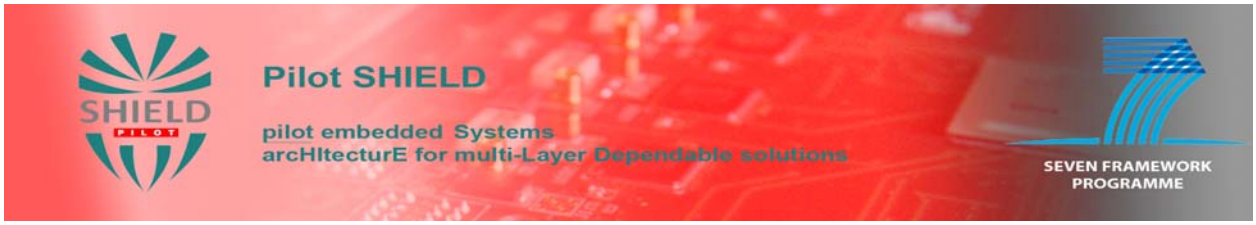


Project no: 100204

**pSHIELD**

**p**ilot embedded **S**ystems arc**HI**tectur**E** for multi-**L**ayer **D**ependable solutions

Instrument type: Capability Project

Priority name: Embedded Systems / Rail Transportation Scenarios

# System Requirements and Specifications

## For the pSHIELD-project

Deliverable D2.1.2

### Partners contributed to the work:

Ansaldo STS, Italy
Center for Wireless Innovation, Norway
Critical Software, Portugal
Elsag Datamat, Italy
Eurotech, Italy
SESM, Italy
THYIA, Slovenia
UNIROMA1, Italy (reviewer)

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

# Document Authors and Approvals

| Authors | | Date | Signature |
|---|---|---|---|
| **Name** | **Company** | | |
| Vlado Drakulovski | THYIA | | |
| Ljiljana Mijić | THYIA | | |
| Spase Drakul | THYIA | | |
| Nastja Kuzmin | THYIA | | |
| Antonio Di Marzo | SESM | | |
| Przemysław Osocha | SESM | | |
| João Cunha | SESM | | |
| Alfio Pappalardo | ASTS | | |
| Flammini Francesco | ASTS | | |
| Antonio Ruggieri | ASTS | | |
| Mohammad Mushfiqur Rahman Chowdhury | CWIN | | |
| Simone Cataldi | ED | | |
| Fabrizio Maria de Seta | ED | | |
| Andrea Morgagni | ED | | |
| Renato Baldelli | ED | | |
| Paolo Azzoni | ETH | | |
| **Reviewed by** | | | |
| **Name** | **Company** | | |
| Francesco Delli Priscoli | UNIROMA1 | | |
| Andrea Fiaschetti | UNIROMA1 | | |
| Claudio De Persis | UNIROMA1 | | |
| Vincenzo Suraci | UNIROMA1 | | |
| Andi Palo | UNIROMA1 | | |
| **Approved by** | | | |
| **Name** | **Company** | | |
| Josef Noll | MAS | | |

# Modification History

| Issue | Date | Description |
|---|---|---|
| **Draft A** | 15.09.2010 | First issue for comments |
| **Issue 1** | 30.10.2010 | Incorporates comments from Draft A review |
| **Issue 2** | 15.12.2010 | Incorporates comments from issue 1 review |
| **Issue 3** | 09.02.2011 | Incorporates comments from issue 2 review |
| **Issue 4** | 15.03.2011 | Incorporates comments from issue 3 review and corrections from THYIA after first review of 2.3.1 Preliminary System Architecture Design |
| **Issue 5** | 24.06.2011 | Incorporates comments from issue 4 review |

# Contents

# Figures



# Tables

There are no tables in the document.

# Glossary

| | |
|---|---|
| AFR | Automatic Firmware Recovery |
| ECC | Elliptic Curve Cryptography |
| ESs | Embedded Systems |
| HLR | High Level Requirements |
| IPv4 | Internet Protocol version 4 |
| IPv6 | Internet Protocol version 6 |
| IPSec | Internet Protocol Security |
| LPC | Low Pin Count, TPM bus interface |
| MRS | Main Requirements Specification |
| QoS | Quality of Service |
| SNMP | Simple Network Management Protocol |
| SOA | Service-Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| SPD | Security Privacy Dependability |
| SPDT | Security Privacy Dependability Trust |
| SRS | System Requirements Specification |
| TPM | Trust Platform Module |
| XML | Extensible Markup Language |
| WSDL | Web Services Description Language |
| W3C | World Wide Web Consortium |

This Page is intentionally left blank

# 1 Executive Summary

The scope of the present document is to provide top-level requirements extracted from the pSHIELD scenario defined in the internal deliverable and from the Technical Annex approved by the Artemis JU[1]. The System Requirements and Specification (SRS) document covers all requirements on the overall pSHIELD system.

The methodology for specifying the requirements based on a common agreement described in deliverables ID2.1 titled “Requirements and Specifications Definitions and Rules – Quality Manual for WP2”.

In chapter 6 a set of high level requirements (HLR) for the selected scenario are defined. Section 7 is providing SPD HLR for the pSHIELD system. These means, the system requirements can be considered as requirements that define the system before it will be specified in the deliverable D2.3.1, where the whole pSHIELD system architecture will be defined. This document covers all requirements on the pSHILED system functionalities. Special focus is given on SPD (security, privacy, dependability), and composability. Therefore, for each SPD technology and for each functional layer, a formal set of high level requirements for the functional architecture are defined. Chapter 8, 9, 10 and 11 covers requirements and specifications for the node, network, middleware and overlay functional layers. In Chapter 12 the conclusions are summarized.

In this document the concepts of Security, Privacy and Dependability are formalized via a reference taxonomy that allows describing them with six different attributes. The formal definition of each attribute is a fundamental step towards the correct finalization of requirements and specification. This will be the input for the other WPs where the pSHIELD system will be designed with additional details.

The description of the scenario, the overall system and its four fundamental functional layers (i.e., node, network, middleware and overlay) requires also a careful consideration of the elements that are targeted for pSHIELD.

The final requirements specification will be refined on the basis of the results of the validation phase and on the detailed description of the application scenario from task 6.4

[1] Technical Annex of pSHIELD project, 12.11.2010.

# 2 Introduction

This deliverable is the main output of Task 2.1 from WP2. The role of the task is to identify the requirements and provides inputs for the specifications of the overall pSHIELD system that will be performed by task 2.3. Since the project is a “pilot”, its results will be mostly tailored on the application scenario: for that reason this task will be strongly influenced by the selected application scenario.

Requirements and specification have been be also influenced by the liaisons activated in WP1.

For each layer a formal set of high level, architectural and interface requirements will be identified. The *rail transportation scenario* [9] will be taken as a reference for defining the SPD requirements of each architectural layer; however the conceived architecture will be able to support any Embedded System (ES) scenario, as well as of the overall system with reduced and clearly identifiable tailoring effort.

An iterative approach will be adopted. A preliminary set of requirements and specification will be provided in this phase of the project. The preliminary outcome of this task will be used by WP3, WP4 and WP5 to develop potential prototypes and by WP6 to validate them. The requirements and specification will be refined on the basis of the results of the validation phase and on the detailed description of the application scenarios from Task 6.4.

The application scenario of reference for the task 2.1 of pSHIELD project is the ***monitoring of freight trains transporting hazardous material***. The detection of abnormal operating or environmental conditions on board of vehicles represents an example application of great interest for the freight train monitoring. In particular, in this use case, the following requirements have to be fulfilled:

- Secure handling of the critical information of the hazardous material;
- Secure and dependable monitoring of the hazardous material.

On the bases of the real-worlds SPD requirements for the specific application, SPD specifications can be defined.

This Deliverable D2.1.1 aims at giving precise definitions characterizing the various concepts that come into play when addressing the security, dependability and privacy of complex systems resulting from the composition of elementary Embedded Systems.

Before going further, it is important to clarify that the pSHIELD project is only a preliminary investigation of the *SHIELD Framework* concepts and possibilities (that subsequent projects are in charge of exploring more in detail). For that reason most of the results obtained in this phase will be natively tailored on the selected scenario, but this doesn’t mean that the potentiality of the framework are limited to it. The reusability of the SHIELD solution with a minimum tailoring effort is indeed one of its main features.

This will be reflected also in the formalization of System Requirements that will be divided in two sets.

- The first set is about System Requirements specific for the rail transportation scenario (i.e. requirements that are valid only in the scope of the application);
- The second set is about general System Requirements that characterize the SHIELD framework independently from the application

# 3 Terms and Definitions

## 3.1 The pSHIELD System: General Definitions

**[pSHIELD System]** - The pSHIELD system (a whole composed by several parts) is a set of interacting and/or interdependent system components forming an integrated and more complex system.

The main characteristics of the pSHIELD system are:

1) The *architecture* defined by components and the results of their composition,
2) The *behavior*, that involves collecting inputs, processing them and producing outputs,
3) The *relations* that the various parts of the system have between each other, both functional and structural
4) The *functionalities* or group of functionalities that the system offers and/or realises.

The pSHIELD system aims to guarantee the following attributes:

- Security,
- Privacy,
- Dependability

For itself and for the application scenario on which it is applied. These attributes are indeed the main goals to be addressed for the new generation Embedded Systems (see next definition).

The pSHIELD system is organized according to the following layering functional architecture:

I. Node Layer: includes the hardware components that constitute the physical part of the system
II. Network Layer: includes the communication technologies (specific for the rail transportation scenarios) that allow the data exchange among pSHIELD components, as well as the external world. These communication technologies, as well as the networks to which pSHIELD is interconnected can be (and usually are) heterogeneous.
III. Middleware Layer: includes the software functionalities that enable the discovery, composition and execution of the basic services necessary to guarantee SPD as well as to perform the tasks assigned to the system (for example, in the railway scenario, the monitoring functionality)
IV. Overlay Layer : the “embedded intelligence” that drives the composition of the pSHIELD components in order to meet the desired level of SPD. This is a software layer as well.

**[Component/Sub-system]** - A component or sub-system is a smaller, self-contained part of a system. In particular for the pSHIELD system the “interacting components” are Embedded Systems.

[Embedded System/Device] - The Embedded System (or Device) is an electronic system (or device) dedicated to a specific and reduced set of functionalities. It could be an integrated circuit that has input, output and processing capabilities or more commonly it is a small programmable chip. The embedded systems are controlled by one or more main processing cores that are typically either microcontrollers (PIC) or digital signal processors (DSP).

**[Asset categories] -** An asset can be grouped in two categories: logical and physical assets. Information, services and software are logical assets, whilst human beings, hardware or particular physical objects are physical assets.

