sensor Networks (WSN)	Software part that can be layered into three levels: sensor software, node software and sensor network software.	5.1.1.3 Specific SPD Considerations for Wireless Sensor Networks
SPD	Security Privacy Dependability	
SPD Component	Is defined as a functionality which (i) offers a given SPD service through an interface which can be semantically described according to the provided SPD Metrics, (ii) can be accessed through the pSHIELD Middleware Core SPD Services for being configured (if necessary) and activated (or deactivated).	4.4 Composability -M01-
SPD Metrics	Is the possibility to identify and quantify the SPD properties of each component, as well as the SPD properties of the overall system. SPD Metrics allow (i) users to define in an univocal way the requirements for the specific application, (ii) to describe the SPD level provided by the components, and (iii) to compute the SPD level achieved by the system through the Composability mechanism.	4.6 SPD Metrics-M01
SPD Node	It is composed by the following sub- blocks: FM Signal Acquisition: this blocks principally handles the receiving of data samples from “FM Signal Generator” and pre-processes the data to feed to the “Demodulation Processing” block. This block provides also periodic status & metrics information to the “Node Metrics/Health Status” block. Demodulation Processing: it is responsible for the demodulation processing of the data coming from the “FM Signal Acquisition“	2 SPD Node Internal Demonstrator Structural Description SPDDemosv7-EB
SPD Security Agent	Consists of two key elements: (i) the Semantic Knowledge Repository (i.e. a database) storing the dynamic, semantic, enriched, ontological aggregated representation of the SPD functionalities of the pSHIELD subsystem (ii) the Control Algorithms generating, on the grounds of the above representation, key SPD-relevant decisions (consisting, as far as the Composability feature is concerned, in a set of discovery, configuration and composition rules).	
SPD Status	It represents the current SPD level associated to the function.	4.4.2 Formalized Conceptual Model – M01-
Special Purpose Processor	(SPP) module for any pre- or post-processing, such as compression/decompression, conversion, etc.	5.1 Pshield Node –M01

Concept	Description	Source
Stable Storage	Used for storing the status of the system, a bit stream to program an FPGA, and/or the software for system start-up, operating system and application. it receives from each block check-pointing data. It is able to perform a stable write with this data (write on a circular buffer on flash memory, and then validate the just written data). On system restart, this module is able to recover the last valid data.	5.1 Pshield Node –M01
Switch	Is a computer networking device that connects network segments.	-
System Health Status Monitoring (SHSM)	Monitoring for checking the status of the whole system.	5.1 Pshield Node –M01
The Security/Privacy controller	Consists of one or more modules able to perform different security-related functionalities, such as Data Encryption, Data Decryption, Generation of Cryptographic Keys, etc	5.1.1.2 –Detailded Module description –M01-
Threat	Include faults, errors and failures, as well as their causes, consequences and characteristics.	5.2-Fault Errors Failure - System Requirement Specification D 2.1.1-
TinyOS	This operating system (OS) is a free and open source operating system and platform that is designed for WSNs.	5.1.3.2 Nano, Micro and Personal Node operating systems -M01-
Tolerance (Fault)	System architecture that deals with putting mechanisms in place that will allow a system to still deliver the required service in the presence of faults, although that service may be at a degraded level	-
WSDL	Web Services Description Language	-

Table 2 pSHIELD Glossary

Annex 2 - Core SPD Services implementation: OSGi Source Code

package eu.artemis.shield.composition.compositionmanager.ICompositionManager.html

```
package eu.artemis.shield.composition.compositionmanager;

import java.util.HashMap;
import java.util.Hashtable;

/**
 * This it the main interface to manage the User Agent service discovery.
 * This interface supplies a method to retreive the XML files describing the service composition
 *
 * @author <a>Davide Migliacci</a>
 *
 * @todo ...
 */

public interface ICompositionManager {

  public void runBundle ( Hashtable bundle_properties, int level, HashMap ht );

}
```

package eu.artemis.shield.composition.compositionmanager.impl.Activator.html

```
/**
 * pSHIELD
 * Service Discovery
 *
 * @author Vincenzo Suraci
 * Department of System and Computer Science (DIS)
 * University of Rome "Sapienza"
 * Via Ariosto, 25
 * 00184, Rome, IT
 *
 * phone: +39 340 156 22 58
 * email: vincenzo.suraci@dis.uniroma1.it
 *
 * Created on 16-May-2007
 * Version 1.0
 *
 */

package eu.artemis.shield.composition.compositionmanager.impl;

import org.osgi.framework.BundleContext;
import org.osgi.framework.BundleActivator;

import eu.artemis.shield.composition.compositionmanager.ICompositionManager;

public class Activator implements BundleActivator
{
  private static BundleContext bc = null;
  private CM cm = null;

  public void start(BundleContext bc) throws Exception
  {
    Activator.bc = bc;
    cm = new CM(Activator.bc);

    bc.registerService(ICompositionManager.class.getName(),cm,null);

    System.out.println("Composition Manager Started");
  }

  public void stop(BundleContext bc) throws Exception
  {
    Activator.bc = null;
    cm.exit();

    System.out.println("Composition Manager Stopped");
  }

}
```

package eu.artemis.shield.composition.compositionmanager.impl.CM.html

```
/**
 * pSHIELD
 * Service Composition
 *
 * @authors Davide Migliacci, Vincenzo Suraci
 * Department of System and Computer Science (DIS)
 * University of Rome "Sapienza"
 * Via Ariosto, 25
 * 00184, Rome, IT
 *
 * Updated on 19-May-2011
 * Version 1.0
 *
 */

package eu.artemis.shield.composition.compositionmanager.impl;
/**
 *
 * The present class shows how a pSHIELD OSGi component
 * could use the potentiality offered by the pSHIELD
 * Service Discovery Framework. It interfaces with the
 * Generic Discovery Manager to discover the services
 * available in the (pSHIELD) network.
 */
import java.util.Dictionary;
import java.util.HashMap;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Vector;

import org.osgi.framework.Bundle;
import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceReference;

import eu.artemis.shield.composition.compositionmanager.ICompositionManager;
import eu.artemis.shield.discovery.gdm.interfaces.IGenericDiscovery;
import eu.artemis.shield.discovery.pdm.IService;
import eu.artemis.shield.discovery.pdm.IServiceList;
import eu.artemis.shield.discovery.pdm.IServiceProperty;
import eu.artemis.shield.discovery.pdm.IServicePropertyList;

public class CM implements ICompositionManager
{
  private BundleContext bc;
  private CMGUI gui = null;

  private final int API_TYPE = 1;
  private final int IMPL_TYPE = 2;

  int SPD_lvl;

  private Vector impl_bundles_to_start = new Vector();

  private IGenericDiscovery gd = null;

  public CM(BundleContext bc)
  {
    this.bc = bc;

    gui = new CMGUI(bc,this);
    gui.setVisible(true);
  }

    /**
   * This function takes an Api Bundle symbolic name and create its Impl Bundle symbolic name.
   * @param bundleApiName Symbolic API name
   * @return Symbolic IMPL name
   */
  private String getBundleImplName(String bundleApiName)
  {

    int indexStart = bundleApiName.lastIndexOf(".");
    int indexEnd = bundleApiName.lastIndexOf("-");

    if (indexEnd > 0){
```

```
      String bundleImplName = bundleApiName.substring(indexStart+1, indexEnd) + "-IMPL";
      return bundleImplName;

    }
    else
      // SLP Bundle
      return bundleApiName.substring(1, bundleApiName.length()-1)+"-IMPL";
  }

  private Bundle installApi( Hashtable ht, String bundle_type, HashMap security ) {

    Vector implBundleNames = new Vector();

    String initial_impl_bundle = ((String) ( (Vector) ht.get("Project Name")).elementAt(0))+"-
IMPL";

    // Display counter
    int j = 1;

    // Get the imported api
    Vector imp_packages = ( Vector ) ht.get( "Import" );

    gui.append("\n");
    gui.appendts("Installing Import Api of " + initial_impl_bundle);

    boolean found = false;

    Iterator it = imp_packages.iterator();

    // For each import bundle
    while ( it.hasNext() ){

      found = false;

      String imp_package = ( String ) it.next();

      gui.append("\n");
      gui.appendts ("API : " + j++ + "/" + imp_packages.size() + " -
 looking for : " + imp_package );

      // Get all installed bundles in the framework
      Bundle[] installed_bundles = bc.getBundles();

      if ( installed_bundles != null ) {

        // For each installed bundle
        for ( int i = 0; i < installed_bundles.length; i++ ) {

          Bundle temp = ( Bundle ) installed_bundles[i];

          Dictionary bundle_manifest = temp.getHeaders();

          // Check the exported bundles of the only API Bundles
          if ( bundle_manifest.get( "Bundle-Category" ) != null &&  bundle_manifest.get( "Bundle-
Category" ).equals( "API" ) ) {

            String imported_packages = ( String ) bundle_manifest.get( "Export-Package" );

            // If there is an installed bundle that export the requested package, we can procede
whit the other requested packages
            if ( imported_packages.contains( imp_package ) ) {

              gui.appendts("The installed bundle [" + bundle_manifest.get( "Bundle-
Name" ) +"] exports the requested package.");

              String name_bundle_found = getBundleImplName( ( String ) bundle_manifest.get( "Bund
le-Symbolicname" ) );

              if( !name_bundle_found.equals( initial_impl_bundle ) ){
                // Add the IMPL bundle Symbolic Name into the vector if security level is satisfi
ed

                implBundleNames.add( name_bundle_found );
              }

              found = true;
```

```
              break;

            }

          }
        }

        // If there aren't installed bundle that export the requested package, we need to search
one in the SLP
        if ( !found ) {

          gui.appendts("There aren't installed bundles that export the requested package : " + im
p_package);

          gui.appendts("Trying to search the package into the SLP...");

          Vector bundles = getBundleByExportedPackage( imp_package, bundle_type, API_TYPE );

          if( bundles != null && !bundles.isEmpty()) {

              Iterator it2 = bundles.iterator();

              // Use the first package founded
            if ( it2.hasNext() ){

                Hashtable ht_tmp = (Hashtable) it2.next();

                gui.appendts ( "Found : " + ht_tmp.get( "Service Name" ).toString() );
                gui.appendts ( "> It exports : " + ht_tmp.get( "Export" ).toString() + "\n");

                Bundle tmp = installBundle( ht_tmp );

                if ( tmp != null ){

                  gui.appendts ( tmp.getSymbolicName() + " : Installation completed." );

                  String name_bundle_found = getBundleImplName( ht_tmp.get( "Project Name" ).toSt
ring() );

                  String service_name = ht_tmp.get("Project Name").toString().substring(1,ht_tmp.
get("Project Name").toString().length()-1 );

                  if( !name_bundle_found.equals( initial_impl_bundle ) ){
                  // Add the IMPL bundle Symbolic Name into the vector

                    if (security.containsKey(service_name)){

                      gui.appendts("Security Level" + security.get("Cryptography") + " " + securi
ty.get("Accounting") + "\n" );

                      gui.appendts("Security Level" + (Integer)ht_tmp.get("SPD Level"));

                    }

                    implBundleNames.add( name_bundle_found );
                }

                } else {

                  gui.appendts ( "ERR: Installation aborted !!!" );

                }

              }

          } else {

            gui.appendts ( "ERR: There aren't available bundles in SLP that export the " + imp_pa
ckage + " package" );

          }
        }
      }
    }


    if ( installImpl( implBundleNames, bundle_type, security ) ){
```

```
      gui.appendts ( initial_impl_bundle + " import installation completed.\n" );
    }
    else{
      gui.appendts ( initial_impl_bundle + " import installation aborted.\n" );
      return null;
    }

    Bundle bundle_inst = installBundle( ht );

    if ( bundle_inst != null ){

      gui.appendts ( "IMPL installed : " + ht.get( "Service Name" ) );

    } else {


      gui.appendts ( "IMPL installation ABORTED : " + ht.get( "Service Name" ) );
      return null;

    }

    return bundle_inst;

  }

  /**
   * This function installs
   * @param bundles_name
   * @param bundle_type
   * @return
   */
  private boolean installImpl( Vector bundles_name, String bundle_type, HashMap parameter ){

    Bundle tmp = null;

    Iterator it = bundles_name.iterator();

    int y = 1;

    // For each import bundle
    while ( it.hasNext() ){

      boolean found = false;

      String impl_bundle =  ( String ) it.next();

      gui.appendts ("IMPL : " + y++ + "/" + bundles_name.size() + " -
 looking for : " + impl_bundle );

      // Get all installed bundles in the framework
      Bundle[] installed_bundles = bc.getBundles();

      if ( installed_bundles != null ) {

        // For each installed bundle
        for ( int i = 0; i < installed_bundles.length; i++ ) {

          tmp = ( Bundle ) installed_bundles[i];

          Dictionary bundle_manifest = tmp.getHeaders();

          // Check the IMPL bundles
          if ( bundle_manifest.get( "Bundle-Category" ) == null ) {

            String imp_bundles = ( String ) bundle_manifest.get( "Bundle-Symbolicname" );

            // If there is an installed bundle that export the requested package, we can procede
 whit the other requested packages
            if ( imp_bundles != null && imp_bundles.contains( impl_bundle ) ) {

              gui.appendts ( "The installed bundle [" + bundle_manifest.get( "Bundle-
 Name" ) +"] implements the requested bundle." );

              impl_bundles_to_start.add( tmp );

            }

            else found = false;
```

```
          }

        }
      }

      // If there aren't installed bundle that implements the requested package, we need to searc
h one in the SLP
      if ( !found ) {

        gui.appendts ( "There aren't installed bundles that implements the requested package : "
+ impl_bundle );

        gui.appendts ( "Trying to search the package into the SLP...\n" );

        Vector bundles = getBundleByExportedPackage( impl_bundle, bundle_type, IMPL_TYPE );

        if( bundles != null && !bundles.isEmpty()) {

            Iterator it2 = bundles.iterator();

            // Use the first package founded
          if ( it2.hasNext() ){

              Hashtable ht = (Hashtable) it2.next();

              gui.appendts ( "Found : " + ht.get( "Service Name" ).toString() );
              gui.appendts ( "> It implements : " + impl_bundle + "\n");

              tmp = installApi( ht, bundle_type, parameter );

              if ( tmp != null ){

                gui.appendts ( "Installation completed." );
                impl_bundles_to_start.add( tmp );

              } else {

                gui.appendts ( "ERR: Installation aborted !!!" );

                return false;

              }

            }

        } else {

          gui.appendts ( "ERR: There aren't available bundles in SLP that implement the " + impl_
bundle );

          return false;

        }

      }

    }

    return true;

  }

  /**
   * This function try to install and run a bundle from its jar url
   */

  public void runBundle ( Hashtable bundle_properties, int level, HashMap bundles){

    String bundle_name = ( String ) ( ( Vector ) bundle_properties.get( "Service Name" ) ).elemen
tAt(0) ;

    String bundle_jar_url = ( String ) ( ( Vector ) bundle_properties.get( "Jar Url" ) ).elementA
t(0) ;

    String bundle_type = ( String ) ( ( Vector ) bundle_properties.get( "Type" ) ).elementAt(0);

    Integer bundle_security = (Integer)((Vector) bundle_properties.get("SPD Level")).elementAt(0)
```

```
;

    Bundle initial_bundle = installApi ( bundle_properties, bundle_type, bundles );

    Iterator it = impl_bundles_to_start.iterator();

    int y = 1;

    while ( it.hasNext() ){

      Bundle tmp =  ( Bundle ) it.next();
      gui.append( "[" + y++ + "]" );
      try{

        tmp.start();

      } catch (Exception e){

        gui.appendts ( "ERR: Run command aborted !!!" );
        System.out.println ( e.getMessage() );
      }

    }

    if ( initial_bundle != null ){

      Dictionary bundle_manifest = initial_bundle.getHeaders();

      Integer parameter = (Integer) ( ( Vector ) bundle_properties.get( "SPD Level" ) ).elementAt
(0);

      String s = (String) bundle_manifest.get("Bundle-Name");

      gui.appendts ( "Initial bundle installed." );
      gui.appendts ( "Running : " + bundle_properties.get( "Service Name" ) );

      try
      {
        initial_bundle.start();

      } catch (Exception e){

        gui.appendts ( "ERR: Run command aborted !!!" );
        System.out.println ( e.getMessage() );
      }

    } else {

      gui.appendts ( "ERR: Run command aborted !!!" );

    }

  }

  /**
   * This function install the bundle using its JAR URL
   * @param hashtable of the bundle to be installed
   * @return the bundle if the installation is completed, null if the installation crashed
   */

  private Bundle installBundle ( Hashtable bundle ){

    // Try to install the selected bundle from its jar url
    try{

      gui.appendts("Trying to install "+ bundle.get( "Service Name" ) +" from its JAR URL...");

      Bundle b = bc.installBundle( ( ( String ) ( (Vector) bundle.get( "Jar Url" ) ).elementAt(0)
 ) );

      return b;

    }catch (Exception e){

      System.out.println( e.getCause() );
      return null;

    }
```

```
  }


  /**
   * @authors Davide Migliacci, Vincenzo Suraci
   *
   * This function try to discover a bundle that exports the requested package
   *
   * @param package_name : The name of the requested package
   * @param service_type : The service type
   * @param bundle_type : The bundle type ( API or IMPL ) to find
   * @return : A vector of Hashtables. One Hashtable for each founded bundle that exports the req
uested package. Return null if there are not bundles
   */
  private Vector getBundleByExportedPackage (String package_name, String service_type, int bundle
_type)
  {
    Vector services = serviceDiscovery(service_type, null, false);
    Vector services_discovered = new Vector();

    if ( services != null && !services.isEmpty() ) {

      Iterator it = services.iterator();

      // For each API bundles available into the SLP
        while ( it.hasNext() ){

          Hashtable ht = ( Hashtable ) it.next();

          switch ( bundle_type ) {
            case API_TYPE :
              if (   ht.containsKey( "Export" ) && ht.containsKey( "Api" ) && ht.containsKey( "Im
pl" )

                  // Search only into the simple API Bundles and the API&IMPL Bundles
                    && ( ( Boolean ) ( (Vector) ht.get( "Api" ) ).elementAt(0) ).booleanValue()

                ){
                String exported_packages = ( String ) ( ( Vector ) ht.get( "Export" ) ).elementAt
(0) ;

                if ( exported_packages.contains( package_name ) ) {

                  services_discovered.add( ht );

                }
                }
              break;
            case IMPL_TYPE :
              if (   ht.containsKey( "Export" ) && ht.containsKey( "Api" ) && ht.containsKey( "Im
pl" )

                  // Search only into the simple API Bundles and the API&IMPL Bundles
                    && ( ( Boolean ) ( (Vector) ht.get( "Impl" ) ).elementAt(0) ).booleanValue(
)

                  ){

                if ( ht.get( "Project Name" ) != null ){

                  String implement_packages = ( String ) ( ( Vector ) ht.get( "Project Name" )
).elementAt(0) ;

                    if ( implement_packages.equals( package_name.substring(0, package_name.leng
th()-5 ) ) ) {

                      services_discovered.add( ht );

                    }
                }
                  }
                break;
            }
          }

      }

    return services_discovered;
```

```
  }


  /**
   * This function find the service available for a specified service type and an array of keywor
ds to do a better filter
   * @param type : the service type to search
   * @param kw : an array of keywords to search
   * @param filter_api : set false if you need all the available bundles, true if you want only t
he API bundles
   * @return a vector of found services ( with no API bundles )
   */
  private Vector serviceDiscovery(String type, String[] kw, boolean filter_api)
  {
    Vector discovered_services = new Vector();
    try
    {

      gui.append("- Service discovery with a specified service type\n");
      ServiceReference[] gdmList = findGenericDiscoveryModuleImplementations();

      if (gdmList != null)
      {
        /*
         * We ignore the possbility to have more than one GDM implementation.
         * JUST USE THE FIRST ONE...
         */
        gd = (IGenericDiscovery) bc.getService(gdmList[0]);

        /*
         * We are ready to start the discovery process!
         */

        /*
         * Set an array of keywords, useful to better filter
         * the services
         */
        if ( kw == null )
          kw = new String[0];
          //kw[0]="gui=false";

        for(int i = 0; i<kw.length; i++)
        {
          gui.append("KEYWORDS " + i + "--> " + kw[i].toString() + "\n");
        }
        /*
         * LET pSHIELD DISCOVER THE SERVICES
         */

        gui.append("- Looking for Services ...");

        // LinkedList ll = gd.findServices(vid, type, kw);
        String CDQL =
          "SELECT default" +
          "FROM default" +
          "SERVICETYPE " + type +
          "LANGUAGE en_gb" +
          "WHERE " +
          "USING slp";
        String SPARQL = "";
        LinkedList query_output = null;
        IServiceList sl = gd.findServices(CDQL, SPARQL, query_output);

        if ( sl != null )
        {
          if ( sl.isEmpty() )
          {
            // NO SERVICES HAVE BEEN DISCOVERED
            gui.append("- No services found !");
          }
          else
          {
              //Iterator it = sl.iterator();
              //while (it.hasNext())
              for (int i = 0; i < sl.size(); i++)
              {
                try
```

```
                {
                  IService s = sl.getService(i);
                  String url = s.getOWLSURL();
                  System.out.println("service URL #" + i + " = " + url);
                  IServicePropertyList spl = s.getProperties();
                  Hashtable ht = new Hashtable();
                  for (int j=0; j<spl.size(); j++)
                  {
                    try
                    {
                      IServiceProperty sp = spl.getServiceProperty(j);
                      System.out.print("> attribute #" + j + " --> " + sp.getName() + "=");
                      Vector values = sp.getValues();
                      if (values != null)
                      {
                        for (int k=0; k<values.size(); k++)
                        {
                          Object obj = values.get(k);
                          if (obj != null)
                          {
                            if (k > 0) System.out.print(",");
                            System.out.print(obj.toString());
                          }
                        }
                      }
                      ht.put(sp.getName(), values);
                      System.out.println("");
                    }
                    catch (IndexOutOfBoundsException ioobe)
                    {
                      ioobe.printStackTrace();
                    }
                  }
                  if (ht.containsKey("Impl") && ht.containsKey("Api"))
                  {
                    if (filter_api)
                    {
                      boolean has_impl = ((Boolean)((Vector)ht.get("Impl")).elementAt(0)).boolean
Value();
                      if (has_impl) discovered_services.add(ht);

                    }
                    else
                    {
                      // Insert an HashTable for each discovered service
                      discovered_services.add( ht );
                    }
                  }
                }
                catch (IndexOutOfBoundsException ioobe)
                {
                  ioobe.printStackTrace();
                }
              }
            }
          }
          else
          {
            //NO SERVICES HAVE BEEN DISCOVERED
            gui.append("NO SERVICE FOUND!");
          }

        }
        else
        {
          /*
           * It was not possible to find a suitable implementation of IGenericDiscovery
           */
          gui.append("No Bundles implement the IGenericDiscovery interface!\n");
        }
      }
      catch (Exception e)
      {
        /*
         * Something went wrong!
         * It was not possible to find a suitable implementation of IGenericDiscovery
         */
        gui.append(e.getMessage());
```

```
      e.printStackTrace();
    }
    return discovered_services;
  }

  /**
   * @author Vincenzo Suraci
   *
   * This function uses the internal OSGi service discovery to find a suitable implementation
   * of the IGenericDiscovery interface.
   */
  private ServiceReference[] findGenericDiscoveryModuleImplementations() throws Exception
  {
    ServiceReference[] gdmi = null;
    try
    {
      gdmi = bc.getServiceReferences("eu.artemis.shield.discovery.gdm.interfaces.IGenericDiscover
y", null);
    }
    catch (Exception e)
    {
      throw e;
    }
    return gdmi;
  }

  public void exit()
  {
    /*
     * Close GUI
     */
    gui.setVisible(false);
    gui.dispose();
  }
}
```

