Contract No. IST 2005-034891

Hydra

Networked Embedded System middleware for
Heterogeneous physical devices in a distributed architecture

Validation Report for SDK Prototype

Integrated Project
SO 2.5.3 Embedded systems

Project start date: 1st July 2006 **Duration: 52 months**

Published by the Hydra Consortium **14.11.2008**

Project co-funded by the European Commission
within the Sixth Framework Programme (2002-2006)

Dissemination Level: PUBLIC

Document file: Hydra - D10.2 Validation Report for SDK Prototype

Work package: WP 10 – Validation & business modelling

Task: T10.1 – User validation

Document owner: INN

Document history:

Version	Author(s)	Date	Changes made
0.1	A. Guarise, M. De Bona (INN)	04-07-2008	Document organisation, objectives and first structure
0.2	P. Antolin (TID); M. Ingstrup (UAAR); J. Schütte (SIT); M. Ahlsén, P. Rosengren, P. Kool (C-Net)	29-09-2008	Input from TID, UAAR, SIT and C-Net
0.3	A. Guarise, V. Grosso (INN); J. Schütte (SIT)	05-10-2008	Section 1, 2 and 4 input and revision of content
0.4	P. Antolín, F. Milagro (TID); M. Ingstrup (UAAR); M. Ahlsén, P. Rosengren (C-Net)	10-10-2008	Section 3 and 4 input (WP4, 5, 6)
0.5	M. Ingstrup (UAAR); André Brinkmann (UPB)	15-10-2008	Section 3 and 4 input (WP4, 5)
0.6	A. Zimmermann (FIT); M. Ahlsén (C-Net)	16-10-2008	Section 3 and 4 input (WP3, 6)
0.7	A. Guarise, V. Grosso (INN); J. Schütte (SIT)	28-10-2008	Section 3, 4 and 5 (conclusions) finalisation
1.0	A. Guarise, V. Grosso, M. De Bona (INN)	28-10-2008	Document first edition, ready for Peer Review. Submitted to Peer Reviewers.
1.1	A. Guarise, V. Grosso, M. De Bona (INN)	14-11-2008	Document revised after Peer Review comments. Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Comments
T. Sabol, M. Mach (TUK)	03-11-2008	Accepted. See report file (inserted in the BSCW repository with the final document)
S. Zickau, A. Adetoye (UR)	14-11-2008	Accepted. See report file (inserted in the BSCW repository with the final document)

Index

List of figures

List of tables

1. Introduction

1.1 Purpose and context

This deliverable provides the results of the first iteration validation phase focussing on the SDK prototype. The objective is to show the major outcomes found after applying the validation concepts described in the Deliverable 10.1 "Validation Plan for prototypes" in order to understand better the middleware and SDK prototype and be able to feed the lessons learnt back into the next development iteration. The validation tests have been fulfilled during the implementation phase and in a specific testing activity and it involved both the first Hydra prototypes (SDK) but also the middleware which is under continuous development. The critical outcomes and those not planned or foreseen to occur during the software code writing are also highlighted.

It is important to underline at this stage that the level of implementation gathered for the SDK is not reaching the release phase, as the iterative approach followed in Hydra consists in successive improvements of the software package, so to say that the validation fulfilled during this first loop of the validation shall involve the software components so far developed and the group of requirements that have been considered as a reference and guiding specification for the actual implementation.

1.2 Outline

The validation report represents the first document of a series of three different assessment studies, one per prototype and iteration, organised and structured as explained in the validation plan (D10.1), which is considered as an input document.

Section 2 recalls the objects of the validation, the targeted users and the reasoning for explaining the requirements selection. As Hydra foresees to meet more than 450 requirements, it is necessary to limit the number with a careful selection of the most important ones, given the fact that not all of them have been already implemented at this point of the project and that a large part of the technical (functional) specifications are considered to be met at debug level.

Section 3 briefly describes the assessment methodology applied to each requirement and summarises them into groups divided per work package. The tables in this section (selection of requirements) are revised in respect to those that were indicated in the Deliverable 10.1, because a short list has been made considering the most important ones among those so far implemented.

Section 4 reports the results obtained while applying the assessment procedures for the evaluation of the fit criteria fulfilment. Each WP leader and participants decided together how to give proof of the requirement verification (fit criterion) and the threshold level below which the requirement is not met. In the worst case (requirement not reaching the threshold) the requirement is marked to be re-evaluated at the next validation iteration, so that all requirements are submitted to a continuous improvement.

Section 5 draws the major conclusions of the report. It indicates the open issues and expected progress in the assessment procedure and how the validation process shall improve the implementation part.

2. Object of the validation

In the first validation cycle the object of the evaluation is the SDK prototype. The foreseen planning from the validation plan is sketched in Figure 1, but due to the application development which run longer than expected and the need to refine the requirements to be assessed (not all requirements are implemented at a time, but in a recursive approach), the first validation has been shifted by two months, even if the first action, the preparation phase, started at due date, while second and third steps were prolonged as requested.

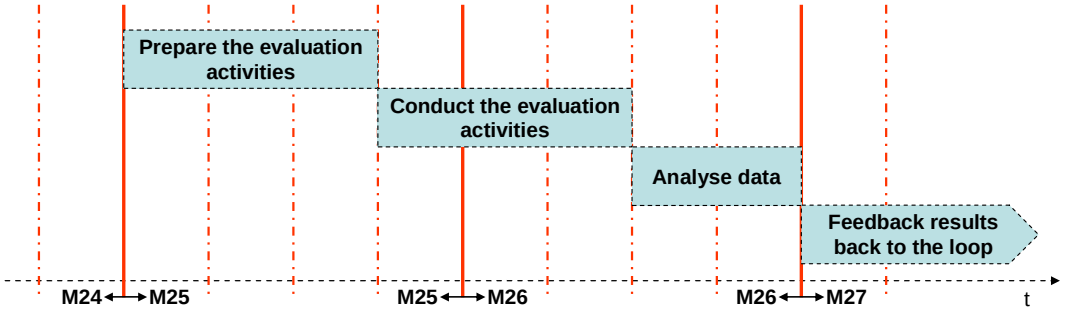


Figure 1: User validation first iteration - time plan

The first iteration considers both the middleware, building block for all prototypes, and the Software Development Kit.

As the iterative approach foresees, the SDK and the middleware software packages analysed in the first validation cycle are intermediate results constituted from those components implemented during the first and second project iteration. During the first project year, when the basic middleware was built, the software debugging was more suitable than a validation process, which shall be applied when the software code is more mature and ready to be tested. So the first validation cycle is assessing components that are not considered as the final ones, but partial release of a subsequent delivery of improved prototypes. Moreover also the requirements selection, even if based on the initial Deliverable 10.1 list, has been updated with the intention first of all to fine tune the group of requirements towards a more reasonable number, and secondly to consider those already implemented so to make possible the testing. The tables in the next section eventually consider new important specifications introduced after the completion of the Validation Plan.

2.1 Target users

Hydra identified along the previous deliverables two main groups of users:

- developers that will use the middleware, considered as the major focus for the validation report due to their direct involvement in the SW development process, which is the aim and the reason why Hydra middleware has been conceived;
- end-users that will benefit from the Hydra enabled services created by the previous group, the developers, and also considered as a major source of feedback due to their role in the value chain and their fundamental part in the product successful commercialisation.

Therefore we differentiate the terms developer-user from end-users, as the same difference existing from those who create a product (developers, first group) from the real users of the good itself. The validation plan divided the task activity into three different moments, related to each project iteration conclusion and depicted in the next table.

Type of user	Object of the evaluation	Start of the user validation (month)
Developer user	SDK + middleware vers. 1	M24
Developer user	DDK + middleware vers. 2	M36
Developer user	IDE + middleware vers. 3	M48
End user	Applications	M48

Table 1: Validation plan milestones

As the actual object of the assessment are the middleware and the SDK, during this validation cycle the target users are the application developers, from the first group indicated above.

The developer users are identified among Hydra internal resources where possible. This is done mainly because it is difficult to find the commitment from companies not directly involved in the Hydra consortium, especially from an economical point of view (external experts who are not Hydra partners asking for a fee shall be paid with mean of subcontracting). This is also a challenge because we must consider that evaluation with developer-users may or may not lead to new issues if compared to traditional user validation. For diminishing this risk the selected developers are preferably chosen among those who were not directly involved in the Hydra implementation, otherwise their judgement would be biased.

2.2 Quality dimensions and assessment criteria

The validation is made through the comparison between an expected impact (requirement) and how the real application works. In Hydra the expected impact is described with mean of the user requirements, derived and collected in WP2 and WP3. The user requirements consist of a list of features and properties of the Hydra middleware including quality criteria, which are considered relevant by the users. Deliverable 3.2 "Updated system requirements report" contains an updated overview of the requirements that shall be necessary to the Hydra developed system as emerged in several focus groups with developer users.

Every requirement statement is composed of six fields to briefly describe it, as shown in the next example.

ID:	31
Type:	Non-functional / look and feel
Priority:	Critical
(Short) description:	An easy-to-use programming framework should be provided
Rationale:	The programming framework provided by the prototype should be easy to use in the sense that it is intuitive
Fit Criteria:	9 out of 10 developers recognise the IDE as intuitive

As quality is a relative or personal issue to be measured, a value must be attached to the cost and benefit of quality-oriented actions. Features and properties requested by stakeholders have to determine on how to implement and what the optimal investment is.

There are different frameworks analysing quality attributes, with differing vocabulary, metrics etc. that are relevant to software architecture design. Quality attributes are essential to the design of software architecture, but it is a challenge to describe quality attribute (requirements) on a common form. For this reason, together with the Volere schema for drafting user requirements, the SEI quality framework (Bass et al., 2003) and the ISO 9126 (2001) international standard have been studied. The SEI quality framework, also known as Quality Attribute Scenarios, is a well-established way of defining architectural requirements in a uniform way and introduces the concept of

considering quality attribute requirements on a fixed and precise scenario form. This approach has been integrated in the context of the Hydra project with the ISO 9126 international standard defining a comprehensive quality model for software products. Deliverable 6.1 "Quality Attribute Scenarios" gives a detailed and clear overview of the two frameworks.

The first validation report conceives a general scheme in the validation framework on how to measure the fit criteria pertaining to each different requirement, which is relevant in respect to different possible quality dimensions: performance, robustness, durability, security, safety and privacy, but also subjective assessment quality, learning effort, cognitive workload and added value.

The assessment procedures identified in this report, summarised in the next chapter tables and then applied in Section 4, will be used where possible in the next validation cycles, so to simplify the evaluation effort and for improving also the single requirement evaluation, in case some of them were not satisfying the threshold condition.

2.3 Requirements for the first iteration

Developer-users are interested in requirements fulfilment, the technical aspects related to the software instrument they want to use: a middleware, SDK, or another prototype. For this Deliverable the validation is applied through requirements technical tests and assessment fulfilled at the end of the SDK cycle implementation.

The first group of requirements was identified in Deliverable 10.1, as the total number of specifications had reached a large quantity. In the Validation Plan all major functional and not functional requirements were chosen, but the overall tables have been revised in this report. As a major observation the largest part of functional requirements were considered to be verified during the debugging phase, otherwise the middleware component would not work, so just the most important among them were taken into account for the validation process. The specifications have been confirmed depending on their implementation status at the time of the validation, and eventually substituted with those that have been already considered at this stage of the project.

The final selection of requirements was performed by each work package leader in agreement with WP participants. Starting from the initial group, each WP first confirmed the possibility to assess or not each requirement and then identified the major ones to which apply the testing procedure, eventually integrating or substituting the initial list in case new requirements were added, old important ones had been left out or the previous selected group was not adequate or sufficient. The need to have a short list of final requirements was due to the large number of entries so far identified during the project course (more than 450) as the validation shall be completed into a defined time frame for allowing the provision of the results back into the loop.

The requirements refinement is strongly related to two factors: the software development process, which requires different needs for different components, and the iterative approach, which adds latest requirements at every implementation update.

The final list of the requirements selected for the fist iteration is presented in the next section, through tables divided depending on the pertaining WP.

3. Description of the validation methods

Once the validation testing procedures are depicted, the tester has to follow the indications given to perform the validation, which can be a laboratory test or a trial of the middleware/SDK. Different expert evaluators do not find the same defects, and not in the same order. It is therefore advisable to use at least two or three experts (even more if available).

The developer user can be assisted by colleagues actively involved in the Hydra project in case something is not clear or misleading. The conduction of the validation by the software developer should be linear if the planning is done carefully and the validation procedures are prepared with sufficient details.

Experience shows that the more immature an implementation is, the faster defects will be found. Users who are confronted with incomplete and faulty software become frustrated and can not provide much constructive feedback. So it is preferable to proceed with the first middleware evaluation at an advanced stage, when the implementation of software has already reached certain robustness. As the prototypes are recursively improved, the middleware assessment is repeated in all iterations. The collected feedback allows having a constant improvement of the implemented system.

First there will be a collection of data as a result of laboratory test by considering each requirement referring to the middleware. This will be the case for those quality dimensions that need a specific measurement (for example, an efficiency performance test). On the other hand requirements that need a special evaluation, not feasible with a simple measurement, will be assessed through a complete description of the reasoning developer users.

The SDK assessment is performed in the same way as it is done for the middleware, but differentiating the domain applicability. The assessment used laboratory measurements, software procedures and an assessment analysis completed by the developer users who exploited the Hydra components.

Assessment procedure for verifying the fit criteria fulfilment

The assessment procedures for the requirement evaluation were deployed by the WP leader in agreement with other WP partners. The testing has been decided in order to assure that the methodology is able to verify that the fit condition is met with limited uncertainties. In case of functional requirements usually this is proved by means of a (numerical) threshold level; in case of a non functional requirement where there is no clear indication of the expected result, the assessment procedure contains the background methodology and the proper conditions able to demonstrate the criterion verification.

As an example, requirement n. 31 mentioned above has already a fit criterion identifying the numerical indication for which the requirement is considered as met.