**[User] –** User is any entity internal or external, human or IT that interacts with the pSHIELD system.

## 3.2 The pSHIELD System: Application-Oriented Definitions

**[Information] –** Information is measured data, real-time streams from audio/video-surveillance devices, smart-sensors, alarms, etc.

**[pSHIELD Asset] -** The pSHIELD assets are information and services.

**[SPD Audit]** - SPD auditing involves recognizing, recording, storing, and analyzing information related to SPD relevant activities (i.e. activities controlled by the TSF). The resulting audit records can be examined to determine which SPD relevant activities took place and whom (which user) is responsible for them [8].

**[Non-repudiation]** - Assuring the identity of a party participating in a data exchange.

**[Access control]** - Is the process of mediating every request of access to pSHIELD assets determining whether the request should be granted or denied according to the security policies established.

**[Identification]** - Determining the identity of users.

**[Authentication]** - Verifying the identity of users and determining their authority to interact with the system.

**[Trusted channel]** - A communication channel that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

**[Software Failures]** – Software failures include crashes, incompatibilities, computation errors, etc.

**[Hardware Failures]** – Hardware Failures include transient faults due to radiation, overheating or power spikes.

**[Transmission Failures]** – Transmission Failures include:

- Repetition (a message is received more than once)
- Deletion (a message is removed from a message stream)
- Insertion (a new message is implanted in the message stream)
- Re-sequencing (messages are received in an unexpected sequence)
- Corruption (the information contained in a message is changed, casually or not)
- Delay (messages are received at a time later than intended)
- Masquerade (a non-authentic message is designed thus to appear to be authentic)

**[Failure Mitigation Mechanisms]** – Failure Mitigation Mechanisms includes:

- hardware redundancy and diversity
- firewall and intrusion detection systems
- self checking and diagnostics routines
- message sequence numbers
- data checksums
- shared or public key cryptography
- vitality checks through watchdog timers
- software rejuvenation

**[Reasoning]** – Reasoning is related to finding one or more solutions (HW/SW configuration) that satisfy the desired SPD level.

**[Composition]** – Composition is related to verifying the possibility of composing the individual elements and composing them logically

**[Configuration]** – Configuration is the translation of logical configuration into a physical configuration.

# 4 The Methodology for pSHIELD System Requirements Specification

The pSHIELD requirements elicitation, as well as the overall work carried out in Task 2.1, is structured according to a number of steps: each step is hereinafter presented following a question/answer approach.

### *Step 1 – What is the pSHIELD System?*

The pSHIELD System is a set of interacting and interconnected Embedded Systems with specific composability and SPD Functionalities.

Reference in this document: Chapter 3 – Terms and Definitions and Chapter 7-11 – System Requirements

### *Step 2 – What is the role of the pSHIELD System?*

pSHIELD aims at assuring SPD for a certain asset or goods in a specific scenario. In the following, for convenience, we will replace the expression “assuring SPD” with the generic expression “protecting”, even if its real meaning is different. In **Błąd! Nie można odnaleźć źródła odwołania.** this concept is represented: the green boxes represent the interconnected ESs and the gray box is the addressed asset/goods (the SPD functionalities are still missing from this graphical representation because they are identified in the following steps).

Moreover, since the terms SPD seem too general, a set of specific attributes will be introduced to better specify the meaning of “Assuring SPD” (for example assuring integrity, reliability, confidentiality, and so on). These attributes are specified in the SPD Taxonomy.

Reference in this document: Section 5 – Reference SPD taxonomy

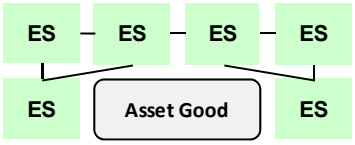


**Figure 4-1: pSHIELD System applied to a specific asset/good.**

### *Step 3 – What are the asset/goods and the scenario addressed by the System?*

The selected scenario is railways transportation and the asset/goods is the secure and dependable monitoring of freight trains transporting hazardous materials. Moreover, since pSHIELD should protect itself, it is an asset/good as well.

Reference in this document: Chapter 6 – High level requirements for Scenario

### *Step 4 – What are the possible menace and/or attack that could affect the protected asset/goods*

Once the assets and goods, as well as the application scenarios are clearly identified (from Technical Annex for example), it is easy to enumerate all the possible menaces and attacks that could affect the level of Security, Privacy and Dependability of the system. The output of this activity is a fundamental input for next step. The logical step is represented in **Błąd! Nie można odnaleźć źródła odwołania.**.

Reference in this document: Chapter 6 – High level requirements for Scenario

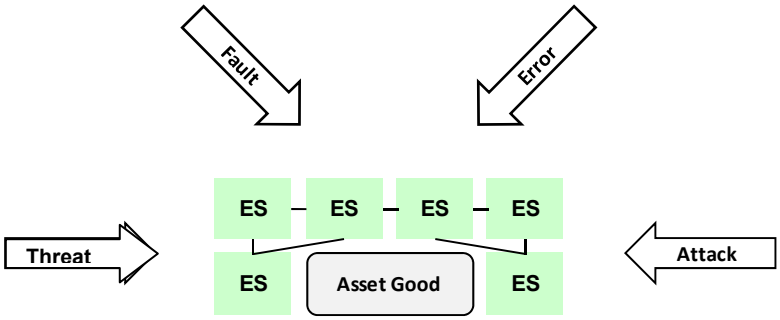


**Figure 4-2: pSHIELD attacks and menaces.**

### *Step 5 – What are the SPD functionalities that can prevent or minimize the effect of the previously identified menace and/or attack?*

Starting from the identified menaces and attacks, a set of SPD Functionalities is identified that are able to prevent or mitigate them (see **Błąd! Nie można odnaleźć źródła odwołania.**). The functionalities are translated in a set of Functional Requirements that are at the basis of the pSHIELD framework. Since these requirements are strictly related to the scenario, they will be listed as Scenario's Requirements.

Reference in this document: Chapter 6 – High level requirements for Scenario

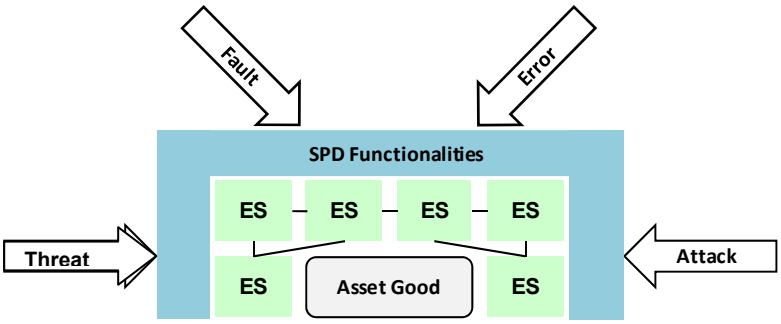


**Figure 4-3: pSHIELD system with SPD functionalities enabled.**

### *Step 6 – What are the features of the system that allow to realize the SPD functionalities?*

Once the required SPD Functionalities are captured (in Step 5), in Step 6 a set of System Requirements can be identified allowing to realize these SPD functionalities. These requirements provide the guidelines for the design and development of the four pSHIELD layers: so, these requirements are divided into four categories corresponding to the four different layers.

Reference in this document: Chapter 7-11 – System Requirements

The result of these six steps is the definition, formalization and translation into requirements of the pSHIELD system.

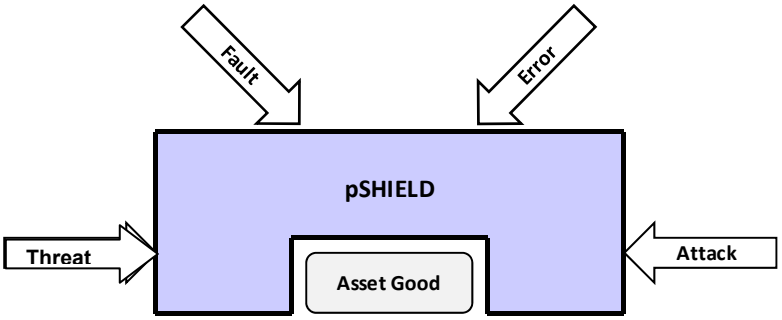


**Figure 4-4: ESs + SPD Functionalities = pSHIELD system.**

# 5 Reference SPD Taxonomy

## 5.1 Security, Privacy and Dependability: from concepts to attributes

Privacy is almost a stand alone concept with no specific attributes. As far as the “privacy” concept is concerned, the definition for privacy arrives from Latin: privatus "separated from the rest, deprived of something, esp. office, participation in the government", from privo "to deprive". It means the ability of an individual or group to seclude themselves or information about themselves and thereby reveal themselves selectively. The boundaries and content of what is considered private differ among cultures and individuals, but share basic common themes. Privacy is sometimes related to anonymity, the wish to remain unnoticed or unidentified in the public realm. When something is private to a person, it usually means there is something within them that is considered inherently special or personally sensitive. The degree to which private information is exposed therefore depends on how the public will receive this information, which differs between places and over time. Privacy is broader than security and includes the concepts of appropriate use and protection of information. Therefore, the pSHIELD scenario should define clearly how privacy enters in the transportation of dangerous materials: in case no specific privacy issues will be identified, then the concept of privacy will be covered by the most adequate “confidentialitiy” (see following section).

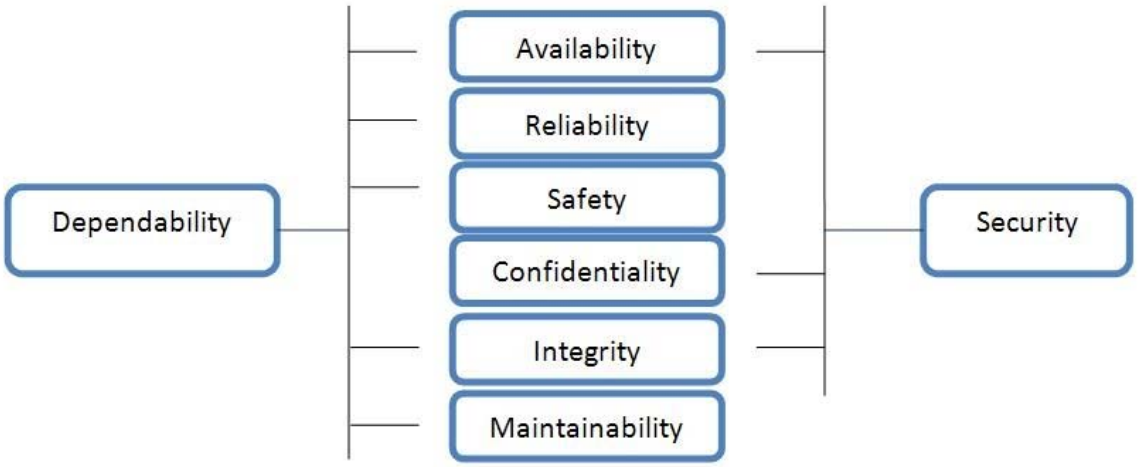


**Figure 5.1 Dependability and security attributes.**

Security and Dependability are two important and complex concepts. In order to better understand and describe them, they will be explored trough their attributes.