package eu.artemis.shield.composition.compositionmanager.impl.CMGUI.html

```
/**
 * pSHIELD
 * Service Discovery
 *
 * @author Davide Migliacci
 * Department of System and Computer Science (DIS)
 * University of Rome "Sapienza"
 * Via Ariosto, 25
 * 00184, Rome, IT
 *
 *
 * Created on 16-May-2007
 * Version 1.0
 *
 */

package eu.artemis.shield.composition.compositionmanager.impl;

import java.awt.BorderLayout;
import java.awt.Button;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.Frame;
import java.awt.TextArea;

import java.net.URL;

import javax.swing.BorderFactory;
import javax.swing.JPanel;


import org.osgi.framework.BundleContext;
import org.osgi.framework.BundleException;

public class CMGUI extends Frame implements ActionListener
{

  private static final int MAJOR = 0;
  private static final int MINOR = 1;

  private static final int width = 440;
  private static final int height = 440;

  private boolean working = true;

  private BundleContext bc = null;
  private CM cm = null;

  private TextArea ta = null;

  private JPanel main_panel = null;
  private JPanel textarea_panel = null;
  private JPanel buttons_panel = null;

  private static final int intNumBtn = 3;
  private static String[] strBtn = new String[intNumBtn];

  // ----------------------------------------------
  //  CONSTRUCTORS
  // ----------------------------------------------

  public CMGUI(BundleContext bc, CM cm)
    {
      super("pSHIELD - Composition Engine v" + MAJOR + "." + MINOR);

      strBtn[0] = "Start/Stop";
      strBtn[1] = "Clear Log";
      strBtn[2] = "Hide Me";
```

```
      this.bc = bc;
      this.cm = cm;

      addWindowListener(windowExit);
      createGUI();

      setSize(width,height);
        setForeground(Color.black);
        setBackground(Color.lightGray);

        /*
         * ENABLE / DISABLE THE GUI...
         */
        setVisible(true);
    }

  //  ------------------------------------------------
  //  EVENT HANDLER
  // ------------------------------------------------

    WindowAdapter windowExit = new WindowAdapter()
  {
      public void windowClosing(WindowEvent e)
      {
        actionExit();
        }
    };

    public void actionExit()
    {
        // Exiting...
      if (bc != null)
      {
        try
        {
          bc.getBundle().stop();
        }
        catch (BundleException BE)
        {
          BE.printStackTrace();
        }
      }
  }

    public void actionPerformed(ActionEvent e)
    {
      if (e.getActionCommand().equals(strBtn[0]))
    {
        if ( working ){
          append("\n");
          appendts("STOP \n");
          working = false;
        }
        else {
          append("\n");
          appendts("START \n");
          working = true;
        }
    }
    else if (e.getActionCommand().equals(strBtn[1]))
    {
      ta.setText("- Bundle ready\n");
    }
    else if (e.getActionCommand().equals(strBtn[1]))
    {
      setVisible(false);
    }
    else
    {
      //Unknown Command
    }
    }

    // ------------------------------------------------
  //  DESIGN GRAPHICS
  // ------------------------------------------------

    private void createGUI()
```

```
    {

      // Initialize the main Panel
    main_panel = new JPanel ( );
    main_panel.setLayout( new BorderLayout() );
    main_panel.setBorder( BorderFactory.createCompoundBorder(BorderFactory.createRaisedBevelBorde
r(), BorderFactory.createLoweredBevelBorder()) );

    // Initialize the Combo Panel
    textarea_panel = new JPanel();

    GridBagLayout gridbag = new GridBagLayout();
        GridBagConstraints c = new GridBagConstraints();
        c.fill = GridBagConstraints.BOTH;

        textarea_panel.setLayout( gridbag );

    c.gridwidth = 1;    // The cell occupies 1 column
        c.gridheight = 1;  // The cell occupies 1 row
      c.gridx = 0;     // The cell is located in 1 column
        c.gridy = 0;     // The cell is located in 1 row
        c.weightx = 0.1;  // The cell occupies the minimum row length
    c.weighty = 0.1;  // The cell occupies the entire column length


      ta = new TextArea();
      ta.setEditable(false);
        gridbag.setConstraints( ta, c);
        textarea_panel.add(ta);

    // Initialize the Buttons Panel
    buttons_panel = new JPanel();

    for (int i = 0; i < intNumBtn; i++)
      {
        /*
           * ADD BUTTONS TO START SEVERAL TEST
           */
      c.gridx = i;       // The cell is located in (i+1) column
          Button b = new Button(strBtn[i]);
            b.addActionListener(this);
            buttons_panel.add(b);
      }


    // Add the subpanels to the main panel

    main_panel.add( textarea_panel, BorderLayout.CENTER );
    main_panel.add( buttons_panel, BorderLayout.PAGE_END );

    // Add the main panel to the Frame
    add( main_panel );

      /*
         * CENTER FRAME ON THE SCREEN
         */
        Dimension dialogSize = getSize();
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    setLocation(screenSize.width/2 - dialogSize.width/2, screenSize.height/2 -
 dialogSize.height/2);

        /*
         * pSHIELD ICON
         */
        try
      {
          /*
           * Check if we are in a JAR file...
           */
          URL url = this.getClass().getResource("logo_pSHIELD_16x16.jpg");
          if (url != null)
          {
            this.setIconImage(Toolkit.getDefaultToolkit().createImage(url));
          }
          else
          {
            /*
               * We are not in a JAR file...
```

```
                */
            }
        }
          catch(Exception e)
          {
            e.printStackTrace();
          }

          appendts("Bundle started");

    }

    public void appendts(String str)
    {
      append("- " + str + "\n");
    }

    public void append(String str)
    {
      /*
       * Add text to the textArea
       */
      ta.setForeground(Color.BLACK);
      ta.append(str);
    }

}
```

package eu.artemis.shield.composition.middlewareadapter.impl.AuthenticationServiceAppRegistration.html

```
package eu.artemis.shield.composition.middlewareadapter.impl;

import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceReference;

import eu.artemis.shield.discovery.gdm.interfaces.IGenericDiscovery;

import java.util.Hashtable;
import java.util.Vector;

public class AuthenticationServiceAppRegistration extends AppRegistration {

  public AuthenticationServiceAppRegistration(BundleContext bc)
  {
    super(bc);

    String project_name = "Authentication";
    String service_name = "Authentication";
    String service_description = "Authentication API";
    String vid = null;
    String type = "service:eu.artemis.shield:http";
    String url = "http://localhost:8080/applications/jars/Authentication/Authentication_api-
0.0.1.jar";
    String export = "eu.artemis.shield.functionalities.authentication";
    long lifetime = 43200; // 1 day
    registerAPI(url, vid, type, lifetime, export, "", project_name, service_name, service_descrip
tion);

    String Import = "eu.artemis.shield.functionalities.cryptography,eu.artemis.shield.functionali
ties.authentication";
    url = "http://localhost:8080/applications/jars/EAPAuthentication/EAPAuthentication-
0.0.1.jar";
    String SPD = "1";
    String owl = "resources/data/data_7_Pilota.owl";
    service_description = "Authentication mechanism based on EAP";
    register(url, vid, type, lifetime, export, Import, project_name, service_name, service_descri
ption, SPD, owl);

    url = "http://localhost:8080/applications/jars/PAPAuthentication/PAPAuthentication-
0.0.1.jar";
    SPD = "1";
    owl = "resources/data/data_8_Pilota.owl";
    service_description = "Authentication mechanism based on PAP";
    register(url, vid, type, lifetime, export, Import, project_name, service_name, service_descri
ption, SPD, owl);

    url = "http://localhost:8080/applications/jars/CHAPAuthentication/CHAPAuthentication-
0.0.1.jar";
    SPD = "8";
    owl = "resources/data/data_9_Pilota.owl";
    service_description = "Authentication mechanism based on CHAP";
    register(url, vid, type, lifetime, export, Import, project_name, service_name, service_descri
ption, SPD, owl);



  }
}
```

package eu.artemis.shield.discovery.gdm.impl.CServiceDiscoveryGDM.html

```
/**
 * pSHIELD  (A4.2)
 * Service Discovery
 *
 * @author Silvano Mignanti
 * Department of System and Computer Science (DIS)
 * University of Rome "Sapienza"
 * Via Ariosto, 25
 * 00184, Rome, IT
 *
 * phone: +39 329 11 38 610
 * email: silvano.mignanti@dis.uniroma1.it
 *
 *
 * Created on 16-May-2007
 * Version 1.0
 *
 */
package eu.artemis.shield.discovery.gdm.impl;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Vector;
import java.util.Hashtable;
import java.util.Enumeration;

import java.net.MalformedURLException;
import java.net.URL;

import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceReference;

import eu.artemis.shield.discovery.filter.IServicesFilter;
import eu.artemis.shield.discovery.gdm.impl.Const;
import eu.artemis.shield.discovery.gdm.interfaces.IGenericDiscovery;
import eu.artemis.shield.discovery.pdm.ISDParameter;
import eu.artemis.shield.discovery.pdm.IService;
import eu.artemis.shield.discovery.pdm.IServiceDiscovery;
import eu.artemis.shield.discovery.pdm.IServiceList;
import eu.artemis.shield.discovery.pdm.IServiceType;
import eu.artemis.shield.discovery.pdm.IServiceTypeList;
import eu.artemis.shield.discovery.qp.interfaces.IQueryPreprocessor;

public class CServiceDiscoveryGDM implements IGenericDiscovery {

  /*
   * PARAMTERS
   */
  private BundleContext bc;

  /*
   * CONSTRUCTORS
   */
  public CServiceDiscoveryGDM(BundleContext bc) {

    this.bc = bc;
  }

  /*
   * IGENERICDISCOVERY METHODS
   */

  public LinkedList findServiceTypes(String VID)
  {
    if (Const.DEBUG_ENABLED) System.out.println("findServices Method of Generic Discovery Module"
);
    //ServiceReference[] srqpp = null;
    ServiceReference[] srSD = null;
    LinkedList result = new LinkedList();

    try
    {
      /*
       * FIND ALL THE PDMs
       */
      srSD = findPDMs();
```

```
      for (int k = 0; k < srSD.length; k++)
      {
        /*
         * FOR EACH PDM
         * FIND THE SERVICES USING THE CDQL QUERY
         */
        Object o = bc.getService(srSD[k]);
        //if (DEBUG) System.out.println(o2.getClass().getName() + srSD.length
        //    + "        " + k + "SDCDQL length: " + sdpcdql.length);
        IServiceDiscovery isDiscovery = (IServiceDiscovery) o;
        try
        {
          IServiceTypeList temp = isDiscovery.findServiceTypes(new ISDParameter[0]);
          if (temp!=null)
          {
            if (Const.DEBUG_ENABLED) System.out.println("TEMP SIZE: " + temp.size());
            for(int j = 0; j<temp.size(); j++)
            {
              IServiceType sType = (IServiceType)temp.get(j);
              result.add(sType.toString());
            }
          }
        }
        catch (Exception e)
        {
          System.out.println("PDM Exception!");
          e.printStackTrace();
        }
      }
    }
    catch (Exception e)
    {
      System.out.println("No PDMs found");
      //e.printStackTrace();
    }
    return result;
  }


  public LinkedList findServices(String VID, String type, String[] keywords)
  {
    if (Const.DEBUG_ENABLED) System.out.println("findServices Method of Generic Discovery Module"
);
    ServiceReference[] srqpp = null;
    ServiceReference[] srSD = null;
    LinkedList owluri = new LinkedList();
    LinkedList result = new LinkedList();
    ISDParameter[] sdpcdql = null;

    try
    {
      /*
       * FIND ALL THE QUERY PREPROCESSORS
       */
      srqpp = findQueryPreprocessors();

      String UserProfileQuery = "SELECT * WHERE{?x ?y ?z.}";
      String UserOWLQuery = "SELECT * WHERE{?x ?y ?z.}";
      String UserRequirementsOnServiceContext = getPreferencesOnUserRequirementsOnServiceContext(
VID, type);

      for (int i = 0; i < srqpp.length; i++)
      {
        /*
         * FOR EACH QUERY PREPROCESSOR
         * CREATE A QUERY
         */
        Object o = bc.getService(srqpp[i]);
        Class[] ifs=o.getClass().getInterfaces();
        if (Const.DEBUG_ENABLED)
        {
          for (int k =0; k<ifs.length; k++)
          {
            System.out.println("Name="+ifs[k].getName()+"\tCanName="+ifs[k].getName()+"\tHash"+if
s[k].hashCode());
          }
        }
        Class iqpp = IQueryPreprocessor.class;
```

```
        if (Const.DEBUG_ENABLED) System.out.println("Name="+iqpp.getName()+"\tCanName="+iqpp.getN
ame()+"\tHash"+iqpp.hashCode());
        IQueryPreprocessor iQPP = (IQueryPreprocessor) o;
        try
        {
          /*
           * CREATE A QUERY
           */
          sdpcdql = iQPP.createQuery(VID, type, keywords);
          if (Const.DEBUG_ENABLED) System.out.println("CServiceDiscoveryGDM::findservices --
> Query created");

          try
          {
            /*
             * FIND ALL THE PDMs
             */
            srSD = findPDMs();
            if (Const.DEBUG_ENABLED) System.out.println("CServiceDiscoveryGDM::findservices --
> " + srSD.length + " PDM found!");
            for (int k = 0; k < srSD.length; k++)
            {
              /*
               * FOR EACH PDM
               * FIND THE SERVICES USING THE CDQL QUERY
               */
              IServiceDiscovery isDiscovery = (IServiceDiscovery) bc.getService(srSD[k]);
              if (Const.DEBUG_ENABLED) System.out.println("CServiceDiscoveryGDM::findservices --
> PDM #" + k + " --> java type = " + isDiscovery.getClass().getSimpleName());
              try
              {
                if (Const.DEBUG_ENABLED) System.out.println("CServiceDiscoveryGDM::findservices -
-> PDM #" + k + " --> service discovering...");
                IServiceList temp = isDiscovery.findServices(sdpcdql);
                if (temp!=null)
                {
                  if (Const.DEBUG_ENABLED) System.out.println("CServiceDiscoveryGDM::findservices
 --> PDM #" + k + " --> discovered " + temp.size() + " services!");
                  for(int j = 0; j<temp.size(); j++)
                  {
                    IService serv = (IService)temp.get(j);
                    String owlurl = serv.getOWLSURL();
                    owluri.add(owlurl);
                  }
                }
              }
              catch (Exception e)
              {
                System.out.println("PDM Exception!");
                e.printStackTrace();
              }
            }
          }
          catch (Exception e)
          {
            System.out.println("No PDMs found");
            //e.printStackTrace();
          }
        }
        catch (Exception e)
        {
          System.out.println("QPPException: " + e.getLocalizedMessage());
          e.printStackTrace();
        }
      }

      /*
       * FILTER THE DISCOVERED OWL URLs
       * USING THE FILTER
       *
       * NOTE: THIS IS JUST A WORKAROUND TO SOLVE ASAP THE FILTERING ISSUES
       * WE USED A MODIFIED INTERFACE INVENTED BY SILVANO WHICH IS NOT THE
       * STANDARD ONE!!!!!!!!
       */

      if (Const.SEMANTIC_FILTER_ENABLED)
      {
        try
```

```
        {
          // DISCOVER ALL THE LOCAL FILTERS
          ServiceReference[] filterSR = findFilters();

          // DISCOVER TAKE THE FIRST ONE
          IServicesFilter iSF = (IServicesFilter) bc.getService(filterSR[0]);

          // LET FILTER SEMANTICALLY THE ALREADY DISCOVERED SERVICES
          result = iSF.filterServices(owluri, VID, UserProfileQuery, UserOWLQuery, UserRequiremen
tsOnServiceContext);
        }
        catch(Exception e)
        {
          System.out.println("Filter Exception: ");
          e.printStackTrace();
        }
      }
      else
      {
        result = owluri; // We pass all the discovered services...
      }
    }
    catch (Exception e)
    {
      e.printStackTrace();
    }

    return result;
  }


  /**
   * @param CDQL the query expressed in CDQL sintax used for the first step of the discovery proc
ess
   * @param SPARQL the query expressed in SPQRQL syntax used for the second step(filtering) of di
scovery process
   * @param query_output a linkedlist of values,if any, returned by the execution of the filterin
g step.
   * @return a list of service discovered and filtered on the besis of the two query submitted.
   */
  public IServiceList findServices(String CDQL, String SPARQL, LinkedList query_output) {

      /*
       * FIND ALL THE PDMs
       */
    LinkedList owluri = new LinkedList();
    LinkedList query_result = new LinkedList();
    IServiceList temp = null;

      ServiceReference[] srSD;
      try {
        srSD = findPDMs();
        for (int k = 0; k < srSD.length; k++)
        {
          /*
           * FOR EACH PDM
           * FIND THE SERVICES USING THE CDQL QUERY
           */
          //sdpcdql = createQuery(CDQL);

          Object o2 = bc.getService(srSD[k]);
          if (Const.DEBUG_ENABLED) System.out.println(o2.getClass().getName() + srSD.length
              + "        " + k + "CDQL length: " + CDQL.length());
          IServiceDiscovery isDiscovery = (IServiceDiscovery) o2;

            temp = isDiscovery.findServices(CDQL);
            if (temp!=null)
            {
              if (Const.DEBUG_ENABLED) System.out.println("TEMP SIZE: " + temp.size());
              //il loop seguente should be eliminato
              for(int j = 0; j<temp.size(); j++)
              {
                IService serv = (IService)temp.get(j);
                String owlurl = serv.getOWLSURL();
                owluri.add(owlurl);
              }
            }
        }
```

```
      }
      catch (Exception e)
      {
        System.out.println("PDM Exception!");
        e.printStackTrace();
      }

      if (Const.SEMANTIC_FILTER_ENABLED)
      {
        try
        {
          // DISCOVER ALL THE LOCAL FILTERS
          ServiceReference[] filterSR = findFilters();
          if (Const.DEBUG_ENABLED) System.out.println("Filters found: " + filterSR.length);

          // DISCOVER TAKE THE FIRST ONE
          IServicesFilter iSF = (IServicesFilter) bc.getService(filterSR[0]);
          if (Const.DEBUG_ENABLED) System.out.println("First filter selected");

          // LET FILTER SEMANTICALLY THE ALREADY DISCOVERED SERVICES
          //result = iSF.filterServices(owluri, null, SPARQL, "");
          query_result=iSF.filterServices(temp, null, SPARQL, "");
        }
        catch(Exception e)
        {
          System.out.println("Filter Exception: ");
          e.printStackTrace();
        }

      }
      if(query_output!=null)
        query_output.addAll(query_result);
      if (Const.DEBUG_ENABLED && temp!=null) System.out.println("TEMP AFTER FILTER SIZE: " + temp
.size());
      return temp;
  }


  public boolean registerService(String VID, String description, Hashtable parameters, long timeo
ut, String type)
  {
    /*
     * We use SLPTool...
     * > We ignore the VIDID
     * > We use the parameters hashtable to register the SLP parameters
     */

    if (Const.DEBUG_ENABLED)
    {
      System.out.println("CServiceDiscoveryGDM --> registerService");
      System.out.println("CServiceDiscoveryGDM --> URL description " + description);
      System.out.println("CServiceDiscoveryGDM --> String type " + type);
    }

    String[] args = new String[4];

    int index = 0;

    /*
     * REGISTER COMMAND
     */

    /*args[index] = "-debug";
    index++;
    */

    args[index] = "register";
    index++;

    /*
     * SERVICE TYPE
     */
    if (type != null)
    {
      int i = type.lastIndexOf(":");
      if (i >= 0)
      {
        try
```

```
        {
          if (new URL(description).getProtocol().equalsIgnoreCase(type.substring(i+1)))
          {
            //args[index] = type + "://" + description.toString().substring(description.getProtoc
ol().length());
            args[index] = type + description.toString().substring(new URL(description).getProtoco
l().length());
          }
          else
          {
            /*
             * Service type is not compliant with the URL... Different protocols!?!
             * Register them separately...
             */

            args[index] = type;
          }
        }catch(MalformedURLException ex){
            ex.printStackTrace();
          }
      }
      else
      {
        try
        {
          if (new URL(description).getProtocol().equalsIgnoreCase(type))
          {
            args[index] = description.toString();
          }
          else
          {
            /*
             * Service type is not compliant with the URL... Different protocols!?!
             * Register them separately...
             */
            args[index] = type;
          }
        }catch(MalformedURLException ex){
            ex.printStackTrace();
          }
      }
    }
    if (Const.DEBUG_ENABLED) System.out.println("CServiceDiscoveryGDM--
> serviceURL " + args[index]);
    index++;

    /*
     * ATTRIBUTES
     *
     * Modified by Vincenzo Suraci on 08/09/2011
     * The OntologyURI parameter was automatically inserted in each service registration.
     * Now the OntologyURI paramter is no longer automatically added.
     */

    //args[index] = "(OntologyURI="+description.toString()+")";
    args[index] = "";

    if (parameters != null)
    {
      Enumeration enukey = parameters.keys();
      while (enukey.hasMoreElements())
      {
        String attr_tag = (String)enukey.nextElement();
        if (args[index].length() > 0)
          args[index] += ",(" + attr_tag;
        else
          args[index] += "(" + attr_tag;
        Vector attr_values = (Vector)parameters.get(attr_tag);
        if (attr_values != null)
        {
          if (attr_values.size() > 0)
          {
            args[index] += "=";
            for (int i = 0; i < attr_values.size(); i++)
            {
              if (i > 0) args[index] += ",";
              Object o = attr_values.elementAt(i);
              if (o.getClass().getName().equals("java.lang.String"))
```

```
              {
                args[index] += (String)o;
              }
              else if (o.getClass().getName().equals("java.lang.Integer"))
              {
                args[index] += (Integer)o;
              }
              else if (o.getClass().getName().equals("java.lang.Boolean"))
              {
                Boolean b = (Boolean)o;
                if (b.booleanValue()) args[index] += "true"; else args[2] += "false";
              }
              else
              {
                /*
                 * CONSIDER THIS VALUE AS A STRING...
                 */
                args[index] += o.toString();
              }
            }
          }
        }
        args[index] += ")";
      }
    }
    if (Const.DEBUG_ENABLED) System.out.println("CServiceDiscoveryGDM--
> attributes " + args[index]);
    index++;

    /*
     * TIMEOUT
     */
    int to = (int)timeout;
    args[index] = "" + to;

    /*
     * FIND ALL THE PDMs
     */
    try
    {
      ServiceReference[] srSD = findPDMs();
      for (int k = 0; k < srSD.length; k++)
      {
        /*
         * FOR EACH PDM
         * REGISTER THE SERVICES USING THE REGISTERSERVICE METHOD
         */
        Object o2 = bc.getService(srSD[k]);
        if (Const.DEBUG_ENABLED){
          System.out.print(o2.getClass().getName());
          System.out.print(srSD.length);
          System.out.println("        " + k);
          //System.out.println("SDCDQL length: " + sdpcdql.length);
        }
        IServiceDiscovery isDiscovery = (IServiceDiscovery) o2;
        try
        {
          /*
           * If the service is registered at least with one PDM, than the
           * method exits successfully.
           */
          if (isDiscovery.registerService(null, args)) return true;

        }
        catch (Exception e)
        {
          System.out.println("PDM Exception!");
          e.printStackTrace();
        }
      }
    }
    catch (Exception e)
    {
      System.out.println("CServiceDiscoveryGDM::registerService ERROR!!!");
      e.printStackTrace();
    }

    return false;
```

```
}

public boolean registerService (Hashtable parameters, long timeout, String type) {

  /*
   * We use SLPTool...
   * > We ignore the VIDID
   * > We use the parameters hashtable to register the SLP parameters
   */

  if (Const.DEBUG_ENABLED)
  {
    System.out.println("CServiceDiscoveryGDM --> registerService");
    System.out.println("CServiceDiscoveryGDM --> String type " + type);
  }

  String[] args = new String[4];

  int index = 0;

  /*
   * REGISTER COMMAND
   */

  /*args[index] = "-debug";
  index++;
  */

  args[index] = "register";
  index++;

  /*
   * SERVICE TYPE
   */
  if (type != null)
  {
    args[index] = type;
  }
  if (Const.DEBUG_ENABLED)
    System.out.println("CServiceDiscoveryGDM--> serviceURL " + args[index]);
  index++;

  /*
   * ATTRIBUTES
   */
  if (parameters != null)
  {
    Enumeration enukey = parameters.keys();
    while (enukey.hasMoreElements())
    {
      String attr_tag = (String)enukey.nextElement();
      if (args[index]!= null)
        args[index] += ",(" + attr_tag;
      else
        args[index] = "(" + attr_tag;
      Vector attr_values = (Vector)parameters.get(attr_tag);
      if (attr_values != null)
      {
        if (attr_values.size() > 0)
        {
          args[index] += "=";
          for (int i = 0; i < attr_values.size(); i++)
          {
            if (i > 0) args[index] += ",";
            Object o = attr_values.elementAt(i);
            if (o.getClass().getName().equals("java.lang.String"))
            {
              args[index] += (String)o;
            }
            else if (o.getClass().getName().equals("java.lang.Integer"))
            {
              args[index] += (Integer)o;
            }
            else if (o.getClass().getName().equals("java.lang.Boolean"))
            {
              Boolean b = (Boolean)o;
              if (b.booleanValue()) args[index] += "true"; else args[2] += "false";
            }
```

```
              else
              {
                /*
                 * CONSIDER THIS VALUE AS A STRING...
                 */
                args[index] += o.toString();
              }
            }
          }
        }
        args[index] += ")";
      }
    }
    if (Const.DEBUG_ENABLED)
      System.out.println("CServiceDiscoveryGDM--> attributes " + args[index]);
    index++;

    /*
     * TIMEOUT
     */
    int to = (int)timeout;
    args[index] = "" + to;

    /*
     * FIND ALL THE PDMs
     */
    try
    {
      ServiceReference[] srSD = findPDMs();
      for (int k = 0; k < srSD.length; k++)
      {
        /*
         * FOR EACH PDM
         * REGISTER THE SERVICES USING THE REGISTERSERVICE METHOD
         */
        Object o2 = bc.getService(srSD[k]);
        if (Const.DEBUG_ENABLED){
          System.out.print(o2.getClass().getName());
          System.out.print(srSD.length);
          System.out.println("        " + k);
          //System.out.println("SDCDQL length: " + sdpcdql.length);
        }
        IServiceDiscovery isDiscovery = (IServiceDiscovery) o2;
        try
        {
          /*
           * If the service is registered at least with one PDM, than the
           * method exits successfully.
           */
          if (isDiscovery.registerService(null, args)) return true;

        }
        catch (Exception e)
        {
          System.out.println("PDM Exception!");
          e.printStackTrace();
        }
      }
    }
    catch (Exception e)
    {
      System.out.println("CServiceDiscoveryGDM::registerService ERROR!!!");
      e.printStackTrace();
    }

    return false;
  }

  public boolean registerService(String description, Hashtable parameters, long timeout, String t
ype)
  {
    return registerService(null, description, parameters, timeout, type);
  }

  /*
   * ADDITIONAL METHODS
   */
```

```
  public ServiceReference[] findQueryPreprocessors() throws Exception{
    ServiceReference[] srqpp = null;
    try {
      srqpp = bc.getServiceReferences("eu.artemis.shield.discovery.qp.interfaces.IQueryPreprocess
or", null);
    } catch (Exception e) {
      throw e;
    }
    return srqpp;
  }

  public ServiceReference[] findPDMs() throws Exception{
    ServiceReference[] srpdms = new ServiceReference[0];
    try {
      srpdms= bc.getServiceReferences("eu.artemis.shield.discovery.pdm.IServiceDiscovery", null);
    } catch (Exception e) {
      throw e;
    }
    return srpdms;
  }

  public ServiceReference[] findFilters() throws Exception{
    ServiceReference[] srfilters = null;
    try {
      srfilters= bc.getServiceReferences("eu.artemis.shield.discovery.filter.interfaces.IServices
Filter", null);
    } catch (Exception e) {
      throw e;
    }
    return srfilters;
  }


  private String getPreferencesOnUserRequirementsOnServiceContext(String VID, String type)
  {
    /*
     * This function takes the VID and the Service Type to look for the User Preferences
     * on the Service Context, regarding that specific Service Type.
     */

    /*
     * The service type is in the form:
     * service:x.y.z:protocol
     *
     * The pSHIELD preference manager wants only simple types
     *  so we take the "z" from the String type
     */
    if (Const.DEBUG_ENABLED) System.out.println("CServiceDiscvoeryGDM::getPreferencesOnUserRequir
ementsOnServiceContext --> Service Type = " + type);
    if (type.indexOf("service:") >= 0)
    {
      // Delete "service:"
      type = type.substring(8);
    }
    int index = type.indexOf(":");
    if (index >= 0)
    {
      // Delete the protocol
      type = type.substring(0,index);
    }
    index = type.indexOf(".");
    while (index >= 0)
    {
      type = type.substring(index+1);
      index = type.indexOf(".");
    }
    if (Const.DEBUG_ENABLED) System.out.println("CServiceDiscvoeryGDM::getPreferencesOnUserRequir
ementsOnServiceContext --> Service Type = " + type);

    /*
     * TEMPORARLY DISABLED
     */
    ArrayList ar = new ArrayList();
    ar.add("maxdistance");
    ar.add("maxqueuetime");
    //requestOutcomes(VID, "servicediscovery42", type, ar, null);

    return "";
```

```
    }
}
```