ID:	31
Type:	Non-functional / look and feel
Priority:	Critical
(Short) description:	An easy-to-use programming framework should be provided
Rationale:	The programming framework provided by the prototype should be easy to use in the sense that it is intuitive
Fit Criteria:	**9 out of 10** developers recognise the IDE as intuitive

3.1 WP3 – Evaluated requirements

ID	Description	Rationale	Fit Criteria	Middle ware	SDK	Assessment procedure	Outcome
18	Support for different software architectural patterns	The Hydra architecture should not prescribe one way to structure applications. Thus several architectural patterns, for example MVC and PAC should be supported.	Hydra allows at least two different architectural patterns for applications.	x	x	Implementation of applications based on the middleware, which show different architectural patterns. Analyse the current architecture design of the middleware.	Supported
31	An easy-to-use programming framework should be provided	The programming framework provided by the SDK should be easy to use in the sense that it is intuitive.	9 out of 10 developers recognise the SDK as intuitive.		x	Conduction of a software-walkthrough and a validation session with developers specifically addressing the ease of use.	Not yet supported
33	Enable manufacturers to develop devices and applications that can be connected to Hydra	The Hydra SDK should provide the manufacturers with an API to develop devices that can be connected to the Hydra network.	APIs are available to develop devices that can be connected to the Hydra network		x	Develop applications based on the Hydra middleware and connect multiple third party devices to a Hydra network.	Supported
41	Hydra Developer's Companion	Complete and comprehensible documentation is very important to the Hydra software developer.	Complete documentation is available. It is at least considered "very helpful" by at least 8 out of 10 developers.		x	Conduction of a technical review of the documentation. Run a software walkthrough as a preparation for the training activities.	Partly supported
136	Dynamic architecture	An architecture of a running Hydra system can be easily modified by increasing or decreasing the degree of centralisation in order to balance utilisation of available resources.	In 95% of all cases, Hydra supports dynamic migration of components to realise centralised and decentralised systems.	x		Implement and run a test application and test whether it is able to be reconfigured or not.	Not yet supported
185	Middleware provides basic services	In order to program AmI applications, the middleware must provide basic services. This makes life easier for application developers. Basic services provide e.g. methods to query available devices and services or to pass messages between components	Middleware provides a set of basic services that at least contain basic functionality that is needed by all services, like communication and a service / device registry.	X		Conduct a technical review of the core Hydra services with developers. This review aims at the setup of a basic Hydra infrastructure, querying available devices and passing messages between devices.	Partly supported
186	GUI for configuring middleware parameters	To make the configuration of the parameters of the middleware easier for the developer	A GUI exists for configuring the middleware	x	x	The fit criteria of this requirement needs to be revised, since GUIs for the configuration of parts of the middleware already exist, but their contribution to the facilitation of the development needs to be assessed.	Supported

ID	Description	Rationale	Fit Criteria	Middle ware	SDK	Assessment procedure	Outcome
199	Modules should be extendable	Hydra modules should be extendable in their functionality by 3rd-party solutions	80% of all Hydra modules are extendable in their functionality by integrating 3rd-party code via a standard interface or replaceable by 3rd-party modules with equivalent functionality.	x	x	An assessment procedure that measures the extensibility of software is part of current research. One approach could be to count the number of hooks that allow for the modification of existing modules or the addition of new ones. Another approach could be to let a number of developers implement extensions to the Hydra middleware and to assess the result. Thus, a formal assessment procedure remains an open issue.	Partly supported
207	Service selection by context	In order to select an appropriate service for a specific task, contextual information, like the spatial position, must be taken into account. Hydra must provide a method to specify a desired service by contextual parameters. For example, if a certain room in a building is specified in a search request for a service, only services are returned that are relevant in the current user's location and context.	In search requests for a specific service, contextual information like a spatial position is allowed.	x		Build a prototype which combines location and other context constraint to select an appropriate service. An example scenario would be: A user wishes to print a coloured document to the nearest printer during a presentation.	Partly supported
217	The middleware should ensure high robustness of services	In order to ensure the service support of important components in the system, the middleware should provide a highly robust service structure.	Breakdown of crucial services of the middleware in less than 1 case per 100 hours of operation.	x		Identify the crucial services of the Hydra middleware, build a test application that bases on that set of services and conduct a long term operation stress test.	Partly supported
234	The middleware should be self descriptive	The developer should be enabled to understand all components and their interplay of the system in order to take full advantage of the Hydra Middleware	Nine out of ten developer have a clear understanding of the Hydra middleware after one week of experience	x	x	Conduct a software peer review with developers.	Not yet supported
320	Separate domain-oriented services and user interface services architecturally	This is a standard architectural design tactic to enhance modifiability	90% of the modules of the architecture properly separate layers for domain services and interfaces.	x		Analyse the SVN repository which contains all Hydra managers and modules and identify those that mesh interface and control logic.	Not yet supported

ID	Description	Rationale	Fit Criteria	Middleware	SDK	Assessment procedure	Outcome
327	The Hydra middleware should be flexible as to allow for opt-in and opt-out on parts	Not all parts of Hydra will make sense in all situations (it will not always be beneficial to use higher layers of communication such as a service composition protocol or maybe a device may be too resource-constrained to use parts). One should be able to take the parts of the Hydra middleware that makes sense for a certain application. For example, it should be possible to for embedded devices with few resources (see other requirements) to take part in a Hydra application without having to install or run all Hydra components. Another example may be that one may want to use just point-to-point communication of Hydra without having to use the context-awareness part. (Werner Vogels, CTO/Amazon at JAOO 2006: "Middleware is evil!", referring to that if one chooses a certain middleware such as CORBA one makes too many decisions (not only on communication in the CORBA case but also, e.g., on transactions) that may not be appropriate for the case at hand)	Hydra is able to support the exact subset of services required by a client (user or service) in 70 % of all cases. In 20 % of all cases the middleware is able to provide a service package that includes the required service. In 10% of all cases Hydra is not able to provide service similar to the desired service.	x		Build several test applications that are based on a different set of managers of the Hydra middleware.	Supported
329	Middleware provides domain-independent services	A lot of the services needed in the apartment scenario are also needed in other scenarios (persistence, logging, visualization, ...). These should be abstracted and built and provided as part of Hydra	Large parts of the building-automation scenario can be built by reusing configurable services from across other application domains.	x		Build several test applications that are based on a similar set of managers provided by the Hydra middleware.	Supported
335	Location awareness / positioning support	Hydra should enable developers to write applications that depend on context, especially spatial context.	A component for acquiring spatial context exists. At any time, min. 75% of all devices attached to a Hydra system can be spatially located. Also, there is a programming model for using spatial context.	x		Build a location-aware application based on the Hydra middleware.	Partly supported

Table 2: WP3 selected requirements

3.2 WP4 – Evaluated requirements

ID	Description	Rationale	Fit Criteria	Middle ware	SDK	Assessment procedure	Outcome
312	Support profiling of devices' performance	The middleware should contain services that allow monitoring and reaction on what devices are doing. This includes monitoring response time, device load (e.g., CPU), and message interchanges per second	Said services available in Hydra	x		See § 4.2	Partly supported
314	Faults should be intercepted by middleware, notified to interested services	To create reliable and available systems it is essential to catch faults/partial failures before they become failures/complete failures. There needs to be uniformity in how this is done; thus it should be supported by the middleware.	The middleware has support (through components/services) for sending and receiving notifications for partial failures	x		Experiment with behaviour when services become available	Partly supported
317	Support runtime reconfiguration	To supporting monitoring leading to adaptation, the architecture should be dynamic in the sense that components/services should be connectable in new ways at runtime	Services and devices can be connected in new ways during runtime in Hydra-based applications	x	x	Test an example application's ability to be reconfigured according to specific scenarios	Not yet supported
318	Devices should be able to be added to the system at runtime	It should not be necessary, e.g., to shut a building complex down to add a new device to a room	Devices can be installed, discovered, and used while the Hydra runtime is running	x		See § 4.2	Supported
366	Web services should run on embedded devices	Service-orientation is a good match for many embedded devices. Web services will provide a gateway to many applications and it would be beneficial to be able to structure all of the communication in a system using the same primitives.	Hydra supports web services on embedded device (Initial target should be Develco's DevCom 02 ZigBee module)	x		See § 4.2	Not yet supported
479	Event prioritisation	The EventManager should handle events according to their priorities. Some events are critical to the health of the system and should be prioritized over others when there are a high number of events being routed through the system	Stress test of the event notification system. If the volume of events exceeds the capacity, events with high priority should be delivered first, and only be discarded as a last resort	x		See § 4.2	Supported

Table 3: WP4 selected requirements

3.3 WP5 – Evaluated requirements

ID	Description	Rationale	Fit Criteria	Middle ware	SDK	Assessment procedure	Outcome
264	Common message protocol	Devices communicate with a common message protocol. The protocol has to follow existing standards, be machine readable and interoperable.	100% of devices in the Hydra network use a common message protocol for service invocation and consumption	x		Use a network sniffer to assert that the message protocol used is the same in all cases being machine readable and interoperable (devices must understand each other)	Supported
276	New communication technologies	New communication technologies might be added to the system, so that Hydra should provide means to facilitate this inclusion	80% of new technologies are supported	x		Integration of the ZigBee protocol and discovery mechanism	Supported. Although percentage yet to be validated.
336	Discovery protocol should support multiple networks	There is a need for discovery of services across multiple physical networks (e.g., BT, Zigbee, RF...). Hydra should enable developers to create applications that have discoverable services on different types of networks	The service discovery protocol of Hydra able to work over multiple physical networks, finding at least 90% of the services available	x		Validation session with developers.	Supported.
407	Storage Manager – Gateways information stored synchronization	The information stored in the Gateway must be synchronized with the information inside the devices. The dumping of devices information could be either initiated by the device or controlled by the Gateway.	90% of the information stored in the Gateway is synchronized with the information stored inside the devices	x		Data will be annotated with timing information, which will be used to evaluate applied (soft) real-time constraints	Not yet supported
419	Device services and resources provision through its Gateway	Each device either Hydra-enabled or non-Hydra-enabled (through proxies) can offer services and resources in the overlay network using a common mechanism (SOAP) through its Gateway.	90% of devices can offer services and resources in the overlay network through its Gateway	x		Perform tests with a non-HED and two Hydra Enabled Devices and assert that the non-HED is discovered and its services can be consumed	Supported
425	D2D communication Overlay Hydra network	Using the D2D communication, any HED is able to participate in the Hydra network and communicate with other HEDs even if they are located behind firewalls or NATs	90% of the HEDs are able to communicate with each other even if they are located behind firewalls or NATs, and to participate in the Hydra network.	x		Perform tests with two Hydra Enabled Devices behind firewalls/NAT's and assert if they can communicate	Supported
445	The level of protection should be independent from the currently used low-layer protocol	The network layer for Hydra is defined as the SOAP layer. Since many protocols can use the SOAP layer, it is essential that the security mechanisms are independent from the protocol layer or can at least be applied on various layers. Thus, interoperability will be achieved.	In 95% of all cases the security model should be applicable to multiple protocol below SOAP layer	x		Evaluation of the current design of communication security	Supported

ID	Description	Rationale	Fit Criteria	Middle ware	SDK	Assessment procedure	Outcome
455	Identity - Update of the correspondences between identifier and physical addresses	To avoid incoherencies, when an entity changes its physical address in the network, the identifier management manager should be advised of or be able to detect that change in order to update mapping table between logical and physical identifiers.	In 100% of the times, when a physical address is changed, the identifiers management module is notified and updates its information.	x		Perform tests with two Hydra Enabled Devices. If one of it changes of IP, the HID table of the other has to be updated accordingly.	Supported
465	Networks overlapping	If two users of the Hydra system wear a personal Hydra Body Area Network (HBAN) and meet themselves in the same place, the HBAN of one user don't have to add the devices of the HBAN of the other user. The middleware must provides criteria to distinguish when a "new" device is authorized to be added to an existing Hydra network and when it belongs to another Hydra network which is temporary near to the previous one	Device are not to be added to an existing Hydra network if it is unauthorised when it belongs to another Hydra network which is temporary near to the previous one	x		Validation session with developers.	Not yet supported. Security not in place.
475	Multimedia streaming in the Hydra network	In order to integrate multimedia devices (UPnP AV, DLNA) in the Hydra middleware, a new communication mechanism for streaming of audio and video between Hydra enabled devices is needed.	80% of multimedia devices are able to exchange multimedia streams using the Hydra overlay network	x		Perform tests with an UPnP AV server and renderer running in different Hydra Enabled Devices and assert that the content is streamed between them	Supported
476	Network Manager Configuration and Testing	The Network Manager should provide a GUI for configuring the different parameters and testing tools	80% of the configuration parameters and functionalities are available for configuration and testing		x	Check that the application developer can use a visual tool to change the configuration parameters of the Network Manager	Supported

Table 4: WP5 selected requirements

3.4 WP6 – Evaluated requirements

ID	Description	Rationale	Fit Criteria	Middle ware	SDK	Assessment procedure	Outcome
91	Any Hydra device should have an associated description	For management, search and discovery purposes, all Hydra enabled devices should be described (classified) according to the Hydra device ontology.	Any device associated to a Hydra application is also included in the Hydra device ontology, and its description can be retrieved.	x	x	Check that a newly discovered device has/gets a corresponding representation in the Device Ontology.	Not yet supported.

ID	Description	Rationale	Fit Criteria	Middle ware	SDK	Assessment procedure	Outcome
101	Manual device ontology definition	The developer should be able to define and extend device ontologies. The IDE is required to provide descriptors for devices and device classes	The Hydra IDE supports the manual editing of devices in the framework of a device ontology.	x	x	Interface exists for entering and updating device descriptions in the device ontology.	Supported. Tool exists.
108	Device discovery	Middleware should be able to detect new device that enters the network	7 of 10 devices are discovered	x	x	Enter new devices into a Hydra network, locally and remotely.	Supported. Devices executable in application
110	Device Categorisation in runtime	Middleware should after discovery of device be able to categorise a device based on device ontology information.	7 of 10 devices are correctly categorised and described.	x	x	Enter new devices into a Hydra network, locally and remotely.	Partly supported. Devices executable in application
111	Dynamic Web Service Binding	Middleware should be able to after device discovery and categorisation expose a new device as a web service that can be called without re-compilation.	New devices are callable and controllable in 7 out of 10 cases.	x	x	Enter new devices into a Hydra network, locally and remotely.	Supported. Devices executable in application
114	Semantic enabling of device web services	Middleware should be able to attach semantic descriptions to device web services based on device ontology.	7 of 10 device are semantically enabled.	x	x	Enter new devices into a Hydra network, locally and remotely.	Not yet supported. Requires manual intervention.
122	Configurable and easy to install middleware	The middleware should be configurable and easy to install/deploy.	The average installation time is less than 1 hour.	x	x	Time a middleware installation.	Not yet supported. Installation still manual.
129	Support for Semantic Web Standards for Device Communication	Middleware should support different semantic web standards, including OWL-S, WSMO, and selected parts of WS-*	Support for at least OWL-S and WSMO	x	x	The use of the relevant standards (OWL, OWL-s and WS-*) are documented and can be shown.	Supported.
210	Middleware should support different architectural styles	It must be possible to build systems with different architectures such as fully decentralised vs centralised. De/centralization can pertain to: - data/knowledge - control - computation	Supports at least two different architecture styles	x	x	Implement two prototype applications, one with a centralized architecture and the other with a distributed approach, which handles two or more Application Device Managers.	Supported. Multiple DACs and P2P device access.
376	Security requirements must be part of the Hydra MDA	Security must be defined to be resolved semantically	Security model can be defined semantically	x	x	A semantic security model exists, check resolution process.	Not yet supported. The resolution process is not in place.

ID	Description	Rationale	Fit Criteria	Middle ware	SDK	Assessment procedure	Outcome
389	Service browsing in device ontology	It must be possible to view services as central building blocks, thus an application developer should be able to browse the device ontology from a service perspective, in addition to a device perspective.	A developer can find services and use them in development, without an a priori knowledge of the devices that implement the services.	x	x	Test with developer.	Supported indirectly thru the DAC. Also by using Ontology Browser tool.