**Dependability** is a composite concept that encompasses the following ***attributes***:

- Availability: readiness for correct service.
- Reliability: continuity of correct service.
- Safety: absence of catastrophic consequences on the user(s) and the environment.
- Integrity: absence of improper system alterations.
- Maintainability: ability to undergo modifications and repairs.

When addressing the **security** concept, an additional attribute has great prominence, confidentiality, i.e., the absence of unauthorized disclosure of information. Security is a composite of the attributes of confidentiality, integrity (in the security context, “improper” means “unauthorized"), and availability (for authorized actions only).

Figure 5.1 summarizes the relationship between dependability and security in terms of their main attributes. The picture should not be interpreted as indicating that, for example, security developers have no interest in maintainability, or that there has been no research at all in the dependability field related to confidentiality—rather it indicates where the main balance of interest and activity lies in each case.

The dependability and security specification of a system must include the requirements for the attributes in terms of the acceptable frequency and severity of service failures for specified classes of faults and a given use environment. One or more attributes may not be required at all for a given system.

Below, we provide a short description of the common attributes for security and dependability. This description is helpful for clarifying importance of distinction between these common attributes when they are more related to security and less to dependability and vice versa.

**A general view on integrity and availability**

Integrity and availability are two competing dependability/security attributes. While some applications require strict integrity, other applications exist, e.g., safety or mission critical systems, where - depending on the specific situation - availability is more important for dependability than strict integrity. Within our work, we focus on data-centric systems, where availability can be increased by temporarily relaxing data integrity, thereby allowing for certain inconsistencies. Potential inconsistencies are accepted based on constraint validation on replicated copies that are possibly stale in the face of network partitions. Such consistency threats need to be bound and eventually resolved during reconciliation to re-establish a consistent system state.

**Integrity related to Security**

The **integrity** of data means its accuracy and completeness. Data have integrity if they have not been corrupted in any way.

In information security, **integrity** means that data cannot be modified undetectably. Integrity is violated when a message is actively modified in transit. Most cipher systems provide message integrity along with privacy as part of the encryption process. Messages that have been tampered with in flight will not be successfully decrypted.

**Integrity** means assurance that the information is authentic and complete. Ensuring that information can be relied upon to be sufficiently accurate for its purpose. The term Integrity is frequently used when considering Information Security, as it represents one of the primary indicators of security (or lack of it). The integrity of data is not only whether the data is 'correct', but whether it can be trusted and relied upon. For example, making copies (say by e-mailing a file) of a sensitive document, threatens both confidentiality and the integrity of the information. Why? Because, by making one or more copies, the data is then at risk of change or modification.

**Integrity related to Dependability**

Dependability of a structural system is a comprehensive concept that - by definition - describes the quality of the system as its ability to perform as expected in a way that can be justifiably trusted. One of the attributes of dependability is integrity, which can be interpreted as the absence of improper alterations of

the structural configuration. The assessment of the integrity during the whole life-cycle can be carried out efficiently by implementing a monitoring system able to detect and diagnose any fault at its onset.

**Availability related to Security**

**Availability** is the area of information security that requires services and components to be continuously available for the user community. If a service or component is unavailable, confidentiality and integrity are meaningless. Network availability is the underlining attribute that must be existent in order to guarantee that services are accessible for end users.

For any information system to serve its purpose, the information must be available when it is needed. This means that the computing systems used to store and process the information, the security controls used to protect it, and the communication channels used to access it, must be correctly functioning. High availability systems aim to remain available at all times, preventing service disruptions due to power outages, hardware failures, and system upgrades. **Ensuring availability also involves preventing denial-of-service attacks**.

**Availability related to Dependability**

**Availability** refers to the accessibility of the system to users. A system is available if its users' requests for service are accepted at the time of their submission. Unlike reliability, availability is instantaneous. The former focuses on the duration of time a system is expected to remain in continuous operation - or effectively so in the case of recovery-enhanced reliability - starting in a normal state of operation. The latter concentrates on the fraction of time instants where the system is operational in the sense of being accessible to the end user.

Based on the above considerations, it is clear that in some cases integrity and availability are more security oriented, whilst in other cases these two attributes are more related to dependability. So, in the former cases, these attributes will be dealt with in the security section, whilst, in the latter cases, in the dependability section.

## 5.2 Faults, Errors and Failures

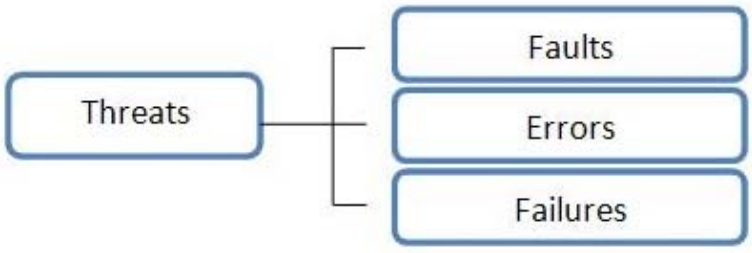


**Figure 5-2: Threats classification.**

At this point we will describe the generic threats that could affect a system.

**Threats** include *faults*, *errors* and *failures*, as well as their causes, consequences and characteristics. An *error* is defined as the part of a system's total state that may lead to a failure. A *service failure* occurs when an error causes the delivered service to deviate from correct service. The cause of an error is called a *fault*: a fault may arise from physical imperfections in the system, physical influence and damage from

the outside, human logical faults made during specification, design, development, installation and operation, etc.

### 5.2.1 Faults – a taxonomy

All faults that may affect a system during its life are classified according to eight basic viewpoints, leading to the elementary fault classes, as shown in Figure 5.3.

If all combinations of the eight elementary fault classes were possible, there would be 256 different combined fault classes. However, not all criteria are applicable to all fault classes; for example, natural faults cannot be classified by objective, intent and capability.

The combined fault classes belong to three major partially overlapping groupings:

- development faults that include all fault classes occurring during development,
- physical faults that include all fault classes that affect hardware,
- interaction faults that include all external faults.

Knowledge of all possible fault classes allows the user to decide which classes should be included in a dependability and security specification.

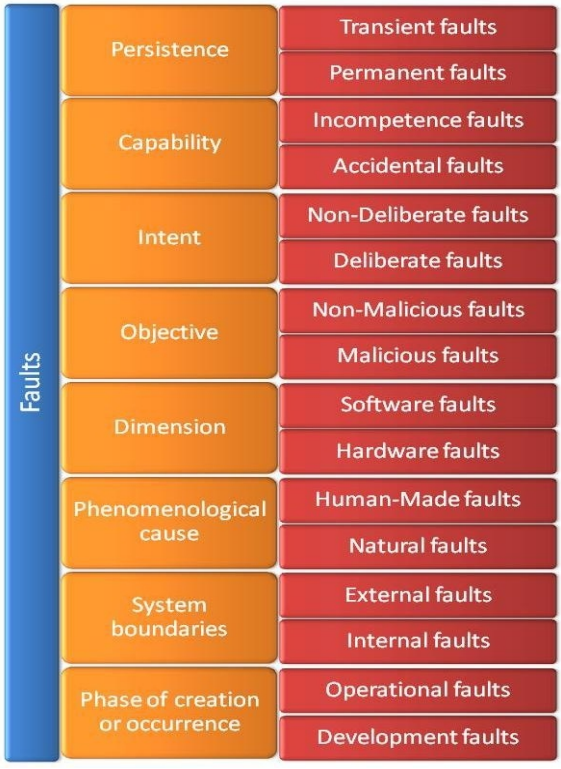


**Figure 5.3 The elementary fault classes.**

Particular attention will be devoted to malicious human-made faults. They are introduced with the malicious objective to alter the functioning of the system during it use. The goals of such faults are:

1) to disrupt or halt service, causing denials of service;
2) to access confidential information; or
3) to improperly modify the system.

They are grouped into two classes:

1. Malicious logic faults that encompass development faults such as Trojan horses, logic or timing bombs, and trapdoors, as well as operational faults such as viruses, worms, or zombies. Definitions for these faults **Błąd! Nie można odnaleźć źródła odwołania.**are given in [1],[2].
2. Intrusion attempts that are operational external faults. The external character of intrusion attempts does not exclude the possibility that they may be performed by system operators or administrators who are exceeding their rights, and intrusion attempts may use physical means to cause faults: power fluctuation, radiation, wire-tapping, heating/cooling, etc.

***Logic bomb:*** *malicious logic that remains dormant in the host system till a certain time or an event occurs, or certain conditions, are met, and then deletes files, slows down or crashes the host system etc.*
***Trojan horse:*** *malicious logic performing, or able to perform an illegitimate action while giving the impression of being legitimate; the illegitimate action can be the disclosure or modification of information (attack against confidentiality or integrity) or a logic bomb;*
***Trapdoor:*** *malicious logic that provides a means of circumventing access control mechanisms;*
***Virus:*** *malicious logic that replicates itself and joins another program when it is executed, thereby turning into a Trojan horse; a virus can carry a logic bomb;*
***Worm:*** *malicious logic that replicates itself and propagates without the users being aware of it; a worm can also carry a logic bomb;*
***Zombie:*** *malicious logic that can be triggered by an attacker in order to mount a coordinated attack.*

**Figure 5.4 Example of Malicious logic faults**

### 5.2.2 Failures – a taxonomy

A service failure is defined as an event that occurs when the delivered service deviates from correct service. The different ways in which the deviation is manifested are a system's service failure modes. Each mode can have more than one service failure severity. The service failure modes characterize incorrect service according to four viewpoints:

1. the failure domain,
2. the detectability of failures,
3. the consistency of failures, and
4. the consequences of failures on the environment.

The **failure domain** viewpoint leads us to distinguish:

- **Content failures**. The content of the information delivered at the service interface (i.e., the service content) deviates from implementing the system function.
- **Timing failures**. The time of arrival or the duration of the information delivered at the service interface (i.e., the timing of service delivery) deviates from implementing the system function.

These definitions can be specialized:

1) the content can be in numerical or non numerical sets (e.g., alphabets, graphics, colors, sounds), and
2) a timing failure may be early or late, depending on whether the service is delivered too early or too late.