package eu.artemis.shield.discovery.slpdaemon.impl.Database.html

```
/**
 * SLPv2 DA Database Management
 *
 * We implemented the database function within Java2. We didn't use
 * a standard DBMS for simplicity and efficiency. The whole database
 * is organized as red-black tree (provided by Java)
 *
 *  (1) the key is ltag+URL, assume each service has a unique URL,
 *      and the same service can be registered using different
 *      languaage (ltag)
 *  (2) the value at each node is an Entry class, which keeps all the
 *      information about the service.
 *
 */

package eu.artemis.shield.discovery.slpdaemon.impl;

import java.io.*;
import java.util.*;
import java.security.interfaces.*;

public class Database {

  public static final int DELETED = 0;
  public static final int LANGUAGE_TAG = 1;
  public static final int NAMING_AUTHORITY =2;
  public static final int TYPE = 3;
  public static final int URL = 4;
  public static final int LIFE_TIME = 5;
  public static final int SCOPE = 6;
  public static final int VERSION_TS = 7; // version TS from the SA for an update
  public static final int ARRIVAL_TS = 8; // arrival TS at the DA for an update
  public static final int ACCEPT_DA = 9; // the accept DA for the update
  public static final int ACCEPT_TS = 10; // the accept TS for the update
  public static final int ATTRIBUTES = 11;

  da daf;
    TreeMap table;
    slpMsgComposer composer;
    Vector matchedEntry;
    ByteArrayOutputStream b;
    DataOutputStream d;
    int totalMatch,i=0;

    public Database(da daf) {
      this.daf = daf;
        table = new TreeMap();
        composer = new slpMsgComposer();
        matchedEntry = new Vector(10);
        b = new ByteArrayOutputStream();
        d = new DataOutputStream(b);
    }

    // return the number of entries in the database
    public  int size() {
        return table.size();
    }

    public TreeMap table() {
        return table;
    }

    //-----------------------------------------------
    // save database to either the stdout or the file
    //-----------------------------------------------
    public void saveDatabase(BufferedWriter o)
    {
        Iterator values = table.values().iterator();
        while (values.hasNext())
        {
            Entry e = (Entry) values.next();
            e.prtEntry(/*daf, */o);
        }
    }

    //-----------------------------------------------
```

```
    // return the database as Vector of Vector
    //-----------------------------------------------
    public Vector getDatabase()
    {
      Vector db = new Vector();
      Vector row;
        Iterator values = table.values().iterator();
        while (values.hasNext())
        {
          row = new Vector();
            Entry e = (Entry) values.next();
            row.add(new String("" + e.getDeleted()));
            row.add(e.getLtag());
            row.add(e.getType());
            row.add(e.getNA());
            row.add(e.getURL());
            row.add(new Long(e.getLifetime())); //int
            row.add(e.getScope());
            row.add(new Long(e.getVersionTS())); //long
            row.add(new Long(e.getArrivalTS())); //long
            row.add(e.getAcceptDA());
            row.add(new Long(e.getAcceptTS())); //long
            row.add(e.getAttr(""));
            db.add(row);
        }
        return db;
    }

    //-----------------------------------------
    // load data form file to internal database
    //-----------------------------------------
    public void loadDatabase(String dbase) {
        String line, ltag, type, url, scope, acceptDA, attr = "";
        int lifetime;
        long versionTS, arrivalTS, acceptTS;
        boolean deleted = false;
        try {
            BufferedReader in =
                new BufferedReader(new FileReader(dbase));
            while ((line = in.readLine()) != null) {
                StringTokenizer st = new StringTokenizer(line);
                deleted = Boolean.valueOf(st.nextToken()).booleanValue();
                ltag = st.nextToken();
                type = st.nextToken();
                url  = st.nextToken();
                lifetime = Integer.parseInt(st.nextToken());
                scope = st.nextToken();
                versionTS = Long.parseLong(st.nextToken());
                arrivalTS = Long.parseLong(st.nextToken());
                acceptDA = st.nextToken();
                acceptTS = Long.parseLong(st.nextToken());
                // Now the DA registers all the values of a given attribute
                st.nextToken(",");
                StringBuffer SB = new StringBuffer(2048);
                while (st.hasMoreTokens()){
                  SB.append(st.nextToken(","));
                  SB.append(",");
                }
                attr = SB.toString();

                addEntry(deleted, ltag, type, url, lifetime, scope, attr,
                         Const.fresh_flag, versionTS, arrivalTS,
                         acceptDA, acceptTS);
            }
            in.close();
        } catch (Exception e) {
            da.appendDebug("Database::loadDatabase");
            da.appendDebug(e);
        }
        daf.refreshDatabaseTable();
    }


    //-----------------------------------------
    // check lifetime and remove expired entries
    //-----------------------------------------
    public void rmExpiredEntries() {
      boolean dbChanged = false;
```

```
        long currtime = System.currentTimeMillis();
        Iterator keys = table.keySet().iterator();
        while (keys.hasNext()) {
            String k = (String) keys.next();
            Entry  e = (Entry)  table.get(k);
            int lif = e.getLifetime() *1000;
            if ((lif>0) && (currtime > (e.getArrivalTS() + e.getLifetime()*1000))) {
              dbChanged = true;
                keys.remove();
            }
        }
        if (dbChanged) daf.refreshDatabaseTable();
    }

    //-----------------------------------------------------------
    // for "SrvReg"
    // add a new entry of service registration to the database
    // or replace/update its previous registration
    //-----------------------------------------------------------
    public int addEntry(boolean deleted, String ltag, String type,
        String url, int lifetime, String scope, String attr, int reg_flag,
        long versionTS, long arrivalTS, String acceptDA, long acceptTS) {
      if (Const.DEBUG_MATCHING_ENABLED)
      {
        da.appendDebug("Database::addEntry");
        da.appendDebug("\t"+deleted+"\t"+ltag+"\t"+type+"\t"+url);
        da.appendDebug("\t"+lifetime+"\t"+scope+"\t"+attr+"\t"+reg_flag+"\n\n");
      }
      if (!table.containsKey(ltag+url)) {
            if ((reg_flag & Const.fresh_flag) == 0) { // incremental SrvReg
                return Const.INVALID_UPDATE;
            }
            Entry ent = new Entry();
            table.put(ltag+url, ent);
        }
        Entry e = (Entry) table.get(ltag+url);
        int result = e.update(deleted, ltag, url, type, lifetime, scope, attr,
                        reg_flag, versionTS, arrivalTS, acceptDA, acceptTS);
        daf.refreshDatabaseTable();
        return result;
    }

    //-----------------------------------------------------
    // for "SrvDeReg"
    // remove the entry with the key: ltag+url (tag=="")
    // or delete some attributes of this entry (tag!="")
    //-----------------------------------------------------
    public int rmEntry(String ltag, String url, String scope, String tag, long versionTS, String
acceptDA, long acceptTS)
    {
        if (table.containsKey(ltag+url))
        {
            Entry e = (Entry) table.get(ltag+url);
            if (!scope.equalsIgnoreCase(e.getScope()))
            {
                return Const.SCOPE_NOT_SUPPORTED;
            }
            e.deletion(tag, versionTS, acceptDA, acceptTS);
        }
        daf.refreshDatabaseTable();
        return Const.OK;
    }

    //-------------------------------------------------------------------------
    // for "SrvTypeRqst"
    // get the list of service types for specified scope & naming authority
    //-------------------------------------------------------------------------
    public String getServiceTypeList(String na, String scope)
    {
        if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug(
            "\nSrvTypeRqst" +
            "\n-----------" +
        "\nDatabase::getServiceTypeList(Naming Authority,Scope)" +
        "\n- Naming Authority = "+ na +
        "\n- Scope = "+ scope);
    Vector typelist = new Vector();
        Iterator values = table.values().iterator();
        while (values.hasNext())
```

```
        {
          Entry e = (Entry) values.next();
          if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Analysing database service --
> " + e.getURL());
            if (!e.getDeleted()) // entry not deleted
            {
              if (scope.equalsIgnoreCase(e.getScope())) // match scope
              {
                if (na.equals("*") || na.equals("") || na.equalsIgnoreCase(e.getNA()))
                {
                  if (!typelist.contains(e.getType()))  // has not been already listed
                  {
                    if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("> OKAY --
> Service type " + e.getType() + " added to the list\n");
                    typelist.addElement(e.getType());
                  }
                  else
                  {
                    if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("> DISCARDED --
> This service type has been already added to the list\n");
                  }
                }
                else
                {
                  if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("> DISCARDED --
> Different naming authority (" + e.getNA() + " vs " + na + ")\n");
                }
              }
              else
              {
                if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("> DISCARDED --
> Different scope (" + e.getScope() + " vs " + scope + ")\n");
              }
            }
            else
            {
              if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("> DISCARDED --
> Service has been deleted\n");
            }
        }
        StringBuffer tl = new StringBuffer();
        for (int i=0; i<typelist.size(); i++)
        {
            String s = (String)typelist.elementAt(i);
            if (tl.length() > 0) tl.append(",");
            tl.append(s);
        }
        if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("\n-
 Returned Service Type List = " + tl.toString() + "\n");
        return tl.toString();
    }

    //-------------------------------------------------------------
    // for "SrvRqst"
    // find the matched URLs with (type, scope, predicate, ltag)
    // return: error code (short)
    //         number of matched URLs (short)
    //         URL blocks (decided by previous #URL)
    //-------------------------------------------------------------

    public int getTotalMatch() {
        return totalMatch;
    }

    public Vector getMatchedEntry() {
        return matchedEntry;
    }

    public byte[] getMatchedURL(String type, String scope,
                String pred, String ltag, Vector ssExtList, int ecode,int ID, RSAPublicKey public
Key) {

        if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("\n*** SrvRqst ***" +
        "\nDatabase::getMatchedURL(Type,Scope,Predicate,Ltag)" +
        "\n- Type = "+ type +
        "\n- Scope = "+ scope +
        "\n- Predicate = "+ pred +
        "\n- Ltag = "+ ltag+"\n");
```

```
        byte[] buf = null;

      // if scope == "" it means it's the default scope.
      if (scope == "") scope = Const.defaultScope;

        if (!Util.shareString(daf.getScope(), scope, ","))
        {
            ecode = Const.SCOPE_NOT_SUPPORTED;
            if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getMatchedURL() -
> SCOPE NOT SUPPORTED!");
        }

        // obtain matched entries
        matchedEntry.clear();
        Iterator values = table.values().iterator();

        while (values.hasNext())
        {
            Entry e = (Entry) values.next();

            if (e.match(type, scope, pred, ltag, (ecode!=Const.AUTHENTICATION_ABSENT)&&(ecode!=Co
nst.AUTHENTICATION_FAILED)&& ID!=Const.SrvRqst))
            {
              if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getMatchedURL() -
> Service " + e.getURL() + "  ADDED\n");
              matchedEntry.addElement(e);
            }
            else
            {
              if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getMatchedURL() -
> Service " + e.getURL() + " FILTERED\n");
            }

        }

      totalMatch = matchedEntry.size();
        if (Const.DEBUG_MATCHING_ENABLED)
        {
          da.appendDebug("Database::getMatchedURL() -> Filtering Process Terminated.");
          da.appendDebug("Database::getMatchedURL() -> " + totalMatch + " services found.");
        }

        // filter matched entries
        for (int i=0; i<ssExtList.size(); i++) {
            SelectSortExt ss = (SelectSortExt) ssExtList.elementAt(i);
            if (ss.getID() == Const.SelectExt) {
                int bound = ss.getBound();
                if (bound < matchedEntry.size()) matchedEntry.setSize(bound);
            } else if (ss.getID() == Const.SortExt) {
                sortFilter(ss.getKey());
            }
        }

        // write matched URLs to buffer
        b.reset();
        try {
            d.writeShort(ecode);                // error code
            d.writeShort(matchedEntry.size());  // URL count
                        // fill in matched URLs
                for (int i=0; i<matchedEntry.size(); i++) {
                    Entry e = (Entry) matchedEntry.elementAt(i);
                    d.writeByte(0);
                    d.writeShort(e.getLifetime());
                    if (ID==Const.SrvRqstAuth)
                    {
                      d.writeShort(e.getCryptoURL(publicKey).length);
                      d.write(e.getCryptoURL(publicKey));
                    }
                    else
                    {
                      d.writeShort(e.getURL().length());
                         d.writeBytes(e.getURL());
                    }
                         d.writeByte(0);
                }

            buf = b.toByteArray();
```

```
        } catch (Exception ex) {
            da.appendDebug("Databse::getMatchedURL");
            da.appendDebug(ex);
        }
        return buf;
    }

    private void sortFilter(String key) {
        int size = matchedEntry.size();
        if (size <= 1) return; // no need to sort
        StringTokenizer st = new StringTokenizer(key, ",");
        int nkey = st.countTokens();
        String[]  keys   = new String[nkey];
        int[]     types  = new int[nkey];
        int[]     orders = new int[nkey];
        Integer[] vals   = new Integer[nkey];
        for (int i=0; i<nkey; i++) {
            String unit = st.nextToken();
            StringTokenizer st1 = new StringTokenizer(unit, ":");
            if (st1.countTokens() < 3) {
              da.append("Incorrect sort key list");
            }
            keys[i] = st1.nextToken().trim();  // key
            String s = st1.nextToken().trim(); // type
            if (s.equalsIgnoreCase("s")) {
                types[i] = Const.StringSort;
            } else {
                types[i] = Const.IntegerSort;
            }
            s = st1.nextToken().trim();        // order
            if (s.equals("+")) {
                orders[i] = Const.IncreasingOrder;
            } else {
                orders[i] = Const.DecreasingOrder;
            }
            if (st1.hasMoreTokens()) {         // reference value
                vals[i] = new Integer(st1.nextToken().trim());
            } else {
                vals[i] = null;
            }
        }
        Vector se = new Vector(size);
        for (int i=0; i<size; i++) {
            Entry e = (Entry)matchedEntry.elementAt(i);
            se.addElement(new SortEntry(e, keys, types, orders, vals));
        }
        Collections.sort(se);
        matchedEntry.clear();
        for (int i=0; i<size; i++) {
            SortEntry s = (SortEntry) se.elementAt(i);
            matchedEntry.addElement(s.getEntry());
        }
    }

    //-------------------------------------------------------
    /**
     * @author Vincenzo Suraci
     *
     * @param URLorType: is a ServiceURL or a Service Type
     * @param scope: is a comma separated scope list
     * @param tag: is a comma separated attribute tag list
     * @param ltag: is a language tag
     *
     * @return a String containing all the attribute tags that match with the Service URL or Type
,
     *  with the scope list, with the attribute tag list and with the language tag
     */
    //-------------------------------------------------------
    public String getAttrList(String URLOrType, String scope, String tag, String ltag)
    {
      if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("\n\nDatabase::getAttrList(url, scope, tag
, ltag)" +
        "\n- URL or Service Type = "+ URLOrType +
        "\n- Scope List = "+ scope +
        "\n- Attribute Tag List = " + tag +
        "\n- Language Tag = " + ltag + "\n");

      /*
```

```
        * Check if we have a ServiceURL or a ServiceTpye
        */
       String url = URLOrType;
       if (table.containsKey(ltag+url))
     {
         //if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getAttrList() -
> We have a URL!");
       Entry e = (Entry) table.get(ltag+url);
       if (e.getDeleted())
       {
         if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getAttrList() -
> Service " + e.getURL() + " has been DELETED");
         return "";
       }
       else if (e.isExpired())
       {
         if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getAttrList() -
> Service " + e.getURL() + " is EXPIRED");
         return "";
       }
       else if (!Util.containsString(scope, e.getScope(), ","))
             {
         if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getAttrList() -
> Service " + e.getURL() + " has a different scope (" + e.getScope() + " is not contined in " + s
cope + ")");
         return "";
             }
             return e.getAttr(tag);
       }
       String result = "";
       String type = URLOrType;
       if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getAttrList() -
> We have a TYPE!");
       /*
        * FROM RFC 2614
        * For the type and scope, return a Vector of all ServiceLocationAttribute objects whose id
s match the String
        * patterns in the attributeIds Vector regardless of the Locator's locale. The request is m
ade independent of
        * language locale. If no attributes are found, an empty vector is returned.
        */
       Iterator it = table.keySet().iterator();
       while (it.hasNext())
       {
         Entry e = (Entry)table.get(it.next());
         if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getAttrList() -
> Evaluating Service " + e.getURL());
         if (e.getDeleted())
         {
           if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getAttrList() -
> Service " + e.getURL() + " has been DELETED");
           return "";
         }
         else if (e.isExpired())
         {
           if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getAttrList() -
> Service " + e.getURL() + " is EXPIRED");
           return "";
         }
         else if (type.equalsIgnoreCase(e.getType()))
         {
           if (!Util.containsString(scope, e.getScope(), ","))
                 {
             if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getAttrList() -
> Service " + e.getURL() + " has a different scope (" + e.getScope() + " is not contined in " + s
cope + ")");
             return "";
                 }

               if (result.equals("")) result = e.getAttr(tag);
                else result += "," + e.getAttr(tag);
               if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getAttrList() -
> adding these attributes: " + e.getAttr(tag));

         }
         else
         {
           if (Const.DEBUG_MATCHING_ENABLED) da.appendDebug("Database::getAttrList() -
```

```
> Service " + e.getURL() + " has different type (\"" + e.getType() + "\" differes from \"" + type
 + "\")");
        }
      }
      return result;

    }

    /*
    private String typeAttrList(String type, String scope,
                                String tag, String ltag) {
        StringBuffer attrList = new StringBuffer();
        Iterator values = table.values().iterator();
        while (values.hasNext()) {
            Entry e = (Entry) values.next();
      if (Const.DEBUG_ENABLED)
      {
        da.append("DEBUG->Database::typeAttrList(...) Found a entry:");
              da.append("entry.type="+e.getType()+"?=?"+type);
        da.append("entry.scope="+e.getScope()+"?=?"+scope);
        da.append("entry.ltag="+e.getLtag()+"?=?"+ltag);
        da.append("entry:deleted="+e.getDeleted());
        da.append("entry:attrs="+e.getAttr(tag));
      }
            if (!e.getDeleted() &&
                type.equalsIgnoreCase(e.getType()) &&
                scope.equalsIgnoreCase(e.getScope()) &&
                ltag.equalsIgnoreCase(e.getLtag())) {
                String s = e.getAttr(tag);
                if (attrList.length() > 0) attrList.append(",");
                attrList.append(s);
            }
        }
        return attrList.toString();
    }
    */

    //-----------------------------------------------------------------
    // find new states based on selective/complete & (rdaList, rtsList)
    // sort new states on their accept IDs and return in TreeMap
    //-----------------------------------------------------------------
    public Vector findNewStates(String rscope,
                 Vector rdaList, Vector rtsList, int etrpType) {

        Vector tmp = new Vector();
        Iterator values = table.values().iterator();
        while (values.hasNext()) {
            Entry  e     = (Entry) values.next();



            String ada   = e.getAcceptDA();
            long   ats   = e.getAcceptTS();

            int    index = rdaList.indexOf(ada); // a requested subset?
            long   rts   = 0;
            if (index != -1) { // it is a requested subset, find rts
                rts = ((Long) rtsList.get(index)).longValue();
            }
            if (Util.shareString(rscope, e.getScope(), ",")) {
                if (index != -1 && ats > rts ||
                    index == -1 && etrpType == Const.complete) {
                    tmp.addElement(e);
                }
            }
        }
        Collections.sort(tmp);
        return tmp;
    }

    public String toString(){
      String t ="\n\n----- Database Contents -----\n\n";
      Iterator i = table.values().iterator();
      int n =1;
      while (i.hasNext()){
        t += "\nEntry "+n+")\n"+((Entry)i.next()).toString();
      }
      return t;
```

```
    }
}
```