Table 5: WP6 selected requirements

3.5 WP7 – Evaluated requirements

ID	Description	Rationale	Fit Criteria	Middle ware	SDK	Assessment procedure	Outcome
308	The Security Level of an existing network should be determinable	For a device entering an existing network it can be useful to determine the security level of that network. Depending on the provided security level the device can decide to enter the network or not.	Hydra middleware provides at least one mechanism enabling devices to determine the security level of an existing network.	x		Evaluation of the current status of the middleware architecture.	Not yet supported
468	Different levels of security must be supported	In the healthcare scenario there are 2 communication types: - the inter-BAN communication - the internet communication Each of them could implement a different security criterion. The middleware could support different security levels during communications with wireless devices. For example, a simple accounting procedure for devices near to the user (a BAN in the healthcare scenario) and an harder codification for long distance communications where identity data are transmitted are supported.	It must always be possible to implement at least two different security levels for an application.	x		Evaluation of the current status of the middleware architecture.	Not yet supported
472	Provide application developers with the functionality of checking tokens against a trust model	Public keys can only be used if the identity of the key owner is trustworthy	End-users (=application developers) are provided with an interface to check tokens (consisting of a public key and an identity) against a trust model	x		Evaluation of the current status of the middleware architecture.	Supported

ID	Description	Rationale	Fit Criteria	Middle ware	SDK	Assessment procedure	Outcome
473	Support of arbitrary trust models	Evaluation of "trust" is required in Hydra but the underlying trust model should not be predetermined in order to support the greatest possible range of applications.	Hydra provides mechanisms to register and use arbitrary trust models.	x		Evaluation of the current status of the middleware architecture.	Supported
474	Core Hydra security mechanisms should run on embedded devices	Core Hydra security is essential for protecting communication between managers of a virtual device. Thus, it should be scalable down to resource-restricted platforms.	Core Hydra security handlers perform sufficiently fast on resource-restricted platforms	x		Test Core Hydra security mechanisms on a resource-restricted platform, estimate scalability in general.	Supported

Table 6: WP7 selected requirements

4. Validation results

This section contains the description of the applied assessment procedures and outcomes, highlighting the major findings emerged during the validation fulfilment. The results are divided depicting the analysis carried on for each single requirement evaluated and grouped depending on their relative work package.

4.1 WP3 validation results

Req. ID: **18**

Description: Support for different software architectural patterns.

Fit criteria: Hydra allows at least two different architectural patterns for applications.

Assessment procedure: Implementation of applications based on the middleware, which show different architectural patterns. Analyse the current architecture design of the middleware.

Description of the assessment result

Supported. The middleware currently supports two architectural patterns: peer-to-peer and a service-orientation.

Req. ID: **31**

Description: An easy-to-use programming framework should be provided.

Fit criteria: 9 out of 10 developers recognise the SDK as intuitive.

Assessment procedure: Conduction of a software-walkthrough and validation sessions with developers specifically addressing the ease of use.

Description of the assessment result

Not supported yet. The SDK is currently in the design phase.

Req. ID: **33**

Description: Enable manufacturers to develop devices and applications that can be connected to Hydra.

Fit criteria: APIs are available to develop devices that can be connected to the Hydra network.

Assessment procedure: Develop applications based on the Hydra middleware and connect multiple third party devices to a Hydra network.

Description of the assessment result

Supported. The current demonstrators show that third party devices can be connected to a Hydra network e.g. through Gateways. The procedure is described in the documents provided by WP5 and in deliverable D3.9.

Req. ID: **41**

Description: Hydra Developer's Companion.

Fit criteria: Complete documentation is available. It is at least considered "very helpful" by at least 8 out of 10 developers.

Assessment procedure: Conduction of a technical review of the documentation. Run a software walkthrough as a preparation for the training activities.

Description of the assessment result

Partly supported. Currently, several documents exist that describe and illustrate the use and handling of the Hydra middleware. Deliverable D3.9 claims to be complete regarding the description of the functional aspects of the middleware and the deployment. However, documents that might contribute to the developer's companion need to be identified, and additional documentation still needs to be produced before it can be presented to developers.

Req. ID: **136**

Description: Dynamic architecture.

Fit criteria: In 95% of all cases, Hydra supports dynamic migration of components to realise centralised and decentralised systems.

Assessment procedure: Implement and run a test application and test whether it is able to be reconfigured or not.

Description of the assessment result

Following the assessment result of the associated requirement n. 317 (WP4, next section), respective work addressing this requirement is underway.

Req. ID: **185**

Description: Middleware provides basic services.

Fit criteria: Middleware provides a set of basic services that at least contain basic functionality that is needed by all services, like communication and a service / device registry.

Assessment procedure: Conduct a technical review of the core Hydra services with developers. This review aims at the setup of a basic Hydra infrastructure, querying available devices and passing messages between devices.

Description of the assessment result

Partly supported. The core Hydra services exist and consolidate, and current demonstrators show that these core elements work together properly. However, the formal technical review of this aspect has not been conducted, yet.

Req. ID: **186**

Description: GUI for configuring middleware parameters.

Fit criteria: A GUI exists for configuring the middleware.

Assessment procedure: The fit criterion of this requirement needs to be revised, since GUIs for the configuration of parts of the middleware already exist, but their contribution to the facilitation of the development needs to be assessed.

Description of the assessment result

Supported. However, the utility of these GUIs still remains to be validated.

Req. ID: **199**

Description: Modules should be extendable.

Fit criteria: 80% of all Hydra modules are extendable in their functionality by integrating 3rd-party code via a standard interface or replaceable by 3rd-party modules with equivalent functionality.

Assessment procedure: An assessment procedure that measures the extensibility of software is part of current research. One approach could be to count the number of hooks that allow for the modification of existing modules or the addition of new ones. Another approach could be to let a number of developers implement extensions to the Hydra middleware and to assess the result. Thus, a formal assessment procedure remains an open issue.

Description of the assessment result

Partly supported. The Hydra SDK will be published under an open source licence, which guarantees at least maximum modifiability. Furthermore, the software architecture follows several design patterns (see deliverable D3.9) that aim at a good extensibility. However, a formal assessment procedure could not be conducted so far.

Req. ID: **207**

Description: Service selection by context.

Fit criteria: In search requests for a specific service, contextual information like a spatial position is allowed.

Assessment procedure: Build a prototype which combines location and other context constraint to select an appropriate service. An example scenario would be: A user wishes to print a coloured document to the nearest printer during a presentation.

Description of the assessment result

Partly supported. The work is currently conceptual and an accordant realisation of the scenario needs to be done. Currently, a similar work in the building automation prototype from the CeBIT event has to do with the selection whether to display an error message on a screen at home or to send an SMS to the user depending on whether he is at home or not (which is in fact a location-based decision).

Req. ID: **217**

Description: The middleware should ensure high robustness of services.

Fit criteria: Breakdown of crucial services of the middleware in less than 1 case per 100 hours of operation.

Assessment procedure: Identify the crucial services of the Hydra middleware, build a test application that bases on that set of services and conduct a long term operation stress test.

Description of the assessment result

Partly supported. The building automation demonstrator, which addresses only a part of the available Hydra Managers, did not show any breakdown. Its current time of operation at the CeBit 2008 did already exceed 50 hours. Further formal tests need to be conducted.

Req. ID: **234**

Description: The middleware should be self descriptive.

Fit criteria: Nine out of ten developers have a clear understanding of the Hydra middleware after one week of experience.

Assessment procedure: Conduct a software peer review with developers.

Description of the assessment result

This requirement has not been fulfilled, yet. Such a software peer review will be scheduled after the developments on the Hydra middleware are finished.

Req. ID: **320**

Description: Separate domain-oriented services and user interface services architecturally.

Fit criteria: 90% of the modules of the architecture properly separate layers for domain services and interfaces.

Assessment procedure: Analyse the SVN repository which contains all Hydra managers and modules and identify those that mesh interface and control logic.

Description of the assessment result

No assessment results so far, since the set of Hydra managers is not complete yet. Currently, no user interface components exist in the architecture.

Req. ID: **327**

Description: The Hydra middleware should be flexible as to allow for opt-in and opt-out on parts.

Fit criteria: Hydra is able to support the exact subset of services required by a client (user or service) in 70 % of all cases. In 20 % of all cases the middleware is able to provide a service package that includes the required service. In 10% of all cases Hydra is not able to provide service similar to the desired service.

Assessment procedure: Build several test applications that base on a different set of managers of the Hydra middleware.

Description of the assessment result

Supported. The demonstrators already show that the Hydra middleware allows for a flexible bundling of services that make up the core functionality for the demonstrator. Furthermore, it has been shown, that extremely resource-constrained devices can be Hydra-enabled, and therefore, join a Hydra network. The results of this assessment as well as the concrete description of the dependencies among the Hydra managers need to be provided.

Req. ID: **329**

Description: Middleware provides domain-independent services.

Fit criteria: Large parts of the building-automation scenario can be built by reusing configurable services from across other application domains.

Assessment procedure: Build several test applications that base on a similar set of managers provided by the Hydra middleware.

Description of the assessment result

Supported. The demonstrators show that particularly for the core Hydra components this requirement is fulfilled. Though the realisation of further demonstrators and test applications the current set of Hydra managers will further consolidate and each manager will experience a further phase of generalisation.

Req. ID: **335**

Description: Location awareness / positioning support.

Fit criteria: A component for acquiring spatial context exists. At any time, min. 75% of all devices attached to a Hydra system can be spatially located. Also, there is a programming model for using spatial context.

Assessment procedure: Build a location-aware application based on the Hydra middleware.

Description of the assessment result

Partly supported. Please refer to requirement n. 207 description.

4.2 WP4 validation results

Req. ID: **312**

Description: The middleware should contain services that allow monitoring and reaction on what devices are doing. This includes monitoring response time, device load (e.g., CPU), and message interchanges per second.

Fit criteria: Said services available in Hydra. The monitoring should be pluggable so it does not induce a performance overhead when not used.

Assessment procedure: Experiments with the diagnostics manager.

Description of the assessment result

The web-services generated by the Limbo compiler are described as state machines. They emit state-change events which facilitate monitoring. As such there is currently support for monitoring what devices are doing.

In addition, there are probes generated for client-server code, which publish events to the Event Manager upon invocation-initiation (client) and invocation-effectuation (server). This allows for measurement of the throughput of messages, although experiments show that this capability comes with a heavy performance penalty.

In addition, this monitoring of client-server invocations allows detection of service failure.

Req. ID: **314**

Description: To create reliable and available systems it is essential to catch faults/partial failures before they become failures/complete failures. There needs to be uniformity in how this is done; thus it should be supported by the middleware.

Fit criteria: The middleware has support (through components/services) for sending and receiving notifications for partial failures.

Assessment procedure: Experiment with behaviour when services become available.

Description of the assessment result

See the description of Req. 312. The requirement is partially supported, in that some failures can be detected. Because the state machines and probes in the web services generated by limbo can emit events notifying both the client-side and server-side of web service invocations, it is possible to infer service failure when a client-side invocation event (emitted by the invoking service) is not matched by a corresponding server-side event (emitted by the invoked service).

Req. ID: **317**

Description: To supporting monitoring leading to adaptation, the architecture should be dynamic in the sense that components/services should be connectable in new ways at runtime.

Fit criteria: Services and devices can be connected in new ways during runtime in Hydra-based applications.

Assessment procedure: Test an example application's ability to be reconfigured according to specific scenarios.

Description of the assessment result

This requirement has not been implemented yet. Work is underway to support reconfiguration through an architectural scripting language that is used to describe reconfigurations, and its interpreter that can effectuate them in the Hydra middleware.

Req. ID: **318**

Description: Devices should be able to be added to the system at runtime. It should not be necessary, e.g., to shut a building complex down to add a new device to a room.

Fit criteria: Devices can be installed, discovered, and used while the Hydra runtime is running.

Assessment procedure: Test if devices can be added at runtime.

Description of the assessment result

The use of UPnP service discovery, implemented in the Device Discovery Manager, enables dynamic addition and removal of devices, because devices that are added at runtime may be discovered and subsequently used through the service discovery mechanism. A newly started device can be allocated a HID after it has been discovered through UPnP, or by allocating it through the Network Manager if it has client code for doing so.

Req. ID: **366**

Description: Web services should run on embedded devices. Service-orientation is a good match for many embedded devices. Web services will provide a gateway to many applications and it would be beneficial to be able to structure all of the communication in a system using the same primitives.

Fit criteria: Hydra supports web services on embedded device (Initial target should be Develco's DevCom 02 ZigBee module).

Assessment procedure: Test if a web-service can be used in a system running on embedded devices.

Description of the assessment result

Currently the Limbo web services compiler generates code for the Eclipse Equinox OSGi implementation, JME, or stand-alone JSE. It has not been tested on embedded devices, although the OSGi platform is used in many smart-home appliances, and JME is also supported on a range of embedded devices. The assessment has not been carried out yet.

Req. ID: **479**

Description: The Event Manager should handle events according to their priorities. Some events are critical to the health of the system and should be prioritized over others when there are a high number of events being routed through the system.

Fit criteria: Stress test of the event notification system. If the volume of events exceeds the capacity, events with high priority should be delivered first, and only be discarded as a last resort.

Assessment procedure: Stress test of the event notification system.

Description of the assessment result

The system has been tested and found to satisfy the fit criteria. There is a lower threshold in that the first events in a series of mixed-priority events published may not be handled according to priority, but at most one event out of order is handled this way.

4.3 WP5 validation results

Req. ID: **264**

Description: Common message protocol

Fit criteria: 100% of devices in the Hydra network use a common message protocol for service invocation and consumption.

Assessment procedure: Use a network sniffer to assert that the message protocol used is the same in all cases being machine readable and interoperable (devices must understand each other).

Description of the assessment result

A network sniffer (wireshark) listens to communications between different devices in the Hydra network. The message protocol used must be the same in all cases, while being machine readable and interoperable (devices must understand each other). In the test carried out, a HED (HED2) has a client (application) running in Spain that tries to consume the services provided by a weather station device and a radio frequency switch, discovered by another HED in a different location (Sweden) (HED1). The test shows that the format of the messages exchanged is the same in both cases, as SOAP is used as the standard protocol for message interchanges in Hydra. The next figure shows the actors in this test case.

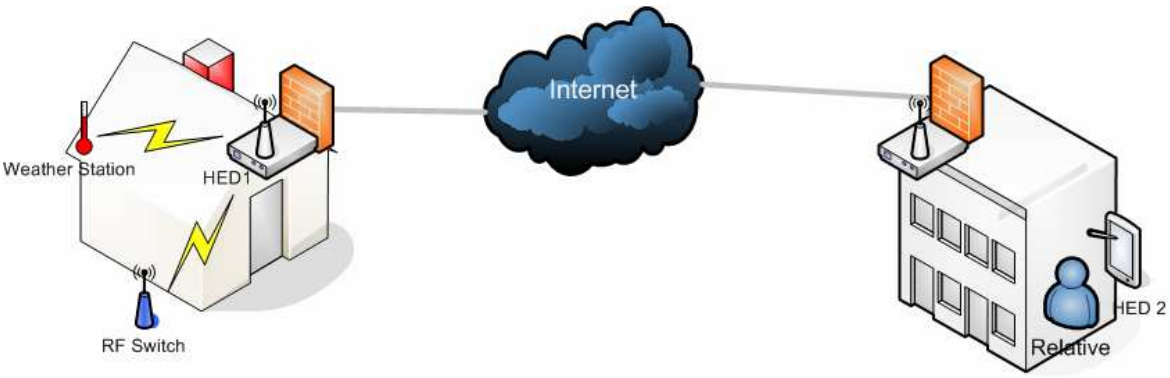


Figure 2: Requirement 264 test case actors

The first test deals with the RF switches. The client application from Spain calls the WS provided by the RF switch in Sweden. Next figure shows the application that accesses the RF switches.

