SEVENTH FRAMEWORK PROGRAMME
Challenge 1
Information and Communication Technologies

Trusted Architecture for Securely Shared Services

| | |
|---|---|
| Document Type: | Deliverable |
| Title: | Pilot evaluation report |
| Work Package: | WP9 |
| Deliverable Nr: | D9.2 |
| Dissemination: | Public |
| Preparation Date: | |
| Version: | 1.0 |



## The TAS³ Consortium

| | Beneficiary Name | Country | Short | Role |
|---|---|---|---|---|
| 1 | KU Leuven | BE | KUL | Proj. Coord. |
| 2 | Synergetics NV/SA | BE | SYN | Partner |
| 3 | University of Kent | UK | KENT | Partner |
| 4 | University of Karlsruhe | DE | KARL | Partner |
| 5 | Technische Universiteit Eindhoven | NL | TUE | Partner |
| 6 | CNR/ISTI | IT | CNR | Partner |
| 7 | University of Koblenz-Landau | DE | UNIKOL | Partner |
| 8 | Vrije Universiteit Brussel | BE | VUB | Partner |
| 9 | University of Zaragoza | ES | UNIZAR | Partner |
| 10 | University of Nottingham | UK | NOT | Partner |
| 11 | SAP Research | DE | SAP | S&T Coord. |
| 12 | EIfEL | FR | EIF | Partner |
| 13 | Intalio | UK | INT | Partner |
| 14 | Risaris | IR | RIS | Partner |
| 15 | Kenteq | NL | KETQ | Partner |
| 16 | Oracle | UK | ORACLE | Partner |
| 17 | Custodix | BE | CUS | Partner |
| 18 | Medisoft | NL | MEDI | Partner |
| 19 | Symlabs | PT | SYM | Partner |

## Contributors

| | Name | Organisation |
|---|---|---|
| 1 | Eda Marchetti | CNR-ISTI |
| 2 | Sandra Winfield | University of Nottingham |
| 3 | Louis Schilders | Custodix |
| 4 | | |
| 5 | | |

# Contents

# List of Figure

# 1 Executive Summary

Testing related activities encompass the entire development process and may consume a large part of the effort required for producing software. In this deliverable we provide two test strategies, and the derived test plan, for the Pilot evaluation report. The test strategies have been selected according to the Pilots documentation available so to reduce the effort for test case derivation and maximise the effectiveness of testing phase.
In this first iteration of the D9.2 we focus on the integration stage and we adopt black box testing approaches. The former test strategy is based on Sequence Diagrams called (UIT_SD), the latter on the XACML specification. However the application of the policy based testing for test cases derivation, has been postponed to the second iteration of this deliverable, because the effort for the manual derivation of the test cases was not compatible with the deliverable schedule. For the second iteration, thanks to the automatic facilities that will be provided in the deliverable D10. 3, the application of this kind of methodology will be faster and more effective.
Moreover, among the various Pilot applications, in this deliverable we first dealt with the evaluation of the employability scenario because currently it is the most complete and ready to use from an application implementation point of view, and, because it's intuitive features, it can be a good benchmark for evaluating the goodness of test strategy selected in terms of effort required to derive and execute test cases and fault detection effectiveness.
In this deliverable we therefore report the evaluation of the employability pilot considering only the UIT_SD methodology. For aim of completeness we include in the test plan some negative testing, derived applying the same test strategy, so to verify the robustness of Pilot in case of wrong or malicious inputs.
Secondly we applied the test strategy to the Healthcare integration trial because this trial focuses primarily on the integration of TAS3 components in an existing (legacy) application. It illustrates clearly the multi-applicability of the test strategy adopted in different circumstances.

## 1.1 Reading Guide

In the next chapter we introduce the basic concepts of testing (Sec 2.1.) and test strategies (Sec 2.2). In particular we describe the classical approaches for integration testing ( Sec 2.3)

In Chapter 3 we propose a selection of test strategies for Pilots applications. In particular we present the UIT_SD method, based on sequence diagrams analysis in Section 3.1, while in Section 3.2 we propose a test strategy based on XACML policies.

In Chapter 4 we present the test plan for employability scenario. In particular in section 4,1 for each Sequence Diagram available we list the test procedures derived and in section we provide details about the test cases derived and result obtained from the test cases execution. Discussion and lesson learned are in section 4.3.

In Chapter 5 we present the testplan for the Healthcare scenario following the same outline asthe one used for the employability scenario.

Finally Chapter 6 comprises the conclusions and the outline of future work.

# 2 Introduction

With the purpose of making this deliverable self-contained, we provide here a brief introduction to testing methodologies, with emphasis on the level of integration which is the focus of this deliverable. References to the literature are provided for further reading.(Chapter 7)

Testing is a crucial part of the software life cycle, and recent trends in software engineering evidence the importance of this activity all along the development process. The success of a test phase relies on well-organized testing stages which includes a planning of test strategies and procedures, and propagate down, with derivation and refinement of test cases along the various development steps since the code-level stage, at which the test cases are eventually executed, and even after deployment, with logging and analysis of operational usage data and customer's reported failures.

Testing involves several high-demanding tasks: at the forefront is the task of deriving an adequate suite of test cases, according to a feasible and cost-effective test selection technique. However, test selection is just a starting point, and many other critical tasks face test practitioners with technical and conceptual difficulties: the ability to launch the selected tests (in a controlled host environment, or worse in the tight target environment of an embedded system); deciding whether the test outcome is acceptable or not (which is referred to as the test oracle problem); if not, evaluating the impact of the failure and finding its direct cause (the fault), and the indirect one (via Root Cause Analysis); judging whether testing is sufficient and can be stopped, which in turn would require having at hand measures of test effectiveness: one by one, each of these tasks presents tough challenges to testers, for which their skill and expertise always remain of topmost importance.

We provide here a test plan for a specific phase of software development: the integration stage. We provide for the pilot applications a test plan developed and tuned according with the requirements and specifications available within the WP9 and described in the previous deliverables D9.1 first and second iteration. The test plan includes the selection of test strategies for test case derivation, the description of testing environment and constraints, the report of test case execution and the final analysis of tests results.

## 2.1 Basic Concepts

There exist many types of testing and many test strategies, however all of them share a same ultimate purpose: increasing the software engineer confidence in the proper functioning of the software. Towards this general goal, a piece of software can be tested to achieve various more direct objectives, all meant in fact to increase confidence, such as amongst others exposing potential design flaws or deviations from user's requirements, measuring the operational reliability and evaluating the performance characteristics. To serve each specific objective, different techniques can be adopted. Generally speaking, test techniques can be divided into two classes:

- Static analysis techniques, where the term “static” does not refer to the techniques themselves (one can use automated analysis tools), but is meant not to involve the execution of the tested system. Static techniques are applicable throughout the lifecycle to the various developed artifacts for different purposes, such as to check the adherence of the implementation to the specifications or to detect flaws in the code via inspection or review.
- Dynamic analysis techniques investigate the software in order to expose possible failures. The behavioural and performance properties of the program are also observed.

The focus of this deliverable is on dynamic test techniques, and where not otherwise specified testing is used as a synonym for “dynamic testing”.

In this deliverable we adopt the following general definition for software testing[4]: Software testing consists of the dynamic verification of the behaviour of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the specified expected behaviour.

This short definition attempts to include all essential testing concerns: the term *dynamic* means, as said, that testing implies executing the program on (valued) inputs; *finite* indicates that only a limited number of test cases can be executed during the testing phase, chosen from the whole test set, that can generally be considered infinite; *selected* refers to the test techniques adopted for selecting the test cases (and testers must be aware that different selection criteria may yield vastly different effectiveness); expected points out to the decision process adopted for establishing whether the observed outcomes of program execution are acceptable or not.

It is also important to clarify the terms “fault”, “error” and “failure” used in the rest of this deliverable. A failure is the manifested inability of the program to per-form the function required, i.e., a system malfunction evidenced by incorrect output, abnormal termination or unmet time and space constraints. The cause of a failure, e.g., a missing or incorrect piece of code, is a fault. A fault may remain undetected long time, until some event activates it. When this happens, it first brings the program into an intermediate unstable state(“error”), which, if and when propagates to the output, eventually causes the failure. The process of failure manifestation can be therefore summed up into a chain [6]:

Fault->Error->Failure

which can recursively iterate: a fault in turn can be caused by the failure of some other interacting system.

In any case what testing reveals are the failures and a consequent analysis stage is needed to identify the faults that caused them.

## 2.2 Test Strategies

The test strategies proposed for the pilot applications refer to the set of dynamic techniques [4], obtaining information of interest about a program by observing some executions. Because the set of input values can be considered infinite, an exhaustive exercise to test every input value becomes an impossible task. Thus

the only possible alternative is to observe only some samples of the program behaviour during testing. A test strategy therefore must be adopted to find a trade-off between the number of chosen inputs and overall time and effort dedicated to testing purposes. Different techniques can be applied depending on the objective and the effect that should be reached. Here we will describe only those more suitable according to the specifications available within WP9.

In particular the objective here will be verifying during the integration phase that the functional specifications are implemented correctly. Once the tests are selected and run, another crucial aspect of this phase is the so-called oracle problem, which means deciding whether the observed outcomes are acceptable or not.

### 2.2.1 Integration Test

During the development lifecycle of a software product, testing is performed at different levels and can involve the whole system or parts of it. Depending on the process model adopted, then, software testing activities can be articulated in different phases, each one addressing specific needs relative to different portions of a system.

Generally speaking, integration is the process by which software pieces or components are aggregated to create a larger component. Integration testing is specifically aimed at exposing the problems that can arise at this stage. Even though the single units are individually acceptable when tested in isolation, in fact, they could still result in incorrect or inconsistent behaviour when combined in order to build complex systems. For example, there could be an improper call or return sequence between two or more components[3] . Integration testing thus is aimed at verifying that each component interacts according to its specifications as defined during preliminary design. In particular, it mainly focuses on the communication interfaces among integrated components.

There are not many formalized approaches to integration testing in the literature, and practical methodologies rely essentially on good design sense and the tester's intuition. Integration testing of traditional systems was done substantially in either a non-incremental or an incremental approach.

In a non-incremental approach the components are linked together and tested all at once ("big-bang" testing)[5]. In the incremental approach, we find the classical "top-down" strategy, in which the modules are integrated one at a time, from the main program down to the subordinated ones, or "bottom-up", in which the tests are constructed starting from the modules at the lowest hierarchical level and then are progressively linked together up-wards, to construct the whole system. Usually in practice, a mixed approach is applied, as determined by external project factors (e.g., availability of modules, release policy, availability of testers and so on)[8].