package eu.artemis.shield.discovery.slpdaemon.impl.slpdaemon.html

```
package eu.artemis.shield.discovery.slpdaemon.impl;

import java.net.InetAddress;

public class slpdaemon {

  public static String _version = Const.MAJOR + "." + Const.MINOR;

  private static SlpDaemonCommandLine sdcm = null;

  private static da f = null;

  static int width = 600;
  static int height = 350;
  //static String scope = "default";

  public static void main(String args[])
  {
    init();
    if (parse(args))
      {
        //Start the SLP Daemon
        startDaemon();
      }
     else
      {
        printInfo();
      }
  }

  public static void stop()
  {
        f.actionExit();
        da.append("SLP Service Registry v" + _version + " stopped");
  }

  private static void init()
  {
    /*
     * STANDARD CONFIGURATION
     */

    /*
    Const.DEBUG_ENABLED = false;

    Const.IP_LINKLOCAL = true;  // (IPv4) 169.254.0.0/16
    Const.IP_SITELOCAL = true;  // (IPv4) 10.0.0.0/8 or 172.10.0.0/16 or 192.168.0.0/15
    Const.IP_LOOPBACK = false;  // (IPv4) 127.0.0.0/8
    Const.IP_GLOBAL = false;    // (IPv4) all the others

    //Const.IPv6_ENABLED = false;

    Const.GUI_ENABLED = true;
      Const.OSGI_BUNDLE = false;
    Const.SERVLET_ENABLED = false;

    Const.USE_JDBC = false;
    */

    Const.FEDERATION_ENABLED = false;
  }

  private static void printInfo()
  {
     System.out.println("\npSHIELD Project - slpdeamon v" + _version);
     System.out.println("Author: Vincenzo Suraci\n");
     System.out.println("USAGE: slpdeamon [-load] [-save] [-debug] [-gui] [-servlet] [-ipv4] [-
nomcast] [-nomesh] [-port port_number] [-global/site/link/lo] [-scope scope] [-dbase database] [-
sum summary] [-etrp type]\n");
     System.out.println("debug   - enables debug information");
     System.out.println("nogui   - disables slpdaemon gui");
     System.out.println("silent  - disables the slpdaemon messages on the console");
     System.out.println("secure  - enables authentication and authorization");
     //System.out.println("servlet- enables slpdaemon servlet");
     System.out.println("ipv4    - forces the slpdaemon to bind an ipv4 address");
     System.out.println("ipv6    - forces the slpdaemon to bind an ipv6 address");
```

```
    System.out.println("port    -
forces the slpdaemon to use the specified port_number");
    System.out.println("global  - forces the slpdaemon to use a global address");
    System.out.println("site    - (default) slpdaemon uses a site local address");
    System.out.println("link    - forces the slpdaemon to use a link local address");
    System.out.println("lo      - forces the slpdaemon to use a loopback address");
    System.out.println("jdbc    -
enables the slpdaemon to use a SQL database to store services");
    System.out.println("scope   - forces the slpdaemon to use the specified scope");
    System.out.println("dbase   - forces the slpdaemon to use the specified database");
    System.out.println("sum     - forces the slpdaemon to use the specified summary");
    System.out.println("nomcast - forces the slpdaemon to not use multicast messages");
    System.out.println("nomesh  - forces the slpdaemon to not use mesh capabilities");
    System.out.println("load    - forces the slpdaemon to load previously saved services");
    System.out.println("save    -
forces the slpdaemon to save registered services when exiting");
    System.out.println("etrp    -
forces the slpdaemon to used the specified type of anti entropy algorithm");

 }

 private static boolean parse(String[] args)
 {
   for (int i = 0; i < args.length; i++)
   {
     if (args[i].toLowerCase().equals("-debug"))
     {
       Const.DEBUG_ENABLED = true;
     }
     else if (args[i].toLowerCase().equals("-link"))
     {
       Const.IP_LINKLOCAL = true;
       Const.IP_SITELOCAL = false;
       Const.IP_LOOPBACK = false;
       Const.IP_GLOBAL = false;
     }
     else if (args[i].toLowerCase().equals("-global"))
     {
       Const.IP_LINKLOCAL = false;
       Const.IP_SITELOCAL = false;
       Const.IP_LOOPBACK = false;
       Const.IP_GLOBAL = true;
     }
     else if (args[i].toLowerCase().equals("-site"))
     {
       Const.IP_LINKLOCAL = false;
       Const.IP_SITELOCAL = true;
       Const.IP_LOOPBACK = false;
       Const.IP_GLOBAL = false;
     }
     else if (args[i].toLowerCase().equals("-lo"))
     {
       Const.IP_LINKLOCAL = false;
       Const.IP_SITELOCAL = false;
       Const.IP_LOOPBACK = true;
       Const.IP_GLOBAL = false;
     }
     else if (args[i].toLowerCase().equals("-ipv4"))
     {
       Const.IPv6_ENABLED = false;
     }
     else if (args[i].toLowerCase().equals("-ipv6"))
     {
       Const.IPv6_ENABLED = true;
     }
     else if (args[i].toLowerCase().equals("-nogui"))
     {
       Const.GUI_ENABLED = false;
     }
     else if (args[i].toLowerCase().equals("-silent"))
     {
       Const.CONSOLE_ENABLED = false;
     }
     else if (args[i].toLowerCase().equals("-secure"))
     {
       Const.SERVICE_REGISTRATION_AUTH_REQUIRED = true;
       Const.USER_AUTH_REQUIRED = true;
     }
```

```
      else if (args[i].toLowerCase().equals("-servlet"))
      {
        Const.SERVLET_ENABLED = true;
      }
      else if (args[i].toLowerCase().equals("-jdbc"))
      {
        Const.USE_JDBC = true;
      }
      else if (args[i].toLowerCase().equals("-scope"))
      {
        Const.SCOPE = args[++i];
      }
      else if (args[i].toLowerCase().equals("-dbase"))
      {
        Const.DBASE = args[++i];
      }
      else if (args[i].toLowerCase().equals("-summary"))
      {
        Const.SUMMARY = args[++i];
      }
      else if (args[i].toLowerCase().equals("-nomesh"))
      {
        Const.MESH_ENHANCED_ENABLED = false;
      }
      else if (args[i].toLowerCase().equals("-nomcast"))
      {
        Const.MULTICAST_ENABLED = false;
      }
      else if (args[i].toLowerCase().equals("-load"))
      {
        Const.LOAD_DATA_ENABLED = true;
      }
      else if (args[i].toLowerCase().equals("-save"))
      {
        Const.SAVE_DATA_ENABLED = true;
      }
      else if (args[i].toLowerCase().equals("-port"))
      {
        try
        {
          i++;
          Const.port = Integer.parseInt(args[i]);
        }
        catch (Exception e)
        {
          return false;
        }
      }else if (args[i].toLowerCase().equals("-etrp"))
      {
        try
        {
          i++;
          Const.ETRP_TYPE = Integer.parseInt(args[i]);
        }
        catch (Exception e)
        {
          return false;
        }
      }
      else
      {
        return false;
      }
    }
    return true;
  }

  private static void startDaemon()
  {

    try
    {
        /*boolean mcast = true;
        boolean mesh_enhanced = true;
        boolean loaddata = false;
        boolean savedata = false;
        int     etrpType; */
```

```
        // set InetAddress for this DA
        InetAddress ia = Util.getLocalInetAddress();

        /*if (ia == null){
          ia = InetAddress.getLocalHost();
          System.out.println("Bound to localhost");
        }*/
        // set scope for this DA
        /*String scope = System.getProperty("eu.artemis.shield.discovery.slpdaemon.scope");
        if (scope == null) scope = Const.defaultScope;

        // database file
        String dbase = System.getProperty("eu.artemis.shield.discovery.slpdaemon.dbase");
        if (dbase == null) dbase = Const.defaultDbase;

        // summary file
        String summary = System.getProperty("eu.artemis.shield.discovery.slpdaemon.summary");
        if (summary == null) summary = Const.defaultSummary;

        // multicast DAAdvert or not?
        String s = System.getProperty("eu.artemis.shield.discovery.slpdaemon.mcast");
        if (s != null && s.equalsIgnoreCase("no")) mcast = false;

        // carry "mesh-enhnaced" attribute keyword in DAAdvert or not?
        s = System.getProperty("eu.artemis.shield.discovery.slpdaemon.mesh");
        if (s != null && s.equalsIgnoreCase("no")) mesh_enhanced = false;

        // use selective or complete anti-entropy
        String s = System.getProperty("eu.artemis.shield.discovery.slpdaemon.mode");
        if (s != null && s.equalsIgnoreCase("complete")) {
            etrpType = Const.complete;
        } else {
            etrpType = Const.selective;
        }*/

        // load data or not?
        /*s = System.getProperty("eu.artemis.shield.discovery.slpdaemon.load");
        if (s != null && s.equalsIgnoreCase("yes")) loaddata = Const.LOAD_DATA_ENABLED;

        // save data or not?
        s = System.getProperty("eu.artemis.shield.discovery.slpdaemon.save");
        if (s != null && s.equalsIgnoreCase("yes")) savedata = Const.SAVE_DATA_ENABLED;*/

        f = new da(ia);

        sdcm = new SlpDaemonCommandLine(f);
        sdcm.start();

        da.append("SLP Service Registry v" + _version + " started");

    }
    catch (Exception e)
    {
      da.append(e);
    }
  }
}
```