Figure 3: Requirement 264 RF switch application application

The Wireshark application is launched and starts capturing the HTTP packets in the network. These are packets sniffed for the call to the RF switch, both the request and the response:

```
content-type: text/xml; charset=utf-8
connection: Keep-Alive
host: 127.0.0.1:8082
content-length: 223
soapaction: "http://tempuri.org/IHydraEnhancedSwitchWSService/GetSwitchStatus"
user-agent: PHP-SOAP/5.2.5

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://tempuri.org/">
<SOAP-ENV:Body>
<ns1:GetSwitchStatus/>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body>
<GetSwitchStatusResponse xmlns="http://tempuri.org/">
<GetSwitchStatusResult>on</GetSwitchStatusResult>
</GetSwitchStatusResponse>
</s:Body>
</s:Envelope>
```

The second test deals with the Thermometer of the Weather Station. The client application from Spain calls the WS provided by the Thermometer in Sweden. The next figure shows the application that accesses the Thermometer.

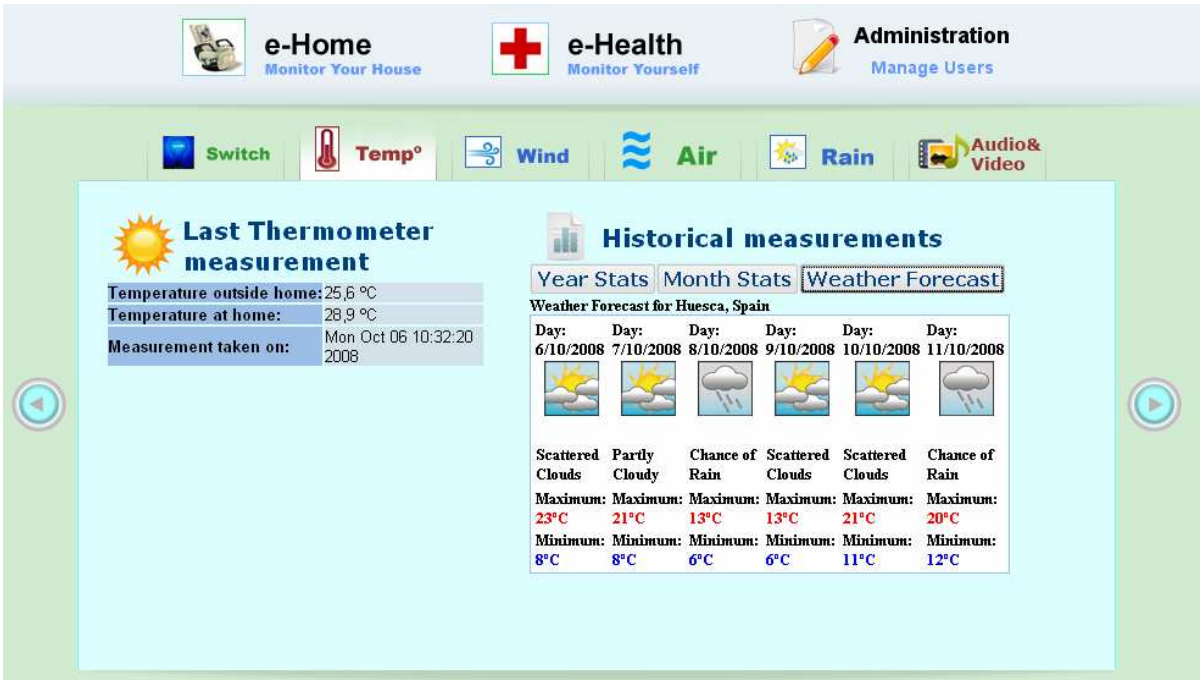


Figure 4: Requirement 264 RF switch application application

The Wireshark application is launched and starts capturing the HTTP packets in the network. These are packets sniffed for the call to the Thermometer, both the request and the response:

```
content-type: text/xml; charset=utf-8
connection: Keep-Alive
host: 127.0.0.1:8082
content-length: 228
soapaction: "http://tempuri.org/IHydraThermometerWSService/GetIndoorTemperature"
user-agent: PHP-SOAP/5.2.5

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://tempuri.org/">
<SOAP-ENV:Body>
<ns1:GetIndoorTemperature/>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body>
<GetIndoorTemperatureResponse xmlns="http://tempuri.org/">
<GetIndoorTemperatureResult>28,9</GetIndoorTemperatureResult>
</GetIndoorTemperatureResponse>
</s:Body>
</s:Envelope>
```

A comparison of the exchanged in case one and two packets proves the format uniformity, by the use of SOAP as the standard protocol for message interchanges in Hydra. This is the common protocol chosen for communication between Hydra devices in the Hydra network.

Req. ID: **276**

Description: New communication technologies

Fit criteria: 80% of new technologies are supported.

Assessment procedure: Integration of the ZigBee protocol and discovery mechanism

Description of the assessment result

The validation of this requirement was done based on the integration of the ZigBee protocol and discovery mechanism into the existing set of technologies already in the Hydra middleware (at the time of validation), i.e., Bluetooth, Radio, Serial, RFID, UPnP/DLNA, IP/WIFI. Although this requirement is considered supported, the percentage expressed in the fit criteria is yet to be met.

Req. ID: **336**

Description: Discovery protocol should support multiple networks

Fit criteria: The service discovery protocol of Hydra able to work over multiple physical networks, finding at least 90% of the services available.

Assessment procedure: Validation session with developers.

Description of the assessment result

This requirement is validated by letting developers applications discover devices connected to a remote network, and then having the possibility to access the services of these devices. The discovery information (for a specific device) is retrieved from the remote networks' network manager via the local network manager, to a local discovery manager. This local, protocol specific, discovery

manager will process this information in order to resolve it into a Hydra device. When resolved, the device will have a set of Hydra web services generated, and these may be invoked through web service calls from the developer application, using the soap tunnelling for transparent addressing of endpoints and subsequently service invocation.

Req. ID: **407**

Description: Storage Manager – Gateways information stored synchronization

Fit criteria: 90% of the information stored in the Gateway is synchronized with the information stored inside the devices.

Assessment procedure: Data will be annotated with timing information, which will be used to evaluate applied (soft) real-time constraints.

Description of the assessment result

The assessment procedure is strongly dependent on the implemented environment, because the requested soft real-time constraints cannot be guaranteed in arbitrary environments with arbitrary storage access patterns.

Therefore, the Storage Manager will contain means to ensure an online-analysis of the real-time behaviour. It is possible to parameterise the real-time requirements for each stored object. The parameter-check is different, whether the information is pulled by the Gateway or pushed from the client node.

In the first case, the Gateway can simply check the local modification time of the data object. The Gateway will increment an internal violation counter if this modification time is smaller than the actual time minus the threshold. This violation counter will be used to trigger an event to the application, if the percentage of violations becomes bigger than a predefined constant (e.g. 10 %).

The assessment becomes more difficult, if the device pushes information to the gateway node. In this case, data is only pushed, if the information actually changes. Therefore, the Gateway cannot only rely on the modification time inside its repository. Furthermore, it is necessary that the internal clocks of the device and the Gateway are synchronized (e.g. by means provided by the Network Manager). If this synchronization can be ensured than the Gateway checks the difference between the modification time inside arriving data and the arrival time at the Gateway. If this difference exceeds the threshold time, then the violation counter will be incremented. Again, this violation counter will be used to trigger an event to the application, if the percentage of violations becomes bigger than a predefined constant

The assessment procedure will be part of the standard implementation of the Storage Manager.

Req. ID: **419**

Description: Device services and resources provision through its Gateway

Fit criteria: 90% of devices can offer services and resources in the overlay network through its Gateway.

Assessment procedure: Perform tests with two Hydra Enabled Devices. Test the registration, discovery and consumption of services offered by a device.

Description of the assessment result

In order to validate this requirement, a scenario has been set up consisting on a Thermometer, two Hydra Enabled Devices, HED1 in Sweden and HED2 in Spain, and an application running on top of HED2 (see Figure 5).

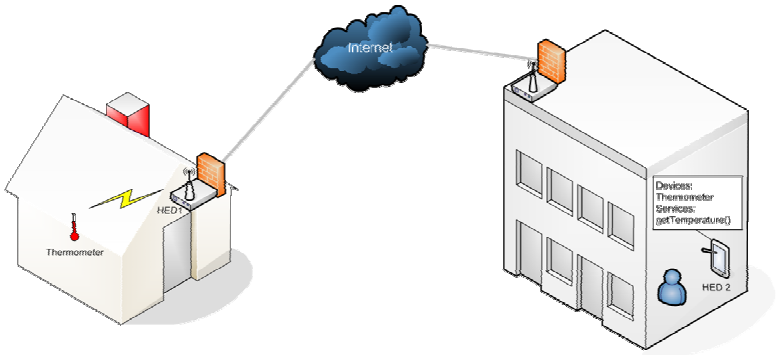


Figure 5: Requirement 419 test scenario

Figure 6 shows the process of service registration, discovery and consumption in Hydra.

First (1), HED1 is responsible of discovering the thermometer device in the local network and registering its services in the Hydra network using the local Network Manager. This is performed through the Discovery Manager and the Device Application Manager (see Figure). In order to register the service, the Discovery Manager creates an HID in the local Network Manager providing the local endpoint of the service provided by the thermometer.

Once the service has been registered, the Network Managers involved exchange the local HIDs registered (2). Then, the Network Manager in HED2 is aware of the new HID created for the thermometer, and an application running on top of HED2 is able to transparently consume the services offered by the thermometer, using the local SOAP tunnel (4) component and the Network Manager (5). In order to do that, the application invokes the service using the endpoint of the local SOAP tunnel instead of the real endpoint of the service (not known by HED2 as it is only available in HED1). The SOAP tunnel and the Network Manager are responsible of routing the SOAP message generated by this invocation to the Network Manager (5, 6) responsible of the thermometer. Finally, the SOAP message is delivered to the device (or proxy) using the local SOAP tunnel (7, 8) and the response of the service is routed back to the application.

Therefore, the Network Manager and the SOAP tunnel provide the mechanisms for service registration, discovery and invocation for any device discovered by the Discovery Manager.

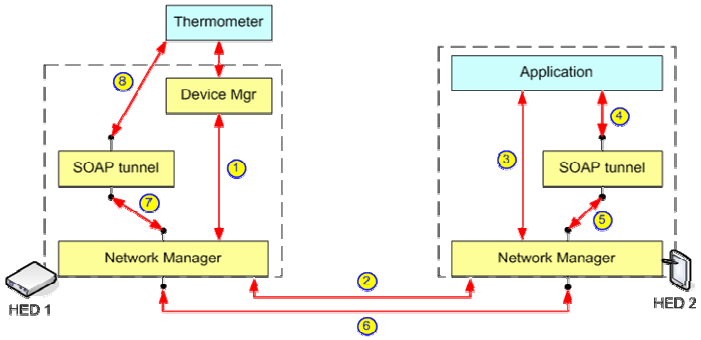


Figure 6: Service consumption process in Hydra

Req. ID: **425**

Description: D2D communication – Overlay Hydra network

Fit criteria: 90% of the HEDs are able to communicate with each other even if they are located behind firewalls or NATs, and to participate in the Hydra network.

Assessment procedure: Perform tests with two Hydra Enabled Devices behind firewalls/NAT's and assert if they can communicate, by consuming services offered by devices controlled by them.

Description of the assessment result

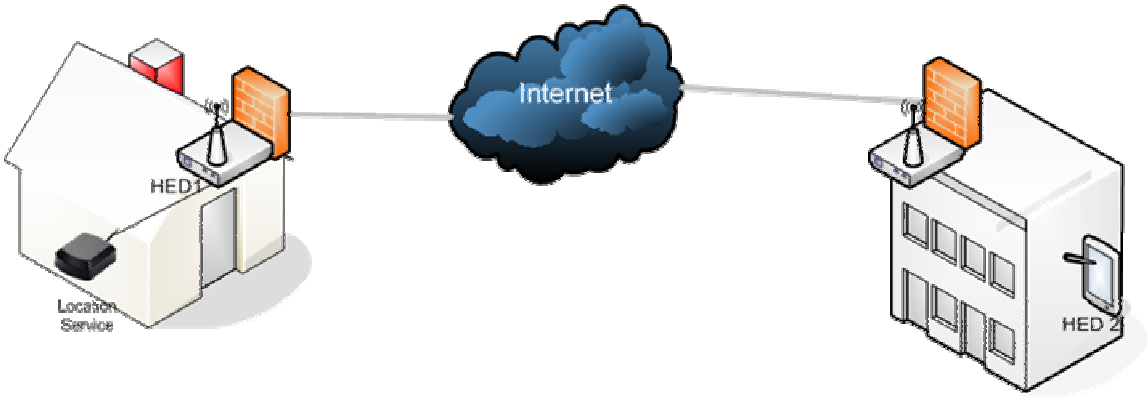


Figure 7: Device to Device communication test scenario

In order to validate this requirement, two Hydra Enabled Devices, HED1 in Telefónica (Spain) and HED2 in CNET (Sweden) have been set behind firewalls and NATs, as shown in Figure 7. A GPS device is offering a location service in HED1 using the Network Manager. A client built on top of HED2 will attempt to consume the Web Service from the GPS device, using the Network Manager and SOAP tunnel components in HED2.

In order to fulfil the requirement, 90% of service invocations should be performed correctly. A set of 100 tests have been performed. For comparing the results obtained, another battery of tests has been performed in a scenario without firewalls or NATs and using direct WS communication. The results are shown in Figure 8.

In both cases, the 100% of service invocations is performed successfully due to the reliability of communications. The average Round Trip Time represents the time since the service is invoked until the response of the service is received. The results show a higher RTT in the firewall scenario, because of the overhead introduced by JXTA. However, this overhead is adequate taking into account that direct WS calls are not possible in the firewall scenario. Therefore, this requirement is fulfilled.

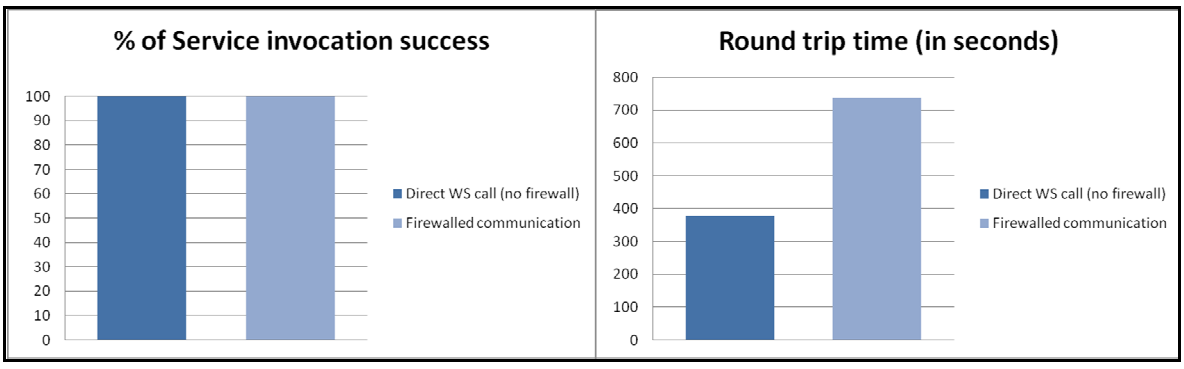


Figure 8: Results from the D2D communication tests

Req. ID: **445**

Description: The level of protection should be independent from the currently used low-layer protocol

Fit criteria: In 95% of all cases the security model should be applicable to multiple protocols below SOAP layer.