Failures when both information and timing are incorrect fall into two classes:

- **halt failure**, or simply halt, when the service is halted (the external state becomes constant, i.e., system activity, if there is any, it is no longer perceptible to the users); a special case of halt is silent failure, or simply silence, when no service at all is delivered at the service interface (e.g., no messages are sent in a distributed system).
- **Erratic failures** otherwise, i.e., when a service is delivered (not halted), but it is erratic (e.g., babbling).

The **detectability** viewpoint addresses the signaling of service failures to the user(s). Signaling at the service interface originates from detecting mechanisms in the system that check the correctness of the delivered service. When the losses are detected and signaled by a warning signal, then **signaled failures** occur. Otherwise, they are **unsignaled failures**. The detecting mechanisms themselves have two failure modes:

1) signaling a loss of function when no failure has actually occurred, that is a **false alarm**,
2) not signaling a function loss, that is an unsignaled failure.

When the occurrence of service failures result in reduced modes of service, the system signals a degraded mode of service to the user(s). Degraded modes may range from minor reductions to emergency service and safe shutdown.

The **consistency** of failures leads us to distinguish, when a system has two or more users:

- **consistent failures**. The incorrect service is perceived identically by all system users.
- **inconsistent failures**. Some or all system users perceive differently incorrect service (some users may actually perceive correct service); inconsistent failures are usually called, after **Błąd! Nie można odnaleźć źródła odwołania.**, **Byzantine failures**.

### 5.2.3 Errors – a taxonomy

An error has been defined as the part of a system's total state that may lead to a failure—a failure occurs when the error causes the delivered service to deviate from correct service. The cause of the error has been called a fault.

An error is **detected** if its presence is indicated by an error message or error signal. Errors that are present but not detected are **latent** errors.

Since a system consists of a set of interacting components, the total state is the set of its component states. The definition implies that a fault originally causes an error within the state of one (or more)

components, but service failure will not occur as long as the external state of that component is not part of the external state of the system. Whenever the error becomes a part of the external state of the component, a service failure of that component occurs, but the error remains internal to the entire system.

Whether or not an error will actually lead to a service failure depends on two factors:

1. The structure of the system, and especially the nature of any redundancy that exists in it:

    - protective redundancy, introduced to provide fault tolerance that is explicitly intended to prevent an error from leading to service failure.
    - unintentional redundancy (it is in practice difficult if not impossible to build a system without any form of redundancy) that may have the same—presumably unexpected—result as intentional redundancy.

2. The behavior of the system: the part of the state that contains an error may never be needed for service, or an error may be eliminated (e.g., when overwritten) before it leads to a failure.

### 5.2.4 The Pathology of Failure: Relationship between Faults, Errors, and Failures

The creation and manifestation mechanisms of faults, errors, and failures are summarized as follows:

1. A fault is *active* when it produces an error; otherwise, it is *dormant*. An active fault is either

    1) an internal fault that was previously dormant and has been activated by the computation process or environmental conditions, or
    2) an external fault. **Fault activation** is the application of an input (the activation pattern) to a component that causes a dormant fault to become active. Most internal faults cycle between their dormant and active states.

2. Error propagation within a given component (i.e., *internal* propagation) is caused by the computation process: An error is successively transformed into other errors. Error propagation from component A to component B that receives service from A (i.e., *external* propagation) occurs when, through internal propagation, an error reaches the service interface of component A. At this time, service delivered by A to B becomes incorrect, and the ensuing service failure of A appears as an external fault to B and propagates the error into B via its use interface.

3. A service failure occurs when an error is propagated to the service interface and causes the service delivered by the system to deviate from the correct service. The failure of a component causes a permanent or transient fault in the system that contains the component. Service failure of a system causes a permanent or transient external fault for the other system(s) that receive service from the given system.

## 5.3 Means to attain Dependability and Security

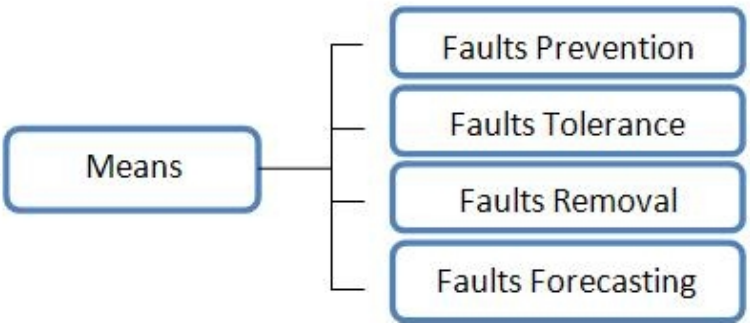


**Figure 5.5 Means to attain Dependability and Security**

Many *means* have been developed to attain the various attributes of dependability and security. Those means can be grouped into four major categories:

- *Fault prevention means* to prevent the occurrence or introduction of faults.
- *Fault tolerance means* to avoid service failures in the presence of faults.
- *Fault removal means* to reduce the number and severity of faults.
- *Fault forecasting means* to estimate the present number, the future incidence, and the likely consequences of faults.

Means like testing and evaluation by modeling and mathematical analysis or simulation, are also needed for the cost-efficient design and dimensioning of systems with respect to their required attributes and to validate and reach confidence in the result.

**The development phase** includes all activities from presentation of the user's initial concept to the decision that the system has passed all acceptance tests and is ready to deliver service in its user's environment. During the development phase, the system interacts with the development environment and development faults may be introduced into the system by the environment. The development environment of a system consists of the following elements:

1. the physical world with its natural phenomena,
2. human developers, some possibly lacking competence or having malicious objectives,
3. development tools: software and hardware used by the developers to assist them in the development process,
4. production and test facilities.

**The use phase** of a system's life begins when the system is accepted for use and starts the delivery of its services to the users. Use consists of alternating periods of correct service delivery (to be called **service delivery**), service outage, and service shutdown. A service outage is caused by a service failure. It is the period when incorrect service (including no service at all) is delivered at the service interface. A **service shutdown** is an intentional halt of service by an authorized entity.

**Maintenance** actions may take place during all three periods of the use phase.

During the use phase, the system interacts with its use environment and may be adversely affected by faults originating in it. The use environment consists of the following elements:

1. the physical world with its natural phenomena;

2. administrators (including maintainers): entities (humans or other systems) that have the authority to manage, modify, repair and use the system; some authorized humans may lack competence or have malicious objectives;
3. users: entities (humans or other systems) that receive service from the system at their use interfaces;
4. providers: entities (humans or other systems) that deliver services to the system at its use interfaces;
5. the infrastructure: entities that provide specialized services to the system, such as information sources (e.g., time, GPS, etc.), communication links, power sources, cooling airflow, etc.
6. intruders: malicious entities (humans and other systems) that attempt to exceed any authority they might have and alter service or halt it, alter the system's functionality or performance, or to access confidential information. Examples include hackers, vandals, corrupt insiders, agents of hostile governments or organizations, and malicious software.

As used here, the term maintenance, following common usage, includes not only repairs, but also all modifications of the system that take place during the use phase of system`s life. Therefore, maintenance is a development process, and the preceding discussion of development applies to maintenance as well.

In the scope of the project, pSHIELD could be considered as the system in charge of the Maintenance of SPD properties. This is summarized in Figure 5.6.

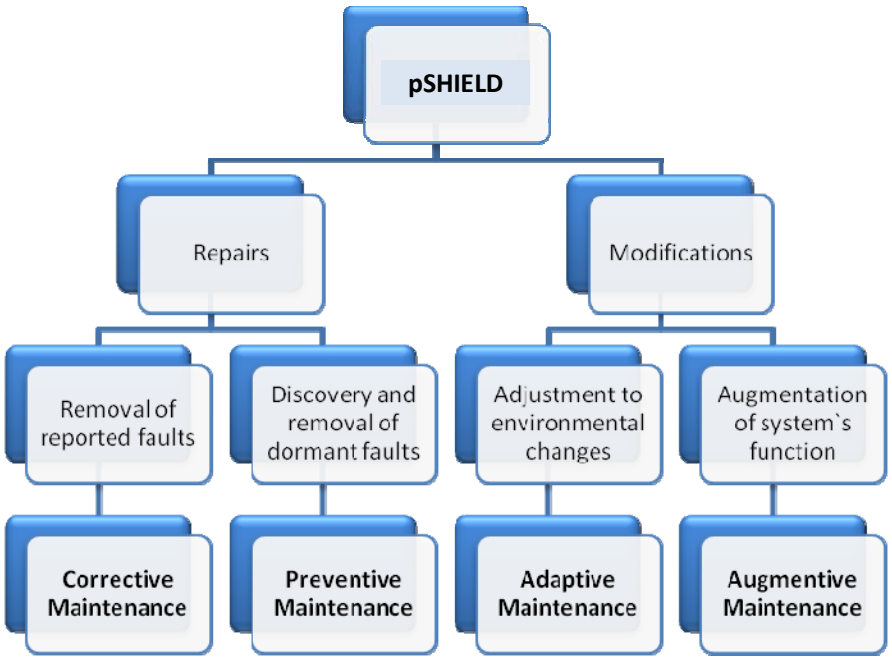


**Figure 5.6 - pSHIELD actions to preserve SPD**

It is noteworthy that repair and fault tolerance are related concepts; the distinction between fault tolerance and maintenance in this paper is that maintenance involves the participation of an external agent, e.g., a repairman, test equipment, remote reloading of software. Furthermore, repair is part of fault removal (during the use phase), and fault forecasting usually considers repair situations. In fact, repair can be seen as a fault tolerance activity within a larger system that includes the system being repaired and the people and other systems that perform such repairs.

# 6 High Level Requirements for Scenario

## 6.1 Functional Requirements

{REQ_D2.1_0301.A Application scenario – Information availability

Information shall be provided continuously according to soft or real-time constraints.

}

{REQ_D2.1_0302.A Application scenario – Information integrity

Sensed, stored or transmitted data shall not be corrupted accidentally or in a malicious way.

}

{REQ_D2.1_0303.A Application scenario – Information privacy

Information shall be accessed only by authorized users.

}

{REQ_D2.1_0304.A Application scenario – ESs integration/expansion

Whenever any new ES needs to be integrated into the system, that should be possible in a straightfoward way.

}

{REQ_D2.1_0305.A Application scenario – ESs integration/expansion

Whenever any new ES needs to be integrated into the system, it should be possible to easily evaluate the impact of the modification on the overall system dependability.

}

{REQ_D2.1_0306.A Application scenario – SPD parameters assurance/evaluation

The holistic assurance and evaluation of SPD parameters (e.g. for assessment/certification purposes) shall be possible in a straighforward way.

}

{REQ_D2.1_0307.A Application scenario – Mechanisms for failure mitigation

The pSHIELD system shall provide mechanisms to mitigate the effects on the system of the following logical threats: software failures, hardware failures, transmission failures.