package eu.artemis.shield.discovery.slpdaemon.impl.slpMsgComposer.html

```
/**
 * Compose various SLPv2 messages (protocol stack)
 * Return the message in a byte array
 *
 */

package eu.artemis.shield.discovery.slpdaemon.impl;

import java.io.*;
import java.util.*;

public class slpMsgComposer {

    DataOutputStream      d;
    ByteArrayOutputStream b;

    slpMsgComposer() {
        b = new ByteArrayOutputStream();
        d = new DataOutputStream(b);
    }

/**
 * Compose SLP common message header, flags may be set, there
 * may exist extensions, language tag is NOT included here
 *+--------------+-----------------+----------------------------------+
 *|  Version     |   Function-ID   |            Length                |
 *+--------------+-+-+-+-----------+---------------+------------------+
 *| Length cont. |O|F|R|       Reserved            | Next Ext. Offset |
 *+--------------+-+-+-+-----------+---------------+------------------+
 *|     Next Ext. Offset  Cont.    |              XID                 |
 *+--------------------------------+----------------------------------+
 */
    private void Header(int type, int len, int flag, int xid) {
        try {
            b.reset();
            d.writeByte(Const.version); // SLP version
            d.writeByte(type);          // Function type
            d.writeByte(0);             // len
            d.writeShort(len);          // length
            d.writeShort(flag);         // flag bits
            d.writeByte(0);             // next ext. offset
            d.writeShort(0);            // next ext. offset
            d.writeShort(xid);          // XID
        } catch (Exception e) {
            da.append(e);
        }
    }

/**
 * Put a string in the byte[], precede with its length.
 * The string could be empty, with a length of 0.
 * If the string is null, then no action is taken.
 */
    private void putString(String s) {
        if (s == null) return;
        try {
            d.writeShort(s.length());
            if (s.length() > 0) {
                d.writeBytes(s);
            }
        } catch (Exception e) {
          da.append(e);
        }
    }

/**
 * put an integer as a byte (normally zero) in the byte[]
 */
    private void putByte(int z) {
        try {
            d.writeByte(z);
        } catch (Exception e) {
          da.append(e.toString());
        }
    }
```

```
/**
 * put an integer as a short (normally as error code) in the byte[]
 */
    private void putShort(int z) {
        try {
            d.writeShort(z);
        } catch (Exception e) {
          da.append(e.toString());
        }
    }

/**
 * put an integer as an integer in the byte[]
 */
    private void putInt(int z) {
        try {
            d.writeInt(z);
        } catch (Exception e) {
          da.appendDebug("slpMsgComposer::putInt()");
          da.appendDebug(e);
        }
    }

    private void putLong(long z) {
        try {
            d.writeLong(z);
        } catch (Exception e) {
          da.appendDebug("slpMsgComposer::putLong()");
          da.appendDebug(e);
        }
    }

/**
 * put URL entry in the byte[], assume "# of URL auths" is zero
 *+---------------+---------------------------------+-----------------+
 *|    Reserved   |          Lifetime               |   URL length    |
 *+---------------+---------------------------------+-----------------+
 *| URL len cont. |          URL (variable length)                     \
 *+---------------+-----------------------------------------------------+
 *| # of URL auths|          Auth. blocks (if any)                     \
 *+-------------------------------------------------------------------+
 */
    private void putURL(String url, int lifetime) {
        try {
            d.writeByte(0);                     // reserved
            d.writeShort(lifetime);             // lifetime
            d.writeShort(url.length());         // len of URL
            d.writeBytes(url);                  // URL string
            d.writeByte(0);                     // # of authenticate
        } catch (Exception e) {
          da.append(e.toString());
        }
    }

/**
 * calculate string length, precede with a short integer length field
 */
    private int strlen(String s)
    {
      if (s != null)
      {
        return (2 + s.length());
      }
        return 2;
    }

/**
 * calculate URL-entry length, assume "# of url auths" is zero
 */
    private int urllen(String url) {
        return (6 + url.length());
    }

/**
 * service request <#1>
 *+---------------------------+--------------------------+
 *| Length of <PRList>        |   <PRList> string        \
 *+---------------------------+--------------------------+
```

```
 *| Length of <service-type>   | <service-type> string     \
 *+----------------------------+---------------------------+
 *| Length of <scope-list>     | <scope-list> string       \
 *+----------------------------+---------------------------+
 *| Length of predicate string | service request predicate \
 *+----------------------------+---------------------------+
 *| Length of <SLP SPI> string |   <SLP SPI> string        \
 *+----------------------------+---------------------------+
 */
    public byte[] SrvRqst(int xid, int flag, String ltag, String pr,
                String type, String scope, String pred, String spi) {
        int len = Const.header_len + strlen(ltag) + strlen(pr) +
                strlen(type) + strlen(scope) + strlen(pred) + strlen(spi);
        Header(Const.SrvRqst, len, flag, xid);
        putString(ltag);                // language tag
        putString(pr);                  // PRList
        putString(type);                // service type
        putString(scope);               // scope list
        putString(pred);                // predicate
        putString(spi);                 // SPI
        if (Const.MESSAGE_LOG_ENABLED) da.append("\nOUTGOING SRVRQST MESSAGE" +
        "\n- Xid = "+ xid +
        "\n- Ltag = "+ ltag +
        "\n- PRList = "+ pr +
        "\n- Type = " + type +
        "\n- Scope = " + scope +
        "\n- Predicate = " + pred +
        "\n- SPI = " + spi +
        "\n");
        return b.toByteArray();
    }

/**
 * service reply (reply for service request) <#2>
 *+----------------------------+---------------------------+
 *|       Error Code           |    URL entry count        |
 *+----------------------------+---------------------------+
 *|   <URl entry 1>          ...     <URL entry N>         \
 *+----------------------------+---------------------------+
 */
    public byte[] SrvReply(int xid, String ltag, byte[] buf) {
        int len = Const.header_len + strlen(ltag) + buf.length;
        Header(Const.SrvRply, len, Const.normal_flag, xid);
        putString(ltag);                        // language tag
        try {
            d.write(buf, 0, buf.length);        // ErrCode + #URL + each entry
        } catch (Exception e) {
          da.append(e.toString());
        }
        if (Const.MESSAGE_LOG_ENABLED) da.append("\nOUTGOING SRVREPLY MESSAGE" +
        "\n- Xid = "+ xid +
        "\n- Ltag = "+ ltag +
        "\n");
      //  if (Const.OSGI_BUNDLE) daOSGi.visual_log("(PSM): SLP SERVICE DISCOVERY RESPONSE");
        return b.toByteArray();
    }

    /**
     * secure service reply (reply for secure service request) <#14>
     *+----------------------------+---------------------------+
     *|       Error Code           |    URL entry count        |
     *+----------------------------+---------------------------+
     *|   <URl entry 1>          ...     <URL entry N>         \
     *+----------------------------+---------------------------+
     */
        public byte[] SrvReplyAuth(int xid, String ltag, byte[] buf) {
            int len = Const.header_len + strlen(ltag) + buf.length;
            int ecode=0;
            Header(Const.SrvRplyAuth, len, Const.normal_flag, xid);
            putString(ltag);                        // language tag
            try {

              d.write(buf, 0, buf.length);        // ErrCode + #URL + each entry
                for (int i=0; i<2; i++) {
                    ecode <<=  8;
                    ecode += buf[i] & 0xff;
                }
            } catch (Exception e) {
```

```
              da.appendDebug(e);
            }
            if (ecode==Const.AUTHENTICATION_FAILED) da.displayMessage ("Authentication Failed", C
onst.EXCLAMATION, "Authentication check");
            if (ecode==Const.AUTHENTICATION_ABSENT) da.displayMessage ("Authentication Absent", C
onst.EXCLAMATION, "Authentication check");

            if (Const.MESSAGE_LOG_ENABLED) da.append("\nOUTGOING SRVREPLYAUTH MESSAGE" +
            "\n- Xid = "+ xid +
            "\n- Ltag = "+ ltag +
            "\n- Ecode= "+ ecode +
                    "\n");
          //  if (Const.OSGI_BUNDLE) daOSGi.visual_log("(PSM): SLP SERVICE DISCOVERY RESPONSE");
            return b.toByteArray();
        }

/**
 * service registration <#3>
 *+-----------------------------------------------------------------+
 *|                          <URL-Entry>                            \
 *+---------------------------------+-------------------------------+
 *|  Length of service type string  |     <service-type>            \
 *+---------------------------------+-------------------------------+
 *|  Length of <scope-list>         |     <scope-list>              \
 *+---------------------------------+-------------------------------+
 *|  Length of attr-list string     |     <attr-list>               \
 *+----------------+----------------+-------------------------------+
 *| # of AttrAuths | (if present) Attribute Authentication Blocks  \
 *+----------------+------------------------------------------------+
 */
    public byte[] SrvReg(int xid, int flag, String ltag, String url,
                int lifetime, String type, String scope, String attr) {
        int len = Const.header_len + strlen(ltag) + urllen(url) +
                  strlen(type) + strlen(scope) + strlen(attr) + 1;
        Header(Const.SrvReg, len, flag, xid);
        putString(ltag);                          // language tag
        putURL(url, lifetime);
        putString(type);                          // service type
        putString(scope);                         // scope list
        putString(attr);                          // attr list
        putByte(0);                               // num of attrAuths
        if (Const.MESSAGE_LOG_ENABLED) da.append("\nOUTGOING SRVREG MESSAGE"+
        "\n- Xid = "+ xid +
        "\n- Ltag = "+ ltag +
        "\n- Type = " + type +
        "\n- Scope = " + scope +
        "\n- Url = " + url +
        "\n- Lifetime = " + lifetime +
        "\n- Attribute List = " + attr +
        "\n");
        return b.toByteArray();
    }

/**
 * service De-registration <#4>
 *+-----------------------------+---------------------------+
 *|  Length of <scope-list>     |        <scope-list>       \
 *+-----------------------------+---------------------------+
 *|                         <URL-entry>                     \
 *+-----------------------------+---------------------------+
 *|  Length of <tag-list>       |        <tag-list>         \
 *+-----------------------------+---------------------------+
 */
    public byte[] SrvDeReg(int xid, String ltag, String scope, String url,
                int ltime, String tag) {
        int len = Const.header_len + strlen(ltag) + strlen(scope) +
                        urllen(url) + strlen(tag);
        Header(Const.SrvDeReg, len, Const.normal_flag, xid);
        putString(ltag);                          // language tag
        putString(scope);                         // scope list
        putURL(url, ltime);                       // URL
        putString(tag);                           // tag list
        if (Const.MESSAGE_LOG_ENABLED) da.append("\nOUTGOING SRVDEREG MESSAGE" +
        "\n- Xid = "+ xid +
        "\n- Ltag = "+ ltag +
        "\n- Scope = " + scope +
        "\n- Url = " + url +
        "\n- Tag List = " + tag +
```

```
        "\n");
        return b.toByteArray();
    }

/**
 * service ack (reply for SrvReg & SrvDeReg) <#5>
 *+----------------------------+
 *|        Error Code          |
 *+----------------------------+
 */
    public byte[] SrvAck(int xid, String ltag, int errcode) {
        int len = Const.header_len + strlen(ltag) + 2;
        Header(Const.SrvAck, len, Const.normal_flag, xid);
        putString(ltag);                        // language tag
        putShort(errcode);                      // ErrCode
        if (Const.MESSAGE_LOG_ENABLED) da.append("\nOUTGOING SRVACK MESSAGE" +
        "\n- Xid = "+ xid +
        "\n- Ltag = "+ ltag +
        "\n- Error = " + errcode +
        "\n");
       // if (Const.OSGI_BUNDLE) daOSGi.visual_log("(PSM): SLP SERVICE REGISTRATION ACK");
        return b.toByteArray();
    }

/**
 * attribute request <#6>
 *+-----------------------------+----------------------------+
 *|  Length of PRList           |   <PRList> string          |
 *+-----------------------------+----------------------------+
 *|  Length of URL              |         URL                |
 *+-----------------------------+----------------------------+
 *|  Length of <scope-list>     |   <scope-list> string      |
 *+-----------------------------+----------------------------+
 *|  Length of <tag-list> string|   <tag-list> string        |
 *+-----------------------------+----------------------------+
 *|  Length of <SLP SPI> string |   <SLP SPI> string         |
 *+-----------------------------+----------------------------+
 */
    public byte[] AttrRqst(int xid, String ltag, String pr, String url,
                           String scope, String tag, String spi) {
        int len = Const.header_len + strlen(ltag) + strlen(pr) +
                strlen(url) + strlen(scope) + strlen(tag) + strlen(spi);
        Header(Const.AttrRqst, len, Const.normal_flag, xid);
        putString(ltag);
        putString(pr);
        putString(url);
        putString(scope);
        putString(tag);
        putString(spi);
        if (Const.MESSAGE_LOG_ENABLED) da.append("\nOUTGOING ATTRRQST MESSAGE"+
        "\n- Xid = "+ xid +
        "\n- Ltag = "+ ltag +
        "\n- PRLIst = " + pr +
        "\n- Url = " + url +
        "\n- Scope = " + scope +
        "\n- Tag List = " + tag +
        "\n- SPI = " + spi +
        "\n");
        return b.toByteArray();
    }

/**
 * attribute reply (reply for attribute request) <#7>
 *+-----------------------------+----------------------------------+
 *|         Error Code          |   Length of <attr-list>          |
 *+-----------------------------+----------------------------------+
 *|                          <attr-list>                           \
 *+----------------+-----------------------------------------------+
 *| # of AttrAuths |  Attribute authentication block (if present) \
 *+----------------+-----------------------------------------------+
 */
    public byte[] AttrReply(int xid, String ltag, int ecode, String buf) {
        int len = Const.header_len + strlen(ltag) + 3 + strlen(buf);
        Header(Const.AttrRply, len, Const.normal_flag, xid);
        putString(ltag);                // language tag
        putShort(ecode);                // ErrCode
        putString(buf);                 // attr-list
        putByte(0);                     // # of AttrAuths
```

```
        if (Const.MESSAGE_LOG_ENABLED) {
          if (Const.FORMAT_MESSAGE_LOG_ENABLED) {
            String temp = "";
            int i = 0;
            StringTokenizer st = new StringTokenizer(buf,")");
            while (st.hasMoreTokens()) {
              if (i!=0)
                temp += st.nextToken().substring(2) + "\n                    ";
              else
                temp += st.nextToken().substring(1) + "\n                    ";
              i++;
            }
            buf = temp;
          }
          da.append("\nOUTGOING ATTRREPLY MESSAGE"+
        "\n- Xid = "+ xid +
        "\n- Ltag = "+ ltag +
        "\n- Error = " + ecode +
        "\n- Attribute List = " + buf +
        "\n");
        }
        return b.toByteArray();
    }

/**
 * directory agent advertisement <#8>
 *+--------------------------------+--------------------------------+
 *|     Error Code                 |   DA Stateless Boot Timestamp  |
 *+--------------------------------+--------------------------------+
 *| DA Stateless Boot Time cont.   |      Length of URL             |
 *+--------------------------------+--------------------------------+
 *|                              URL                                \
 *+--------------------------------+--------------------------------+
 *|   Length of <scope-list>       |      <scope-list>              \
 *+--------------------------------+--------------------------------+
 *|   Length of <attr-list>        |      <attr-list>               \
 *+--------------------------------+--------------------------------+
 *|   Length of SLP <SPI>          |      SLP <SPI> string          \
 *+----------------+---------------+--------------------------------+
 *| # Auth Blocks |       Authentication blocl (if any)             \
 *+----------------+---------------+--------------------------------+
 */
    public byte[] DAAdvert(int xid, int flag, String ltag, int ts,
                String url, String scope, String attr, String spi) {
        int len = Const.header_len + 7 + strlen(ltag) + strlen(url) +
                strlen(scope) + strlen(attr) + strlen(spi);
        Header(Const.DAAdvert, len, flag, xid);
        putString(ltag);                // language tag
        putShort(0);                    // ErrCode
        putInt(ts);                     // boot timestamp
        putString(url);                 // URL
        putString(scope);               // scope list
        putString(attr);                // attribute list
        putString(spi);                 // SLP SPI
        putByte(0);                     // # Auth blocks
        if (Const.MESSAGE_LOG_ENABLED && Const.DAAdvert_LOG_ENABLED) da.append("\nOUTGOING DAADVE
RT MESSAGE"+
        "\n- Xid = "+ xid +
        "\n- Ltag = "+ ltag +
        "\n- Boot TS = "+ ts +
        "\n- Url = " + url +
        "\n- Scope = " + scope +
        "\n- Attribute List = " + attr +
        "\n- SPI = " + spi +
        "\n");
        return b.toByteArray();
    }

/**
 * service type request <#9>
 *+--------------------------------+--------------------------------+
 *|     Length of PRList           |     <PRList> string            |
 *+--------------------------------+--------------------------------+
 *| Length of Naming Authority     |  <Naming Authority String>     |
 *+--------------------------------+--------------------------------+
 *| Length of <scope-list>         |    <scope-list> string         |
 *+--------------------------------+--------------------------------+
 */
```

```
    public byte[] SrvTypeRqst(int xid, String ltag, String pr, String na,
                              String scope) {
        int len;
        if (na.equals("-1")) {
            len = Const.header_len + strlen(ltag) + strlen(pr) +
                  2 + strlen(scope);
        } else {
            len = Const.header_len + strlen(ltag) + strlen(pr) +
                  strlen(na) + strlen(scope);
        }
        Header(Const.SrvTypeRqst, len, Const.normal_flag, xid);
        putString(ltag);                // language tag
        putString(pr);                  // PRList
        if (na.equals("-1")) {          // Naming Authority
            putShort(0xFFFF);           // -1 for all
        } else {
            putString(na);
        }
        putString(scope);               // scope list
        if (Const.MESSAGE_LOG_ENABLED) da.append("\nOUTGOING SRVTYPERQST MESSAGE"+
        "\n- Xid = "+ xid +
        "\n- Ltag = "+ ltag +
        "\n- PRList = " + pr +
        "\n- Naming Authority = " + na +
        "\n- Scope = " + scope +
        "\n");
        return b.toByteArray();
    }

/**
 * service type reply (reply for service type request) <#10>
 *+--------------------------------+--------------------------------+
 *|          Error Code            |    Length of <srvType-list>    |
 *+--------------------------------+--------------------------------+
 *|                       <srvType-list>                            \
 *+-----------------------------------------------------------------+
 */
    public byte[] SrvTypeReply(int xid, String ltag, int ecode, String buf) {
        int len = Const.header_len + strlen(ltag) + 2 + strlen(buf);
        Header(Const.SrvTypeRply, len, Const.normal_flag, xid);
        putString(ltag);                // language tag
        putShort(ecode);                // ErrCode
        putString(buf);                 // srvtype-list
        if (Const.MESSAGE_LOG_ENABLED) da.append("\nOUTGOING SRVTYPEREPLY MESSAGE"+
        "\n- Xid = "+ xid +
        "\n- Ltag = "+ ltag +
        "\n- Error = " + ecode +
        "\n- Type List = " + buf +
        "\n");
        return b.toByteArray();
    }

/**
 * DataRqst <#12> and DataRplyCmpl <#13> message
 *+--------------------------------+--------------------------------+
 *|      Anti-entropy type ID      |   Number of Accept ID entries  |
 *+--------------------------------+--------------------------------+
 *|    Accept ID Entry 1         ...          Accept ID Entry k     \
 *+-----------------------------------------------------------------+
 */
    public byte[] AntiEtrpRqst(int xid, String ltag, int type,
                               Vector adaList, Vector atsList) {
        int size = 0;
        for (int i=0; i<adaList.size(); i++) {
            size += 10 + ((String)adaList.elementAt(i)).length();
        }
        int len = Const.header_len + strlen(ltag) + 4 + size;
        Header(Const.AntiEtrpRqst, len, Const.normal_flag, xid);
        putString(ltag);                // language tag
        putShort(type);
        putShort(adaList.size());
        for (int i=0; i<adaList.size(); i++) {
            putLong(((Long)atsList.elementAt(i)).longValue());
            putString((String)adaList.elementAt(i));
        }
        if (Const.MESSAGE_LOG_ENABLED) da.append("\nOUTGOING ANTIETRPRQST MESSAGE"+
        "\n- Xid = "+ xid +
        "\n- Ltag = "+ ltag +
```

```
        "\n- Entropy Type = " + type +
        "\n- ATS List = " + atsList.toString() +
        "\n- ADA List = " + adaList.toString() +
        "\n");
        return b.toByteArray();
    }

/**
 * append MeshFwd extension & adjust original message
 *+--------------------------------+----------------------------------+
 *|  MeshFwd Extension ID = 0x0006 |  Next Extension Offset (NEO)     |
 *+--------------+-----------------+----------------------------------+
 *|  NEO Contd.  |     Fwd-ID      |         Version Timestamp        |
 *+--------------+-----------------+----------------------------------+
 *|                     Version Timestamp, contd.                     |
 *+--------------------------------+----------------------------------+
 *|    Version Timestamp, contd.   |            Accept ID             \
 *+--------------------------------+----------------------------------+
 */
    public byte[] MeshFwdExt(byte[] buf, int id, long versionTS,
                             String ada, long ats) {
        if (ada == null) {
            //da.appendDebug("Null acceptDA!");
            return buf;
        }
        int len = Util.parseInt(buf, 2, 3);
        int alen = 14 + 10 + ada.length();
        adjustMesg(buf, alen);
        try {
            b.reset();
            d.write(buf, 0, len);
            d.writeShort(Const.MeshFwdExt);      // mesh-forwarding extension
            d.writeShort(0);                     // next ext. offset
            d.writeByte(0);                      // next ext. offset cont.
            d.writeByte(id);                     // Fwd-ID
            d.writeLong(versionTS);              // version timestamp
            d.writeLong(ats);                    // accept TS
            d.writeShort(ada.length());          // length of accept DA URL
            d.writeBytes(ada);                   // accept DA URL
        } catch (Exception e) {
          da.append(e.toString());
        }
        return b.toByteArray();
    }

/**
 * append Select extension & adjust original message
 *+--------------------------------+----------------------------------+
 *|  Select Extension ID = 0x4002  |  Next Extension Offset (NEO)     |
 *+--------------+-----------------+----------------+-----------------+
 *|  NEO Contd.  |       Number of URL Entries      |
 *+--------------+----------------------------------+
 */
    public byte[] SelectExt(byte[] buf, int num) {
        int len = Util.parseInt(buf, 2, 3);
        int alen = 7;
        adjustMesg(buf, alen);
        try {
            b.reset();
            d.write(buf, 0, len);
            d.writeShort(Const.SelectExt);       // selection extension
            d.writeShort(0);                     // next ext. offset
            d.writeByte(0);                      // next ext. offset cont.
            d.writeShort(num);                   // number of URL entries
        } catch (Exception e) {
          da.append(e.toString());
        }
        return b.toByteArray();
    }

/**
 * append Sort extension & adjust original message
 *+--------------------------------+----------------------------------+
 *|   Sort Extension ID = 0x4003   |  Next Extension Offset (NEO)     |
 *+--------------+-----------------+---------------+------------------+
 *|  NEO Contd.  |      Length of sort key list    |   sort key list  \
 *+--------------+---------------------------------+------------------+
 */
```

```
    public byte[] SortExt(byte[] buf, String key) {
        int len = Util.parseInt(buf, 2, 3);
        int alen = 7 + key.length();
        adjustMesg(buf, alen);
        try {
            b.reset();
            d.write(buf, 0, len);
            d.writeShort(Const.SortExt);      // selection extension
            d.writeShort(0);                    // next ext. offset
            d.writeByte(0);                     // next ext. offset cont.
            d.writeShort(key.length());         // length of key
            d.writeBytes(key);                  // key string
        } catch (Exception e) {
          da.append(e.toString());
        }
        return b.toByteArray();
    }

/**
 * AttrList extension
 */
    public byte[] AttrListExt(byte[] buf, Entry entry) {
        int len = Util.parseInt(buf, 2, 3);
        int alen = 10;
        String url  = "";
        String attr = "";
        if (entry != null) {
            url  = entry.getURL();
            attr = entry.getAttr("");
            alen += url.length() + attr.length();
        }
        adjustMesg(buf, alen);
        try {
            b.reset();
            d.write(buf, 0, len);
            d.writeShort(Const.AttrListExt);    // AttrList extension
            d.writeShort(0);                    // next ext. offset
            d.writeByte(0);                     // next ext. offset cont.
            putString(url);
            putString(attr);
            d.writeByte(0);                     // num auths
        } catch (Exception e) {
          da.append(e.toString());
        }
        return b.toByteArray();
    }
/**
 * adjust source message for the adding extension, need to change:
 *   (1) packet length (add new length)
 *   (2) last extension's NEO links to new one
 */
    private void adjustMesg(byte[] buf, int alen) {
        int plen = Util.parseInt(buf, 2, 3);
        int nextExt = Util.parseInt(buf, 7, 3);
        int lastExtAddr = 7;
        while (nextExt != Const.EndOfExt) {
            lastExtAddr = nextExt+2;
            nextExt = Util.parseInt(buf, lastExtAddr, 3);
        }
        Util.writeInt(buf, lastExtAddr, plen, 3); // new ext. starting point
        Util.writeInt(buf, 2, plen+alen, 3);      // adjust message length
    }
}
```