Assessment procedure: Evaluation of the current design of communication protection.

Description of the assessment result

This requirement states that all mechanisms for communication protection used in Hydra should be independent from the currently used low-layer protocol, whereas "low-layer" refers to all protocols below the SOAP layer.

The Hydra communication protection is separated into "Core Hydra" and "Inside Hydra" message protection. While both mechanisms refer to slightly different security requirements, they are both implemented at the SOAP-layer. That means, protected Hydra messages are always embedded in SOAP messages which makes them completely independent from underlying protocols of OSI layers 1-4. As the SOAP specification clearly states, SOAP can be applied to arbitrary low-layer protocols as long as a SOAP binding for the respective protocol has been defined. Therefore, this requirement can be regarded as fulfilled.

A further implication of this requirement and the resulting implementation is that all messages that are not sent over SOAP can not be protected by the Hydra middleware. For example all messages using plain JXTA or UPnP are not protected and can potentially be read and modified by everybody.

Req. ID: **455**

Description: Identity - Update of the correspondences between identifier and physical addresses.

Fit criteria: In 100% of the times, when a physical address is changed, the identifiers management module is notified and updates its information.

Assessment procedure: Perform tests with two Hydra Enabled Devices. If one of it changes of IP, the HID table of the other has to be updated accordingly.

Description of the assessment result

A HED (HED2) is running in Spain while another HED is running in a different location (Sweden) (HED1). A weather station device is discovered by HED1. The test shows that the discovered information (HID and description) is spread through out the Hydra network. The HID table in HED2 is updated with the new information. Once the weather station devices disappears from the network, HED1 is unable to discover it, and thus, the local table of HID is updated, the information spread again and the HID table in HED2 updated accordingly. Next figure shows the actors involved in this test case.

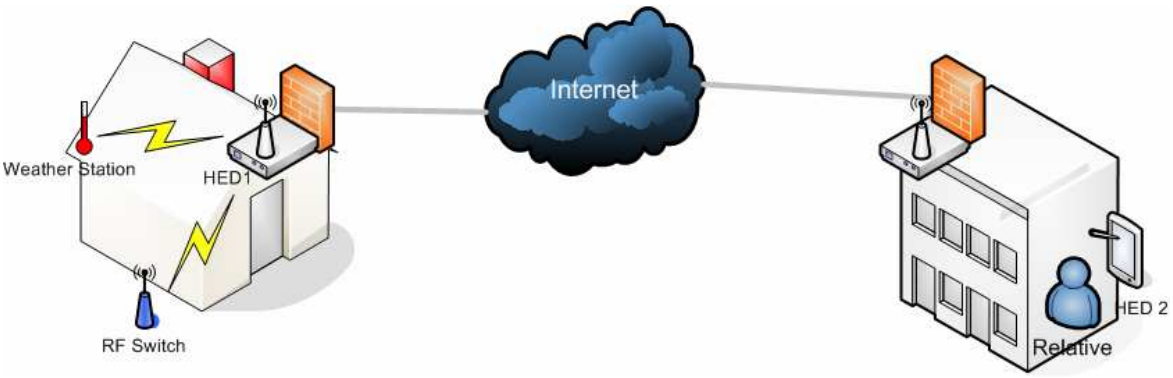


Figure 9: Req. 455 test case actors

The following boxes show what is going on in the HID tables of the involved actors in the test case. The first box shows the status of the HID table of HED2 before the devices are discovered in HED1. Then, HED1 discovers the weather station and a light and publishes the assigned HIDs in the network (second box). The third box shows how the HID table in HED2 is updated with the new information. After that, the weather station is disconnected and HID1 has to update again the its local HID table and announce the changed to the other elements in the network. The new announcement is shown in the forth box. Finally, the last box shows again the updated HID table in HED2. All the information is taken from the logs of the Network Manager in HED1 and HED2.

```
idTable Status:
20.20.20.20 10.95.74.100:8082:NetworkManagerApplication:http://localhost:8082/NMApp
```

```
Publishing NM with the following hids:
0.0.0.97903252919994039;RainSensor:Sweden:
0.0.0.300186204304291141;Windmeter:Sweden:
0.0.0.7501027082104858231;Light1:Sweden:
4.4.4.4;NetworkManager:Sweden:
```

```
idTable Status:
20.20.20.20 10.95.74.100:8082:NetworkManagerApplication:http://localhost:8082/NMApp
0.0.0.979032529199940392 212.128.0.72:8082:RainSensor:Sweden:
0.0.0.300186204304291141 212.128.0.72:8082:Windmeter:Sweden:
0.0.0.7501027082104858231 212.128.0.72:8082:Light1:Sweden:
4.4.4.4 212.128.0.72:8082:NetworkManager:Sweden:
```

```
Publishing NM with the following hids:
0.0.0.97903252919994039;RainSensor:Sweden:
4.4.4.4;NetworkManager:Sweden:
```

```
idTable Status:
20.20.20.20 10.95.74.100:8082:NetworkManagerApplication:http://localhost:8082/NMApp
0.0.0.7501027082104858231 212.128.0.72:8082:Light1:Sweden:
4.4.4.4 212.128.0.72:8082:NetworkManager:Sweden:
```

Req. ID: **465**

Description: Networks overlapping

Fit criteria: Device is not to be added to an existing Hydra network if it is unauthorised when it belongs to another Hydra network which is temporary near to the previous one.

Assessment procedure: Validation session with developers.

Description of the assessment result

This requirement is not yet supported. Resolution and enforcement of authorization is not in place yet.

Req. ID: **475**

Description: Multimedia streaming in the Hydra network

Fit criteria: 80% of multimedia devices are able to exchange multimedia streams using the Hydra overlay network

Assessment procedure: Test this feature with different multimedia devices (UPnP AV)

Description of the assessment result

In order to satisfy this requirement, the Network Manager has been designed to enable multimedia streaming of audio and video over the overlay Hydra network following the UPnP AV standard (version 1.0). Then, it should be logical to say that 100% of UPnP AV devices are compatible with it. However, not all the devices follow the specification strictly, so there is a chance of having problems with some of them. Therefore, in order to pass this requirement, different UPnP AV devices (servers and renderers) will be tested.

The test scenario is shown in Figure 10, where HED1 controls the UPnP renderer and HED2 controls the UPnP server. The test consist on, using an application running on top of HED1, reproduce content from the UPnP server on the UPnP renderer.

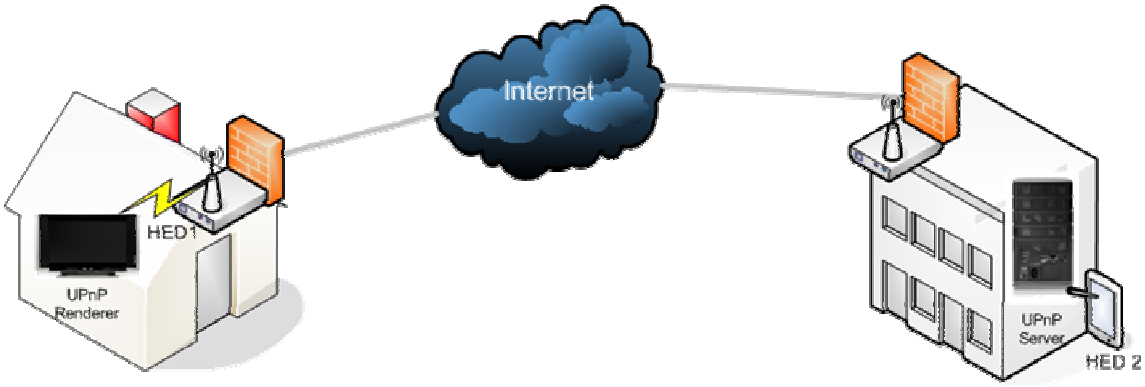


Figure 10: Multimedia streaming test scenario

In order to satisfy the requirement, different UPnP servers and renderers have been tested and the results are shown in Table 7.

From the results we can assure that more of the 80% of UPnP AV devices compatible with 1.0 specification can be used for multimedia streaming using the Network Manager mechanisms. Therefore the requirement is fulfilled.

Device	Audio	Video	Pictures
Intel AV Server	Yes	Yes	Yes
TwonkyVision Media Server	Yes	Yes	Yes
Google Media Server	Yes	Yes	Yes
Intel AV Renderer	Yes	Yes	Yes
Noxon AV Renderer	Yes	No[1]	No[1]
Philips Streamium Renderer	Yes	Yes	Yes

Table 7: UPnP AV servers and renderers compatibility tests

Req. ID: **476**

Description: Network Manager Configuration and Testing

[1] Not implemented in the device

Fit criteria: 80% of the configuration parameters and functionalities are available for configuration and testing

Assessment procedure: Check that the application developer can use a visual tool to change the configuration parameters of the Network Manager

Description of the assessment result

The application developer can use a visual tool to change the configuration parameters of the Network Manager. Next figures show two screenshots of the visual tool, part of the SDK, that manages the configuration of more than 80% of the configurable parameters of the Network Manager. The tool not only lets you change configuration parameters of the Network Manager but to test the functionality of it carrying different tests.

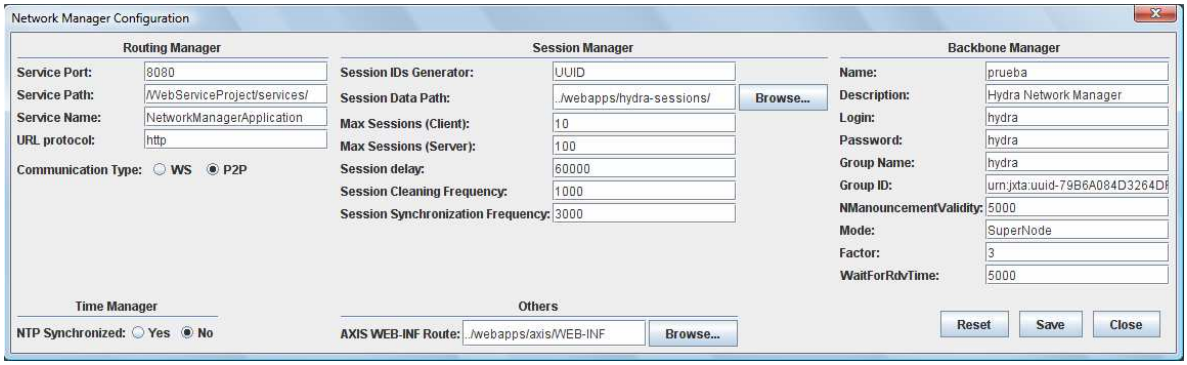


Figure 11: Network Manager configuration

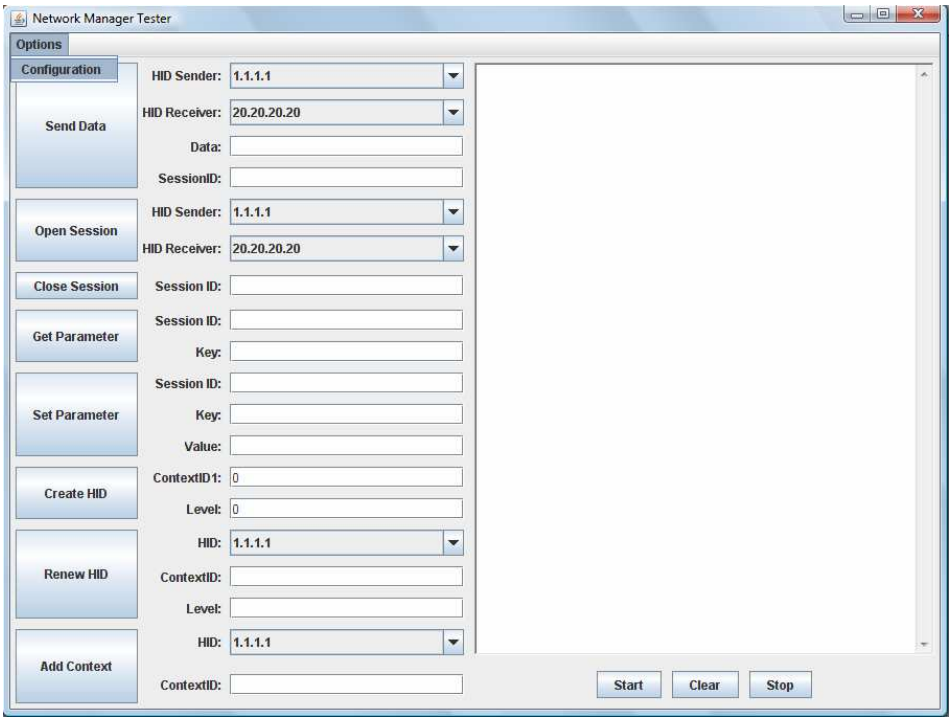


Figure 12: Network Manager tester

4.4 WP6 validation results

The WP6 requirements selected in this document were validated by conducting a validation session with a small set of developer users, in combination with an analysis of the current architecture and manager implementation.

The validation session was performed as a development assignment for users. The group included four developer users, of which three were experienced professionals and one was a student. Two instructors led the session, which was video taped in part.

The developers were given an assignment to write a Hydra application that should integrate and access a number of devices in a home automation setting. A set of sample devices of diverse types were part of the validation scenario.

Figure 13: Hydra enabled devices in the validation scenario setup

The development environment (IDE) consisted of the Hydra SDK, integrated in .NET and Visual Studio, with the developer client machines connected to a local (WIFI) net. A gateway machine in the local net was running the Hydra middleware managers, including a Network Manager for access to a remote network.