In modern Object Oriented, distributed systems, approaches such as top-down or bottom-up integration and their practical derivatives, are no longer usable, as no "classical" hierarchy between components can be generally identified. Some other criteria for integration testing imply integrating the software components based on identified functional threads [5]. In this case the test is focused on those

classes used in reply to a particular input or system event (thread-based testing)[5]; or by testing together those classes that contribute to a particular use of the system.

# 3 Selection of strategies for Test Case Generation for Pilots Applications

Effective testing requires strategies to trade-off between the two opposing needs of amplifying testing thoroughness for which a high number of test cases would be desirable and reducing time and cost for which reducing the test cases would be better. Given that test resources are limited, it is of crucial importance how test cases are selected. Indeed, the problem of test case selection is the main topic in software testing research to the extent that in the literature "software testing" is often taken as a synonym for "test case selection".

A decision procedure for selecting the test cases is provided by a test criterion. A basic criterion is random testing, according to which the test inputs are picked purely randomly from the whole input domain according to a specified distribution, i.e., after assigning to the inputs different "weights" (more properly probabilities). For instance the uniform distribution does not make any distinction among the inputs, and any input has the same probability of being chosen.

In contrast with random testing a broad class of test criteria is referred to as *partition testing*. The underlying idea is that the program input domain is divided into subdomains within which it is assumed that the program behaves the same, i.e., for every point within a subdomain the program either succeeds or fails: we also call this the "test hypothesis". Therefore, thanks to this assumption only one or few points within each subdomain need to be checked, and this is what allows for getting a finite set of tests out of the infinite domain. Hence a partition testing criterion essentially provides a way to derive the subdomains.

A test criterion yielding the assumption that all test cases within a subdomain either succeed or fail is only an ideal, and would guarantee that any fulfilling test set of test cases always detect the same failures: in practice, the assumption is rarely satisfied, and different set of test cases fulfilling a same criterion may show varying effectiveness depending on how the test cases are picked within each subdomain.

Many are the factors of relevance when a test selection criterion has to be chosen. An important point to always keep in mind is that what makes a test a "good" one does not have a unique answer, but changes depending on the context, on the specific application, and on the goal for testing. The most common interpretation for "good" would be "able to detect many failures"; but again precision would require to specify what kind of failures, as it is well known and experimentally observed that different test criteria trigger different types of faults [5].

For the integration phase, among the available selection criteria the approach chosen for the Pilot applications is the *specification-based testing*. The documentation to be analysed is included in the Pilot Specifications described in Deliverable D9.1. In particular we selected two test strategies:

- A test strategy based on the Sequence Diagrams analysis, which combines a standard methodology for partition testing (such as CA Category Partition [7] and a boundary method.

- A test strategy based on Policy analysis

## 3.1 Use of Sequence Diagrams

The constraints taken into consideration for a test strategy selection based on Sequence Diagrams analysis have been:

- usability: use for test planning exactly the same UML diagrams developed for analysis and design, without requiring any additional formalism or ad-hoc effort specifically for testing purposes. Usability means that the test methodology should, as far as possible, adapt itself to the modelling notations and procedures in use, and note vice versa

- timeliness: according to the good software engineering principle that test planning should start as early as possible the test strategy should be applicable even when not all relevant scenarios are yet specified and the Sequence Diagrams are defined at a high abstraction level, with several of them yet sketchy. The plan will be as abstract as the diagrams processed and will be progressively refined as the diagrams become richer and more detailed with more information.

- incrementality: the strategy should be conceived for integration phase where integration testing is typically conducted in an incremental fashion, considering progressively larger parts of the system and addressing, at each incremental step, the functionalities and the interactions that are relevant at the level considered.

- Scale: the test strategy should keep under control the size of the test suite keeping the coverage of functional areas as wide as possible.

Considering these four features the test strategies selected for the Pilot applications have been a customised rework of the *UIT* (Use Interaction Test) methodology proposed in [2], as described in the rest of this section.

### 3.1.1 Use Interaction Test

UIT, largely inspired by the Category Partition method briefly described below, introduces two concepts, RUP process [9]: *Test Cases* and *Test Procedures.*

- A Test Case is the set of actions performed to test a possible objects interaction, with associated test inputs and execution conditions.

- A Test Procedure is a set of detailed instructions for setting up, executing, and evaluating the results of a given Test Case.

The final output of UIT is therefore a set of Test Procedures, derived exclusively by the Sequence Diagrams without requiring the introduction of additional formalisms.

#### 3.1.1.1 Category Partition Method

The Category Partition (CP) is a well-known and quite intuitive method proposed in the late 1980's [7]to derive functional tests from the specifications written in structured, semiformal language.

CP provides a systematic, formalized approach to partition testing that is one standard functional testing methodology. Generally speaking, partition testing is based on the simple idea that the input domain is first divided into several equivalence classes (also called partitions, although in order to be true partitions these should be non-overlapping, which is rarely the case in practice); then one or few tests are selected from within each of the identified partitions, as representative of the behaviour of the entire class.

The first step of the CP method is to analyse the functional requirements in order to divide the analysed system into functional units that can be tested separately. A functional unit can be a high-level function or a procedure of the implemented system. For each identified functional unit, the tester identifies the environmental conditions (the required system properties for a certain functional unit) and the parameters (explicit inputs for the unit) that are relevant for testing purposes: these are called the categories. The test cases are then selected by taking the significant values of each category, which in CP are called the choices. A complete set of test cases is obtained by taking all possible combinations of choices for all the categories. To prevent meaningless combinations or pairs of contradictory choices, the categories can be annotated with constraints, e.g., in a test case a choice from one category cannot occur together with certain choices from other categories.

The CP method has been implemented by the Test Development Environment (TDE), a product developed at Siemens Corporate Research [1]. TDE processes a test design written in the Test Specification Language (TSL). This language is based on the CP method, which identifies behavioral equivalence classes within the structure of a system under test. The CP method has encountered wide interest in the literature, and has inspired the further development of a large number of test methodologies, also using formal languages such as Z.

### 3.1.2 Category Partition applied to Sequence Diagrams

Here we present the modified version of the UIT methodology, called UIT_SD. Similarly to the UIT method, UIT_SD constructs the Test Procedures using solely information retrieved from the Sequence diagrams. UIT_SD is an incremental test methodology; it can be used at diverse levels of design refinement, with a direct correspondence between the level of detail of the scenario descriptions and the expressiveness of the Test Procedures derived. For each selected SD, the algorithm for Test Procedures generation is the following:

Define *Messages_Sequences*. Observing the temporal order of the messages along the vertical dimension of the SD, a Messages_Sequence is defined considering each message with no predecessor association, plus, if any, all the messages belonging to its nested activations. A Messages_Sequence represents a behaviour to be tested and describes the interactions among objects necessary for realizing the corresponding functionality.

Analyse possible sub cases: the messages involved in a derived Messages_Sequence may contain some feasibility conditions (e.g., if/else conditions).. If these feasibility conditions exist, a Messages_Sequence is divided into subcases, corresponding to the different possible choices.

Identify *Settings Categories:* for each resulting Messages_Sequence, we define the Settings Categories as the values or data structures that can influence its execution. In detail, they can be determined from all the messages involved, by considering their input parameters.

Determine *Choices:* for each Settings Category and for each Message belonging to a Messages_Sequence, the possible choices are identified as follows:

- for the Messages, they represent the list of specific situations, or relevant cases in which the messages can occur;
- for the Settings Categories, they are the set or range of input data that parameters or data structures can assume.

Determine *Constraints* among choices: the values of different choices in a Messages_Sequence may turn out to be either meaningless or even contradictory. To avoid this, the Category Partition methodology suggests introducing constraints among choices. These are specified by assigning to choices specific *Properties* used to check the compatibility with other choices belonging to the same Messages_Sequence, and by introducing the *IF Selectors*, which are conjunctions of previously assigned properties.

Derive *Test Procedures*: a Test Procedure is automatically generated for every possible combination of choices, for each category and message involved in a Messages_Sequence. For each analysed SD, a document, called the *Test Suite*, collects all the derived meaningful Test Procedures grouped by Messages_Sequences.

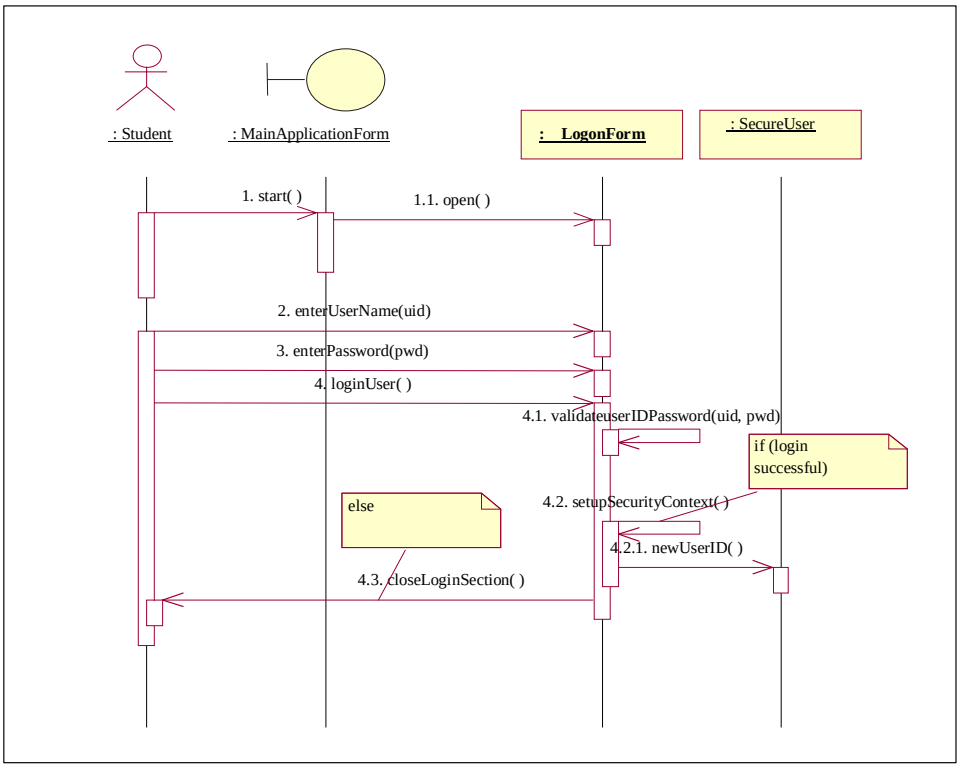


**Figure 1 Sequence Diagram "Login-Main Flow" from CRS project [9]**

Below, we report an example of UIT_SD application to the SD `Login-Main Flow` of Figure 1. Following the sequencing of messages along the vertical axis it is possible to initially define (Step1) four Messages_Sequences (M_S) such as:

- M_S1: 1.start()
  1.1.open()
- M_S2: 2.enterUserName(String)
- M_S3: 3.enterPassword(String)
- M_S4: 4.loginUser(),
  4.1validateUserIDPassword(String, String),
  4.2.setupSecurityContext(),
  4.2.1.newUserID(),
  4.3.closeLoginSection()

As described in Step 2, a feasibility condition in messages 4.2 and 4.3 can be observed: the value of `login successful` determines the execution of messages 4.2.1 or 4.3 so that Messages_Sequence 4 is split into two different subcases:

- M_S4.1: 4.loginUser(),
  4.1.validateIDPassword(String, String),
  4.2.setupSecurityContext(),
  4.2.1.newUserID()

- M_S4.2: 4.loginUser(),
  4.1.validateIDPassword(String, String),
  4.3.closeLoginSection()

For each derived Messages_Sequence, the Settings Categories can be identified (Step 3). In M_S4.1, for example, the categories are: `uid` and `pwd`, representing the

parameters of the messages involved. Then for each message and for each Settings Category it is necessary to determine the Choices (Step 4). Figure 2 shows the definition of Choices for M_S4.1 and the Constraints values (Step 5) associated to the Choices in square brackets. Finally, as described in Step 6, the relevant Test Procedures are generated. In Figure 3 we report, for the SD `Login-Main Flow`,a Test Procedures as derived by the proposed methodology.