}

{REQ_D2.1_0308.A Application scenario – Robust Mechanisms for failure mitigation

The pSHIELD system shall provide robust mechanisms for failures mitigation.

}

## 6.2 Structural Requirements

{REQ_D2.1_20001.A System and network homogeneity

The pSHIELD system shall be composed of at least one homogenous network infrastructure that allows communications, monitoring and control functionality in the transportation of dangerous materials. If more than one network infrastructure, then they could be heterogeneous.

}

{REQ_D2.1_20002.A System and network heterogeneity

The pSHIELD system shall be composed of at least two different homogenous network infrastructures that allow communications, monitoring and control functionality in the transportation of dangerous materials.

}

{REQ_D2.1_20003.A Central control unit

The pSHIELD system shall have a central control unit.

Traceability: {REQ_D2.1_20001.A, {REQ_D2.1_20002.A

}

{REQ_D2.1_20004.A Energy management

The pSHIELD system should support batteries and/or low size power generators.

Traceability: {REQ_D2.1_20001.A

}

{REQ_D2.1_20005.A Sensors

The pSHIELD system shall support the usage of smart wireless sensors.

Traceability: {REQ_D2.1_20001.A, {REQ_D2.1_20002.A

}

{REQ_D2.1_20006.A Gateway node

The pSHIELD system shall have a gateway node.

Traceability: {REQ_D2.1_20001.A, REQ_D2.1_20005.A

}

{REQ_D2.1_20007.A Satellite positioning antenna

The pSHIELD system shall have a satellite positioning antenna.

Traceability: {REQ_D2.1_20001.A, REQ_D2.1_20005.A

}

{REQ_D2.1_20008.A Heterogeneous communications support

The pSHIELD system should support at least one communication standards from GSM, GPRS, UMTS, EDGE, ZigBee, IEEE 802.11x or other WAN/WLAN communications.

Traceability: {REQ_D2.1_20001.A, REQ_D2.1_20005.A

}

{REQ_D2.1_20009.A Interoperability in heterogeneous network environment

The pSHIELD system shall guarantee a certain degree of interoperability for the communication technologies selected,

Traceability: {REQ_D2.1_20001.A, REQ_D2.1_20005.A , {REQ_D2.1_20008.A

}

## 6.3 pSHIELD Monitoring Applications

{REQ_D2.1_06001.A Reliable communication link

Monitoring application shall have reliable communication links between the peripheral nodes and central control units.

Traceability: {REQ_D2.1_20008.A

}

{REQ_D2.1_06002.A Robustness of the communication link

Monitoring application shall have a robust communication link.

}

{REQ_D2.1_06003.A Access control to applications

pSHIELD applications shall have access control mechanisms to regulate access to applications.

}

{REQ_D2.1_06004.A Integrity of data

Monitoring application shall have the provision to ensure the integrity of data while being transmitted (to control center) and also while being shared between the stakeholders.

}

{REQ_D2.1_06005.A Confidentiality aware information delivery

pSHIELD application shall guaranty confidentiality while information being transported to different stakeholders.

}

## 6.4 pSHIELD Services

{REQ_D2.1_20010.A SPD services

The pSHIELD system shall allow SPD services over homogenous or heterogeneous networks.
Traceability: Traceability: {REQ_D2.1_20001.A, {REQ_D2.1_20002.A

}

# 7 SPD High Level Requirements for pSHIELD System

## 7.1 pSHIELD System

{REQ_D2.1_20011.A pSHIELD System (pSS)

The pSHIELD System shall be composed as a heterogeneous system of a group of interacting, interrelated, or interdependent composable embedded devices and sub-systems with SPD functionalities, other sub-systems and elements, e.g., legacy devices (LDs), and external systems, e.g., legacy, public information systems, and other systems forming a complex whole.

Traceability: {REQ_D2.1_20001.A, {REQ_D2.1_20002.A,

}

{REQ_D2.1_20012.A Security, Privacy and Dependability (SPD) functionalities

The pSHIELD system shall implement SPD management functionalities for a chosen scenario.

}

{REQ_D2.1_20013.A SPD transmission

The pSHIELD system should perform SPD smart driven transmission for a chosen scenario.

}

{REQ_D2.1_20014.A Trusted and dependable connectivity

The pSHIELD system shall allow trusted and dependable connectivity for a chosen scenario.

Traceability: {REQ_D2.1_20013.A

}

{REQ_D2.1_20015.A SPD core service

The pSHIELD shall allow exaction of the SPD core services for a chosen scenario.

Traceability: {REQ_D2.1_20012.A

}

{REQ_D2.1_20016.A SPD metrics

The pSHIELD system shall have SPD metrics for a chosen scenario.

Traceability: {REQ_D2.1_20013.A.

}

{REQ_D2.1_20017.A Semantics and ontology

The pSHIELD system should support semantics and ontology technologies for a chosen scenario.

}

{REQ_D2.1_20018.A Policy based management

The pSHIELD system should support policy based management.

}

{REQ_D2.1_20019.A Compatibility

The pSHIELD system shall be compatible with the supported communication standards.

}

{REQ_D2.1_20020.A Redundancy

The pSHIELD shall support redundancy functions.

}

{REQ_D2.1_06006.A Audit Functionalities

The pSHIELD system shall guarantee Audit functionalities

}

{REQ_D2.1_06007.A Cryptographic Support

The pSHIELD system shall guarantee cryptographic support

}

{REQ_D2.1_06008.A Non-repudiation functionalities

The pSHIELD system shall guarantee non-repudiation functionalities

}

{REQ_D2.1_06009.A Access control functionalities

The pSHIELD system shall guarantee access control functionalities on users, assets and operations among them

}

{REQ_D2.1_06010.A Identification and Authentication functionalities

The pSHIELD system shall guarantee users Identification and Authentication functionalities

}

{REQ_D2.1_06011.A Management of Security Functionalities

The pSHIELD system shall guarantee the management of Security Functionalities

}

{REQ_D2.1_06012.A Anti physical tampering functionalities

The pSHIELD system should guarantee anti physical tampering that could compromise the SPD functionalities

}

{REQ_D2.1_06013.A Reliable Timestamps

The pSHIELD system should be able to provide reliable timestamps

}

{REQ_D2.1_06014.A Self test functionalities

The pSHIELD system should run a suite of self-test to demonstrate the correct operation of its SPD functionalities

}

{REQ_D2.1_06015.A Trusted channel functionalities

The pSHIELD system shall provide trusted channel for SPD functionalities

}

## 7.2 pSHIELD Reference System Architecture

{REQ_D2.1_20021.A Web Service

In the pSHIELD system each device or software component should be represented as a web service.

}

{REQ_D2.1_20022.A pSHIELD Service Oriented Architecture (SOA)

The pSHIELD system should be a SOA based system.

}

{REQ_D2.1_20023.A Proxy running on a dedicated gateway

Resource constraint devices in the pSHIELD system such as sensors should interact with the rest of the pSHIELD network through a Proxy running on a dedicated gateway.

}

{REQ_D2.1_20024.A Protocol conversion

Proxies running on a dedicated pSHIELD system gateway shall handle the communication with the resource-constraint devices and manage the protocol conversion to achieve an IP communication.

Traceability: {REQ_D2.1_20023.A

}

{REQ_D2.1_20025.A Usability of pSHIELD embedded devices

Middleware of the pSHIELD system should enable any embedded device to be usable from a pSHIELD application.

}

{REQ_D2.1_20026.A Semantic model-based architecture

The pSHIELD system shall have a generic semantic model-based architecture.

}

{REQ_D2.1_20027.A Semantic Model Driven Architecture

The pSHIELD system shall define

- Device Ontology
- Security Ontology
- Software Components Ontology

Traceability: {REQ_D2.1_20025.A, {REQ_D2.1_20026.A

}

{REQ_D2.1_20028.A Device and service ontology

Device and service ontology shall ensure the secure and fail-safe connection between the different components of the pSHIELD system.

Traceability: {REQ_D2.1_20026.A

}

{REQ_D2.1_20029.A Secure Service Discovery

The pSHIELD middleware shall support standards for secure service discovery.

Traceability: {REQ_D2.1_20026.A

}

# 8 Node Requirements and Specifications

*NOTE: In the following, the expression "pSHIELD node" is referred to all three node levels of increasing complexity: nano node, micro/personal node and power node.*

## 8.1 Security

{REQ_D2.1.1_01001.A Node – built-in security mechanisms

A pSHIELD node should be designed with security built-in mechanisms.

Traceability: Technical Annex - Abstract

}

{REQ_D2.1.1_01002.A Node – integrity to unauthorized accesses

A pSHIELD node should be designed with mechanisms that improve its resilience to unauthorized information alteration (integrity)

Traceability: Technical Annex - 2.2.2. Automatic Access Control and Denial-of-Service

}

{REQ_D2.1.1_01003.A Node – availability for authorised users

A pSHIELD node should be designed with mechanisms that improve its availability for authorized users.

Traceability: Technical Annex - 2.2.2. Automatic Access Control and Denial-of-Service; Task 3.2

}

{REQ_D2.1.1_01004.A Node – integrity protection

A pSHIELD node should provide mechanisms that guarantee data integrity based on hardware "hooks" and secure key installation.

Traceability: Technical Annex – Task 3.1.

}

{REQ_D2.1.1_01005.A Node – secure firmware upgrade

A pSHIELD node should provide a mechanism that allows secure upgrading of the firmware from a remote site as well as local site.

Traceability: Technical Annex – Task 3.1.

}

{REQ_D2.1.1_01006.A Node – secure boot

A pSHIELD node should provide mechanisms that guarantee a secure boot.

Traceability: Technical Annex – Task 3.1.

}

{REQ_D2.1.1_01007.A Node – TPM

A pSHIELD micro/personal and power node should be TPM compliant.

Traceability: Technical Annex – 2.2.2. TPM and Smartcard for Trust ESs; Task 3.1

}

{REQ_D2.1.1_01008.A Node – TPM cryptographic/hash improvement

A pSHIELD node should have an improved TPM architecture to support future evolution of cryptographic/hash functionalities.

Traceability: Technical Annex – 2.2.2. TPM and Smartcard for Trust ESs; Task 3.1

}

{REQ_D2.1.1_01009.A Node – TPM alternative communication interfaces

For a pSHIELD micro/personal and power node, the TPM platform should be extended in order to have alternative communication interfaces, better adapted to the embedded applications than the LPC (low pin count) currently supported.

Traceability: Technical Annex – 2.2.2. TPM and Smartcard for Trust ESs; Task 3.1

}

{REQ_D2.1.1_01010.A Node – TPM new specialized/dedicated commands

For a pSHIELD micro/personal and power node, the TPM platform should be extended in order to add some specialized/dedicated commands (e.g. to further develop on-the-fly encryption).