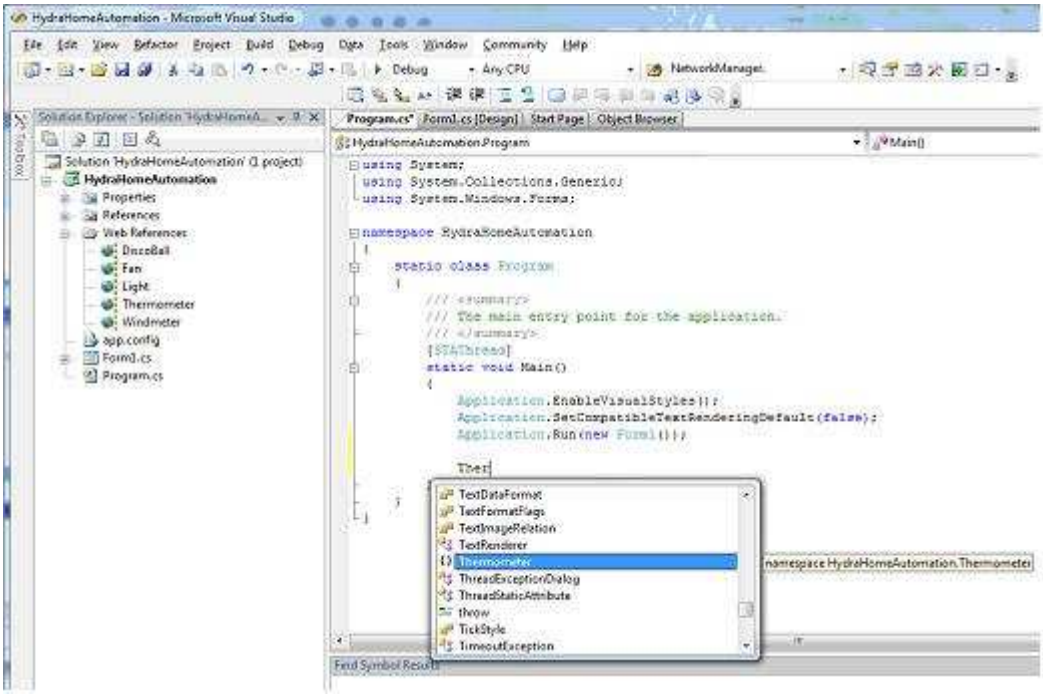


Figure 14: A device object (a Hydra enabled thermometer) is created in the application

The development assignment included finding devices, including them in the Device Application Catalogue (DAC), and writing the necessary code with web service calls to device services. A sample code snippet follows,

```
string windspeedstring = myWindMeter.GetWindSpeed(); //get windspeed from WindMeter
double windspeed = System.Convert.ToDouble(windspeedstring); //Convert to string

if ( windspeed > 2)  // check limit
{
       myLight.Flash(3, true);   // Flash light 3 times
       myFan.TurnOff();
}
```

During the session developers were able to test and debug their applications, while viewing the device real time behaviour in a video take-up of the setup.

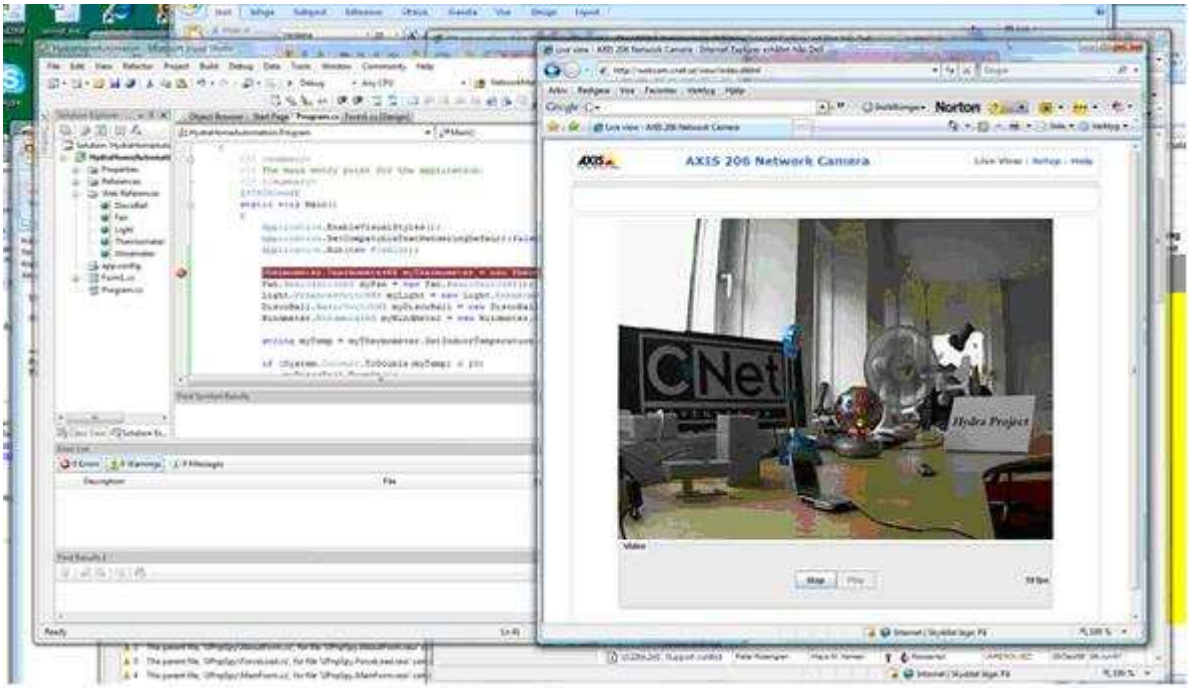


Figure 15: Run-time view of the IDE

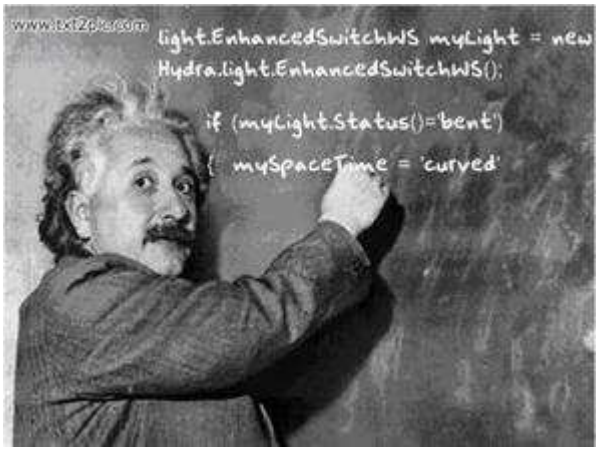


Figure 16: Instructor at work

Req. ID: **91**

Description: Any Hydra device should have an associated description.

Fit criteria: Any device associated to a Hydra application is also included in the Hydra device ontology, and its description can be retrieved.

Assessment procedure: Check that a newly discovered device has/gets a corresponding representation in the Device Ontology.

Description of the assessment result

At the time of validation the devices were not automatically classified in the device ontology as they were discovered. Hence an application could have access to devices not having a corresponding description in the ontology. Devices could however be manually classified.

Req. ID: **101**

Description: Manual device ontology definition.

Fit criteria: The Hydra IDE supports the manual editing of devices in the framework of a device ontology.

Assessment procedure: Interface exists for entering and updating device descriptions in the device ontology.

Description of the assessment result

The Device Ontology can be updated using the web-based administration tool.

Req. ID: **108**

Description: Device Discovery

Fit criteria: 7 of 10 devices are discovered.

Assessment procedure: Enter new devices into a Hydra network, locally and remotely.

Description of the assessment result

The requirement states the need to physically detect new devices in the local network. This is governed by the current set of supported physical discovery managers for the individual device protocols. Among the currently supported (discoverable) device protocols are Bluetooth, RFID, ZigBee, RF, Serial and UPnP.

See also requirement ID 336 above (section 4.3).

Req. ID: **110**

Description: Device categorization in run-time.

Fit criteria: 7 of 10 devices are correctly categorized and described.

Assessment procedure: Enter new devices into a Hydra network, locally and remotely.

Description of the assessment result

The last step in the discovery process is to categorise a device based on device ontology information (aka the semantic discovery). This requirement is yet to be fully verified regarding the fit criteria ratio.

Req. ID: **111**

Description: Dynamic Web service binding.

Fit criteria: New devices are callable and controllable in 7 out of 10 cases.

Assessment procedure: Enter new devices into a Hydra network, locally and remotely.

Description of the assessment result

After a device is discovered the middleware creates a web service interface and exposes this thru the device application catalogue. Each service is then callable from applications (or thru the DAC browser).

Req. ID: **114**

Description: Semantic enabling of web services.

Fit criteria: 7 of 10 devices are semantically enabled.

Assessment procedure: Enter new devices into a Hydra network, locally and remotely.

Description of the assessment result

The intention here is that the system should be able to associate semantic descriptions to device (web) services based on the device ontology. After a device has been discovered (physically and thru UPnP) the discovery process will generate a web service interface for the device (c.f. req. 111). The corresponding web services may then be further classified and described in the service part of the device ontology. This requirement currently only partly supported, it requires manual intervention in terms of updating the device ontology. However, we should also note the system is able to handle device with associated annotated WSDL files.

Req. ID: **122**

Description: Configurable and easy to install middleware.

Fit criteria: The average installation time is less than 1 hour.

Assessment procedure: Time of middleware installation.

Description of the assessment result

The final installation procedures and scripts have not yet been developed. The current Hydra implementation and configuration can probably meet the installation time constraint in most case, but requires manual intervention.

Req. ID: **129**

Description: Support for Semantic Web Standards for device communication.

Fit criteria: Support for at least OWL-S and WSMO.

Assessment procedure: The use of the relevant standards (OWL, OWL-s and WS-*) are documented and can be shown.

Description of the assessment result

Of the de facto standards that can be labelled "semantic web" standards, Hydra implements: OWL, OWL-s (simplified), SAWSDL, SWRL. As a result of previous design decision, WSMO is not supported.

Req. ID: **210**

Description: Middleware should support different architectural styles.

Fit criteria: Supports at least two different architecture styles.

Assessment procedure: Implement two prototype applications, one with a centralized architecture and the other with a distributed approach, which handles two or more Application Device Managers.

Description of the assessment result

We refer to architectural styles here in the general sense as the result of building applications with different architectures with respect to the degree of de/centralization of data/knowledge, control and computation. In Hydra, an application can work with multiple device application catalogues (DACs) and across network boundaries using P2P. This would correspond to a decentralized architecture, whereas a centralized Hydra application would use a single DAC in a local network. The degree of de/centralization also depends on the capabilities of the devices connected to the application.

Req. ID: **376**

Description: Security requirements must be part of the Hydra MDA.

Fit criteria: Security model can be defined semantically.

Assessment procedure: A semantic security model exists, check resolution process.

Description of the assessment result

We can conclude that security requirements can be included in the MDA (the model driven architecture) of Hydra, i.e., the security meta model describes security requirements and policies, and a security ontology is in place. To validate this requirement, it should be possible to define security at the application and device levels and to resolve it semantically. However, the resolution process is not yet implemented, so this cannot be validated in run-time.

Req. ID: **389**

Description: Service browsing in the Device Ontology.

Fit criteria: A developer can find services and use them in development, without an a priori knowledge of the devices that implement the services.

Assessment procedure: Test with developer.

Description of the assessment result

This requirement says that an application developer should be able to browse the device ontology from a service perspective, in addition to a device perspective. This is made possible by the ontology structure (e.g., the device and service taxonomies) and the Ontology administration tool.

4.5 WP7 validation results

Req. ID: **308**

Description: The Security Level of an existing network should be determinable

Fit criteria: Hydra middleware provides at least one mechanism enabling devices to determine the security level of an existing network.

Assessment procedure: Evaluation of the current status of the middleware architecture.

Description of the assessment result

The term "Security level" stated in this requirement is relatively fuzzy. We will therefore briefly describe how currently assurances of certain security mechanisms and protocols can be determined. After that, we will discuss whether the current status fulfils the requirement or not.

Currently, Hydra's security ontology allows to reason about assurances that have been provided by certain institutions for different security algorithms and protocols. As an example, the German

Bundesnetzagentur regularly publishes a list of algorithms and estimations of their strength, depending on different key lengths. Provided that it would be possible to gather a list of security mechanisms that are currently in use in a Hydra network, the security ontology could provide existing assurances for those mechanisms. As an example, Figure 17 shows how the security ontology returns estimations of the strength of the RSA encryption algorithm.

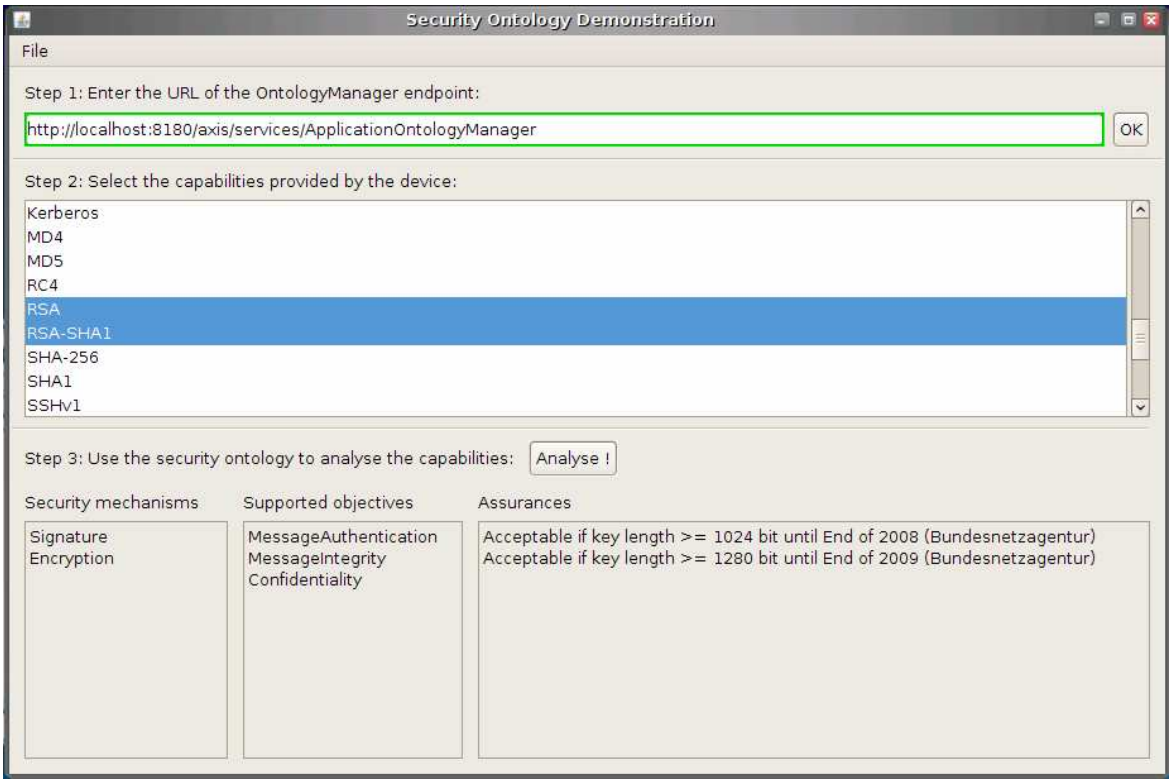


Figure 17: Different security mechanisms assurances in the security ontology

In order to fulfil the requirement, it must be possible to observe the security mechanisms that are used in a network and to estimate their "Security level", i.e. their strength. The second point is fulfilled by the security ontology and the reasoner as shown above. However, at the moment it is not possible to automatically detect all security mechanisms that are used by entities in a network. In theory, this is also only possible up to a certain extend: Core and Inside Hydra messages contain meta information about the protection mechanisms that have been applied to the message. A device could observe the protected communication in a network and use this meta information to query the security ontology to get explanations of the current "level of security". However, this would not work for security mechanisms whose application can't be observed. For example, it would not be obvious whether devices protect data before storing it or how authorisation and access-control is organised. An alternative would be that devices joining the network are presented a list of mechanisms that describe how the security of the network is organised. Unfortunately, revealing such information to a device that has not been authenticated yet would be a serious security flaw.

Therefore, this requirement can't be regarded as fulfilled. It is furthermore questionable whether it is even possible to fulfil it at all. We recommend stating the Fit Criteria more precise so that it becomes clearer which security mechanisms should be evaluated according to which criteria.

Req. ID: **468**

Description: Different levels of security must be supported

Fit criteria: It must always be possible to implement at least two different security levels for an application.

Assessment procedure: Evaluation of the current status of the middleware architecture.