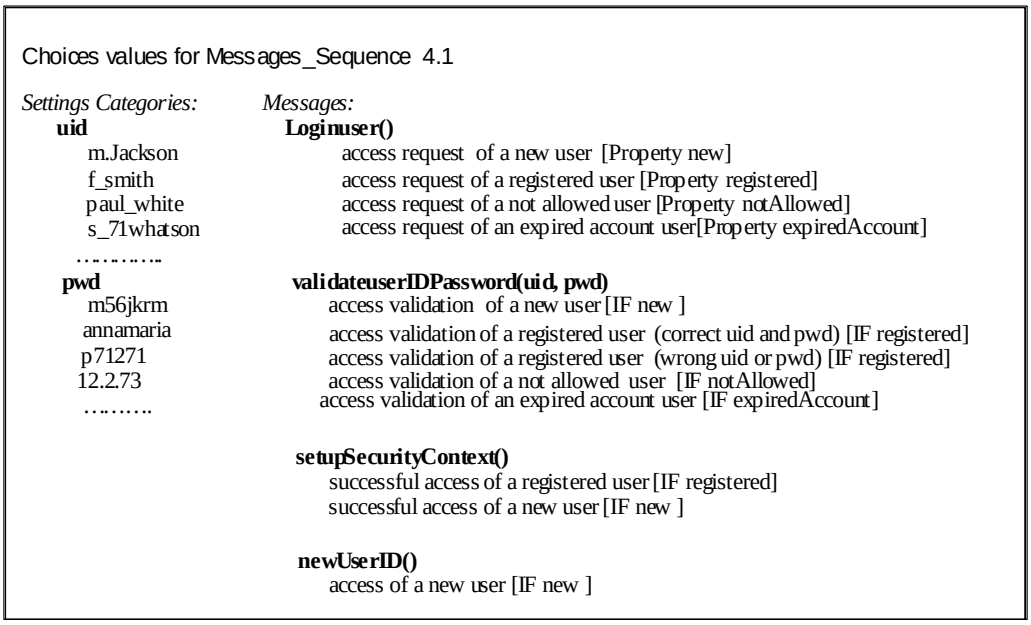


**Figure 2 Choices values for Messages_Sequence 4.1**

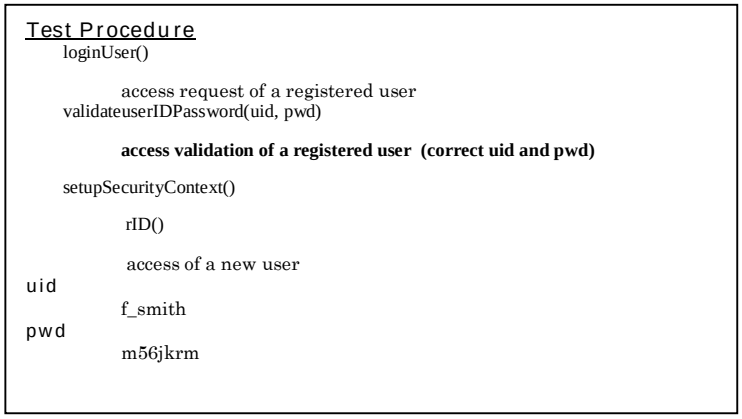


**Figure 3 Test Procedure example**

It is important to note that depending on the choices associated to each Settings Categories the test cases derived can also be used either to test boundary condition or for negative testing.
In the first case a special property called [single] limits the number of occurrence of a given values in the selected combination to 1. This is typically associated to the bounder values of an interval.
In the second case instead incorrect values can be associated to the choices and combined so to derive negative test cases. In this case a special property called [error] indicates a value class that need to be tried only once, in combination with non error values or parameters.

## 3.2 Use of Policies

As detailed in the Deliverable 10.2 chapter 4 the policies specification can be used for deriving requests. The idea is combining the values expressed in the policy for elements and attributes, so to obtain executable and meaningful requests. A simply random value assignment would not be sufficient for exercising all the policy functionality, thus the necessity of a component for the policy analysis.

Specifically the steps required are:

1. Define four values sets: SubjectSet, ResourceSet, ActionSet and EnvironmentSet respectively.
2. Look for the matching attributes and elements that appear in the *target* element of each *rule*, *policy* and *PolicySet* element included in the policy specification under test. They are defined in the *SubjectMatch*, *ResourceMatch*, *ActionMatch* and *EnvironmentMatch* sections of the *Subject*, *Resources*, Actions and *Environments* respectively.

- For each *SubjectMatch*, take the values of the AttributeValue elements and the values of the attributes named *AttributeId, Datatype, Issuer* and *SubjectCategory* of the *SubjectAttributeDesignator* element or the value of the *DataType* attribute of the *AttributeSelector* element and put them in the SubjectSet set. Note that the *Issuer* and *SubjectCategory* attributes are optional.

- For each *ResourceMatch* element take the values of the *AttributeValue* elements and the values of the attributes named *AttributeId*, *Datatype*, and Issuer (optional) of the *ResourceAttributeDesignator* element or the value of the *DataType* attribute of the *AttributeSelector* element and put them in the ResourceSet set.

- For each *ActionMatch* element take the values of the *AttributeValue* elements and the values of the attributes named *AttributeId*, *Datatype*, and *Issuer* (optional) of the *ActionAttributeDesignator* element, or the value of the *DataType* attribute of the *AttributeSelector* element and put them in the ActionSet set.

- For each *EnvironmentMatch* element take the values of the *AttributeValue* elements and the values of the attributes named *AttributeId*, *Datatype*, and *Issuer* (optional) of the *EnvironmentAttributeDesignator* element, or the value of the *DataType* attribute of the *AttributeSelector* element and put them in the EnvironmentSet set.

- For each policy rule, take all values of the *AttributeValue* elements of the *condition* section and put them in the SubjectSet, ResourceSet, ActionSet and EnvironmentSet if they refer to the *SubjectAttributeDesignator*, *ResourceAttributeDesignator*, *ActionAttributeDesignator* or *EnvironmentAttributeDesignator* elements respectively. Take then all values of attributes named *AttributeId*, *Datatype* and *Issuer* (optional) of the *condition* section and put them in the SubjectSet, ResourceSet, ActionSet and EnvironmentSet, if they are specified in the *SubjectAttributeDesignator*,

*ResourceAttributeDesignator*, *ActionAttributeDesignator* or *EnvironmentAttributeDesignator* elements respectively.

In Figure 4 we show the result of the application of the policy analysis component to the policy described below, which is an example of a simplified XACML policy ruling library access. Note that in this case the EnvironmentSet is empty and no values are associated to the Issuer and *SubjectCategory* attributes.

For robustness and negative testing purposes the policy analyzer component adds to each set of data(SubjectSet, ResourceSet, ActionSet, and EnvironmentSet) random values for elements and attributes.

```
<Policy
 xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 PolicyId="urn:va:xacml:2.0:policyExample"
 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:first-applicable">
 <Target/>
 <Rule RuleId="urn:oasis:names:tc:xacml:2.0:rule1"
  Effect="Deny">
  <Target>
   <Resources>
    <Resource>
     <ResourceMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
      <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#anyURI"
       >http://library.com/record/</AttributeValue>
      <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
       DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
     </ResourceMatch>
    </Resource>
   </Resources>
   <Actions>
    <Action>
     <ActionMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#string"
       >write </AttributeValue>
      <ActionAttributeDesignator
       AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
       DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </ActionMatch>
    </Action>
   </Actions>
  </Target>
  <Condition>
   <Apply
    FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-
least-one-member-of">
    <SubjectAttributeDesignator
     AttributeId="urn:oasis:names:tc:xacml:1.0:example:role"
     MustBePresent="false"
     DataType="http://www.w3.org/2001/XMLSchema#string"/>
    <Apply
```

```
     FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
     <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#string"
      >researcher</AttributeValue>
     <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#string"
      >professor</AttributeValue>
     <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#string"
      >staff</AttributeValue>
    </Apply>
   </Apply>
  </Condition>
 </Rule>
 <Rule RuleId="urn:oasis:names:tc:xacml:2.0:rule2"
  Effect="Permit">
  <Target>
   <Subjects>
    <Subject>
     <SubjectMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#string"
       >Julius</AttributeValue>
      <SubjectAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
       DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </SubjectMatch>
    </Subject>
   </Subjects>
   <Resources>
    <Resource>
     <ResourceMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
      <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#anyURI"
       >http://library.com/record/journals/</AttributeValue>
      <ResourceAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
       DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
     </ResourceMatch>
    </Resource>
   </Resources>
   <Actions>
    <Action>
     <ActionMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#string"
       >read</AttributeValue>
      <ActionAttributeDesignator
       AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
       DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </ActionMatch>
    </Action>
   </Actions>
  </Target>
 </Rule>
</Policy>
```

| AttributeId | DataType | AttributeValue |
|---|---|---|
| **SubjectSet** | | |
| role | string | professor<br>researcher<br>staff |
| subject-id | string | Julius |
| **ResourceSet** | | |
| resource-id | anyURI | http://library.com/record/ |
| resource-id | anyURI | http://library.com/record/journals/ |
| **ActionSet** | | |
| action-id | string | read |
| action-id | string | write |
| **EnvironmentSet** | | |
| – | – | – |

**Figure 4 Analysis of the policy presented above**

#### 3.2.1.1 Request values assignment

The final steps is the generation of the requests with the values of the SubjectSet, ResourceSet, ActionSet, and EnvironmentSet sets. Note that, the number of requests to be filled can be established according with the procedure described in the TAS3 Deliverable D10.2 Trustworthiness Architecture and Proof of Concept , by using the TAXI tool. It is out of the scope of this deliverable to go deeply into details of that tool. Here the purpose is to propose a procedure for requests generation that can be applied independently to any application for integration testing purposes.