Traceability: Technical Annex – 2.2.2. TPM and Smartcard for Trust ESs; Task 3.1

}

{REQ_D2.1.1_01011.A Node – TPM additional cryptographic protocols

For a pSHIELD micro/personal and power node, the TPM platform should be extended in order to implement additional cryptographic protocols (e.g. elliptic curves)

Traceability: Technical Annex – 2.2.2. TPM and Smartcard for Trust ESs; Task 3.1

}

{REQ_D2.1.1_01012.A Node – TPM protection against attacks on its integrity

For a pSHIELD micro/personal and power node, the TPM platform should be extended in order to improve protection against attacks on its integrity, particularly against physical attacks

Traceability: Technical Annex – 2.2.2. TPM and Smartcard for Trust ESs; Task 3.1

}

{REQ_D2.1.1_09001.A Node – power node built-in security mechanism

The FPGA engine of the Power Node includes a core logic that monitors the security of the pSHIELD Power Node itself. Tampering with the node triggers a protection mechanism in the security node that:

- Physically disconnects any I/O and network,
- Deletes any data resident on the node,
- Initiates the physical destruction of the device itself by driving the power supply.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09002.A Node – power node security features cryptography

The pSHIELD Power Node provides security features such as cryptographic capabilities through a dedicated core embedded in the Power Node's FPGA; moreover the hardware supports the Intel AES-NI technology.

Traceability: Technical Annex – Task 3.2.

}

## 8.2 Dependability

{REQ_D2.1.1_01013.A Node – dependability mechanism

A pSHIELD node should be designed with built-in mechanisms improving system dependability.

Traceability: Technical Annex – Abstract

}

{REQ_D2.1.1_01014.A Node – system availability

A pSHIELD node should be designed with mechanisms that improve system availability.

Traceability: Technical Annex – 2.2.2. Power Supply Protection «[...] system availability has to be one of the key points.»

}

{REQ_D2.1.1_01015.A Node – system integrity

A pSHIELD node should be designed with mechanisms that improve system integrity.

Traceability: Technical Annex – Task 3.1. «[...] integrity protection of the ES firmware [...]»

}

{REQ_D2.1.1_01016.A Node – safety

A pSHIELD node should be designed with mechanisms that improve safety.

Traceability: Technical Annex – 4.1.1. 3) «[...] Research should take into account the interplay between system properties such as safety, [...]»

}

{REQ_D2.1.1_01017.A Node – system maintainability

A pSHIELD node should be designed with mechanisms that improve system maintainability, such as self-reconfigurability, self-recovery or firmware upgrade mechanisms.

Traceability: Technical Annex – 2.2.2. Self-re-configurability and self-recovery of sensing and processing tasks; Task 3.3

}

{REQ_D2.1.1_01018.A Node – self-recovery

A pSHIELD node shall provide a mechanism of self-recovery from a detected error. Self-recovery may be by re-execution, reassigning functions to a redundant unit running in degrading mode, or re-programming firmware, such as by automatic firmware recovery (AFR).

Traceability: Technical Annex – 2.2.2. Self-re-configurability and self-recovery of sensing and processing tasks; Task 3.3.

}

{REQ_D2.1.1_01019.A Node – preventive and corrective maintenance

A pSHIELD node should be capable to reconfigure itself in order to perform either preventive or corrective maintenance by, for example, optimizing the system performance.

Traceability: Technical Annex – 2.2.2. Self-re-configurability and self-recovery of sensing and processing tasks;

}

{REQ_D2.1.1_01020.A Node – safe-state

A pSHIELD node should have a safe state, in which the harmful consequences of a failure are minor.

Traceability: Technical Annex – 4.1.1. 3) «[...] Research should take into account the interplay between system properties such as safety, [...]»

}

{REQ_D2.1.1_01021.A Node – States ACTIVE, LOW-POWER STANDBY, TEST

A pSHIELD node shall support and always be in one of several node Operational Modes, such as follows:

- ACTIVE: The Node supports operational services
- LOW-POWER STANDBY: Power is applied to the Node and it provides a Heartbeat but it doesn't provide any services
- TEST: The Node can perform all functionality but it is not supporting operational services; this mode is used for test/maintenance activities (e.g. software upgrades; system test).

}

{REQ_D2.1.1_01022.A Node – Self Test

A pSHIELD node should perform a complete self test of all functions.

}

{REQ_D2.1.1_01023.A Node – uninterruptible Power Supply

For a pSHIELD node, power supply should be provided continuously, without any cut in time neither in the power, voltage or current levels, to correctly bias the devices.

Traceability: Technical Annex – 2.2.2. Power Supply Protection; Task 3.1.

}

{REQ_D2.1.1_01024.A Node – Power Supply monitoring

For a pSHIELD node, power supply should be monitored continuously in order to prevent any system power risk, which might affect the node.

Traceability: Technical Annex – 2.2.2. Power Supply Protection; Task 3.1.

}

{REQ_D2.1.1_01025.A Node – Power Supply fault tolerance

A pSHIELD node should support mechanisms to protect itself from any power supply failure.

Traceability: Technical Annex – 2.2.2. Power Supply Protection; Task 3.1.

}

{REQ_D2.1.1_01026.A Node – remote powering

A pSHIELD node should be able to support remote powering, at least to some modules of the device, allowing some functionalities to become operational in case of power failure.

Traceability: Technical Annex – 2.2.2. Power Supply Protection; Task 3.1.

}

{REQ_D2.1.1_09003.A Node – Power Node dependability mechanism

The pSHIELD power node should be designed to provide redundancy by aggregation of multiple identical units. The physical design of the Power Node is compliant to MIL standards. Execution segregation through hardware virtualization allows for protection, monitoring, disabling, and replacement of malfunctioning or compromised system images.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09004.A Node – Power Node fault identification

The pSHIELD power node is equipped with multiple monitoring devices that verify the status of the node. The monitoring of the Power Node is permanent, regardless of the status of the Power Node (operational, powered off, malfunctioning).

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09005.A Node – Power Node fault reaction

In a configuration with multiple pSHIELD power nodes, a malfunction triggers a recovery mechanism that physically disconnects the defective node and automatically activates one spare Power Node.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09006.A Node – Power Node dependability Power Supply

The pSHIELD power node should be powered by a protected Power Supply; continuity for a limited time is ensured by a UPS Group. The Power Supply is redundant with hot swap capability. Both the Power Supply and UPS Unit are monitored by the pSHIELD power node through an internal bus.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09007.A Node – Power Node dependability Self re-configuration

In case of a fault, redundant hardware should provide dependable operations. This is accomplished at the hardware level through duplication of the resource (i.e. redundant Power Supply) and at a functional level through aggregation of resources (spare Power Nodes).

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09008.A Node – Power Node Local and Remote Firmware Upgrade

The pSHIELD power node should allow the remote and local reconfiguration of the firmware (BIOS, FPGA image, etc) by an embedded mechanism. The FPGA coordinates the replacement of the images; in case of failure the system is automatically reverted to a validated set of images that is permanently stored on

board. Remote configuration is performed by sending the images through the network; local configuration is performed through a JTAG or similar mechanism.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_01027.A Node – fault-tolerance

A pSHIELD power node should be designed with hardware and firmware redundancy to implement fault-tolerance.

Traceability: Technical Annex – 2.2.2. Intrinsically secure ES Firmware; Rugged High Performance Computing Node; Task 3.2

}

{REQ_D2.1.1_01028.A Node – fail-controlled

A pSHIELD power node should fail in a controlled way, meaning that node failures are, to an acceptable extent, halting and signalled.

Traceability: Technical Annex – 2.2.2. Intrinsically secure ES Firmware; Rugged High Performance Computing Node

}

{REQ_D2.1.1_01029.A Node – error recovery

A pSHIELD power node should implement concurrent error detection and recovery mechanism; the node should be capable to react to the anomalies through an auto-reconfiguration. The auto-reconfiguration may be hardware reconfiguration, firmware reconfiguration or both.

Traceability: Technical Annex – 2.2.2. Rugged High Performance Computing Node; Task 3.2.

}

{REQ_D2.1.1_01030.A Node – automatic access control

A pSHIELD node should support an automatic Access Control.

Traceability: Technical Annex – 2.2.2 Automatic Access Control and Denial-of-Service

}

{REQ_D2.1.1_01031.A Node – secure authentication

A pSHIELD node should support a secure authentication protocols.

Traceability: Technical Annex – 2.2.2 Asymmetric cryptography for low cost nodes

}

## 8.3 Privacy

{REQ_D2.1.1_01032.A Node – built-in privacy mechanisms

A pSHIELD node should be designed with built-in mechanisms that provide information privacy.

Traceability: Technical Annex – Abstract

}

{REQ_D2.1.1_01033.A Node – asymmetric cryptography

A pSHIELD node should support a hardware implementation of asymmetric cryptography.

Traceability: Technical Annex – 2.2.2 Asymmetric cryptography for low cost nodes

}

{REQ_D2.1.1_09009.A Node – Power Node privacy mechanism

For a pSHIELD power node, the FPGA engine includes a core logic that encrypts any sensitive data prior to moving it across the network or storing it on the embedded storage.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09010.A Node – Power Node privacy

The pSHIELD power node should have a privacy chain that includes BIOS password protection.

Traceability: Technical Annex – Task 3.2.

}

## 8.4 Composability

{REQ_D2.1.1_01034.A Node – composability

A pSHIELD node shall support composability; the nodes shall be assembled (physically as well as logically) in various combinations to satisfy specific user requirements.

Traceability: Technical Annex – 1.1.2. IP1 - Composability

}

{REQ_D2.1.1_01035.A Node – commands for composition and configuration

A pSHIELD node should be able to support static and dynamic composability through commands received from the pShield overlay.

Traceability: Technical Annex – 2.1.2.6.

}

{REQ_D2.1.1_09011.A Node – Power Node composability

The pSHIELD power node is scalable to achieve the degree of computational performance specified by the customer. The I-O and network interfaces are programmable to permit interfacing the system to multiple network and bus technologies and protocols.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09012.A Node – Power Node Static and Dynamic composability

The pSHIELD power node should be configured in order to provide aggregated power and/or redundancy. The configuration mechanism supports both static and dynamic composability. When static composability is used, the configuration of the clustered units is set statically; if dynamic composability is used, the configuration of the system is changed on-the-fly to respond to external or internal events. In this case, a Power Node joins or leaves the group according to a set of rules that are used during a negotiation phase when the resources available are verified through the devices' network.

Traceability: Technical Annex – Task 3.2.