package eu.artemis.shield.discovery.slpdaemon.impl.slpMsgParser.html

```
/**
 * SLPv2 message parser (protocol stack)
 * Use separate get-methods to obtain each field after parsing
 *
 */

package eu.artemis.shield.discovery.slpdaemon.impl;

import java.io.*;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.Signature;
```

```
import java.security.interfaces.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.X509EncodedKeySpec;
import java.util.*;
import java.net.InetAddress;
import java.security.InvalidKeyException;
import java.security.SignatureException;

//import eu.artemis.shield.discovery.pdm_slp.sdc.slpapi.ServiceLocationAttribute;

public class slpMsgParser {


  private static boolean WRITE_LOG_ON_FILE = false;

    da daf;
    Database database;
    int    version, func_id, packet_len, slp_flag, ext_offset, xid,
           ecode, lifetime,attrlength;
    String ltag, scope, attr, pred, spi, tag, type, prlist, url,_attr1;
    String attrList, typeList, urlList;
    int    daBootTS;              // DA boot timestamp
    int    meshFwdID;             // in MeshFwd ext.
    long   versionTS;             // version TS from the SA for the update
    long   arrivalTS;             // arrival TS at the DA for the update
    String acceptDA;              // the accept DA for the update
    long   acceptTS;              // the accept TS for the update
    Vector atsList, adaList;      // used in DataRqst
    Vector ssExtList;             // used in SrvRqst
    boolean hasSelectExt;         // whether has Select extensions
    Vector urlVector, attrVector; // used in AttrList extension
    boolean hasAttrListExt;       // whether has AttrList extensions
    int    totalMatch;            // number of matches before selection
    Vector matchedEntry;          // matched entries in database
    int    etrpType;
    PrintStream ps;
    long after, before;
    InetAddress daTCPServeria;
    RSAPublicKey _publicKey;


    /*
    slpMsgParser(da daf) throws Exception {  // for DA
      init(daf);
    }
    */

    slpMsgParser(da daf, InetAddress ia) throws Exception {  // for DA
      init(daf);
      daTCPServeria = ia;
      url = ia.getHostAddress();
      da.appendDebug("slpMsgParser::slpMsgParser->ia.hostAdrr ="+ ia.getHostAddress());
    }

    public InetAddress getTCPServerInetAddress()
    {
      return daTCPServeria;
    }

    private void init(da daf) throws Exception {  // for DA
        this.daf = daf;
        database = daf.getDatabase();
        atsList = new Vector(10);
        adaList = new Vector(10);
        ssExtList = new Vector(10);
        urlVector  = new Vector(10);
        attrVector = new Vector(10);
        if (WRITE_LOG_ON_FILE) {
          File f = new File(".","logTest.txt");
          FileOutputStream fos = new FileOutputStream(f);
      ps = new PrintStream(fos);
        }
    }


    slpMsgParser() {        // for UA/SA
        ssExtList = new Vector(10);
        urlVector  = new Vector(10);
```

```
        attrVector = new Vector(10);
    }

/**
 * SLP common message header, not including language tag
 *+--------------+-----------------+---------------------------------+
 *|  Version     |   Function-ID   |            Length               |
 *+--------------+-+-+-+-----------+---------------+-----------------+
 *| Length cont. |O|F|R|       Reserved            | Next Ext. Offset |
 *+--------------+-+-+-+-----------+---------------+-----------------+
 *|     Next Ext. Offset  Cont.    |              XID                |
 *+--------------------------------+---------------------------------+
 */
    public void Header(byte[] buf) {  // parse header
        int[] ia = { 0 };
        version    = Util.parseInt(buf, ia, 1);         // index=0
        func_id    = Util.parseInt(buf, ia, 1);         // index=1
        packet_len = Util.parseInt(buf, ia, 3);         // index=2
        slp_flag   = Util.parseInt(buf, ia, 2);         // index=5
        ext_offset = Util.parseInt(buf, ia, 3);         // index=7
        xid        = Util.parseInt(buf, ia, 2);         // index=10
    }

    public void LangTag(byte[] buf, int ia[]) {  // parse language tag
        ltag = Util.parseString(buf, ia);
    }

    public int getPacketLen() {
        return packet_len;
    }

    public int getFuncID() {
        return func_id;
    }

    public int getFlag() {
        return slp_flag;
    }

    public int getXID() {
        return xid;
    }

    public String getLtag() {
        return ltag;
    }

    public String getURL() {
        return url;
    }

    public String getScope() {
        return scope;
    }

    public String getAttr() {
        return attr;
    }

    public String getAttrList() {
        return attrList;
    }

    public String getTypeList() {
        return typeList;
    }

    public String getUrlList() {
        return urlList;
    }

    public int getEcode() {
        return ecode;
    }

    public int getDaBootTS() {
        return daBootTS;
    }
```

```
    public int getMeshFwdID() {
        return meshFwdID;
    }

    public long getVersionTS() {
        return versionTS;
    }

    public String getAcceptDA() {
        return acceptDA;
    }

    public long getAcceptTS() {
        return acceptTS;
    }

    public Vector getAtsList() {          // be careful Object CANNOT be
        return (Vector) atsList.clone();  // shared between two threads
    }

    public Vector getAdaList() {          // CANNOT be shared!
        return (Vector) adaList.clone();
    }

    public int findTotalMatch() {
        if (ssExtList.size() > 0) {
            SelectSortExt ss = (SelectSortExt) ssExtList.elementAt(0);
            if (ss.getID() == Const.SelectExt) return ss.getBound();
        }
        return 0;
    }

    public int getTotalMatch() {
        return totalMatch;
    }

    public int getEtrpType() {
        return etrpType;
    }

    public boolean hasSelectExt() {
        return hasSelectExt;
    }

    public boolean hasAttrListExt() {
        return hasAttrListExt;
    }

    public Vector getMatchedEntry() {
        return matchedEntry;
    }

    public Vector getUrlVector() {
        return urlVector;
    }

    public Vector getAttrVector() {
        return attrVector;
    }

/**
 * parse URL entry, to get the lifetime & URL string
 *+---------------+---------------------------------+-----------------+
 *|    Reserved   |          Lifetime               |   URL length    |
 *+---------------+---------------------------------+-----------------+
 *| URL len cont. |          URL (variable length)                    \
 *+---------------+---------------------------------------------------+
 *| # of URL auths|          Auth. blocks (if any)                    \
 *+-------------------------------------------------------------------+
 */
    public String parseURL(byte[] buf, int[] ia) {
      da.appendDebug("slpMsgParser::parseURL");

      ia[0] += 1;                             // skip one byte for reserved
        lifetime = Util.parseInt(buf, ia, 2);   // lifetime
        url      = Util.parseString(buf, ia);   // URL
        if (Util.parseInt(buf, ia, 1) != 0) {
```

```
            da.appendDebug("slpMsgParser::parseURL -> URL authentication blocks are present");
        }
        return url;
    }

/**
 * service request <#1>
 *+----------------------------+---------------------------+
 *| length of <PRList>         |   <PRList> string         \
 *+----------------------------+---------------------------+
 *| length of <service-type>   | <service-type> string     \
 *+----------------------------+---------------------------+
 *| length of <scope-list>     | <scope-list> string       \
 *+----------------------------+---------------------------+
 *| length of predicate string | service request predicate \
 *+----------------------------+---------------------------+
 *| length of <SLP SPI> string |   <SLP SPI> string        \
 *+----------------------------+---------------------------+
 */
   public byte[] SrvRqst(byte[] buf, int[] ia) {
        prlist = Util.parseString(buf, ia);             // PRList
        type   = Util.parseString(buf, ia);             // service type
        scope  = Util.parseString(buf, ia);             // scope list
        pred   = Util.parseString(buf, ia);             // predicate
        spi    = Util.parseString(buf, ia);             // SLP SPI string
        if (type.equalsIgnoreCase(Const.DAAdvert_Rqst)) return null;
        byte[] tmp = database.getMatchedURL(type, scope, pred, ltag, ssExtList, ecode, getFuncID(
), null);

        totalMatch = database.getTotalMatch();
        matchedEntry = database.getMatchedEntry();
        if (Const.MESSAGE_LOG_ENABLED && Const.SrvRqst_LOG_ENABLED) da.append("\nINCOMING SRVRQST
 MESSAGE" +
                           "\n- Xid = "+ xid +
                         "\n- PRList = "+ prlist +
                           "\n- Type = " + type +
                           "\n- Scope = " + scope +
                           "\n- Predicate = " + pred +
                           "\n- SPI = " + spi +
                           "\n");
        return tmp;
    }
    /**
     * secure service request <#13>
     *+----------------------------+---------------------------+
     *| length of <PRList>         |   <PRList> string         \
     *+----------------------------+---------------------------+
     *| length of <service-type>   | <service-type> string     \
     *+----------------------------+---------------------------+
     *| length of <scope-list>     | <scope-list> string       \
     *+----------------------------+---------------------------+
     *| length of predicate string | service request predicate \
     *+----------------------------+---------------------------+
     *| length of <SLP SPI> string |   <SLP SPI> string        \
     *+----------------+-----------+---------------------------+
     *| # of AttrAuths |      Attribute Authentication Blocks  \
     *+----------------+---------------------------------------+
     */
        public byte[] SrvRqstAuth(byte[] buf, int[] ia) {
            prlist = Util.parseString(buf, ia);             // PRList
            type   = Util.parseString(buf, ia);             // service type
            scope  = Util.parseString(buf, ia);             // scope list
            pred   = Util.parseString(buf, ia);             // predicate
            spi    = Util.parseString(buf, ia);             // SLP SPI string
            int authBlockNum = Util.parseInt(buf, ia, 1);
            da.appendDebug("slpMsgParser::SrvRqstAuth -
> " + authBlockNum + " Attribute Authentication Block(s) found");
            if(authBlockNum!=0){
              AuthBlock(buf,ia,authBlockNum);
              }
            else ecode=Const.AUTHENTICATION_ABSENT;
            if (type.equalsIgnoreCase(Const.DAAdvert_Rqst)) return null;
            byte[]  tmp = database.getMatchedURL(type, scope, pred, ltag, ssExtList, ecode,getFun
cID(), _publicKey);

            totalMatch = database.getTotalMatch();
            matchedEntry = database.getMatchedEntry();
            if (Const.MESSAGE_LOG_ENABLED && Const.SrvRqstAuth_LOG_ENABLED) da.append("\nINCOMING
```

```
 SECURESRVRQST MESSAGE" +
                                    "\n- Xid = "+ xid +
                                "\n- PRList = "+ prlist +
                                    "\n- Type = " + type +
                                    "\n- Scope = " + scope +
                                    "\n- Predicate = " + pred +
                                    "\n- SPI = " + spi +
                                    "\n");
            return tmp;
        }

/**
 * service reply (reply for service request) <#2>
 *+-----------------------------+-------------------------+
 *|       Error Code            |    URL entry count      |
 *+-----------------------------+-------------------------+
 *|   <URL entry 1>           ...     <URL entry N>       \
 *+-----------------------------+-------------------------+
 */
    public void SrvReply(byte[] buf, int[] ia) {
        ecode = Util.parseInt(buf, ia, 2);
        if (Const.MESSAGE_LOG_ENABLED && Const.SrvRply_LOG_ENABLED) da.append("\nINCOMING SRVREPL
Y MESSAGE" +
                                "\n- Xid = "+ xid +
                            "\n- Error Code = " + ecode);
        int n = Util.parseInt(buf, ia, 2);
        if (Const.MESSAGE_LOG_ENABLED && Const.SrvRply_LOG_ENABLED) da.append("\n- URLs = " + n);
        StringBuffer tl = new StringBuffer();
        for (int i=0; i<n; i++) {
            if (tl.length() > 0) tl.append(",");
            String strurl = parseURL(buf, ia);
            if (Const.MESSAGE_LOG_ENABLED && Const.SrvRply_LOG_ENABLED) da.append("\n "+ (i+1) +"
> " + strurl);
            tl.append(url);       // URL only, no lifetime
        }
        urlList = tl.toString();
        if (Const.MESSAGE_LOG_ENABLED && Const.SrvRply_LOG_ENABLED) da.append("\n");
    }

/**
 * service registration <#3>
 *+-------------------------------------------------------------------+
 *|                          <URL-Entry>                              \
 *+---------------------------------+---------------------------------+
 *|  length of service type string  |     <service-type>              \
 *+---------------------------------+---------------------------------+
 *|  length of <scope-list>         |     <scope-list>                \
 *+---------------------------------+---------------------------------+
 *|  length of attr-list string     |     <attr-list>                 \
 *+----------------+----------------+---------------------------------+
 *| # of AttrAuths | (if present) Attribute Authentication Blocks  \
 *+----------------+--------------------------------------------------+
 * Need to set error code (ecode)
 * Extensions have been parsed, so versionTS/acceptDA/acceptTS are known
 */
    public void SrvReg(byte[] buf, int[] ia)
    {
        parseURL(buf, ia);                          // URL
        type  = Util.parseString(buf, ia);          // service type
        scope = Util.parseString(buf, ia);          // scope list
        attrlength=Util.parseInt(buf, ia, 2);       // length of attr-list
        ia[0]=ia[0]-2;
        attr  = Util.parseString(buf, ia);          // attribute list
        _attr1=attr;

        int authBlockNum = Util.parseInt(buf, ia, 1);
        da.appendDebug("slpMsgParser::SrvReg -
> " + authBlockNum + " Attribute Authentication Block(s) found");

        if(authBlockNum!=0)
        {
          AuthBlock(buf,ia,authBlockNum);
        }
        else if (authBlockNum==0)
        {
          if (Const.SERVICE_REGISTRATION_AUTH_REQUIRED)
          {
            da.appendDebug("> Error: Authentication Absent");
```

```
                ecode=Const.AUTHENTICATION_ABSENT;
                da.displayMessage ("Authentication Block NOT found", Const.EXCLAMATION, "Authentica
tion check");
             }
          }

          if (lifetime < 0) //Lifetime == 0 will be considered VALID and as a service with an INFIN
ITE Lifetime
          {
              ecode = Const.INVALID_REGISTRATION;
          }
          else if (!Util.shareString(daf.getScope(), scope, ","))
          {
              ecode = Const.SCOPE_NOT_SUPPORTED;
          }
          else
          {
              if (acceptDA.equalsIgnoreCase(daf.getFQDN()))
              {
                  arrivalTS = acceptTS;
              }
              else
              {
                  arrivalTS = System.currentTimeMillis();
              }

              if(ecode==0)
              {
              ecode = database.addEntry(false, ltag, type, url, lifetime, scope,
                    attr, slp_flag, versionTS, arrivalTS, acceptDA, acceptTS);
              }
          }

          if (Const.MESSAGE_LOG_ENABLED && Const.SrvReg_LOG_ENABLED)
          {
            String token = null;
            String attribute = null;
            StringTokenizer st = new StringTokenizer(attr,",");
            attr = "";
           while (st.hasMoreTokens())
           {
             token = st.nextToken();
             if (token.startsWith("("))
             {
               if (token.endsWith(")"))
               {
                 /*
                  * We have a valid attribute in token, with one value
                  * (attr_tag=attr_values)
                  */
                 attr += "\n                    " + token.substring(1,token.length()-
1);
               }
               else
               {
                 /*
                  * We have the first part of a valid attribute in token, with more than one value
                  * (attr_tag=attr_values)
                  */
                 attribute = token.substring(1);
               }
             }
             else
             {
               if (attribute != null)
               {
                 if (token.endsWith(")"))
                 {
                   /*
                     * We have the last part of a valid attribute in token, with more than one valu
e
                     * (attr_tag=attr_values)
                     */
                   attr += "\n                    " + attribute + "," + token.substring(0,token.leng
th()-1);
                   attribute = null;
                 }
                 else
```

```
                {
                  /*
                    * We have the N part of a valid attribute in token, with more than one value
                    * (attr_tag=attr_values)
                    */
                  attribute += "," + token;
                }
              }
              else
              {
                /*
                 * We have an attribute in token with no values
                 * attr_tag
                 */
                //da.appendDebug("Attribute Name = " + token);
                attr += "\n                    " + token;
              }
            }
          }
                da.append("\nINCOMING SRVREG MESSAGE"+
                          "\n- Xid = "+ xid +
                        "\n- Type = " + type +
                        "\n- Scope = " + scope +
                        "\n- Url = " + url +
                        "\n- Lifetime = " + lifetime +
                        "\n- Attribute List = " + attr +
                        "\n");

          }
    }

/**
* Authentication Block
*+--------------------------------+------------------------------+
*|  Block   Structure  Descriptor  | Authentication Block Length  \
*+--------------------------------+------------------------------+
*|                            Timestamp                          \
*+--------------------------------+------------------------------+
*|           SPI Length           |             SPI              \
*+--------------------------------+------------------------------+
*|                            SIGNATURE                          \
*+---------------------------------------------------------------+
 */

   public void AuthBlock(byte[] buf,int[] ia,int authblocknum){

        for (int i = 1; i <= authblocknum; i++)
        {
          da.appendDebug("slpMsgParser::SrvReg -> Parsing Authentication Block [" + i + "]");
          int BSD = Util.parseInt(buf, ia, 2);
            da.appendDebug(" > BSD = " + BSD);
            int blockLength = Util.parseInt(buf, ia, 2);
            da.appendDebug(" > Block Length = " + blockLength);
            int timeStamp = Util.parseInt(buf, ia, 4);
            da.appendDebug(" > Time Stamp = " + (new Date((long)timeStamp*1000)).toString());

            /*Estrazione chiave Pubblica*/
            ia[0]=ia[0]+2;
             byte[] _spi=new byte[162];
              for(int k=0; k < 162; k++)
              {
                 _spi[k] = buf[ia[0]+k];
              }
              ia[0]=ia[0]+162;
              try{
              X509EncodedKeySpec pubKeySpec = new X509EncodedKeySpec(_spi);
              KeyFactory keyFactory = KeyFactory.getInstance("RSA");
               _publicKey =(RSAPublicKey) keyFactory.generatePublic(pubKeySpec);

              da.append(" > SPI = "+_publicKey.toString());
              }catch (NoSuchAlgorithmException e) {
                   // TODO Auto-generated catch block
                   e.printStackTrace();
              }catch (InvalidKeySpecException e) {
                  // TODO Auto-generated catch block
                   e.printStackTrace();
              }
```

```
            /*Signature*/
            int signLength = blockLength - 10 - _spi.length;
            byte[] _sign = new byte[signLength];

            for (int j = 0; j < signLength; j++)
            {
              _sign[j] = buf[ia[0]+j];
            }
            da.append(" > SIGN = "+ _sign);

            /*Sign Verify*/
          if(func_id==Const.SrvReg)
          {

            try{
            ByteArrayOutputStream bos = new ByteArrayOutputStream();
            DataOutputStream dos = new DataOutputStream(bos);

            dos.writeShort(_spi.length);
            dos.write(_spi);
            dos.writeShort((short) attrlength);

            String token1 = null;
           StringTokenizer st1 = new StringTokenizer(_attr1,",");
           _attr1="";

           while (st1.hasMoreTokens())
          {
            token1 = st1.nextToken();
            if (token1.startsWith("("))
            {
              if (token1.endsWith(")"))
              {

                int indexuguale= token1.indexOf("=");
                dos.write("(".getBytes());
                    dos.write(token1.substring(1,indexuguale).getBytes());
                  dos.write("=".getBytes());
                  dos.write(token1.substring(indexuguale+1,token1.length()-1).getBytes());
                dos.write(")".getBytes());
              }
            }
            else
            {
              dos.write(token1.getBytes());
            }

          }

            dos.writeInt(timeStamp);
            dos.write(getURL().getBytes());

            byte[] data=bos.toByteArray();
            Signature signature = Signature.getInstance("SHA1withRSA");
            signature.initVerify(_publicKey);
            signature.update(data);

            boolean verify=signature.verify(_sign);
               if(verify){
                            da.append("> Athentication Executed Correctly");
                            }
              else{
              da.append("> Error: Authentication Failed");
              da.displayMessage ("Authentication Failed", Const.CRITICAL, "Authentication check")
;
              ecode=Const.AUTHENTICATION_FAILED;
            }

            }catch(IOException e1){
              //DO nothing???
            }
            catch (NoSuchAlgorithmException e2) {
              da.appendDebug("Algorithm not supported");
                da.displayMessage ("Encryption Algorithm NOT supported", Const.EXCLAMATION, "Auth
entication check");

              }
```

```
            catch (InvalidKeyException e3) {
              da.appendDebug("Invalid public key");
                da.displayMessage ("Invalid public key", Const.EXCLAMATION, "Authentication check
");

              }
            catch (SignatureException e4) {
              da.appendDebug("Invalid signature");
                da.displayMessage ("Invalid signature", Const.EXCLAMATION, "Authentication check"
);
              }
         }
        else if(func_id==Const.SrvRqstAuth)
              {
          try{
                ByteArrayOutputStream bos = new ByteArrayOutputStream();
                DataOutputStream dos = new DataOutputStream(bos);

                dos.writeShort(_spi.length);
                dos.write(_spi);
                dos.writeShort(type.getBytes().length);
                dos.write(type.getBytes());
                dos.writeShort(scope.getBytes().length);
                dos.write(scope.getBytes());
                dos.writeShort(pred.getBytes().length);
                dos.write(pred.getBytes());
                dos.writeInt(timeStamp);

                byte[] data=bos.toByteArray();
                Signature signature = Signature.getInstance("SHA1withRSA");
                signature.initVerify(_publicKey);
                signature.update(data);

                boolean verify=signature.verify(_sign);
                   if(verify){
                               da.append("> Athentication Executed Correctly");
                               }
                  else{
                  da.append("> Error: Authentication Failed");
                  ecode=Const.AUTHENTICATION_FAILED;
                }

                }catch(IOException e1){
                  //DO nothing???
                }
                catch (NoSuchAlgorithmException e2) {
                    da.appendDebug("Encryption Algorithm not supported");
                    da.displayMessage ("Encryption Algorithm NOT supported", Const.EXCLAMATION, "
Authentication check");

                  }

                catch (InvalidKeyException e3) {
                  da.appendDebug("Invalid public key");
                    da.displayMessage ("Invalid public key", Const.EXCLAMATION, "Authentication c
heck");

                  }
                catch (SignatureException e4) {
                  da.appendDebug("Invalid signature");
                    da.displayMessage ("Invalid signature", Const.EXCLAMATION, "Authentication ch
eck");
                  }

              }
        }


    }

/**
 * service De-registration <#4>
 *+-----------------------------+--------------------------+
 *|  Length of <scope-list>     |        <scope-list>      \
 *+-----------------------------+--------------------------+
 *|                          <URL-entry>                     \
 *+-----------------------------+--------------------------+
 *|  Length of <tag-list>       |        <tag-list>        \
```

```
 *+-----------------------------+--------------------------+
 * Need to set error code, 0 is for OK
 */
    public void SrvDeReg(byte[] buf, int[] ia) {
        scope = Util.parseString(buf, ia);              // scope list
        parseURL(buf, ia);                              // URL
        tag   = Util.parseString(buf, ia);              // tag list
        if (Util.shareString(daf.getScope(), scope, ",")) {
            ecode = database.rmEntry(ltag, url, scope, tag, versionTS,
                                     acceptDA, acceptTS);
        } else {
            ecode = Const.SCOPE_NOT_SUPPORTED;
        }
        if (Const.MESSAGE_LOG_ENABLED && Const.SrvDeReg_LOG_ENABLED) da.append("\nINCOMING SRVDER
EG MESSAGE" +
        "\n- Xid = "+ xid +
        "\n- Scope = " + scope +
        "\n- Url = " + url +
        "\n- Tag List = " + tag +
        "\n");
    }

/**
 * service ack (reply for SrvReg & SrvDeReg) <#5>
 *+-----------------------------+
 *|        Error Code           |
 *+-----------------------------+
 * return the error code
 */
    public void SrvAck(byte[] buf, int[] ia) {
        ecode = Util.parseInt(buf, ia, 2);
        if (Const.MESSAGE_LOG_ENABLED && Const.SrvAck_LOG_ENABLED) da.append("\nINCOMING SRVACK M
ESSAGE" +
                            "\n-Xid = "+ xid +
                        "\n- Error = " + ecode +
                        "\n");
    }

/**
 * attribute request <#6>
 *+-------------------------------+---------------------------+
 *|  length of PRList             |   <PRList> string          \
 *+-------------------------------+---------------------------+
 *|  length of URL                |         URL                \
 *+-------------------------------+---------------------------+
 *|  length of <scope-list>       |   <scope-list> string      \
 *+-------------------------------+---------------------------+
 *|  length of <tag-list> string  |   <tag-list> string        \
 *+-------------------------------+---------------------------+
 *|  length of <SLP SPI> string   |   <SLP SPI> string         \
 *+-------------------------------+---------------------------+
 */
    public void AttrRqst(byte[] buf, int[] ia) {
      da.appendDebug("slpMsgParser::AttrRqst");
      prlist = Util.parseString(buf, ia);             // PRList
        url    = Util.parseString(buf, ia);             // URL or Service type
        scope  = Util.parseString(buf, ia);             // scope list
        tag    = Util.parseString(buf, ia);             // tag list
        spi    = Util.parseString(buf, ia);             // SLP SPI string

        /*
         * Check if at least one scope is in the list of the DA
         */
        if (!Util.shareString(daf.getScope(), scope, ","))
        {
          // NO SCOPES...
          ecode = Const.SCOPE_NOT_SUPPORTED;
            attrList = "";
        }
        else
        {
            ecode = Const.OK;
            attrList = database.getAttrList(url, scope, tag, ltag);
        }
        if (Const.MESSAGE_LOG_ENABLED && Const.AttrRqst_LOG_ENABLED) {

          da.append("\nINCOMING ATTRRQST MESSAGE"+
        "\n- Xid = "+ xid +
```

```
        "\n- Scope = " + scope +
        "\n- Attribute List = " + tag +
        "\n- Service URL or Type = " + url +
        "\n- Prlist = " + prlist +
        "\n- Spi = " + spi +
        "\n");
        }
    }

/**
 * attribute reply (reply for attribute request) <#7>
 *+-------------------------------+---------------------------------+
 *|         Error Code            |   length of <attr-list>         |
 *+-------------------------------+---------------------------------+
 *|                          <attr-list>                            \
 *+----------------+----------------------------------------------+
 *| # of AttrAuths |  Attribute authentication block (if present) \
 *+----------------+----------------------------------------------+
 */
    public void AttrReply(byte[] buf, int[] ia) {
        ecode = Util.parseInt(buf, ia, 2);
        attrList = Util.parseString(buf, ia);
        if (Const.MESSAGE_LOG_ENABLED && Const.AttrRply_LOG_ENABLED){
          da.append("\nINCOMING ATTRREPLY MESSAGE"+

        "\n- Xid = "+ xid +
        "\n- Error = " + ecode +
        "\n- Attribute List = " + attrList +
        "\n");
        }
    }

/**
 * directory agent advertisement <#8>
 *+---------------------------------+---------------------------------+
 *|     Error Code                  |   DA Stateless Boot Timestamp   |
 *+---------------------------------+---------------------------------+
 *| DA Stateless Boot Time cont.    |      length of URL              |
 *+---------------------------------+---------------------------------+
 *|                               URL                                 \
 *+---------------------------------+---------------------------------+
 *|   length of <scope-list>        |      <scope-list>               \
 *+---------------------------------+---------------------------------+
 *|   length of <attr-list>         |      <attr-list>                \
 *+---------------------------------+---------------------------------+
 *|   length of SLP <SPI>           |      SLP <SPI> string           \
 *+---------------+-----------------+---------------------------------+
 *| # Auth Blocks |       Authentication block (if any)               \
 *+---------------+-----------------+---------------------------------+
 */
    public void DAAdvert(byte[] buf, int[] ia) {
      /*da.appendDebug("\n\n\n\nslpMsgPrs::DAADVERT\n\n\n\n");
      for (int i=0; i<buf.length;i++){
        da.appendDebug(buf[i]);
      }*/
      ecode = Util.parseInt(buf, ia, 2);
        daBootTS = Util.parseInt(buf, ia, 4);           // boot timestamp
        url    = Util.parseString(buf, ia);             // URL
        scope = Util.parseString(buf, ia);              // scope-list
        attr  = Util.parseString(buf, ia);              // attr-list
        da.appendDebug("slpMsgParser::DAAdvert-URL ="+url);

        if (Const.MESSAGE_LOG_ENABLED && Const.DAAdvert_LOG_ENABLED) da.append("\nINCOMING DAADVE
RT MESSAGE"+
        "\n- Xid = "+ xid +
        "\n- Url = " + url +
        "\n- Scope = " + scope +
        "\n- Attribute List = " + attr +
        "\n");
    }

/**
 * service type request <#9>
 *+-------------------------------+-----------------------------+
 *|     length of PRList          |     <PRList> string         |
 *+-------------------------------+-----------------------------+
 *| length of Naming Authority    |  <Naming Authority String>  |
 *+-------------------------------+-----------------------------+
```

```
 *| length of <scope-list>        |    <scope-list> string      |
 *+-------------------------------+-----------------------------+
 */
    public void SrvTypeRqst(byte[] buf, int[] ia) {
        prlist = Util.parseString(buf, ia);             // PRList
        String na = Util.parseString(buf, ia);          // Naming authority
        scope  = Util.parseString(buf, ia);             // scope list
        if (!Util.shareString(daf.getScope(), scope, ","))
        {
            ecode = Const.SCOPE_NOT_SUPPORTED;
            typeList = "";
        }
        else
        {
            ecode = Const.OK;
            typeList = database.getServiceTypeList(na, scope);
        }
        if (Const.MESSAGE_LOG_ENABLED && Const.SrvTypeRqst_LOG_ENABLED)
        {

          da.append("\nINCOMING SRVTYPERQST MESSAGE"+
        "\n- Xid = "+ xid +
        "\n- PRList = " + prlist +
        "\n- Naming Authority = " + na +
        "\n- Scope = " + scope +
        "\n");
        }
    }

/**
 * service type reply (reply for service type request) <#10>
 *+-------------------------------+-------------------------------+
 *|      Error Code               |    length of <srvType-list>   |
 *+-------------------------------+-------------------------------+
 *|                       <srvType-list>                          |
 *+---------------------------------------------------------------+
 */
    public void SrvTypeReply(byte[] buf, int[] ia) {
        ecode = Util.parseInt(buf, ia, 2);
        typeList = Util.parseString(buf, ia);
        if (Const.MESSAGE_LOG_ENABLED && Const.SrvTypeRply_LOG_ENABLED)
        {
          da.append("\nINCOMING SRVTYPEREPLY MESSAGE"+

        "\n- Xid = "+ xid +
        "\n- Type List = " + typeList +
        "\n");
        }
    }

/**
 * AntiEtrpRqst <#12> message
 *+-------------------------------+-------------------------------+
 *|      Anti-entropy type ID     |   Number of Accept ID entries |
 *+-------------------------------+-------------------------------+
 *|    Accept ID Entry 1         ...          Accept ID Entry k   \
 *+---------------------------------------------------------------+
 */
    public void AntiEtrpRqst(byte[] buf, int[] ia) {
        etrpType = Util.parseInt(buf, ia, 2);
        int k  = Util.parseInt(buf, ia, 2);
        atsList.clear();
        adaList.clear();
        for (int i=0; i<k; i++) {
            atsList.addElement(new Long(Util.parseLong(buf, ia)));
            adaList.addElement(Util.parseString(buf, ia));
        }
        if (Const.MESSAGE_LOG_ENABLED && Const.AntiEtrpRqst_LOG_ENABLED) da.append("\nINCOMING AN
TIETRPRQST MESSAGE"+
        "\n- Xid = "+ xid +
        "\n- Entropy Type = " + etrpType +
        "\n- ATS List = " + atsList.toString() +
        "\n- ADA List = " + adaList.toString() +
        "\n");
    }

/**
 * Mesh Forwarding extension parser:
```

```
*    (1) initialize (turn off previous value)
*    (2) if MeshFwdExt,
*           get Fwd-ID & versionTS
*           if Fwd-ID == Const.RqstFwd, change it to Const.Fwded
*           if Fwd-ID == Const.Fwded, get acceptDA & acceptTS
*+--------------------------------+--------------------------------+
*|  MeshFwd Extension ID = 0x0006 |  Next Extension Offset (NEO)   |
*+--------------+-----------------+--------------------------------+
*|  NEO Contd.  |     Fwd-ID      |         Version Timestamp      |
*+--------------+-----------------+--------------------------------+
*|                     Version Timestamp, contd.                   |
*+--------------------------------+--------------------------------+
*|    Version Timestamp, contd.   |           Accept ID            \
*+--------------------------------+--------------------------------+
*/
   public void MeshFwdExt(byte[] buf, String fromPeer) { // buf: whole msg
       meshFwdID = -1;                  // no MeshFwdExt
       acceptDA  = daf.getFQDN();
       acceptTS  = System.currentTimeMillis();
       versionTS  = acceptTS;
       int[] ia = { ext_offset };       // initial extension offset
       while (ia[0] != Const.EndOfExt) { // while has more extensions
           int extID = Util.parseInt(buf, ia, 2);      // extension ID
           int nextExt = Util.parseInt(buf, ia, 3);    // next extension
           if (extID == Const.MeshFwdExt) {    // mesh-forwarding extension
               int idAddr = ia[0];       // may need to MeshFwdID
               meshFwdID = Util.parseInt(buf, ia, 1);
               versionTS  = Util.parseLong(buf, ia);
               if (meshFwdID == Const.Fwded) {
                   acceptTS = Util.parseLong(buf, ia);
                   acceptDA = Util.parseString(buf, ia);
                   daf.setSummary(acceptDA, acceptTS, fromPeer);
               } else if (meshFwdID == Const.RqstFwd) {
                   Util.writeInt(buf, idAddr, Const.Fwded, 1);
                   Util.writeLong(buf, idAddr+9, acceptTS);
               }
               break; // at most one MeshFwd extension
           }
           ia[0] = nextExt;
       }
       if (meshFwdID != Const.Fwded) {         // accepted by local host
           daf.setSummary(acceptDA, acceptTS, acceptDA);
       }
   }

   public void SelectSortExt(byte[] buf) { // buf: whole message
       ssExtList.clear();
       hasSelectExt = false;
       int[] ia = { ext_offset };       // initial extension offset
       while (ia[0] != Const.EndOfExt) { // while has more extensions
           int extID = Util.parseInt(buf, ia, 2);      // extension ID
           int nextExt = Util.parseInt(buf, ia, 3);    // next extension
           if (extID == Const.SelectExt) {     // selection extension
               hasSelectExt = true;
               int num = Util.parseInt(buf, ia, 2);
               ssExtList.addElement(new SelectSortExt(Const.SelectExt, num));
           } else if (extID == Const.SortExt) {  // sort extension
               String key = Util.parseString(buf, ia);
               ssExtList.addElement(new SelectSortExt(Const.SortExt, key));
           }
           ia[0] = nextExt;
       }
   }

   public void AttrListExt(byte[] buf) { // buf: whole message
       urlVector.clear();
       attrVector.clear();
       hasAttrListExt = false;
       int[] ia = { ext_offset };       // initial extension offset
       while (ia[0] != Const.EndOfExt) { // while has more extensions
           int extID = Util.parseInt(buf, ia, 2);      // extension ID
           int nextExt = Util.parseInt(buf, ia, 3);    // next extension
           if (extID == Const.AttrListExt) {   // AttrList extension
               hasAttrListExt = true;
               urlVector.addElement(Util.parseString(buf, ia));  // url
               attrVector.addElement(Util.parseString(buf, ia)); // attr
           }
           ia[0] = nextExt;
```

```
            }
        }
}
```