In particular we define a *subject entity* as a combination of the values of elements and attributes of the SubjectSet set, and similarly the *resource entity*, the *action entity* and the *environment entity* as a combination of the values of the elements and attributes of the ResourceSet, ActionSet, and EnvironmentSet respectively. According to these definitions the values assignment is performed with the following steps:

- Derive the set of subject entities, resource entities, action entities and environment entities.
- Generate the *Values Set* with the combinations of subject entity, resource entity, action entity and environment entity following a combinatorial approach. In particular, the values are inserted in the set applying first the pair-wise combination, then if there are more intermediate requests to be filled the three-wise and finally the four-wise is applied so that all the possible combinations are generated .

Take the values from the *Values Set* one by one and use them to fill the elements and attributes of the *subject, resource, action* and *environment* of the intermediate requests. If the *Values Set* has been entirely used for filling the intermediate requests, then take again the *Values Set* values in the same order.

If there are further elements and attributes in the *subject, resource, action* and *environment* in the intermediate request, then fill them randomly picking values from the SubjectSet, ResourceSet, ActionSet and EnvironmentSet respectively, excluding the values of the subject *entities, resource entities, action entities* and *environment entities* already assigned.

Below an example of requests with the subject entities, resource entities and action entities obtained from the values of Figure 4, considering also the random values. Note that the values of the *AttributeId* and *Datatype* attributes and the value of the *AttributeValue* element of the *Environment* section of the request are random values included in the EnvironmentSet set. Also the value (*datastream:id*) of the *AttributeId* attribute of one Attribute element of the Resource section is a random value included in the ResourceSet.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request
 xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
 access_control-xacml-2.0-context-schema-os.xsd">
 <Subject>
  <Attribute
   AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
   DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>Julius</AttributeValue>
  </Attribute>
 </Subject>
 <Resource>
  <Attribute
   AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
   DataType="http://www.w3.org/2001/XMLSchema#anyURI">
<AttributeValue>http://library.com/record/journals/</AttributeValue
>
  </Attribute>
  <Attribute
   AttributeId="urn:fedora:names:fedora:2.1:resource:datastream:id"
   DataType="http://www.w3.org/2001/XMLSchema#anyURI">
   <AttributeValue>http://library.com/record/</AttributeValue>
  </Attribute>
 </Resource>
 <Action>
  <Attribute
   AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
   DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>read</AttributeValue>
  </Attribute>
 </Action>
 <Environment>
  <Attribute
   AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-
time"
   DataType="http://www.w3.org/2001/XMLSchema#time">
   <AttributeValue>08:23:47-05:00</AttributeValue>
  </Attribute>
 </Environment>
</Request>
```

# 4 Testing of the UK Employability Scenario

In this section we propose the test plan derived applying the UIT_SD test strategy to the pilot developed by the University of Nottingham. In particular in section 4.1 we list the test procedures derived from the various sequence diagrams developed for the employability scenario. As required by th UIT_SD methodology we also provide the list of setting categories and the respective choices used for test case derivation.

Due to the high number of test cases derived by combining all the choice values, in section 4.2 we only provide for each test procedure the number of test cases that have been derived and we show as example the structure of the most important test cases that have been executed. In particular we report some examples of successful test execution and the most important failures detected by a test case run.

The results of the test plan execution are discussed in section 4.3.

### 4.1.1 Login

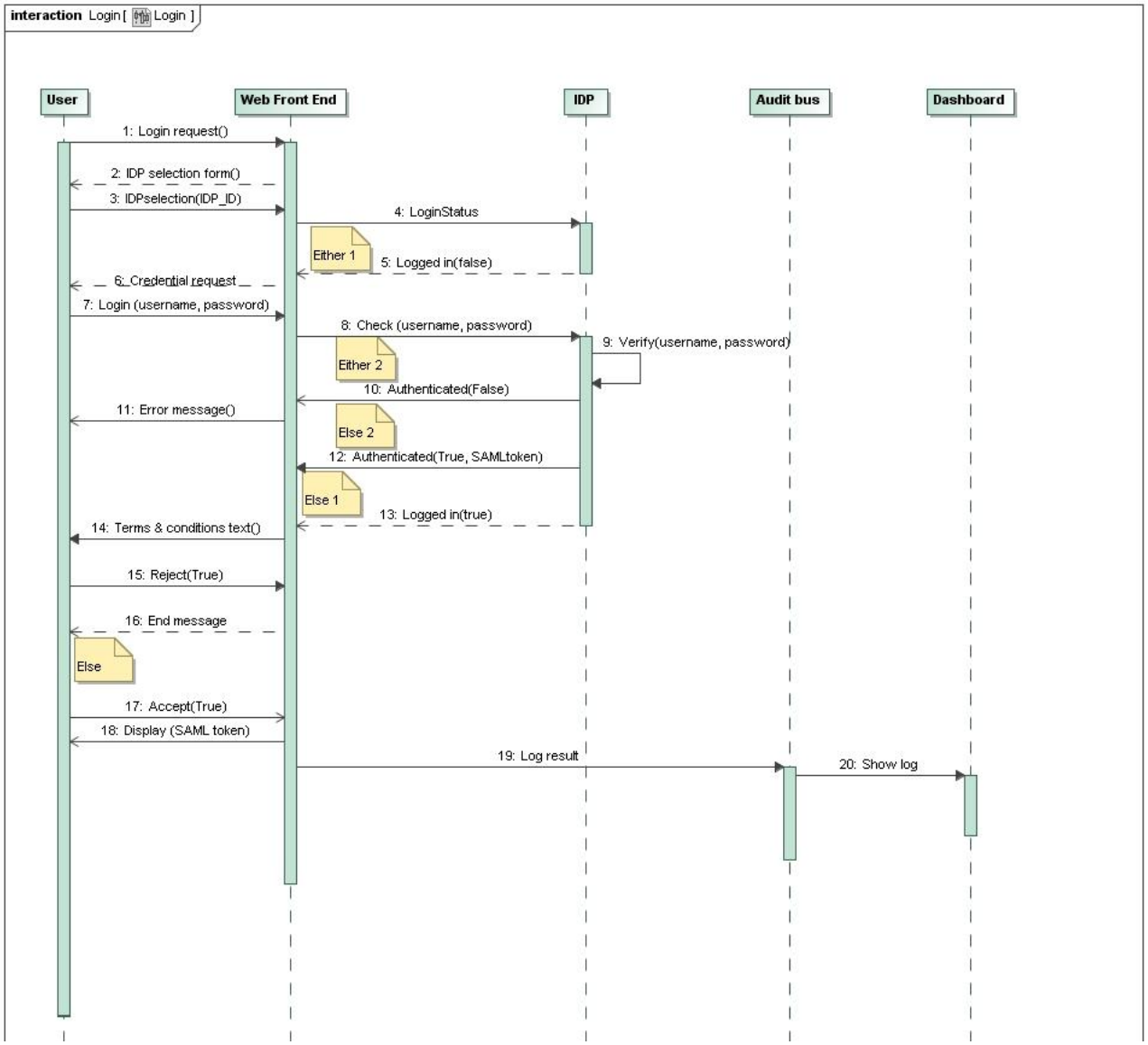


**Figure 5 Login SD**

List of test procedures:

Test Procedure 1
*Precondition:*
*Login request()*
*SettingCategoryValue*
*Post Condition:*
*Comment and results:*

Test Procedure 2.1
*Precondition:*
*IDP selection form(IDP_ID)*
*SettingCategoryValue*
*LoginStatus()*
*SettingCategoryValue*
*Logged in(boolean)*
*SettingCategoryValue*
*Credential request*
*SettingCategoryValue*
IDP_ID
*SettingCategoryValue*
boolvalue
false
*Post Condition: Proceed to credential request*
*Comment and results:*

Test Procedure 2.2
*Precondition:*
*IDP selection form(IDP_ID)*
*SettingCategoryValue*
*LoginStatus()*
*SettingCategoryValue*
*Logged in(boolean)*
*SettingCategoryValue*
*Terms and condition*
*SettingCategoryValue*

IDP_ID
*SettingCategoryValue*
boolvalue
true
*Post Condition:*
*Comment and results:*

Test Procedure 3.1
*Precondition: User not logged in*
*Login (username, password)*
*SettingCategoryValue*
*Check (username, password)*
*SettingCategoryValue*
*Verify (username,password)*
*SettingCategoryValue*
*Authenticated (boolvalue, SAML token)*
*SettingCategoryValue*
*Error message ()*