}

## 8.5 Performance/Metrics

{REQ_D2.1.1_01036.A Node – Performance Parameters monitoring

A pSHIELD node should monitor its Performance Parameters and report alert or alarm conditions to the external systems when the defined thresholds for alarm/alert conditions are exceeded.

}

{REQ_D2.1.1_01037.A Node – parameters for SPD metrics

A pSHIELD node should continuously provide parameters that will be monitored by the pSHIELD overlay..

Traceability: Technical Annex – 2.1.2.6.

}

{REQ_D2.1.1_01038.A Node – low power

HW and SW implementation of a pSHIELD node should take into account power constraints.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09013.A Node – Power Node Performance Parameters monitoring

The independent, embedded controller (BMC) allows the monitoring of each performance parameters, such as temperatures, voltages, etc. Access to these parameters can be done by the pSHIELD power node applications, locally and remotely over the network.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09014.A Node – Power Node Computation power

The pSHIELD power node should offer a maximum processing power of 80GFlops/unit, provided by two X86 high performance, 64 bit, multicore CPUs and a high performance FPGA engine.

Traceability: Technical Annex – Task 3.2.

}

## 8.6 External interfaces

{REQ_D2.1.1_01039.A Node – Middleware Interface

A pSHIELD node should be capable to exchange data, measures as well as parameters with middleware, through a predefined interface.

Traceability:

}

{REQ_D2.1.1_01040.A Node – Remote Commands

A pSHIELD node should accept configuration commands, data, and equipment status form an external system.

Traceability:

}

{REQ_D2.1.1_01041.A Node – SNMP Interface

A pSHIELD node should provide an SNMP interface for monitoring and control by the external maintenance system.

}

{REQ_D2.1.1_01042.A Node – network interface

A pSHIELD node should be able to exchange data, measures as well as parameters with network level, through a predefined interface.

Traceability: Technical Annex – 2.1.2.2.

}

{REQ_D2.1.1_01043.A Node – overlay interface

A pSHIELD node should be able to send measurements and parameters, and receive commands from the Shield overlay through a dependable interface.

Traceability: Technical Annex – 2.1.2.4.; 2.1.2.6.

}

{REQ_D2.1.1_09015.A Node – Power Node Remote Self Test

At boot time, the pSHIELD power node should enter in a Power-On-Self-Test. The POST can be triggered and monitored remotely using the embedded BMC.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09016.A Node – Power Node SNMP Interface

The embedded controller BMC provides an SNMP interface to the pSHIELD power node and allows setting traps for specific events.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09017.A Node – Power Node Remote Commands

The embedded BMC should permit the remote configuration of the pSHIELD power node through the network. Additional remote configurability can be done through the FPGA.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09018.A Node – Interfaces

The pSHIELD power node should provide the following interfaces:

- Ethernet: 10/100/1000Mbps port.
- Serial: 2x RS-232 ports.
- USB: 1 USB 2.0 port.
- I2C: 2 ports.
- High speed networks: Up to 7 high speed (Infiniband, XAUI, 10GE, SRIO) serial lines available through rugged CX4 connectors.
- Video: VGA Analog Video Output.

Traceability: Technical Annex – Task 3.2.

}

## 8.7 Miscellaneous

{REQ_D2.1.1_01044.A Node – self-reconfigurability

A pSHIELD node should provide a mechanism of self-reconfigurability to increase function density, increase security against side-channel attacks, and increase dependability implementing self-healing properties.

Traceability: Technical Annex – 2.2.2. Self-re-configurability and self-recovery of sensing and processing tasks; Task 3.3.

}

{REQ_D2.1.1_01045.A Node – TPM improvement of product endurance and lifespan

For a pSHIELD micro/personal and power node, the TPM platform should be extended in order to implement additional mechanisms to improve product endurance and increase product lifespan.

Traceability: Technical Annex – 2.2.2. TPM and Smartcard for Trust ESs; Task 3.1

}

{REQ_D2.1.1_01046.A Node – TPM inexpensive implementation

For a pSHIELD micro/personal and power node, the TPM platform should be extended in order to have inexpensive implementation to allow widespread use.

Traceability: Technical Annex – 2.2.2. TPM and Smartcard for Trust ESs; Task 3.1

}

{REQ_D2.1.1_01047.A Node – TPM Compliance with global export control regulations in order not to restrict international trade with TC platforms (PCs)

For a pSHIELD micro/personal and power node, the TPM platform must be extended to be compliant with global export control regulations in order not to restrict international trade with TC platforms (PCs)

Traceability: Technical Annex – 2.2.2. TPM and Smartcard for Trust ESs; Task 3.1

}

{REQ_D2.1.1_09019.A Node – States: ACTIVE, LOW-POWER STANDBY, TEST

The pSHIELD power node should support different states:

- Active: this is the normal operating mode
- Low-Power Standby: the on board BMC (independent controller) is active and controls the power state of the Power Node (off, active, power cycle)
- TEST: the Power node is fully active and enters the maintenance

The pSHIELD power node supports also ACTIVE substates, that have different power consumption figures. The Active substates are achieved by setting the CPUs in different power modes and by selectively turning on/off CPUs cores.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09020.A Node – Power Node Saving Power

The maximum power dissipation for the Power Node should be less than 180 Watts. The power supply should work in the range 28-48VDC and should provide protection against: Reverse, Over Voltage, Surge.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_09021.A Node – Power Node Environmental conditions

The pSHIELD power node should be capable to operate in the following environmental conditions:

- Altitude: up to 18000 meters.
- Temperature: Operating: -40ºC to +60ºC ambient, Storage: -40ºC to +85ºC.
- Humidity: 5% to 95% (non-condensing); 100% (condensing).
- Physical: Corrosion Resistant. Anodized per MIL-A-8625, Type II, Class 2.
- Ingress: Resistant to Dust, Water, and Moisture Boards are conformably coated.

Traceability: Technical Annex – Task 3.2.

}

{REQ_D2.1.1_01048.A Node – embedded camera array

A pSHIELD power node should be able to provide the means to build a camera array with auto-calibration and auto-configuration.

Traceability: Technical Annex – 2.2.2. Embedded Camera Array auto-calibration and auto configuration techniques – Task 3.2.

}

# 9 Network Requirements and Specifications

{REQ_D2.1_20030.A Network Security

The pSHIELD network layer shall be secure.

Traceability: {REQ_D2.1_20002.A

}

{REQ_D2.1_20031.A Network Security Protocols

The pSHIELD network layer should have a suitable security protocols.

Traceability: {REQ_D2.1_20002.A, {REQ_D2.1_20030.A

}

{REQ_D2.1_20032.A Network Security Protocols for IP-based

In the pSHIELD network layer should support security protocols to protect entire IP payload or upper-layer protocols.

Traceability: {REQ_D2.1_20002.A, {REQ_D2.1_20030.A, {REQ_D2.1_20031.A.

}

{REQ_D2.1_20033.A Network Security Attributes

The pSHIELD network layer should provide Availability, Confidentiality and Integrity.

Traceability: {REQ_D2.1_20002.A, {REQ_D2.1_20030.A

}

{REQ_D2.1_20034.A Denial of Service Attack

The pSHIELD network layer shall support anti-replay protection to prevent against a denial of service attack.

Traceability: {REQ_D2.1_20002.A, {REQ_D2.1_20030.A

}

{REQ_D2.1_20035.A Confidentiality

The pSHIELD network layer shall support data confidentiality to protect, and to encrypt the entire data.

Traceability: {REQ_D2.1_20002.A, {REQ_D2.1_20030.A

}

{REQ_D2.1_20036.A Encryption Algorithms

The pSHIELD network layer shall support algorithms for encryption.

Traceability: {REQ_D2.1_20002.A, {REQ_D2.1_20030.A

}

{REQ_D2.1_20037.A Data Integrity

The pSHIELD network layer shall support data integrity to ensure that the contents of the packet do not change in transit.

Traceability: {REQ_D2.1_20002.A, {REQ_D2.1_20030.A

}

{REQ_D2.1_20038.A Authentication

The pSHIELD network layer shall support data authentication to verify that the packet received is actually from the claimed sender.

Traceability: {REQ_D2.1_20002.A, {REQ_D2.1_20030.A

}

{REQ_D2.1_20039.A IPSec support

The pSHIELD network layer should use the IPSec protocol.

Traceability: {REQ_D2.1_20002.A, {REQ_D2.1_20032.A

}

{REQ_D2.1_20040.A Efficient Security Algorithms

The pSHIELD network layer should support more efficient algorithms for fast and better speed of execution to satisfy the conditions in the case of limited processing and power capabilities of embedded devices.

Traceability: {REQ_D2.1_20002.A, {REQ_D2.1_20030.A

}

{REQ_D2.1_20041.A Network Security Features Cryptography

The pSHIELD network layer should support symmetrical cryptography.

Traceability: {REQ_D2.1_20002.A, {REQ_D2.1_20030.A

}

{REQ_D2.1_20042.A Dependability

The pSHIELD network layer should provide dependability mechanism.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_20043.A Dependability Attributes

The pSHIELD network layer should provide Reliability, Availability, Safety, Maintainability, and Integrity.

Traceability: {REQ_D2.1_20002.A, {REQ_D2.1_20042.A

}

{REQ_D2.1_20044.A Privacy

The pSHIELD network layer should provide privacy.

Traceability: {REQ_D2.1_20002.A,

}

# 10 Middleware Requirements and Specifications

{REQ_D2.1_20045.A WS-Security

The pSHIELD middleware should support WS-Security.

}

{REQ_D2.1_20046.A Fault recovery

The pSHIELD middleware shall recover from faults.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_0701.A Middleware – Information Translation

The pSHIELD middleware should translate information acquired from nodes and network in a normalized format by means of semantic technologies.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_0702.A Middleware – Interface Capability

The pSHIELD middleware should be capable to interface different kinds of nodes to be specified.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_0703.A Middleware – Discovery Functionality

The pSHIELD middleware should have discovery functionality in order to catalogue the capability available from nodes and network

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_0704.A Middleware – Services

The pSHIELD middleware should expose its services to the application layer.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_0705.A Middleware – Metrics provisioning

The pSHIELD middleware should provide metrics collected from nodes and the network to the overlay.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_0706.A Middleware – Configuration provisioning

The pSHIELD middleware should act towards the nodes and the network the configuration computed from the overlay.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_0707.A Middleware – Internal Interfaces

The internal interfaces between the middleware and overlay modules should be defined.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_0708.A Middleware – External Interfaces

The external interfaces to the application layer should be implemented by web services.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_06016.A Middleware – Interoperability

A pSHIELD Middleware should address the interoperability between different SPD technologies.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_06017.A Middleware – Generic interface

A pSHIELD Middleware should have the generic interfaces to integrate different types of SPD nodes and to retrieve and interpret various types of incoming data.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_06018.A Middleware – Lightweight representation

When required (e.g. lower processing power), a pSHIELD Middleware should exchange lightweight representations of capabilities, interfaces, metrics and other relevant information between the node, network, middleware and overlay layer.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_06019.A Middleware – Conversion to lightweight

When required a pSHIELD middleware should covert the heavyweight representations to lightweight ones to comply with the technical limitations of the components in other layers.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_06020.A Middleware - Knowledge base

The pSHIELD system should have the high level knowledge base containing various information and data of capabilities, interfaces and metrics.