package eu.artemis.shield.overlay.securityagent.impl.DiscoveryServlet.html

```
package eu.artemis.shield.overlay.securityagent.impl;

import java.awt.event.ActionEvent;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.LinkedList;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.osgi.framework.ServiceReference;

import eu.artemis.shield.discovery.gdm.interfaces.IGenericDiscovery;

public class DiscoveryServlet extends HttpServlet{
  private static final long serialVersionUID = 3472683797378538L;
                      //   DiscoveryServlet

  IGenericDiscovery GDM;

  ServiceReference httpSR;

  boolean print = false;

  public DiscoveryServlet(ServiceReference httpSR, IGenericDiscovery GDM) {
    this.httpSR = httpSR;
    this.GDM = GDM;
  }

  public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

    doGet(request, response);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    PrintWriter out = response.getWriter();

    response.setContentType("text/html");

    printHeader(out);

    try {
      if (request.getParameter("GO") != null
          && request.getParameter("GO").equals("GO")) {
        String VID = null;
        String type = "";
        String serviceType = "servicetype=";
        String languageTag = "language=";
        String gui = "gui=";
        if (request.getParameter("servicetype") != null) {
          serviceType += request.getParameter("servicetype");
        }
        if (request.getParameter("languagetag") != null) {
          languageTag += request.getParameter("languagetag");
          if (languageTag.equals("language=")) {
            out
                .println("<br><br><h4>You haven't specified the Language Tag, Default is \"en_GB\
"</h4>");
          }
        }
        String[] keywords = new String[3];
        keywords[2] = null;
        if(!request.getParameter("gui").equals(""))
        {
          gui += request.getParameter("gui");
          keywords[2] = gui;
        }
        keywords[0] = serviceType;
        keywords[1] = languageTag;

        LinkedList result = GDM.findServices(VID, type, keywords);
        // URL image =
```

```
        // ClassLoader.getSystemClassLoader().getResource("./provaIMG.jpg");
        if (result != null) {
          out
              .println("<h2> This is the URL list for OWL Description </h2>");
          out
              .println("<table style=\"border-
collapse: collapse\" BORDERCOLORDARK = #9999CC BORDERCOLORLIGHT = #9999CC BORDER = 1 CELLPADDING
= 5 CELLSPACING = 0>");

          for (int i = 0; i < result.size(); i++) {
            int j = i+1;
            out
                .println("\t<tr BORDERCOLOR = #0000FF BACKGROUND = #0000FF>");
            out
                .println("\t\t<td BGCOLOR = #9999CC BORDERCOLOR = #FFFFFF><p align = center><b>"
+ j + "</b></p></td>");
            out.println("\t\t<td BGCOLOR = #" + "F0F0FF"
                + " BORDERCOLOR = #FFFFFF>" + "<a href=\""
                + result.get(i).toString() + "\">"
                + result.get(i).toString() + "</a></td>"
                + "</tr>");

            /*
             * out.println("<table><tr><td><a href=\"" +
             * result.get(i).toString() + "\">" +
             * result.get(i).toString() + "</a></td>" + "<td></td></tr></table>");
             */
          }
          out.println("</table>");
        }
      }

    } catch (Exception e) {
      out.println("<pre>");
      e.printStackTrace(out);
      out.println("</pre>");
    }
  }

  public void actionPerformed(ActionEvent evt) {
    print = true;
  }

  void printHeader(PrintWriter out) throws IOException {
    out.println("<html>");
    out.println("<head>");
    out.println("<title>pSHIELD - Security Agent</title>");
      out.println("<table><tr><td><img width=100 src=\"../serviceprovider/sapienza_logo.gif\"></t
d><td>");
      out.println("<h1>University of Rome - pSHIELD </h1>");
      out.println("<h1>User Agent</h1>");
    out.println("<h2>Service Discovery Form</h2>");
    out.println("</td></tr><table>");

    out
        .println("<LINK href=\"/knopflerfish.css\" rel=\"stylesheet\" type=\"text/css\">");
    out.println("</head>");
    out.println("<body>");
    out.println("<form method=\"POST\">");
    out
        .println("<strong>Insert keywords to perform service discovery and then click \"GO\"</str
ong><br><br>");
    out.println("<table>");
    out
        .println("<tr><td><font size=\"3\"><strong>ServiceType  </strong></font></td><td><input t
ype=\"text\" name=\"servicetype\" value=\"\" size=\"50\" /></td></tr>");
    out
        .println("<tr><td><font size=\"3\"><strong>LanguageTag  </strong></font></td><td><input t
ype=\"text\" name=\"languagetag\" value=\"\" size=\"50\" /></td></tr>");
    out
        .println("<tr><td><font size=\"3\"><strong>GUI  </strong></font></td><td><input type=\"te
xt\" name=\"gui\" value=\"\" size=\"50\" /></td></tr>");
    out.println("</table>");
    out
        .println("<br><input type=submit value=\"GO\" name=\"GO\" id=\"GO\">");
  }

}
```

package eu.artemis.shield.overlay.securityagent.impl.SA.html

```
/**
 * pSHIELD
 * Service Discovery
 *
 * @author Vincenzo Suraci
 * @author Silvano Mignanti
 * Department of System and Computer Science (DIS)
 * University of Rome "Sapienza"
 * Via Ariosto, 25
 * 00184, Rome, IT
 *
 * phone: +39 340 156 22 58 / +39 329 11 38 610
 * email: vincenzo.suraci@dis.uniroma1.it / silvano.mignanti@dis.uniroma1.it
 *
 *
 * Created on 16-May-2007
 * Version 1.0
 *
 */

package eu.artemis.shield.overlay.securityagent.impl;

/**

 *
 * The present class shows how a pSHIELD 2 OSGi component
 * could use the potentiality offered by the pSHIELD 2
 * Service Discovery Framework. It interfaces with the
 * Generic Discovery Manager to discover the services
 * available in the (pSHIELD 2) network.
 */

import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceReference;

import eu.artemis.shield.discovery.gdm.interfaces.IGenericDiscovery;

import eu.artemis.shield.discovery.pdm.IService;
import eu.artemis.shield.discovery.pdm.IServiceList;
import eu.artemis.shield.discovery.pdm.IServiceProperty;
import eu.artemis.shield.discovery.pdm.IServicePropertyList;

import eu.artemis.shield.overlay.securityagent.ISecurityAgent;
import eu.artemis.shield.overlay.securityagent.impl.SAGUI;
import eu.artemis.shield.overlay.semanticknowledge.ISemanticKnowledge;

import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.Vector;
import java.util.Hashtable;


public class SA implements ISecurityAgent
{
  private BundleContext bc;
  private IGenericDiscovery gd = null;
  private SAGUI gui = null;
  public List fileOWL = new LinkedList();
  public ISemanticKnowledge sk;
  public Hashtable h_bundle = new Hashtable();

  public SA(BundleContext bc)
  {
    this.bc = bc;

    gui = new SAGUI(bc, this);
    gui.setVisible(true);

  }

  public Vector serviceTypeDiscovery()
  {
    Vector types_vector = new Vector();
    try
```

```
    {
      gui.append("Discovering Service Types ...");
      ServiceReference[] gdmList = findGenericDiscoveryModuleImplementations();
      if (gdmList != null)
      {
        /*
         * We ignore the possibility to have more than one GDM implementation.
         * JUST USE THE FIRST ONE...
         */
        gd = (IGenericDiscovery) bc.getService(gdmList[0]);

        /*
         * We are ready to start the discovery process!
         */

        // Obatin in the proper way the user agent VIDID
        String vid = null;

        /*
         * LET pSHIELD DISCOVER THE TYPES OF SERVICES
         */

        LinkedList ll = gd.findServiceTypes(vid);

        if (ll != null)
        {
          if (ll.isEmpty())
          {
            // NO SERVICES HAVE BEEN DISCOVERED
            gui.append("No Service Types Found !");
          }
          else
          {

            Iterator it = ll.iterator();
            while (it.hasNext())
            {
              String types = (String)it.next();
              types_vector.add( types );
            }
          }

        }
        else
        {
          // NO SERVICE TYPES HAVE BEEN DISCOVERED
          gui.append("No Service Types Found !");
        }
      }
      else
      {
        /*
         * It was not possible to find a suitable implementation of IGenericDiscovery
         */
        gui.append("No Bundles implement the IGenericDiscovery interface!\n");
      }
      gui.append("Done.");
      return types_vector;

    }
    catch (Exception e)
    {
      /*
       * Something went wrong!
       * It was not possible to find a suitable implementation of IGenericDiscovery
       */
      gui.append(e.getMessage());
      return types_vector;
    }
  }

  /**
   * This function takes a URL and a service Type and create the Service URL
   *
   * @param type service type
   * @param url URL
   * @return Service URL
   */
```

```
  private String serviceurl(String type, String url)
  {

    int index = type.lastIndexOf(":");
    if (index > 0)
    {
      String protocol = type.substring(index+1);
      if (url.startsWith(protocol))
      {
        String serviceurl = type + url.substring(protocol.length());
        return serviceurl;
      }
      else
      {
        gui.append("SA.java::serviceurl -
> Trying to merge different protocols:\nurl = " + url + "\ntype = " + type);
      }
    }
    return null;
  }

  /**
   * This function find the service available for a specified service type
   * @param type : the service type to search
   * @return : a vector of found services ( with no API bundles )
   * @throws PShieldEngineException
   * @throws Exception
   */
  public Vector serviceDiscovery( String type )  {

    return serviceDiscovery( type, null );

  }


  /**
   * This function find the service available for a specified service type and an array of keywor
ds to do a better filter
   * @param type : the service type to search
   * @param kw : an array of keywords to search
   * @return : a vector of found services ( with no API bundles )
   * @throws PShieldEngineException
   */
  public Vector serviceDiscovery( String type, String[] kw ) {
    return serviceDiscovery(type, kw, true);
  }

  /**
   * This function find the service available for a specified service type and an array of keywor
ds to do a better filter
   * @param type : the service type to search
   * @param kw : an array of keywords to search
   * @param filter_api : set false if you need all the available bundles, true if you want only t
he API bundles
   * @return a vector of found services ( with no API bundles )
   * @throws PShieldEngineException
   * @throws Exception
   */

  public Vector serviceDiscovery( String type, String[] kw, boolean filter_api )
  {

      Vector discovered_services = new Vector();
      try
      {

        gui.append("Service discovery with a specified service type");
        ServiceReference[] gdmList = findGenericDiscoveryModuleImplementations();

        if (gdmList != null)
        {
          /*
           * We ignore the possbility to have more than one GDM implementation.
           * JUST USE THE FIRST ONE...
           */
          gd = (IGenericDiscovery) bc.getService(gdmList[0]);
          /*
           * We are ready to start the discovery process!
```

```
         */

        // Obatin in the proper way the user agent VIDID
        String vid = null;


        /*
         * Set an array of keywords, useful to better filter
         * the services
         */
        if ( kw == null )
          kw = new String[0];

        for(int i = 0; i<kw.length; i++)
        {
          gui.append("KEYWORDS " + i + "--> " + kw[i].toString() + "\n");
        }
        /*
         * LET pSHIELD DISCOVER THE SERVICES
         */

        gui.append("Looking for Services ...");

        String CDQL =
          "SELECT default" +
          "FROM default" +
          "SERVICETYPE " + type +
          "LANGUAGE en_gb" +
          "WHERE " +
          "USING slp";
        String SPARQL = "";
        LinkedList query_output = null;
        IServiceList sl = gd.findServices(CDQL, SPARQL, query_output);

        if ( sl != null )
        {
          if ( sl.isEmpty() )
          {
            // NO SERVICES HAVE BEEN DISCOVERED
            gui.append("No services found !");
          }
          else
          {
              for (int i = 0; i < sl.size(); i++)
              {
                try
                {
                  IService s = sl.getService(i);
                  String url = s.getOWLSURL();
                  System.out.println("service URL #" + i + " = " + url);
                  IServicePropertyList spl = s.getProperties();
                  Hashtable ht = new Hashtable();
                  String owl = null;
                  for (int j=0; j<spl.size(); j++)
                  {
                    try
                    {

                      Object obj = null;
                      IServiceProperty sp = spl.getServiceProperty(j);
                      System.out.print("> attribute #" + j + " --> " + sp.getName() + "=");

                      //sp.getName() --> attribute's name

                      Vector values = sp.getValues();
                      if (values != null)
                      {

                        for (int k=0; k<values.size(); k++)
                        {
                          obj = values.get(k);
                          if (sp.getName().equals("ontologyURI")){
                            System.out.println("Ottengo le chiavi" + obj + " " + sp.getName() +
" " + values.toString());
                            owl = obj.toString();
                          }

                        }
```

```
                        }
                        ht.put(sp.getName(), values);


                        System.out.println("Valore attributo" + values);
                      }


                      catch (IndexOutOfBoundsException ioobe)
                      {
                        ioobe.printStackTrace();
                      }
                    }

                    if (owl != null && ht != null){

                        h_bundle.put(owl, ht);
                        System.out.println("Numero Bundle" + h_bundle.size() + " " + ht.get("Serv
ice Name") + " " + owl);

                    }

                    if (ht.containsKey("Impl") && ht.containsKey("Api"))
                    {
                      if (filter_api)
                      {
                        boolean has_impl = ((Boolean)((Vector)ht.get("Impl")).elementAt(0)).boole
anValue();
                        if (has_impl) discovered_services.add(ht);

                      }
                      else
                      {
                        // Insert an HashTable for each discovered service
                        discovered_services.add( ht );
                      }
                    }
                  }
                  catch (IndexOutOfBoundsException ioobe)
                  {
                    ioobe.printStackTrace();
                  }
                }
              }
            }
            else
            {
              //NO SERVICES HAVE BEEN DISCOVERED
              gui.append("NO SERVICE FOUND!");
            }

          }
          else
          {
            /*
             * It was not possible to find a suitable implementation of IGenericDiscovery
             */
            gui.append("No Bundles implement the IGenericDiscovery interface!\n");
          }
        }
        catch (Exception e)
        {
          /*
           * Something went wrong!
           * It was not possible to find a suitable implementation of IGenericDiscovery
           */
          gui.append(e.getMessage());
          e.printStackTrace();
        }


        List l = getServices(11);
        System.out.println("Numero bundle da avviare" + l.size());
        Hashtable prova = getBundle(l);

        try
        {
```

```
        ServiceReference[] skList = findSemanticKnoledgeImplementations();
        if (skList != null)
        {
          sk = (ISemanticKnowledge) bc.getService(skList[0]);

          return discovered_services;
        }

      }
      catch (Exception e)
      {
        /*
         * Something went wrong!
         * It was not possible to find a suitable implementation of ISemanticKnowledge
         */
        gui.append(e.getMessage());
      }

      return discovered_services;

  }



  /**
   *
   * @param type
   * @param kw
   * @param filter_api
   * @return The list of OWLfiles of bundles with a SPD level
   */
  public List getList( String type, String[] kw, boolean filter_api )
  {

    ServiceReference[] gdmList;
    try {
      gdmList = findGenericDiscoveryModuleImplementations();

      if (gdmList != null)
      {
      gd = (IGenericDiscovery) bc.getService(gdmList[0]);

      String CDQL =
        "SELECT default" +
        "FROM default" +
        "SERVICETYPE " + type +
        "LANGUAGE en_gb" +
        "WHERE " +
        "USING slp";
      String SPARQL = "";
      LinkedList query_output = null;

      IServiceList sl = gd.findServices(CDQL, SPARQL, query_output);

      if ( sl != null )
      {
        if ( sl.isEmpty() )
        {
          // NO SERVICES HAVE BEEN DISCOVERED
          gui.append("No services found !");
        }
        else
        {
            for (int i = 0; i < sl.size(); i++)
            {
                IService s = sl.getService(i);
                String url = s.getOWLSURL();
                IServicePropertyList spl = s.getProperties();
                for (int j=0; j<spl.size(); j++)
                {
                    IServiceProperty sp = spl.getServiceProperty(j);

                    Vector values = sp.getValues();
                    if (values != null)
                    {
                      for (int k=0; k<values.size(); k++)
                      {
                        Object obj = values.get(k);
```

```
                        if (obj != null)
                        {
                          if (k > 0) System.out.print(",");

                          if (sp.getName().equals("OntologyURI") && !obj.toString().equals(null))
                          {
                          System.out.println(sp.getName());
                          System.out.print(obj.toString());
                          System.out.println("Added" + " " + obj.toString());
                          fileOWL.add(obj.toString());

                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
          }


          return fileOWL;
        }

/**
 * This function returns the list of OWL files belonging to the bundles to start
 * @param i = security level
 * @return List of ontologies belonging to the bundles that have an SPD Level that respect securi
ty level i
 */

public List getServices(int i)
{

  List services = new LinkedList();

  services.add("http://www.owl-ontologies.com/Ontology1300273978.owl#CHAP");
  services.add("http://www.owl-ontologies.com/Ontology1300273978.owl#AES");
  services.add("http://www.owl-ontologies.com/Ontology1300273978.owl#PIN");

  return services;
}

/**
 * This function returns an HashMap with type of service and its security level
 * @param l List of ontologies
 * @return HashMap with type of service and its security level
 */

public HashMap getParameters(List l){

  HashMap ht = new HashMap();

  ht.put("Authentication", 8);
  ht.put("Accounting", 2);
  ht.put("Cryptography", 1);

  return ht;


}


  /**
   * @author Vincenzo Suraci
   *
   * This function uses the internal OSGi service discovery to find a suitable implementation
   * of the IGenericDiscovery interface.
   */
  public ServiceReference[] findGenericDiscoveryModuleImplementations() throws Exception
  {
    ServiceReference[] gdmi = null;
    try
```

```
    {
      gdmi = bc.getServiceReferences("eu.artemis.shield.discovery.gdm.interfaces.IGenericDiscover
y", null);
    }
    catch (Exception e)
    {
      throw e;
    }
    return gdmi;
  }

  /**
   * @author Vincenzo Suraci
   *
   * This function uses the internal OSGi service discovery to find a suitable implementation
   * of the ISemanticKnoledge interface.
   */
  public ServiceReference[] findSemanticKnoledgeImplementations() throws Exception
  {
    ServiceReference[] gdmi = null;
    try
    {
      gdmi = bc.getServiceReferences("eu.artemis.shield.overlay.semanticknowledge.ISemanticKnowle
dge", null);
    }
    catch (Exception e)
    {
      throw e;
    }
    return gdmi;
  }



  public void exit()
  {
    /*
     * Close GUI
     */
    gui.setVisible(false);
    gui.dispose();
  }
}
```