*SettingCategoryValue*
username
*SettingCategoryValue*
password
*SettingCategoryValue*
boolvalue
false
SAML token
empty
*Post Condition: login fails*
*Comment and results*:

Test Procedure 3.2
*Precondition: User not logged in*
*Login (username, password)*
*SettingCategoryValue*
*Check (username, password)*
*SettingCategoryValue*
*Verify (username,password)*
*SettingCategoryValue*
*Authenticated (boolvalue, SAML token)*
*SettingCategoryValue*
*Term&condition text ()*
*SettingCategoryValue*
username
*SettingCategoryValue*
password
*SettingCategoryValue*
boolvalue
true
SAML token
*SettingCategoryValue*
*Post Condition:*
*Comment and results*

Test Procedure 4
*Precondition:*
*Reject (Boolvalue)*
*SettingCategoryValue*
*End message ()*
boolvalue
true
*Post Condition:*
*Comment and results*:

Test Procedure 5
*Precondition:*
*Display (SAML token)*
*SettingCategoryValue*
*Accept (Boolvalue)*
*SettingCategoryValue*
*Log result ()*
*SettingCategoryValue*
*Show log ()*
*SettingCategoryValue*
boolvalue
true
SAML token
*SettingCategoryValue*
*Post Condition:*
*Comment and results*:

## Setting Values

### Messages:

Login request():
- User is not already logged in [Property error]
- User is already logged in

IDP selection form
- User has one possible IDP to choose from
- User has more than one possible IDP to choose from
- User has no possible IDP to choose from [Property error]

Login (username, password)
- User is not already logged in [Property error]
- User is already logged in

Check (username, password)
1) username and password sent from web front end to the IDP for validation

Verify (username, password)
- username and pwd correct
- username and pwd not correct [Property error]

Reply (boolvalue, SAML token) SAML
- SAML token is only returned if the boolvalue is True

Error message (text)
- Incorrect username and/or password

Terms&conditions text (web form)
- accept
- reject

Reject (boolvalue)
End message (text)
- Process terminated

Accept (boolvalue)
Display (SAML token)
Log result ()
Show log ()
- data displayed in Dashboard

**Parameters:**

Username:
- Betty
- Sandra [Property error]

Password: Betty
- Betty
- Sandra[Property error]

boolvalue
- false
- true

SAML token
- empty [Property error],
- UserID tas3\betty
  Surname: Hobkins
  Given name: Betty
  DOB: 1974-07-14
  Gender: female
  Affiliation: Nottingham
  Coursename: management
  Entitlement: student

### 4.1.2 Programme matching

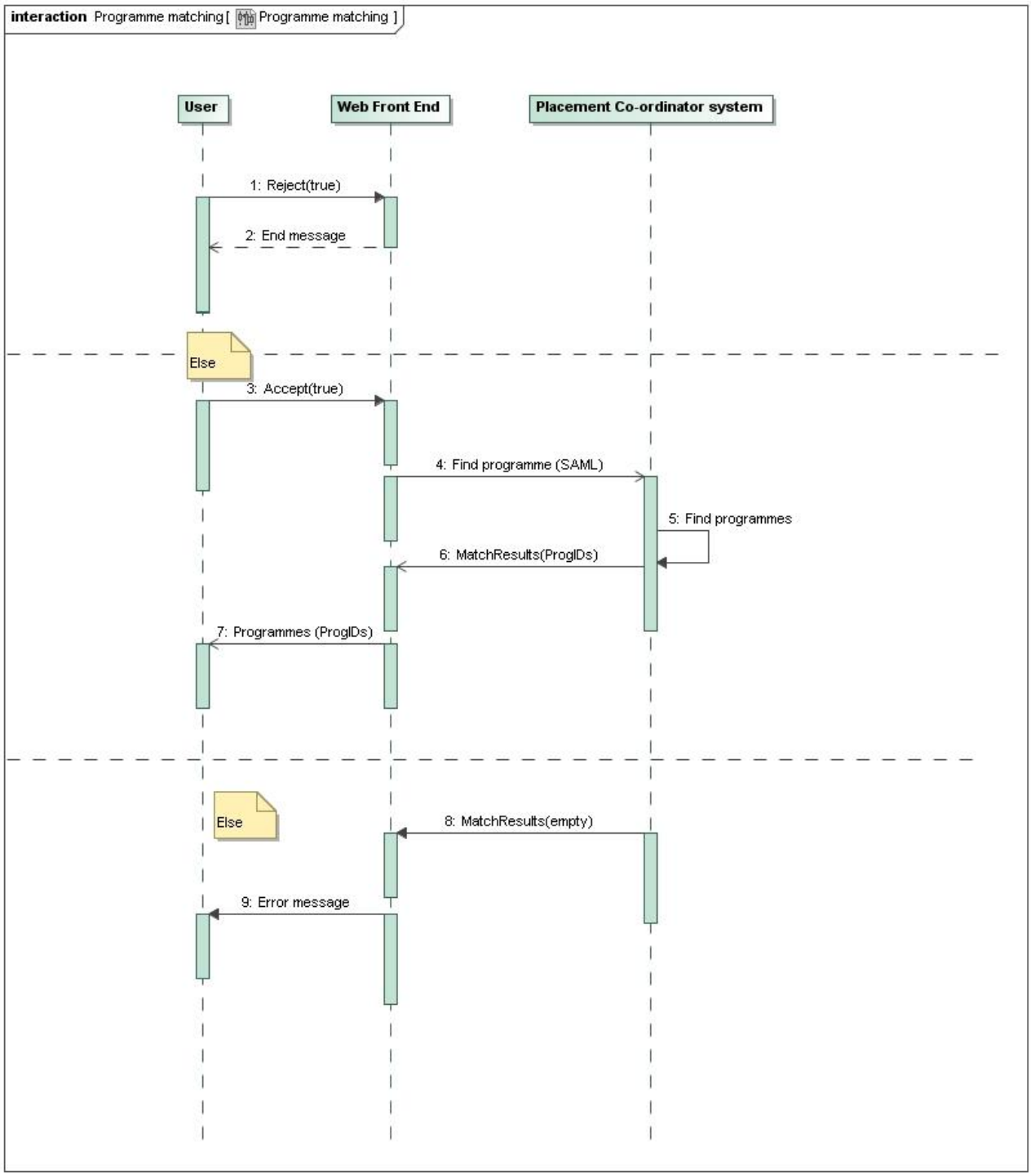


**Figure 6 Programme matching SD**

List of test procedures:

Test Procedure 1.1

*Precondition: User logged in, SAML token returned from IDP and displayed to User by Web Front End*

Reject(boolvalue)
*SettingCategoryValue*
End Message ()
*SettingCategoryValue*
boolvalue
true

*Post Condition: Condition: user remains logged in ; process terminated*
*Comment and results*:

Test Procedure 1.2.1
*Precondition: login successful, SAML token returned from IDP and displayed to User by Web Front End*
Accept(Boolvalue)
*SettingCategoryValue*
Find programme (SAMLtoken)
*SettingCategoryValue*
Find programmes ()
*SettingCategoryValue*
MatchResults(ProgIDs)
*SettingCategoryValue*
Programmes (ProgIDs)
*SettingCategoryValue*
boolvalue
true
SAML token
*SettingCategoryValue*
ProgIDs
*SettingCategoryValue*
*Post Condition: User can see and choose from list of programmes*

## Setting Values

### Messages:

EndMessage()
- Process terminated

FindProgrammes
- One programme found
- More than one programme found
- No programmes found [property error]

**Programmes**
- One programme found
- More than one programme found

**ErrorMessage**
- No programmes found

### Parameters:

**Boolvalue:**
- false,
- true

**SAMLtoken:**
- Affiliation: Nottingham
  Coursename: Management
  Entitlement: Student

**ProgIDs:**
- UK2EU
- EU2UK
- Nottingham

*Comment and results*:

Test Procedure 1.2.2
*Precondition:login : login successful, SAML token returned from IDP and displayed to User by Web Front End*

Accept(Boolvalue)
*SettingCategoryValue*
Find programme (SAMLtoken)
*SettingCategoryValue*
Find programmes ()
*SettingCategoryValue*
MatchResults (ProgIDs)
*SettingCategoryValue*
Error message ()
*SettingCategoryValue*
boolvalue
true
SAML token
*SettingCategoryValue*
ProgIDs
*empty*
*Post Condition: No matching programmes found*
*Comment and results*:

### 4.1.3 Registration

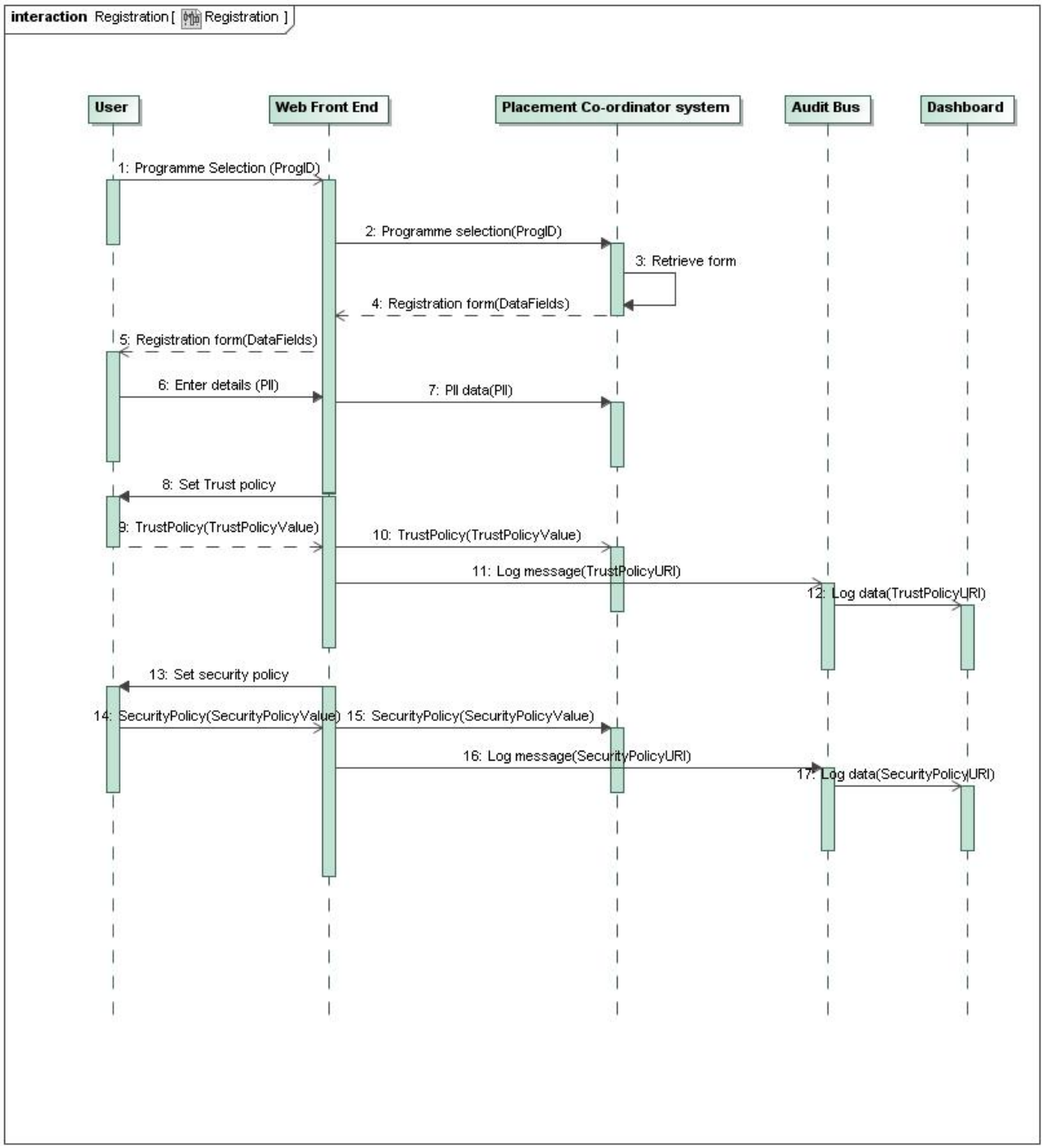


**Figure 7 Registration SD**

List of test procedures:

Test Procedure 1

*Precondition: Programme matching has completed successfully; User has a list of one or more Programmes to choose from*

Programme Selection (ProgID)

*SettingCategoryValue*

Retrieve form (ProgID)

*SettingCategoryValue*

Registration form(DataFields)

*SettingCategoryValue*

ProgID

*SettingCategoryValue*
DataFields
*SettingCategoryValue*
*Post Condition Correct form presented to User*
*Comment and results*:

Test Procedure 2
*Precondition: User has chosen Programme; correct registration form has been presented*
Enter details(PII)
*SettingCategoryValue*

PIIData(PII)
*SettingCategoryValue*
PII
*SettingCategoryValue*
*Post Condition: data logged, User can set policies*
*Comment and results*:

Test Procedure 3
*Precondition: : PII has been entered, Trust policy requested*
Trust policy (TrustPolicyValue)
*SettingCategoryValue*
Trust policy (TrustPolicyValue)
*SettingCategoryValue*
Log message (TrustPolicyURI)
*SettingCategoryValue*
Log data (TrustPolicyURI)
*SettingCategoryValue*
TrutPolicyValue
*SettingCategoryValue*
TrutPolicyURI