Description of the assessment result

As already stated above, the term "level of security" is very fuzzy which makes the evaluation of this requirement hard. In terms of cryptography for message protection, this requirement is fulfilled as the modules for Core Hydra and Inside Hydra communication protection are based on XMLEncryption[2] and XMLSignature[3]. Both standards define a message format for protected data but leave it up to the developer to use a suitable cryptographic algorithm from a list recommended ones. In that way, different "security levels" in terms of "algorithms" and "key lengths" are supported.

Besides, "security level" could also be understood in the sense of a set of access-control policies. The "security level" could be higher, the more access to different services is restricted by those policies. Even in that way, the requirement can be considered fulfilled as Hydra's policy framework will provide the basis for defining and enforcing such access-control rules.

However, it is not absolutely clear what the original intention of that requirement was and thus we recommend splitting it up during the next iteration into several more precise requirements.

Req. ID: **472**

Description: Provide application developers with the functionality of checking tokens against a trust model

Fit criteria: End-users are provided with an interface to check tokens (consisting of a public key and an identity) against a trust model

Assessment procedure: Evaluation of the current status of the middleware architecture.

Description of the assessment result

The term "token" comprises the combination of cryptographic public key and optional additional information, describing the purpose or the owner of the key. Examples for such tokens are X509v3 certificates, PGP certificates or reputation-based credentials. In order for this requirement to validate, a component of the middleware has to provide an interface which allows developer users to validate such tokens at runtime.

The Trust Manager is designed for that purpose. By providing a web service interface with methods `getTrust(token)` and `getTrust(token, trustModel)`it fulfils this requirement.

Req. ID: **473**

Description: Support of arbitrary trust models

Fit criteria: Evaluation of "trust" is required in Hydra but the underlying trust model should not be predetermined in order to support the greatest possible range of applications.

Assessment procedure: Evaluation of the current status of the middleware architecture.

[2] http://www.w3.org/TR/xmlenc-core/
[3] http://www.w3.org/TR/xmldsig-core/

Description of the assessment result

In order to validate this requirement, it has to be split up into two criteria that have to be fulfilled: On the one hand, the middleware has to provide the functionality of evaluating "trust" and on the other hand, it must be possible for developers to choose an arbitrary trust model as a basis.

Regarding the first criteria, the Trust Manager is found to fulfil the requirement. Given a token (consisting of a key and identity information) the Trust Manager evaluates a level of trust for this token. This functionality is provided by two methods, named getTrustValue() and getTrustValueWithIdentitfier(), respectively. Both will return a value between 0 (token is untrusted) and 1 (token is fully trusted). The specific meaning of values between 0 and 1 is determined by the underlying trust model which is not part of the Trust Manager. So, by using the Trust Manager, it is possible to evaluate "trust" as required in the Fit Criteria.

Regarding the second part of the Fit Criteria, the middleware must not predetermine any trust model but has to be open for any arbitrary trust model. A trust model is a methodology that maps tokens to trust values. Tokens have to contain at least a cryptographic key and an identity and may be of any arbitrary format (e.g., X.509v3, PGP certificates, etc.). The trust value describes how likely the validity of the key is, i.e. whether it can be assumed that the key belongs to the identity stated in the token. It is the trust model's task to derive this trust value from a token. To be open for any trust model, the Hydra Trust Manager provides a mechanism for registering arbitrary trust models that can be used. Any trust model that implements a predefined Java interface can be added to the Trust Manager's configuration and will then be made available to the application developer. Thus, all trust models that can be mapped to the Java interface are supported by Hydra. The interface requires the trust model only to:

1. Have an arbitrary name
2. Accept any bytes as a token
3. Return a value between 0 and 1 for every supported token

As these requirements are the least common denominator for all current (and probably even future) trust models, it is feasible to regard also the second criteria of this requirement as fulfilled.

In order to add a practical part to this validation, a prototype of the Trust Manager has been implemented and two trust models have been added: one model is a "Null" trust model that validates every token regardless of its content; the other model validates X509v3 certificates. Depending on the configuration, validation according to the Common-PKI[4] and PKI-X[5] standards are supported. More details on this component can be found in deliverable D7.6.

Req. ID: **474**

Description: Core Hydra security mechanisms should run on embedded devices

Fit criteria: Core Hydra security is essential for protecting communication between managers of a virtual device. Thus, it should be scalable down to resource-restricted platforms

Assessment procedure: Test Core Hydra security mechanisms on a resource-restricted platform, estimate scalability in general.

Description of the assessment result

Core Hydra Security comprises protection of the communication between Hydra managers of the same virtual device. Those components may be located on a single physical platform but will also be distributed across heterogeneous platforms in many cases. As this functionality will be integrated into every secured Hydra manager, it is desirable to keep it as lightweight as possible to support security even for resource-restricted platforms. At first, we will outline the minimum theoretical

[4] http://www.common-pki.org/index.php?id=567&L=1
[5] http://www.ietf.org/rfc/rfc5280.txt

requirements for applying Core Hydra Security and after that provide the results we gained from performance measurements made on an exemplary resource-restricted platform.

Core Hydra Security is based on the approved standards XMLEncryption and XMLSignature. Other protocols like WS-Security are also based on these standards but require additional mechanisms like SOAP headers which are not available on all platforms. As their names imply, XMLEncryption and XMLSignature are based on XML-structured data, i.e. an XML parser is required. Further, they make use of cryptographic algorithms which have to be available on the respective platform. Secret keys are deployed at design time and have to be stored in a keystore file (but could optionally be kept in memory). So, in summary, these are the functionalities that have to be available to a platform to support Core Hydra Security:

- A runtime environment (Java VM or .NET framework)
- A XML parser
- A cryptography library
- (optionally, access to the file system)

A minimum set-up that should theoretically be able to run Core Hydra Security would thus be a home DSL router, a TV set-top box or home gadgets like a Chumby[6]. Those platforms usually comprise RISC processors, and about 16MB to 256MB of memory. Java virtual machines for such platforms are available (e.g., JamVM[7]) as well as SAX-compliant XML parsers (e.g., MinML[8]) and cryptographic APIs (e.g., Bouncycastle's J2ME-compliant API[9]).

For the Hydra home automation prototype, presented at the CeBIT fair 2008, a Sony Playstation 3 was used for applying Core Hydra Security to kSOAP messages. On this platform (a PPC-970 at 3.2 GHz with 256MB of memory), no slowdown due to the security module was noticeable. Further, for considerations regarding the scalability of encryption speed and bandwidth depending on the size of Core Hydra messages, we refer to deliverable D7.6 where approximate linearity is shown (cf. Figure 18 and Figure 19).

Although the term "resource-restricted platform" from the requirement is not absolutely precise, given the fact that practical experiments with the Playstation 3 provided very good results and theoretical considerations showed that Core Hydra Security should run with a very small foot print, it is feasible to regard this requirement as fulfilled. To further assure this validation, experiments with even smaller sized platforms should be conducted.

[6] http://www.chumby.com
[7] http://sourceforge.net/projects/jamvm/
[8] http://www.wilson.co.uk/xml/minml.htm
[9] http://ww.bouncycastle.org

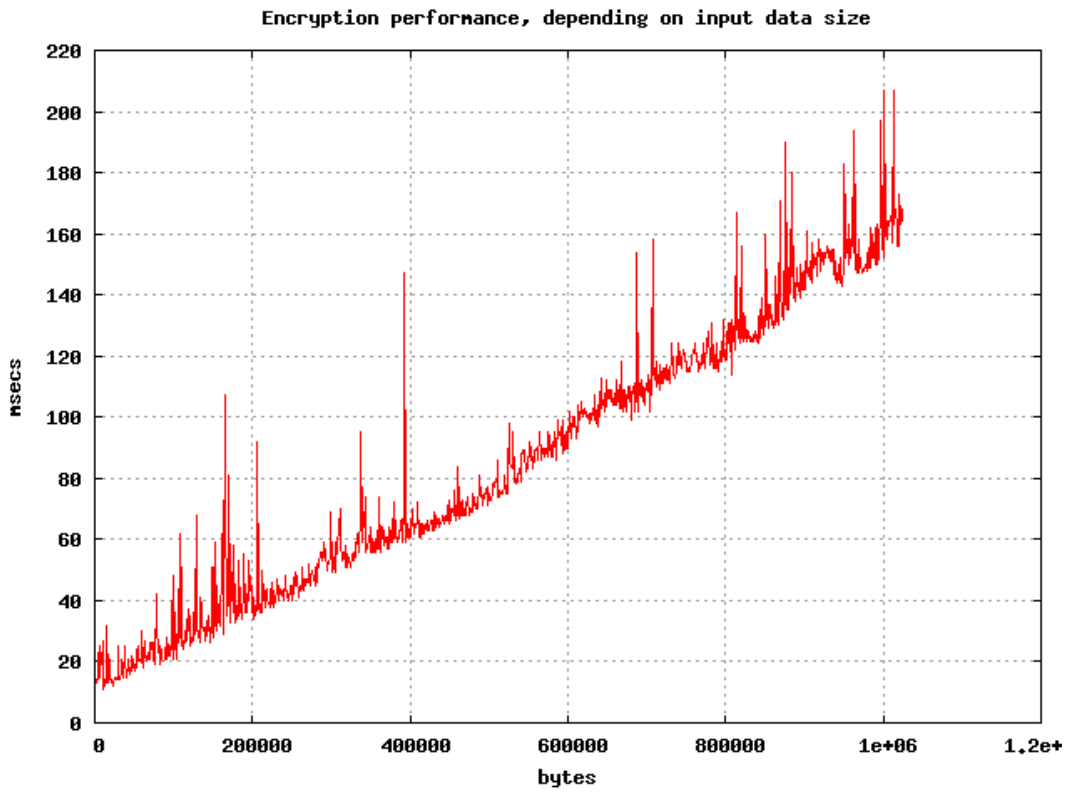


Figure 18: Duration of encrypting a message, depending on message size

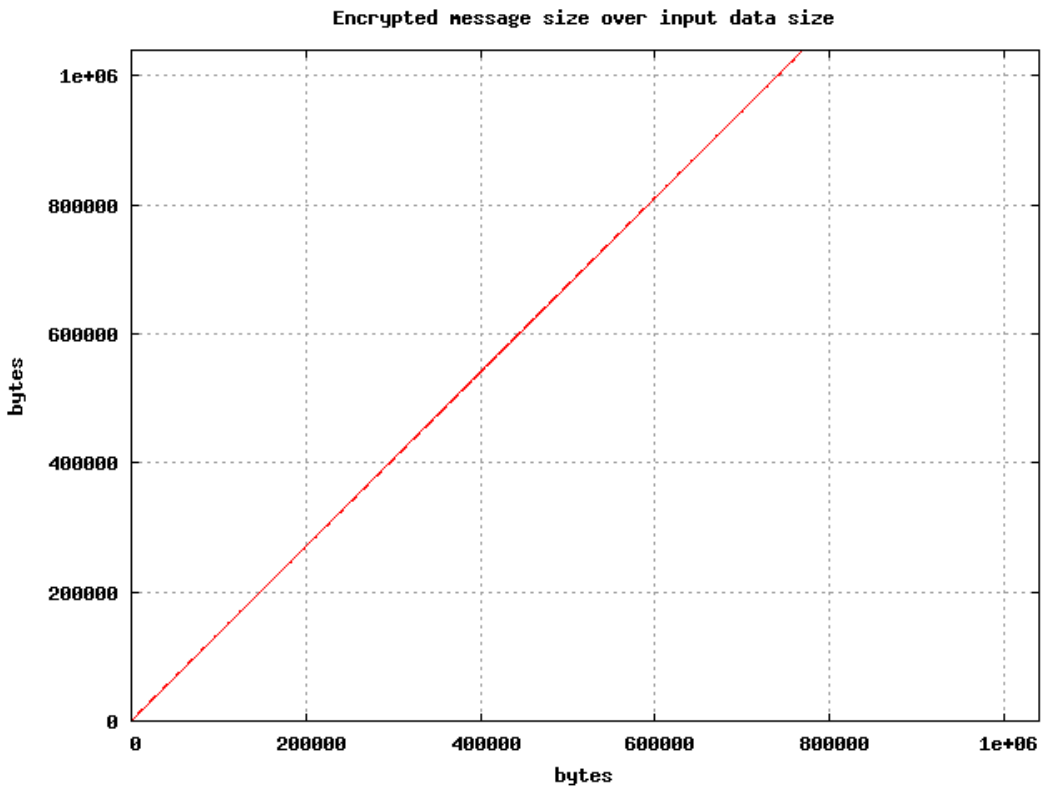


Figure 19: Size of encrypted message depending on input message size

4.6 Summary of the evaluated requirements

In the following table we summarise the results obtained for the validation of the selected requirements.

WP3		
18	Support for different software architectural patterns	Supported
31	An easy-to-use programming framework should be provided	Not yet supported
33	Enable manufacturers to develop devices and applications that can be connected to Hydra	Supported
41	Hydra Developer's Companion	Partly supported
136	Dynamic architecture	Not yet supported
185	Middleware provides basic services	Partly supported
186	GUI for configuring middleware parameters	Supported
199	Modules should be extendable	Partly supported
207	Service selection by context	Partly supported
217	The middleware should ensure high robustness of services	Partly supported
234	The middleware should be self descriptive	Not yet supported
320	Separate domain-oriented services and user interface services architecturally	Not yet supported
327	The Hydra middleware should be flexible as to allow for opt-in and opt-out on parts	Supported
329	Middleware provides domain-independent services	Supported
335	Location awareness / positioning support	Partly supported
WP4		
312	UAAR: Support profiling of devices' performance	Partly supported
314	UAAR: Faults should be intercepted by middleware, notified to interested services	Partly supported
317	UUAR: Support runtime reconfiguration	Not yet supported
318	UAAR: Devices should be able to be added to the system at runtime	Supported
366	Web services should run on embedded devices	Not yet supported
479	Event prioritisation	Supported
WP5		
264	Common message protocol	Supported
276	New communication technologies	Supported
336	Discovery protocol should support multiple networks	Supported
407	Storage Manager – Gateways information stored synchronization	Not yet supported
419	Device services and resources provision through its Gateway	Supported
425	D2D communication Overlay Hydra network	Supported
445	The level of protection should be independent from the currently used low-layer protocol	Supported
455	Identity - Update of the correspondences between identifier and physical addresses	Supported
465	Networks overlapping	Not yet supported
475	Multimedia streaming in the Hydra network	Supported
476	Network Manager Configuration and Testing	Supported
WP6		
91	Any Hydra device should have an associated description	Not yet supported
101	Manual device ontology definition	Supported
108	Device discovery	Supported
110	Device Categorisation in runtime	Partly supported

111	Dynamic Web Service Binding	Supported
114	Semantic enabling of device web services	Not yet supported
122	Configurable and easy to install middleware	Not yet supported
129	Support for Semantic Web Standards for Device Communication	Supported
210	Middleware should support different architectural styles	Supported
376	Security requirements must be part of the Hydra MDA	Not yet supported
389	Service browsing in device ontology	Supported
WP7		
308	The Security Level of an existing network should be determinable	Not yet supported
468	Different levels of security must be supported	Not yet supported
472	Provide application developers with the functionality of checking tokens against a trust model	Supported
473	Support of arbitrary trust models	Supported
474	Core Hydra security mechanisms should run on embedded devices	Supported

Table 8: Summary of evaluation results

From the table it is possible to sketch the graphics on the successfulness rate for the actual validation, in terms of requirements' percentages reaching the threshold.