Traceability: {REQ_D2.1_20002.A, ,

}

{REQ_D2.1_06021.A Middleware - Integration to knowledge base

The pSHIELD system should have automatic integration of relevant information and data into the knowledge base.

Traceability: {REQ_D2.1_20002.A, , {REQ_D2.1_06020.A

}

{REQ_ D2.1_2101.A Middleware – Knowledge model of SPD

An ontology shall be developed with the aim of creating a knowledge model of SPD technologies in Embedded Systems.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_ D2.1_2102.A Middleware – Core concepts of SPD

The knowledge model shall be constituted by a core ontology, including main concepts and relationships of SPD technologies in the context of Embedded Systems.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_ D2.1_2103.A Middleware – Extendibility of the knowledge model

It shall be possible to add further concepts and relationships concerning SPD, as well as instances of SPD, by extending the developed knowledge model.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_ D2.1_2104.A Middleware – Modelling of functional features of SPD

The knowledge model (core ontology) shall enable at least the modelling and representation of functional features of SPD.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_ D2.1_2105.A Middleware – Modelling of structural features of SPD

The knowledge model (core ontology) shall enable the modelling and representation of structural features of HW/SW components and sub-components that perform SPD.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_ D2.1_2106.A Middleware – Modelling of parameter features of SPD

The knowledge model (core ontology) shall enable the modelling and representation of parameter features of SPD, expressing enumerable, environmental or nonfunctional properties of the functionalities.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_ D2.1_2107.A Middleware – Modelling of composition of features of SPD

The knowledge model (core ontology) shall enable the modelling and representation of a generic inter structural / functional / parameter features relations.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_ D2.1_2108.A Middleware – Representation of the knowledge model

The core ontology shall be expressed by a formal notation that shall ensure that reasoning about the model shall provide a suitable degree of soundness and completeness.

Traceability: {REQ_D2.1_20002.A,

}

# 11 Overlay Requirements and Specifications

{REQ_D2.1_2301.A Overlay – Functional Elements

The pSHIELD Overlay should be composed by two functional elements: a set of federated SPD Security Agents and the Information/Data Management.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2302.A Overlay - Security Agent

A pSHIELD Security Agent shall realize one or more Overlay functionalities for the sub-system under its responsibility.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2303.A Overlay - Security Agents communication

A pSHIELD Security Agent shall communicate with other Security Agents that are responsible of other sub-systems.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2304.A Overlay - Security Agents interaction

The federation obtained by the interaction of different pSHIELD Security Agents shall realize all the Overlay functionalities.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2305.A Overlay – Information/data management

The Information/Data-Management shall store and provide to the adequate module the knowledge collected by the middleware and used by the SPD Security Agents.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2306.A Overlay - Overlay interfaces

The pSHIELD Overlay should support two types of interfaces: external interfaces (to communicate with pSHIELD Node and Network Layers) and internal interfaces (to communicate with pSHIELD Middleware).

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2307.A Overlay - Internal interfaces

The pSHIELD Overlay shall communicate and share information with the middleware via internal interfaces. These interfaces support also the information exchange between Security Agents.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2308.A Overlay – Overlay/Node external interfaces

The pSHIELD Overlay shall be able to receive measurements and parameters, and send commands to the pSHIELD Nodes through external interfaces. This interface may be the pSHIELD Middleware itself.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2309.A Overlay – Overlay/Network external interfaces

The pSHIELD Overlay shall be able to receive measurements and parameters, and send commands to the pSHIELD Network through external interface. This interface may be the pSHIELD Middleware itself.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2310.A Overlay – Input: desired SPD and policies

The pSHIELD Overlay shall be able to receive as input, from the application layer, or the end-user, the desired level of SPD and/or the SPD Policies.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2311.A Overlay – Input: effective SPD

The pSHIELD Overlay should support continuous monitoring of the parameters that will be provided by the pSHIELD Nodes and Network.

Traceability: {REQ_D2.1_20002.A, REQ_D2.1.1_01039.A.

}

{REQ_D2.1_2312.A Overlay – Main Functionalities

The pSHIELD Overlay shall realize four main functionalities: discovery, control, composition and configuration. These functionalities could be either strictly separated or partially merged.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2313.A Overlay – Control sub-functionalities

The pSHIELD Overlay control functionality should be obtained through four different sub-functionalities: measurement and quantification, comparison and reasoning. These functionalities could be either strictly separated or partially merged.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2314.A Overlay – Functionality: measurement and quantification

The pSHIELD Overlay shall be able to measure and quantify the SPD level of each SHIELD component as well as the SPD level of the overall system.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2315.A Overlay – Functionality: comparison

The pSHIELD Overlay shall be able to compare the desired SPD level with the effective SPD level.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2316.A Overlay – Functionality: reasoning

The pSHIELD Overlay shall have reasoning capabilities (including policy-based and/or context-aware procedures).

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2317.A Overlay – Functionality: composition

The pSHIELD Overlay shall be able to statically compose (off-line) the available pSHIELD components,according to the indications provided by the reasoning functionality, in order to obtain the overall desired level of SPD.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2318.A Overlay – Functionality: configuration

The pShield Overlay should be able to realize static composition through commands sent to the pShield Nodes and Network.

Traceability:{REQ_D2.1_20002.A, REQ_D2.1.1_01037.A.

}

{REQ_D2.1_2319.A Overlay – Functionality: discovery

The SPD Security Agent shall use the discovery functionalities offered by the middleware to perform the Overlay tasks: monitoring and composition of the SHIELD framework.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2320.A Overlay – Functionality: secure discovery

The pSHIELD Overlay should handle secure discovery of SPD services/components/parameters.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2321.A Overlay – Functionality: policy-based discovery

The pSHIELD Overlay should handle policy-based discovery of SPD services/components/parameters.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_D2.1_2322.A Overlay – Functionality: context-aware discovery

The pSHIELD Overlay should handle context-aware discovery of SPD services/components/parameters.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_ D2.1_0722.A Overlay – Representation of the knowledge model

An OWL representation shall be used to express the ontology model in order to ensure that reasoning about the model shall provide a suitable degree of soundness and completeness.

Traceability: {REQ_D2.1_20002.A,

}

{REQ_ D2.1_0723.A Overlay – Trade off between computation and implementation

Suitable OWL profiles (i.e. sub-languages) shall be used so as to trade off different aspects of OWL's expressive power in return for different computational and/or implementational benefits.

}

{REQ_ D2.1_0724.A Overlay – Programmatic access to the knowledge model

The knowledge model shall be accessed by means of a programmatic environment by Middleware Layer and Overlay Agents.

}

{REQ_ D2.1_0725.A Overlay – Support to interoperability

The ontology shall enable interoperability within Middleware Layer and rule based discovery and composition within Overlay Agents

- When nodes have no semantic capabilities due to computational / power limits, semantic reasoning based on ontology model may carry out a reconciliation of heterogeneous formats of parameters exchanged between different layers (also suitable for interaction with legacy agents).
- The semantic characterization of the behavioural aspect of components makes it suitable for an agent to determine “what the service does”.
- The semantic characterization of the composition of functionalities and of the relations among them makes it suitable for an agent to reason about SPD metrics of the current configuration and – if needed - to carry out reconfigurations of the system at run-time, by means of rule-based combination / composition of components and SPD technologies, in order to achieve the new intended values for SPD metrics.

}

{REQ_D2.1_06022.A Overlay – Context-aware discovery, composition and delivery

A pSHIELD overlay should handle context-aware discovery, composition and delivery of SPD services.

}

{REQ_D2.1_06023.A Overlay – Policy-based security

A pSHIELD overlay shall provide security based on pre-defined policy. Policy should include context-aware provisioning.

}

{REQ_D2.1_06024.A Overlay – Policy-based discovery, composition and delivery

A pSHIELD overlay shall provide policy-based discovery, composition and delivery capabilities.

}

# 12 Conclusions

The document provided requirements and specifications of the pSHIELD overall system or application that benefits of the pSHIELD features, starting from a reference application scenario, related to the monitoring of freight trains transporting hazardous material. However, the approach is general enough to support any application scenario, in order to define proper SPD metrics (Task 2.2) and architectural design (Task 2.3).

Generic issues related to the definitions and SPD taxonomy are addressed in the sections 3 and 4, in particular to clarify the concepts of security, privacy and dependability and their relationships. High level requirements and specifications are identified in sections 5 and 6. Finally, for each pSHIELD layer (node, network, middleware and overlay), a preliminary set of SPD requirements, as well as architectural, interface and performance specifications is proposed in sections 7, 8, 9 and 10.

The preliminary outcome of this task will be used by WP3, WP4 and WP5 to develop potential prototypes and by WP6 to validate them. The requirements and specifications will be refined on the basis of the results of the validation phase and on the detailed description of the application scenarios from Task 6.4.

# 13 References

[1] C.E. Landwher, A.R. Bull, J.P. McDermott, and W.S. Choi, “A Taxonomy of Computer Program Security Flaws,” ACM Computing Survey, vol. 26, no. 3, pp. 211-254, 1994.
[2] “Conceptual Model and Architecture of MAFTIA,”MAFTIA, Project IST-1999-11583, D. Powell and R. Stroud, eds., p. 123, Jan. 2003.
[3] L. Lamport, R. Shostak, and M. Pease, “The Byzantine Generals Problem,” ACM Trans. Programming Languages and Systems, vol. 4, no. 3, pp. 382-401, July 1982.
[4] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, Transactions on Dependable and Secure Computing 1 (1) (2004) 11–33. January–March, ISSN 1545-5971.
[5] www.w3.org/XML/
[6] www.w3.org/TR/soap/
[7] www.oasis-open.org/
[8] Common Criteria for Information Technology Security Evaluation - Part 2: Security functional components.
[9] Flammini, F., Gaglione, A., Ottello, F., Pappalardo, A., Pragliola, C., Tedesco, A.: Towards Wireless Sensor Networks for Railway Infrastructure Monitoring. In: IEEE Proc. ESARS 2010, 19-21 October 2010, Bologna, Italy: pp. 1-6