*SettingCategoryValue*

*Post Condition: Trust policy setting visible in Dashboard log*
*Comment and results*:

Test Procedure 4
*Precondition: PII has been entered, Trust policy has been set, set security requested*
Security policy (SecurityPolicyValue)
*SettingCategoryValue*
Security policy (SecurityPolicyValue)
*SettingCategoryValue*
Log message (SecuritytPolicyURI)
*SettingCategoryValue*
Log data (SecuritytPolicyURI)
*SettingCategoryValue*
SecurityPolicyValue
*SettingCategoryValue*
SecurityPolicyURIalue
*SettingCategoryValue*
*Post Condition: Security policy setting visible in Dashboard log*
*Comment and results*:

# Setting Values

## Messages:

**Programmes:**
- One programme returned
- More than one programme returned
- No programmes returned []

**Trust policy**
- Trust all
- Average Feedback
- Most reputable users
- Top service providers

**Security policy:**
- Allow all – no access restrictions
- Deny all – no access allowed

## Parameters:

**ProgID**:
- UK2EU
- EU2UK
- Nottingham

**DataFields**:
- HTML Post return

**PII:**
- Address,
  Work Experience,
  References,
  Qualifications

**TrustPolicyValue:** Integer 0-10
- 0 Property[single]
- 5
- 10 Property[single]
- 11 Property[error]
- empty Property[error]

**SecurityPolicyValue**:
- trustall
- denyall

## 4.1.4 Service discovery

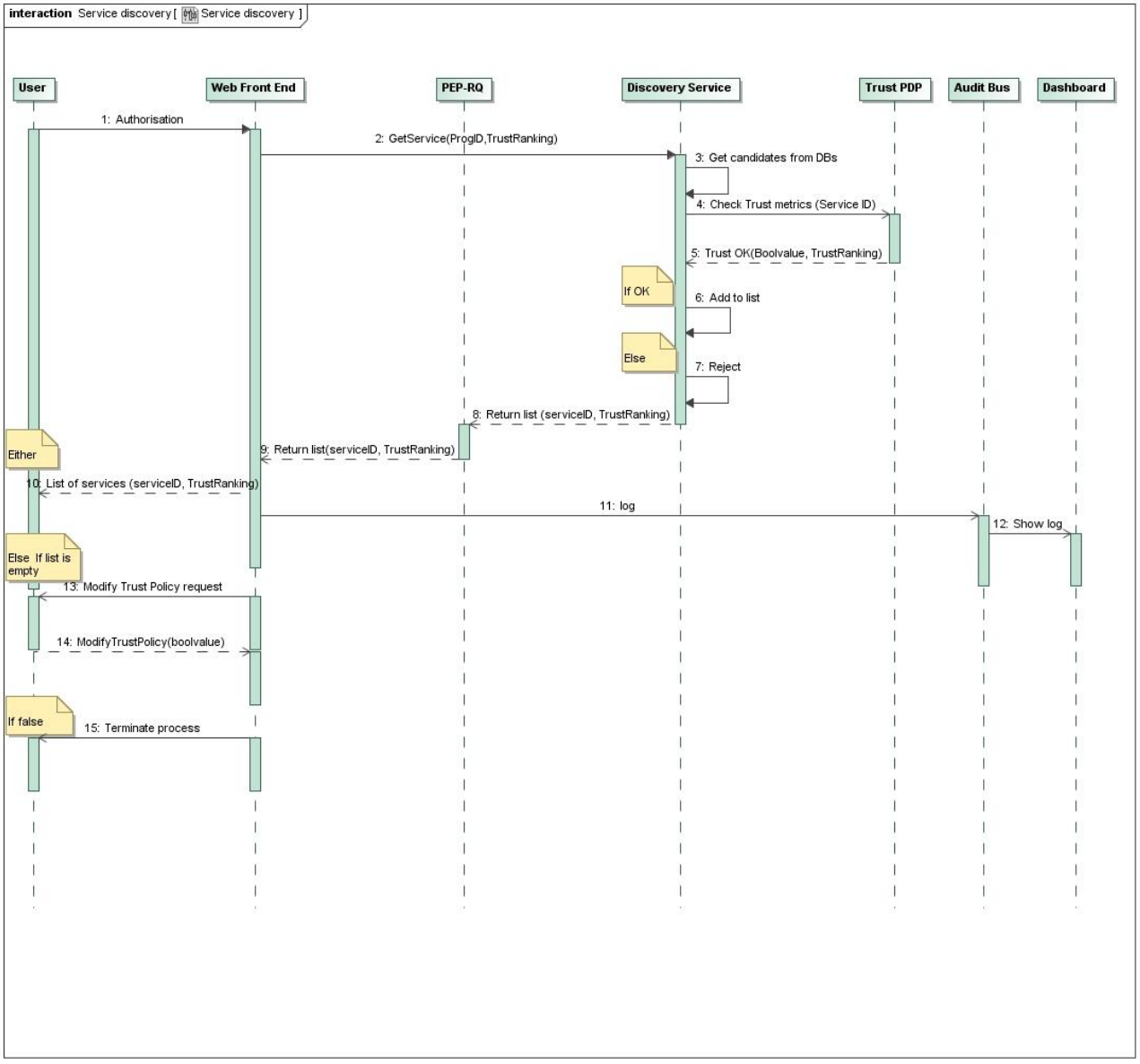


**Figure 8 Service discovery SD**

List of test procedures:

Test Procedure 1.1

*Precondition: User has one or more programmes to choose from; User has set Trust Policy; User has set Security Policy*

Authorization()
*SettingCategoryValue*
Get service (ProgID, TrustRanking)
*SettingCategoryValue*
Get candidates form DBs
*SettingCategoryValue*
Check Trust metrics (Service ID)
*SettingCategoryValue*
Trust metrics (Boolvalue, TrustRanking)
*Trust level acceptable*
Add to list
*SettingCategoryValue*
Return list (TrustRankin)
*SettingCategoryValue*
Return list (TrustRankin)
*SettingCategoryValue*
List of services (service ID, TrustRanking)
*SettingCategoryValue*
log ()
*SettingCategoryValue*
Show log
*SettingCategoryValue*
ProgID
*SettingCategoryValue*
TrustRunking
*SettingCategoryValue*
Service ID
*SettingCategoryValue*

boolvalue
*SettingCategoryValue*
*Post Condition: Service providers added to list to return to user*
*Comment and results*:

Test Procedure 1,2
*Precondition: User has one or more programmes to choose from; User has set Trust Policy; User has set Security Policy, User has to modify the trust policy*

Authorization()
*SettingCategoryValue*
Get service (Type, PII, Trust level)
*SettingCategoryValue*
Get candidates form DBs
*SettingCategoryValue*
Check Trust metrics (Service ID)
*Trust level not acceptable*
Trust metrics (value)
*SettingCategoryValue*
Reject
*SettingCategoryValue*
Modify trust policy request
*SettingCategoryValue*
ProgID
*SettingCategoryValue*
TrustRunking
*SettingCategoryValue*
boolvalue
*SettingCategoryValue*
*Post Condition: Service provider not added to list, user does not see it*
*Comment and results*:

Test Procedure 2.1
*Precondition: User has one or more programmes to choose from; User has set Trust Policy; User has set Security Policy*
Modify trust policy (Boolean)
*SettingCategoryValue*
boolvalue
true
*Post Condition:*
*Comment and results*:

Test Procedure 2.2
*Precondition: User has one or more programmes to choose from; User has set Trust Policy; User has set Security Policy*
Modify trust policy (Boolean)
*SettingCategoryValue*
Terminate process
*SettingCategoryValue*

boolvalue
false
*Post Condition:*
*Comment and results*:

# Setting Values

## Messages:

**Authorisation**
- User starts process

**Get candidates from DBs**
- One entry matches service type
- More than one entry matches service type
- No entries match service type

**Check Trust metrics**
- Trust ranking of SP is within range, added to list
- Trust ranking of SP is out of range, not added to list

**Return list**
- List has one entry
- List has more than one entry
- List has no entries

**ModifyTrustPolicy**
- Select weaker policy (return to policy selection)
- Terminate process
-

## Parameters:

**ProgID:**
- UK2EU
- EU2UK
- Nottingham

**PII:**
- Address,
  Work Experience,
  References,
  Qualifications

**TrustRanking:** Integer 0-10
- 0 Property[single]
- 5
- 10 Property[single]
- 11 Property[error]
- empty Property[error]

**ServiceID:**
- MatchingService1
- MatchingService2,
- MatchingService3

**Boolvalue:**
- True
- false

## 4.1.5 Service execution

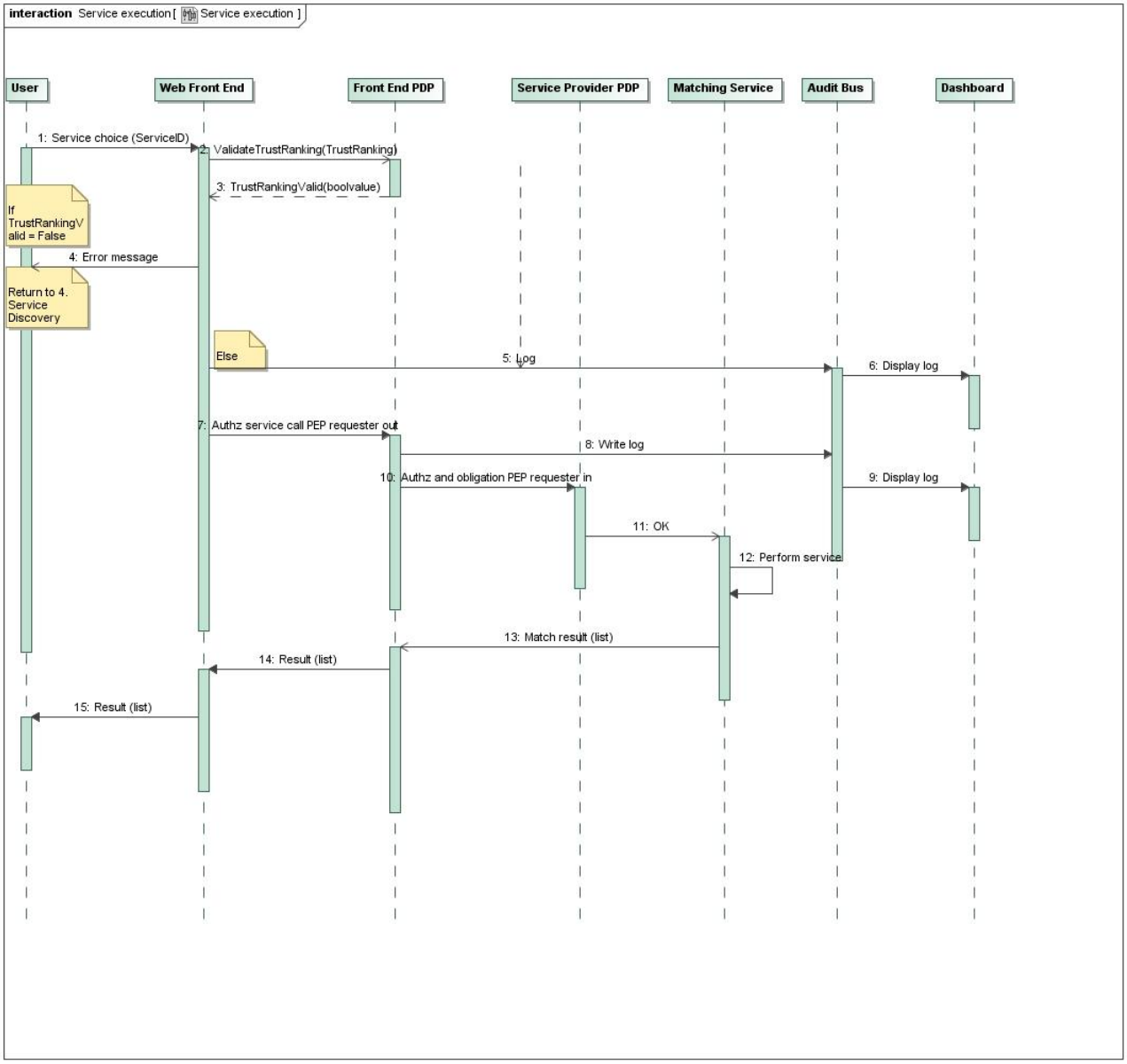


**Figure 9 Service execution SD**

List of test procedures:

Test Procedure 1.1

*Precondition: User has a list of services to choose from*

Service choice (Service ID)
*SettingCategoryValue*
Validate Trust ranking(TrustRanking)
*Trust ranking is still within parameters set by user*
TrustRankingValid (Boolvalue)
*SettingCategoryValue*
log
*SettingCategoryValue*
Display log
*SettingCategoryValue*
Authz service call PEP requester out
*SettingCategoryValue*
Write log
*SettingCategoryValue*
Display log
*SettingCategoryValue*
Authz and obligation PEP requester in
*SettingCategoryValue*
OK
*SettingCategoryValue*
Perform service
*SettingCategoryValue*
Match result (list)
*SettingCategoryValue*
Result (list)
*SettingCategoryValue*
Result (list)
*SettingCategoryValue*
TrustRanking
*SettingCategoryValue*
Boolvalue
True
Service ID

*SettingCategoryValue*
*list*
*SettingCategoryValue*
*Post Condition:*
*User can select vacancy and make manual application;*
*User can leave feedback*
*comment and results*:

Test Procedure 1. 2
*Precondition:*
Service choice (Service ID)
*SettingCategoryValue*
Validate Trust ranking(TrustRanking)
*Trust ranking no longer within parameters set by user*
TrustRankingValid (Boolvalue)
*SettingCategoryValue*
Error message
*SettingCategoryValue*
TrustRanking
*SettingCategoryValue*
Service ID
*SettingCategoryValue*
Boolvalue
False
*Post Condition: Follow the activity described in Service discovery*
*Comment and results*:

## Setting Values

### Parameters:

**TrustRanking:** Integer 0-10
- 0 Property[single]
- 5
- 10 Property[single]
- 11 Property[error]
- empty Property[error]

**ServiceID:**
- MatchingService1
- MatchingService2,
- MatchingService3

**Boolvalue:**
- True
- false

# 4.1.6 Feedback

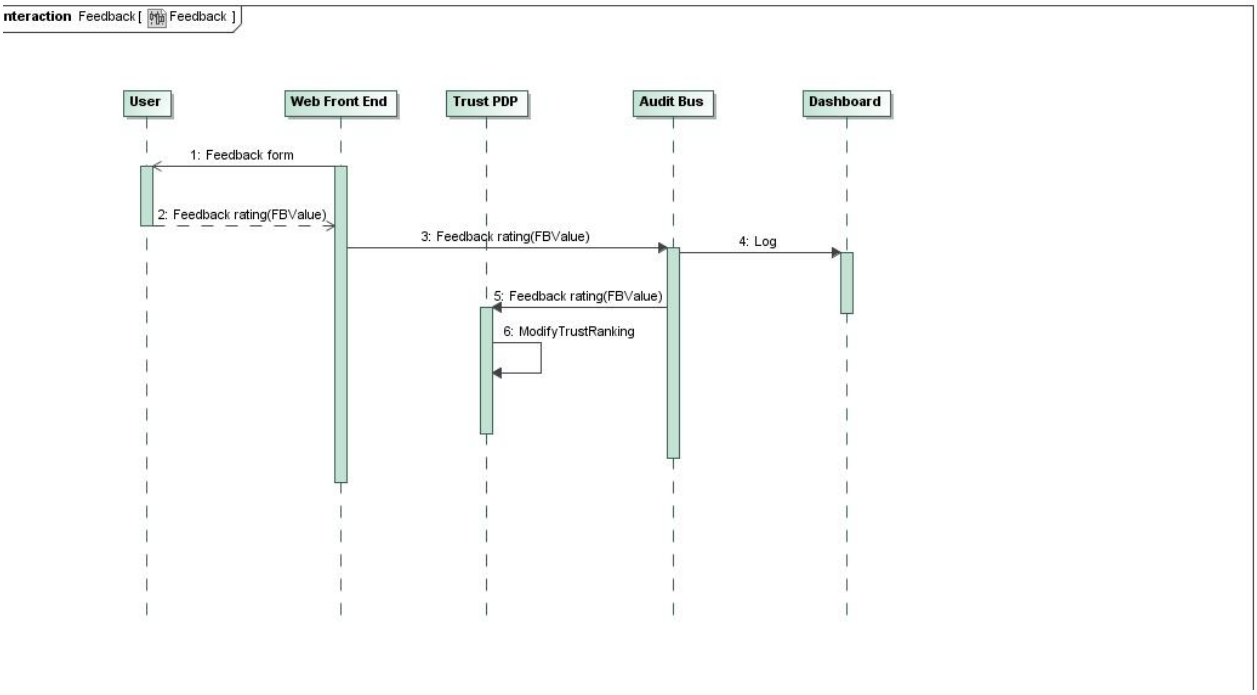


**Figure 10 Feedback SD**

## List of test procedures:

Test Procedure 1
*Precondition: Matching process completed; Feedback form received*
Feedback rating(FBValue)
*SettingCategoryValue*
Feedback rating(FBValue)

*SettingCategoryValue*
Log
*SettingCategoryValue*
Feedback rating (FBValue)
*SettingCategoryValue*
ModifyTrustRanking
*Post Condition: TrustPDP modifies SP trust ranking for future transactions based on new feedback received*
*Comment and results*:

### Setting Values

**Messages:**
**Feedback rating()**
- User leaves feedback
- User chooses not to leave feedback

**Modify Trust Ranking()**
- Trust ranking falls
- Trust ranking increases
- Trust ranking remains the same

**Parameters:**
**FBValue:**
- Positive
- negative
- none

## 4.1.7 Receipt

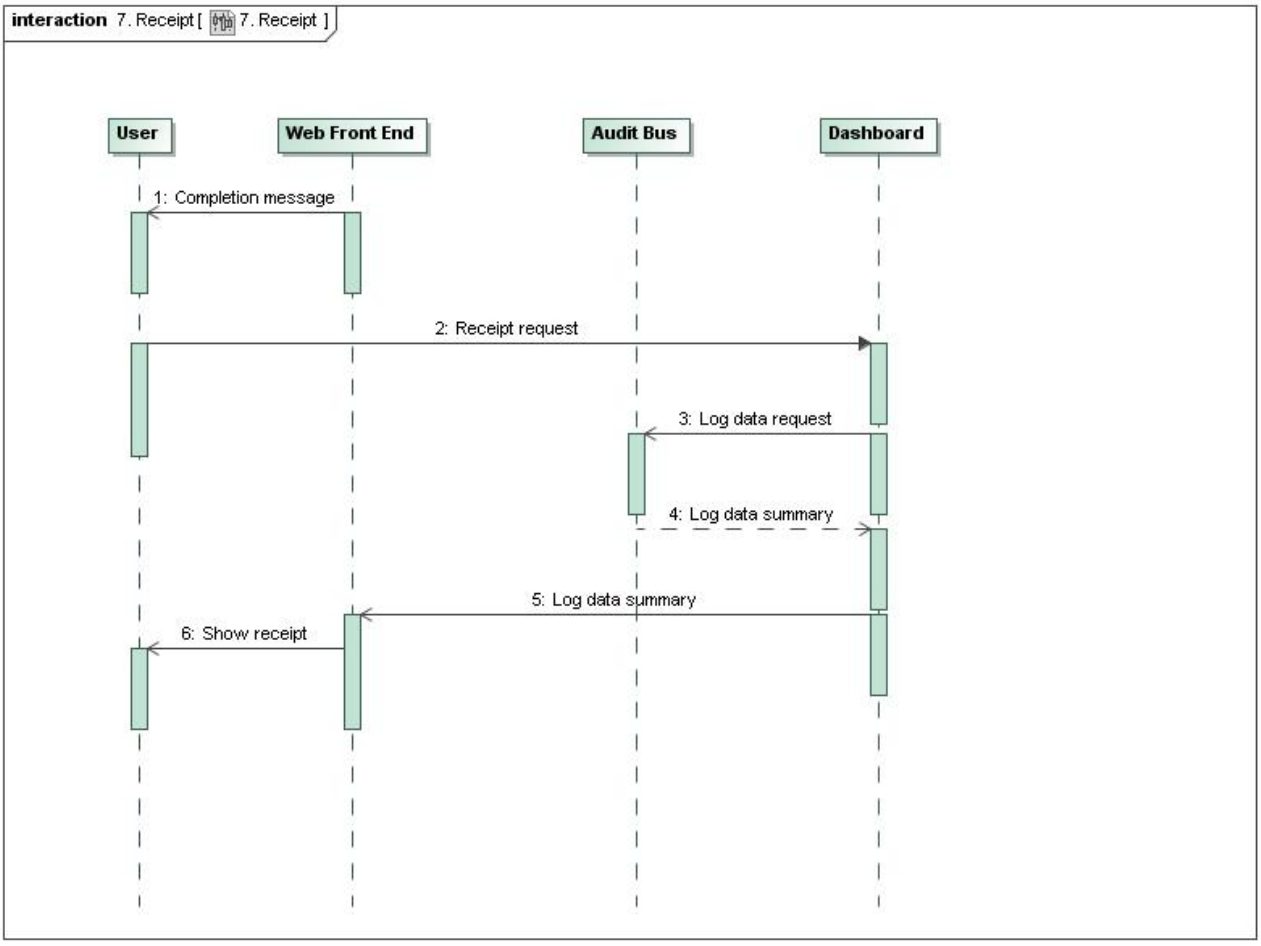


**Figure 11 Receipt SD**

List of test procedures:

Test Procedure 1
*Precondition:*
Receipt request
*A request of completion message has been received from web front end*
Log data request
*SettingCategoryValue*
Log data summary1
*SettingCategoryValue*
Log data summary2
*SettingCategoryValue*
Show receipt
*SettingCategoryValue*
*Post Condition: workflow terminated*
*Comment and results*:

## 4.2 Test case derived

We report in the table below the number of test cases derived by each test procedure combining the setting values provided. Moreover we provide some examples of the test cases that have been executed. In particular we report some examples of successful test execution and the most important failures detected by a test case run.

Note that for reporting the test result we exploited the structure of the test case itself. Thus once executed we report in the field "comment and results" the pass or fail verdict and in this last case the corrective action executed so to fix the observed failure.

| Name of the SD | Test procedure | Valid test cases | Invalid test cases |
|---|---|---|---|
| Login | 1 | 2 | 0 |
| | 2.1 | 2 | 1 |
| | 2.2 | 2 | 1 |
| | 3.1 | 1 | 4 |
| | 3.2 | 2 | 4 |
| | 4 | 1 | |
| | 5 | 1 | |
| Program matching | 1.1 | 1 | |
| | 1.2.1 | 12 | 1 |
| | 1.2.2 | 2 | 1 |
| Registration | 1 | 3 | 1 |
| | 2 | 1 | |
| | 3 | 6 | 2 |
| | 4 | 4 | |
| Service discovery | 1.1 | 326 | 2 |
| | 1.2 | 38 | 2 |

| | | | |
|---|---|---|---|
| | 2.1 | 2 | |
| | 2.2 | 2 | |
| Service execution | 1.1 | 5 | 2 |
| | 1.2 | 5 | 2 |
| Feedback | 1 | 18 | |
| receipt | 1 | 1 | |
| Total | | 437 | 23 |

## 4.2.1 Some of the most meaningful test cases

SD Login: Test Procedure 1: Test case
*Precondition: User is not already logged in*
Login request()
IDP selection form()
*User has more than one possible IDP to choose from*
*Post Condition: User is able to select IDP*
*Comment and results*: Success

SD Login: Test Procedure 3.1: Test case
*Precondition: User is not already logged in*
*Login (username, password)*
*Check (username, password)*
*Verify (username,password)*
*username and pwd correct*
*Authenticated (boolvalue, SAML token)*
*Error message ()*
username
*Sandra*
password
*Sandra*
boolvalue
false
SAML token
empty
*Post Condition: User is not authenticated successfully and not logged in*
*Comment and results*: Success

SD Login: Test Procedure 3.2: Test case
*Precondition:*
*Login (username, password)*
*User is not already logged in*
*Check (username, password)*
*Verify (username,password)*
*username and pwd correct*
*Authenticated (boolvalue, SAML token)*
*Error message ()*
username
*Betty*
password
*Betty*
boolvalue
true
SAML token
UserID tas3\betty
Surname: Hobkins
Given name: Betty
DOB: 1974-07-14
Gender: female
Affiliation: Nottingham
Coursename: management
Entitlement: student

*Post Condition: User is not authenticated successfully and not logged in*
*Comment and results*: Initial failure: parameter in ZXID changed, resulting in success

SD Programme Matching: Test Procedure 1.2.1

*Precondition: login successful, SAML token returned from IDP and displayed to User by Web Front End*

Accept(Boolvalue)
Find programme (SAMLtoken)
Find programmes ()
MatchResults(ProgIDs)
Programmes (ProgIDs)

boolvalue
true

SAML token
Affiliation: Nottingham
Coursename: management
Entitlement: student

ProgIDs
EU2UK, UK2EU, Nottingham

*Post Condition: User can see and choose from list of programmes*

*Comment and results*: Success, all 3 programmes returned. Further tests need to be done with larger numbers of programmes and other users in later iterations

## SD Registration: Test Procedure 3

*Precondition: : PII has been entered, Trust policy requested*

Trust policy (TrustPolicyValue)
Trust policy (TrustPolicyValue)
Log message (TrustPolicyURI)
Log data (TrustPolicyURI)

TrutPolicyValue
Top service providers

TrustPolicyURI
Identifier for trust policy