package eu.artemis.shield.overlay.securityagent.impl.SAGUI.html

```
package eu.artemis.shield.overlay.securityagent.impl;

import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.util.List;

import javax.swing.*;

import javax.swing.table.*;

import java.net.URL;

import org.osgi.framework.BundleContext;
import org.osgi.framework.BundleException;
import org.osgi.framework.ServiceReference;

import eu.artemis.shield.composition.compositionmanager.ICompositionManager;
import eu.artemis.shield.overlay.securityagent.impl.SA;


public class SAGUI extends Frame implements ActionListener
{
  private static final long serialVersionUID = 82484L;
                      //   UAGUI

  private static final int MAJOR = 1;
  private static final int MINOR = 0;

  private static final int width = 440;
  private static final int height = 440;

  private BundleContext bc = null;
  private SA sa = null;
  private HashMap ht = null;

  private ICompositionManager cm = null;

  private Vector types_vector = null;
  private Vector services_discovered = null;

  private JComboBox cb = null;

  private JScrollPane  table_panel = null;
  private JPanel services_panel = null;
  private JPanel combo_panel = null;
  private JPanel buttons_panel = null;
  private JTable services_table = null;

  private DefaultTableModel table_model = null;

  private Vector columnNames = null;

  private static final int intNumBtn = 3;
  private static String[] strBtn = new String[intNumBtn];

  // ------------------------------------------------
  //  CONSTRUCTORS
  // ------------------------------------------------

  public SAGUI(BundleContext bc, SA sa)
    {
      super("pSHIELD - Security Agent MOD v" + MAJOR + "." + MINOR);

      strBtn[0] = "Types Discovery";
      strBtn[1] = "Services Discovery";
      strBtn[2] = "Hide Me";

      columnNames = new Vector();

      columnNames.add("Icon");
      columnNames.add("Name");
      columnNames.add("Run");

      this.bc = bc;
      this.sa = sa;
```

```
      addWindowListener(windowExit);

      createGUI();

      setSize(width,height);
        setForeground(Color.black);
        setBackground(Color.lightGray);

        /*
         * ENABLE / DISABLE THE GUI...
         */

    // this.pack();
    setVisible(true);
    }

  //  ------------------------------------------------
  //  EVENT HANDLER
  // ------------------------------------------------

    WindowAdapter windowExit = new WindowAdapter()
  {
      public void windowClosing(WindowEvent e)
      {
        actionExit();
        }
    };

    public void actionExit()
    {
        // Exiting...
      if (bc != null)
      {
        try
        {
          bc.getBundle().stop();
        }
        catch (BundleException BE)
        {
          BE.printStackTrace();
        }
      }
  }

    public void actionPerformed(ActionEvent e)
    {
      if (e.getActionCommand().equals(strBtn[0]))
    {
        try{
          int i = 0;
              types_vector = sa.serviceTypeDiscovery();
              cb.removeAllItems();
              if ( sa != null ){

            Iterator it = types_vector.iterator();

            while ( it.hasNext() ){
              cb.insertItemAt( (String) it.next(), i );
              i++;
            }
              }
        }catch (Exception ex){
          ex.printStackTrace();
        }

        cb.setEditable(false);
        cb.setEnabled(true);

    }
    else if (e.getActionCommand().equals(strBtn[1]))
    {
      String type_selected = (String) cb.getSelectedItem();
      if ( type_selected != null ) {

        try {
          services_discovered = sa.serviceDiscovery( type_selected );
        } catch (Exception e1) {
```

```
        // TODO Auto-generated catch block
        e1.printStackTrace();
      }

    }
    else JOptionPane.showMessageDialog( null , "Select a Service Type" );

    List l = sa.getServices(11);
    ht = sa.getParameters(l);

    // Initialize the services table with the services discovered
    initializeServicesTable( services_discovered );

  }
  else if (e.getActionCommand().equals(strBtn[2]))
  {
    setVisible(false);
  }
  else
  {
    //Unknown Command
  }
  }

  /**
   * Function that initialize and the services table with a data vector
   * @param data the data vector ( if it is null, the table will be empty )
   */
  private void initializeServicesTable( Vector data ){

    table_model = new DefaultTableModel();
  table_model.addColumn("Name");
  table_model.addColumn("Description");
  table_model.addColumn("");

  if ( data != null && !data.isEmpty() ){

    Iterator it = data.iterator();

      while ( it.hasNext() ){

        Hashtable ht = ( Hashtable ) it.next();

        if ( ht.containsKey( "Service Name" ) && ht.containsKey( "Service Description" ) ) {

        String serv_name = (String)((Vector)ht.get("Service Name")).elementAt(0);
        String description = (String)((Vector)ht.get("Service Description")).elementAt(0);

        if ( serv_name != null && description != null )  {
          table_model.addRow( new Object[] { serv_name, description, "Run"} );
        }

        }
      }
  }

  services_table = new JTable ( table_model ){
          //  Returning the Class of each column will allow different
          //  renderers to be used based on Class
          public Class getColumnClass(int column)
          {
            if ( getValueAt(0, column) == null )
              return null;
            else
              return getValueAt(0, column).getClass();
          }
          public boolean isCellEditable(int row, int col) {
              if ( col == 2 )
                return true;
              else
                return false;
          }

          public String getToolTipText(MouseEvent e) {
              String tip = null;
              java.awt.Point p = e.getPoint();
              int rowIndex = rowAtPoint(p);
              int colIndex = columnAtPoint(p);
```

```
                int realColumnIndex = convertColumnIndexToModel(colIndex);

                if ( services_discovered != null ){

                  Hashtable tmp = (Hashtable) services_discovered.elementAt( rowIndex );
                  String description = ( String ) ( ( Vector ) tmp.get( "Service Description" ) )
.elementAt(0);

                  if ( realColumnIndex < 2 ) {
                    tip = description;
                  }
                }

                return tip;

            }

        };

        //  Create the button column
        ButtonColumn buttonColumn = new ButtonColumn(services_table, 2);

        services_table.setPreferredScrollableViewportSize(new Dimension(500, 70));
    services_table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    services_table.setColumnSelectionAllowed(false);
    services_table.setRowSelectionAllowed(true);

    table_panel = new JScrollPane( services_table );

    services_panel.removeAll();

    services_panel.add( combo_panel, BorderLayout.PAGE_START );
    services_panel.add( table_panel, BorderLayout.CENTER );
      services_panel.add( buttons_panel, BorderLayout.PAGE_END );

    services_panel.validate();

    }

    // ------------------------------------------------
  //  DESIGN GRAPHICS
  // ------------------------------------------------

    private void createGUI()
    {

      // Initialize the main Panel
    services_panel = new JPanel ( );
    services_panel.setLayout( new BorderLayout() );
    services_panel.setBorder( BorderFactory.createCompoundBorder(BorderFactory.createRaisedBevelB
order(), BorderFactory.createLoweredBevelBorder()) );

    // Initialize the Combo Panel
    combo_panel = new JPanel();

    GridBagLayout gridbag = new GridBagLayout();
        GridBagConstraints c = new GridBagConstraints();
        c.fill = GridBagConstraints.BOTH;

    combo_panel.setLayout( gridbag );

    c.gridwidth = 1;    // The cell occupies 1 column
        c.gridheight = 1;  // The cell occupies 1 row
      c.gridx = 0;     // The cell is located in 1 column
        c.gridy = 0;     // The cell is located in 1 row
        c.weightx = 0.0;  // The cell occupies the minimum row length
    c.weighty = 0.1;  // The cell occupies the entire column length

      JLabel lab = new JLabel("Choose a Service Type :");
        gridbag.setConstraints( lab, c);
        combo_panel.add(lab);

        types_vector = new Vector();

        cb = new JComboBox( types_vector );

    cb.setEditable(false);
    cb.setEnabled(true);
```

```
    c.gridwidth = 3;   // The cell occupies 1 column
        c.gridheight = 1;  // The cell occupies 1 row
      c.gridx = 1;    // The cell is located in 1 column
        c.gridy = 0;    // The cell is located in 1 row
        c.weightx = 0.1;  // The cell occupies the entire row length
    c.weighty = 0.1;  // The cell occupies the entire column length

    gridbag.setConstraints( cb, c);
        combo_panel.add(cb);

    // Initialize the Buttons Panel
    buttons_panel = new JPanel();

    for (int i = 0; i < intNumBtn; i++)
      {
        /*
           * ADD BUTTONS TO START SEVERAL TEST
           */
          Button b = new Button(strBtn[i]);
            //gridbag.setConstraints(b, c);
            b.addActionListener(this);
            buttons_panel.add(b);
      }

    // Create an empty table
    initializeServicesTable( null );

    // Add the subpanels to the main panel
    services_panel.add( combo_panel, BorderLayout.PAGE_START );

    services_panel.add( table_panel, BorderLayout.CENTER );

    services_panel.add( buttons_panel, BorderLayout.PAGE_END );

    // Add the main panel to the Frame
    add( services_panel );

      /*
         * CENTER FRAME ON THE SCREEN
         */
        Dimension dialogSize = getSize();
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    setLocation(screenSize.width/2 - dialogSize.width/2, screenSize.height/2 -
 dialogSize.height/2);

        /*
         * pSHIELD ICON
         */
        try
      {
          /*
           * Check if we are in a JAR file...
           */
          URL url = this.getClass().getResource("logo_pSHIELD_16x16.jpg");
          if (url != null)
          {
            this.setIconImage(Toolkit.getDefaultToolkit().createImage(url));
          }
          else
          {
            /*
               * We are not in a JAR file...
               */
          }
      }
        catch(Exception e)
        {
          e.printStackTrace();
        }
    }

    class ButtonColumn extends AbstractCellEditor implements TableCellRenderer, TableCellEditor,
ActionListener {
      JTable table;
      JButton renderButton;
      JButton editButton;
      String text;
```

```
    public ButtonColumn(JTable table, int column) {
        super();
        this.table = table;
        renderButton = new JButton();

        editButton = new JButton();
        editButton.setFocusPainted( false );
        editButton.addActionListener( this );

        TableColumnModel columnModel = table.getColumnModel();
        columnModel.getColumn(column).setCellRenderer( this );
        columnModel.getColumn(column).setCellEditor( this );
    }

    public Component getTableCellRendererComponent(
        JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column)
    {
        if (hasFocus)
        {
            renderButton.setForeground(table.getForeground());
            renderButton.setBackground(UIManager.getColor("Button.background"));
        }
        else if (isSelected)
        {
            renderButton.setForeground(table.getSelectionForeground());
            renderButton.setBackground(table.getSelectionBackground());
        }
        else
        {
            renderButton.setForeground(table.getForeground());
            renderButton.setBackground(UIManager.getColor("Button.background"));
        }

        renderButton.setText( (value == null) ? "" : value.toString() );
        return renderButton;
    }

    public Component getTableCellEditorComponent(
        JTable table, Object value, boolean isSelected, int row, int column)
    {
        text = (value == null) ? "" : value.toString();
        editButton.setText( text );
        return editButton;
    }

    public Object getCellEditorValue()
    {
        return text;
    }

    public void actionPerformed(ActionEvent e)
    {
        fireEditingStopped();

        try
        {
      ServiceReference[] sr = findCompositionManagerImplementations();
      Object o = bc.getService(sr[0]);
      cm = (ICompositionManager) o;

      //Devo avviare i servizi restituiti dall'hashtable
      if ( services_discovered != null) {
        cm.runBundle( ( Hashtable ) services_discovered.elementAt( table.getSelectedRow() ),11,
 ht );
        }
        }
      catch (Exception er)
      {
      er.printStackTrace();
    }

    }
}


  /**
   * @author Davide Migliacci
```

```
   *
   * This function uses the internal OSGi service discovery to find a suitable implementation
   * of the IComunicationManager interface.
   */
  private ServiceReference[] findCompositionManagerImplementations() throws Exception
  {
    ServiceReference[] cmi = null;
    try
    {
      cmi = bc.getServiceReferences(ICompositionManager.class.getName(), null);
    }
    catch (Exception e)
    {
      throw e;
    }
    return cmi;
  }

    public void appendts(String str)
    {
      append("[" + new String((new Date((new Long(System.currentTimeMillis())).longValue())).toSt
ring()) + "]\n" + str + "\n");
    }

    public void append(String str)
    {
      System.out.println( str );
    }

}
```

package eu.artemis.shield.overlay.semanticknowledge.impl.SemanticKnowledge.html

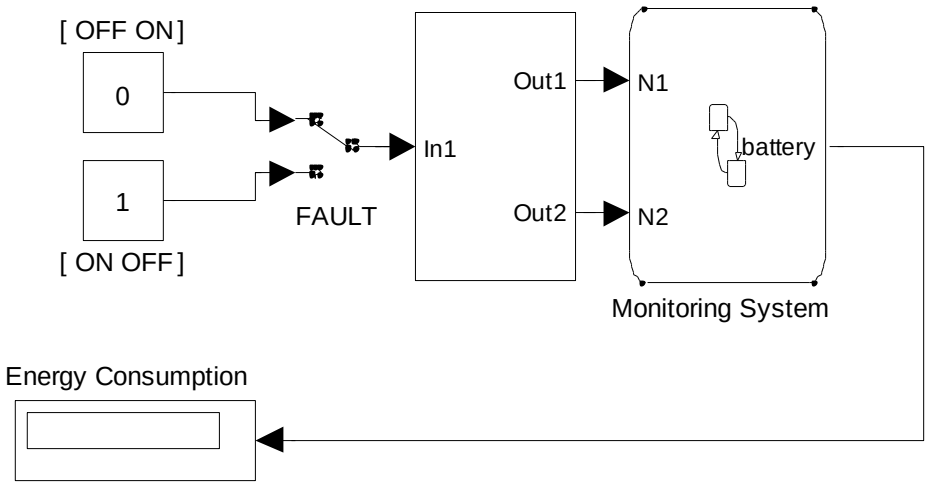
```
package eu.artemis.shield.overlay.semanticknowledge.impl;

import it.trs.rd.pshield.data.response.CompositionData;
import it.trs.rd.pshield.data.response.CompositionResponse;
import it.trs.rd.pshield.engine.PShieldApplicationContex;
import it.trs.rd.pshield.engine.PShieldEngineException;
import it.trs.rd.pshield.engine.PShieldService;

import java.util.LinkedList;
import java.util.List;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;

import eu.artemis.shield.overlay.semanticknowledge.ISemanticKnowledge;

public class SemanticKnowledge implements ISemanticKnowledge {

  public SemanticKnowledge(){

  }

  public List semanticComposition(List list)
  {

    System.out.println("1");

    PShieldApplicationContex ctx;
    try
    {
      System.out.println(System.getProperty("user.dir"));
      ctx = PShieldApplicationContex.getContex();
      System.out.println("2");

      PShieldService service = ctx.getPShieldService();

      System.out.println("3");

      for (int i = 0; i < list.size(); i++)
      {
        try {
          service.addElement(readFileAsString((String)list.get(i)));
          System.out.println(i);
        } catch (Exception e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
        }
      }

      System.out.println("4");

      /*HashMap services = new HashMap();

      for (int i = 1; i<10; i++ ){
        String alfa = url("resources/data/data_"+i+"_Pilota.owl");
        String beta = spdLevel("resources/data/data_"+i+"_Pilota.owl");
        services.put(alfa, beta);
      }*/

      CompositionResponse response = null;
      try {
        response = service.getComposition();
      } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }

      for (CompositionData composition : response.getCompositions()) {

           System.out.println("------");

          for (String elem : composition.getFuncionalities()) {

              System.out.println(elem);
```

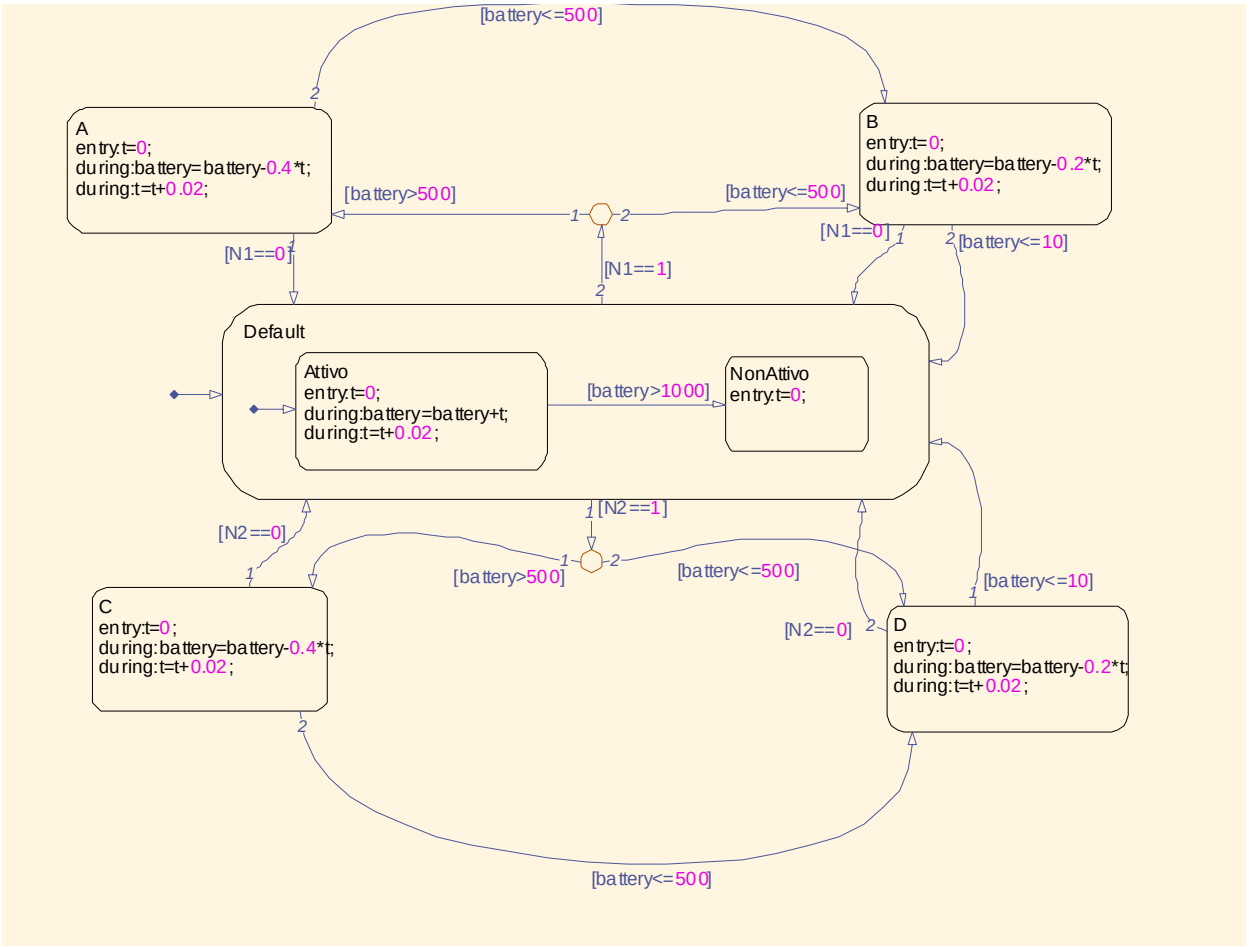
```
        }

      }

      return response.getCompositions();
    }
    catch (PShieldEngineException e1)
    {
      // TODO Auto-generated catch block
      e1.printStackTrace();
    }
    return null;
  }

    private static String readFileAsString(String filePath) throws java.io.IOException {
  StringBuffer fileData = new StringBuffer(1000);
  BufferedReader reader = new BufferedReader(new FileReader(filePath));
  char[] buf = new char[1024];
  int numRead = 0;
  while ((numRead = reader.read(buf)) != -1) {
      String readData = String.valueOf(buf, 0, numRead);
      fileData.append(readData);
      buf = new char[1024];
  }
  reader.close();
  return fileData.toString();
    }

    /**
     * This function return the spdLevel associated to that owl file
     * @param path
     * @return spdLevel
     * @throws IOException
     */
   /* private static String spdLevel(String path) throws IOException{
      String file = "";
      BufferedReader in = new BufferedReader(new FileReader(path));
      String control = "        <SPDStatus rdf:datatype=\"&xsd;int\">";
      while((file = in.readLine())!=null){
        if (file.startsWith(control)){
          file = file.substring(control.length(), control.length()+2);
          String a = file.substring(1);
          if (a.equals("<")){
            file = file.substring(0,1);
          }
          System.out.println("Found" + file);
          return file;
        }
        else {
        System.out.println("Not Found");
        }
    }

      return file;
    }*/

    /**
     * This function returns the url associated to that owl file
     * @param path
     * @return url
     * @throws IOException
     */
  /*  private static String url(String path) throws IOException{
      String file = "";
      BufferedReader in = new BufferedReader(new FileReader(path));
      String control = "<rdf:RDF xmlns=\"";
      while((file = in.readLine())!=null){
        if (file.startsWith(control)){
          file = file.substring(control.length(), file.length()-1);
          System.out.println("Found" + file);
          return file;
        }
        else {
        System.out.println("Not Found");
        }
    }

      return file;
```

```
    }*/
}
```

Annex 3 – Overlay control algorithms – Matlab Source Code prototype

Prototype A





Prototype B

sensormodel_full.hys

```
SYSTEM sensor {


INTERFACE {
             STATE { REAL B [0,100];
            REAL P [0,100];
            REAL BT [0,100];
            BOOL S;
                }
             INPUT { REAL rIN [0,120];
                }
             OUTPUT {REAL y1,y2,y3;
            BOOL y4;
                 }
             PARAMETER {
                REAL Ts, rOUTmax, bMAX, btLOW,pMAX, rINmax;
             }
}
IMPLEMENTATION {
                AUX { REAL rOUT, rOUT2, rOUT3, R, R2;
                      BOOL max, maxB, LOWbt, IDLE, FINbt, EMPb, maxP;
                }
                AD  { max = rIN>=rOUTmax ;
              maxB = B>=bMAX;
              LOWbt = BT<=btLOW;
              IDLE = P<=0;
              FINbt = BT<=1;
              EMPb = B>=0;
              maxP = P>=pMAX;


            }
```

```
        AUTOMATA {S = FINbt;
        }

                DA  { rOUT = {IF (max | maxB) & !LOWbt THEN rOUTmax ELSE rOUT2};
              rOUT2 ={IF S | IDLE THEN 0 ELSE rOUT3};
              rOUT3 ={IF EMPb THEN B ELSE rIN};
              R = {IF (maxB | maxP) & !LOWbt THEN 2 ELSE R2};
              R2 = {IF IDLE | S THEN 0 ELSE 1 };
            }

                CONTINUOUS {
                     B = B+Ts*(rIN-rOUT);
                     P = 100*(Ts)*(rIN+rOUT)/(rINmax+rOUTmax);
                     BT = BT-Ts*R;
                }

                OUTPUT {y1=B;
                y2=P;
                y3=BT;
                y4=S;
                }
            }
}
```

MPC_optimization.m

```
clear all; close all; clc;

%load sensor.hys
Ts=0.5;
bMAX=80;
rOUTmax=100;
rINmax=120;
btLOW=10;
pMAX=90;

clc

% Generate the MLD model
S=mld('sensormodel_full',Ts);

% Generate the equivalent PWA model
P=pwa(S);


 % Design MPC controller
clear Q refs limits

refs.x=1;    % only state x(2) is weighted
Q.x=1;       % weight on state x(2)
Q.rho=Inf;   % hard constraints
%Q.norm=2;    % use quadratic costs
Q.norm=Inf; % use infinity norm
N=2;

limits.xmin=[0;0;0;0];

C=hybcon(S,Q,N,limits,refs);
C.mipsolver='glpk';    % used for MILP
%C.mipsolver='cplex';  % used for MIQP
Tstop=100;
x0=[0;0;40;0]; % initial state
clear r
TS=[0:Ts:99.5]';
U=zeros(length(TS),1)
for i=1:length(TS)

    U(i)=85*(TS(i)>=0)*(TS(i)<=20);
end
r.x=U;


% Simulate Hybrid MPC loop using from command line
[XX,UU,DD,ZZ,TT]=sim(C,S,r,x0,Tstop);
figure
subplot(311);
plot(TT,2*XX(:,2),TT,UU);
axis([0 30 0 130]);
grid
title('CPU (%), Rin(MB/s)')
subplot(312);
plot(TT,XX(:,1),TT,r.x);
axis([0 30 0 100]);
grid
```

```
title('Buffer (MB)')
subplot(313);
plot(TT,XX(:,3));
axis([0 30 0 50]);
grid
title('Battery (%)')
set(gcf,'position',[360    55   385   623]);
```