On the average, almost 70% of the tested requirements have been partly or completely covered, as it appears in Figure 20.

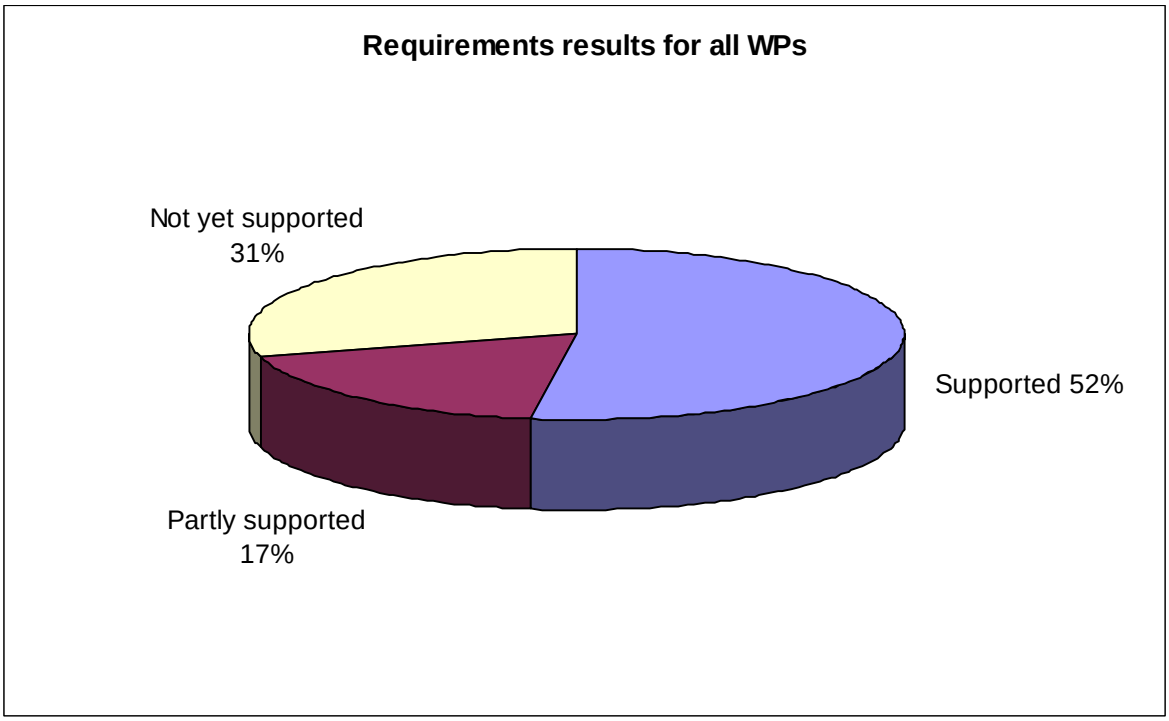
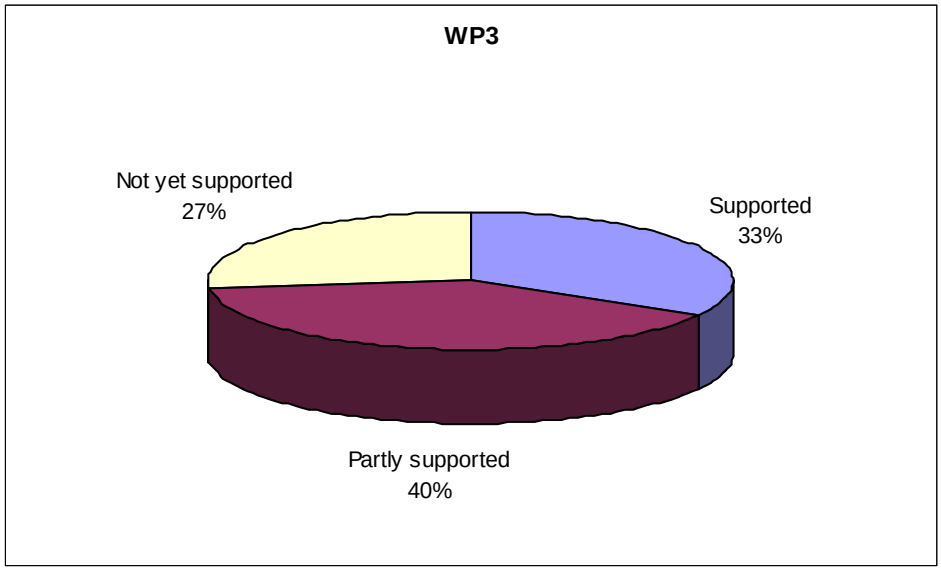
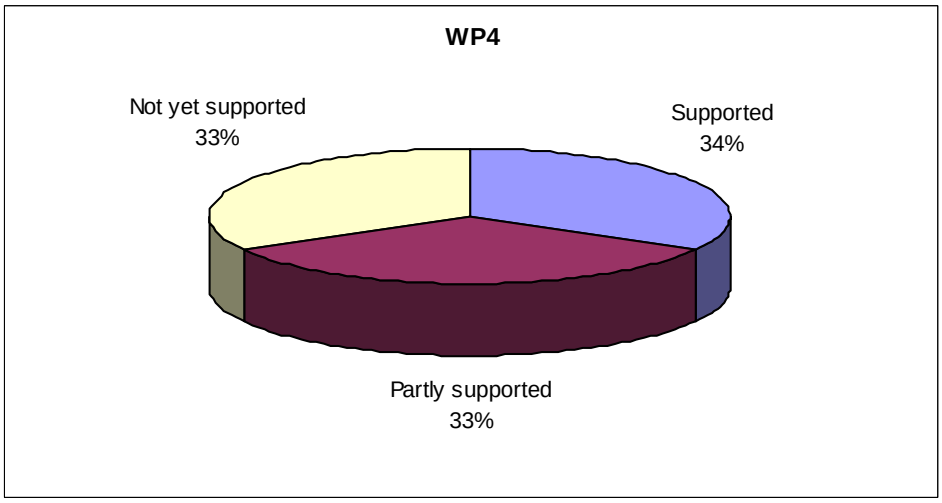
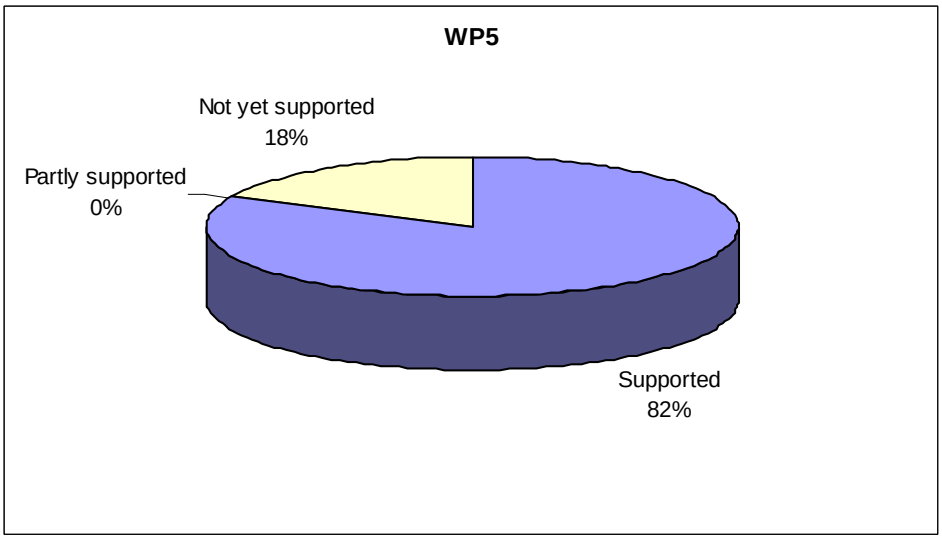


Figure 20: Overall success percentages after 1st validation cycle

In the next summarising table we present the results obtained divided per WP. The indication is not relevant in terms of quantitative aspects, but it is considered as a basic reference for the future development and validation activities to be fulfilled during the next project iteration.


WP3
Not yet supported
27%
Supported
33%
Partly supported
40%


WP4
Not yet supported
33%
Supported
34%
Partly supported
33%


WP5
Not yet supported
18%
Partly supported
0%
Supported
82%

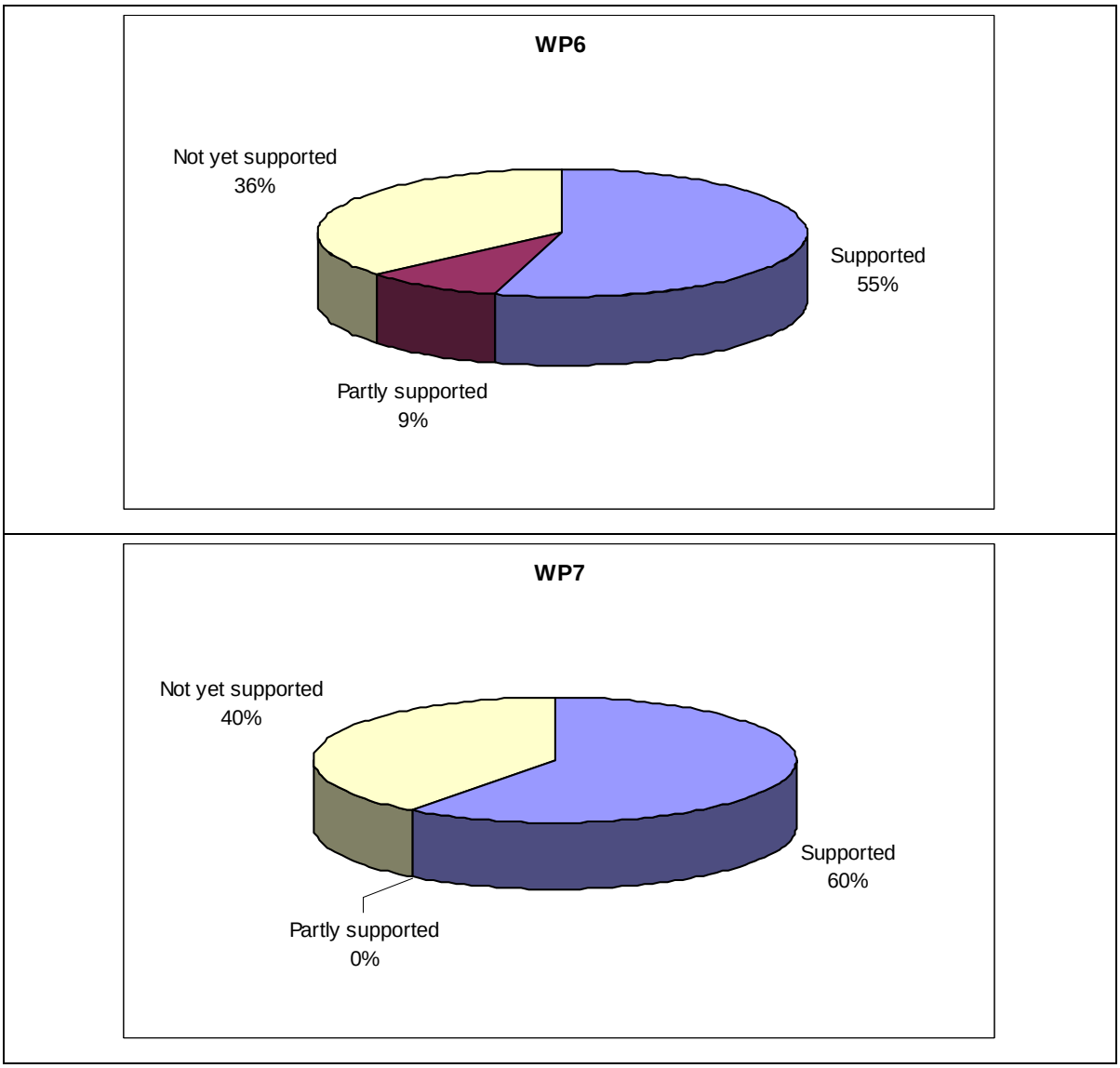


Table 9: Requirements fulfilment per WP

5. Conclusions

The validation methodology has been built and applied by the comparison between an expected impact (requirement) and how the real prototype or application behaves. The assessment procedure was applied from the (potential) Hydra user, who is a developer or a software expert able to recognise if the promised features and properties of the Hydra middleware are met. The environment selected for the validation was the software laboratory of the Hydra partners; where possible developer users not previously working in the Hydra implementation have been selected. As for this last point, some examples for the validation session can be found in the Hydra project web site (www.hydramiddleware.eu) and an interesting video regarding WP6 assessment can be found at: http://Hydra.cnet.se/downloads/Hydravalidationfull.wmv.

More in details, the validation methodology consisted in the verification that each selected requirement fit criterion has reached the threshold level, or if the requirement have been partially or not met. The selection of the requirements to be validated has been fulfilled by considering the following parameters:

- effective implementation or not of the requirements (in respect to the actual timing or status of the project)
- relevance for the overall architecture (cross related features)
- requirement type and priority

In total 48 requirements have been assessed. The results are summarised into the next table and in Figure 20.

Assessment threshold level	N. of requirements fulfilling the threshold
Supported	25 (52%)
Partly supported	8 (17%)
Not yet supported	15 (31%)

Table 10: Success rate

As a primary conclusion, the overall validation outcomes clearly show that the Hydra platform implementation is at a really positive stage and many requirements have been already met at the first SDK prototype development iteration (in total around 70% are partly or completely met). This represents a good basis from which starting the prosecution of the project activities.

As a consequence the next validation report, planned towards the end of the next iteration, will start the investigation from the work fulfilled during the preparation of this Deliverable, as it will be possible to highlight improvements and lesson learnt in respect to this first middleware evaluation. The first validation report can be considered a first step or milestone where the validation process has been tested and improved in order to have a significant basis for analysing the software components and ground the project advancement verification.

As a final observation, the validation results contribute to the project success just if the project plan foresees that all the user feedbacks are given back to the developers of the system, by continuing the iterative approach. The data emerged in the present analysis shall be distributed to the Hydra consortium (starting from each technical Work Package, but also looped back to WP2), as a mean for refining the user requirements, better detailing the project the lessons learnt and continuously improving the system characteristics.

6. References

International Standards Organisation website
http://www.iso.org/

Foglia, T. and Costa, N. (2007). *Validation Framework*. Technical Report D2.6, Hydra Consortium. EU Project IST 2005-034891

Kupries, M. and Hansen, K.M. (2007). *Updated Systems Requirements Report*. Technical Report D3.2, Hydra Consortium. EU Project IST 2005-034891

Zimmermann, A. (2008). *Updated System Architecture Report*. Technical Report D3.9, Hydra Consortium. EU Project IST 2005-034891

Hansen, K.M. (2007). *Quality Attribute Scenarios*. Technical Report D6.1, Hydra Consortium. EU Project IST 2005-034891

Wahl, T., Hoffmann, M. and Schütte, J. (2008). *Impact of architectural security implications*. Technical Report D7.6, Hydra Consortium. EU Project IST 2005-034891

Guarise, A. and De Bona, M. (2008). *Validation Plan for prototypes*. Technical Report D10.1, Hydra Consortium. EU Project IST 2005-034891