*Post Condition: Trust policy setting visible in Dashboard log*

*Comment and results*: Fail: trust policy value set but not communicated to Dashboard. SAWS has some problems with concurrent requests (probably writing requests). 2 clients used the Audit Bus to send messages to SAWS and after 5,6 messages the complete Tomcat Server went down. A new version of SAWS offered a partial solution, but for final demo tests it was replaced by a MySQL database as a placeholder.

## SD Service Discovery: Test Procedure 1.1

*Precondition: User has one or more programmes to choose from; User has set Trust Policy; User has set Security Policy*

Authorization()
Authorised
Get service (ProgID, TrustRanking)
Get candidates from DBs
Check Trust metrics (Service ID)
Trust metrics (Boolvalue, TrustRanking)
Add to list
Return list (TrustRanking)
Return list (TrustRanking)
List of services (service ID, TrustRanking)
log ()
Show log

ProgID
Nottingham

TrustRanking
10

Service ID
MatchingService1, MatchingService2, MatchingService3

boolvalue
True

*Post Condition: Service providers added to list to return to user*

*Comment and results*: Fail: Only 2 out of 3 endpoints returned. Root causes identified as:
1. The service had opted to create metadata manually by editing a file on the web server rather than automatically (o=B).
2. The metadata, corresponding to the ProviderID in the EPR registration, was not available. This was a simple file naming issue. Nevertheless, enough to get computer confused.
3. The metadata, after being named correctly and made available, still had inconsistency between the URL and the entityID field inside the metadata.
4. Some of the metadata was cached by the test participants such that the fixes did not immediately become used by all participants.
5. After fixing metadata (be sure to fix connectivity to the metadata as well if needed) and after everybody had restarted their services, the problem seems to be gone

## SD Service Discovery : Test Procedure 1.2

*Precondition: User has one or more programmes to choose from; User has set Trust Policy; User has set Security Policy, User has to modify the trust policy*

Authorization()
Get service (Type, PII, Trust level)
Get candidates form DBs
Check Trust metrics (Service ID)
Trust metrics (value)
Reject
Modify trust policy request

ProgID
Nottingham
TrustRanking
10
boolvalue
False
*Post Condition: Service provider not added to list, user does not see it*
*Comment and results*: Success

SD Service Execution: Test Procedure 1.1
*Precondition: User has a list of services to choose from*
Service choice (Service ID)
MatchingService1
Validate Trust ranking(TrustRanking)
*Trust ranking is still within parameters set by user*
TrustRankingValid (Boolvalue)
log ()
Display log ()
Authz service call PEP requester out ()
Write log ()
Display log ()
Authz and obligation PEP requester in ()
OK
Boolvalue
Perform service ()
Match result (list)
Result (list)
Result (list)

TrustRanking
10
Boolvalue
True
Service ID
MatchingService1
list
MatchingService1
*Post Condition:*
*User can select vacancy and make manual application; User can leave feedback*

*Comment and results*: Fail: issue with Matching Service timing out beyond timeout threshold of BPEL engine. Resolved by restarting container.

SD Service Execution: Test Procedure 1. 2
*Precondition:*
Service choice (Service ID)
MatchingService2
Validate Trust ranking(TrustRanking)
*Trust ranking no longer within parameters set by user*
TrustRankingValid (Boolvalue)
False
Error message ()

TrustRanking
10
Service ID
MatchingService2
Boolvalue
False
*Post Condition: Follow the activity described in Service discovery*
*Comment and results*: Success. Changing the Trust value of the service in the DB means it is no longer returned.

SD Feedback: Test Procedure 1
*Precondition: Matching process completed; Feedback form received*
Feedback rating(FBValue)
*SettingCategoryValue*
Feedback rating(FBValue)
*SettingCategoryValue*
Log
*SettingCategoryValue*
Feedback rating (FBValue)
*SettingCategoryValue*
ModifyTrustRanking
*Post Condition: TrustPDP modifies SP trust ranking for future transactions based on new feedback received*
*Comment and results*: Success, new ranking carries through to further runs of the entire process

## 4.3 Discussion and lessons learned

Integration testing relies on thorough component testing, but also has to be done iteratively in parallel with component testing. Each informs the other: integration testing reveals bugs and features in individual components as they are combined, and the testing of components reveals features which need to be considered in the integration process or to be accommodated by other components. A rapid prototyping approach to development facilitates this process but places on developers and integrators the requirement to document approaches and changes, as well as decisions made and the reasons for making them.

For the UK Employability integration trial, components were integrated in a lab situation and testing carried out first remotely and then with all developers and component machines in a local environment. We were able to demonstrate that some components were more mature than others, and found that some components appeared completely functional until repeated consecutive test runs were made, revealing issues with timing out and cacheing. Furthermore, as this was a small-scale demonstration where data sets used were limited and static, the next stage of development will need to facilitate greater scalability and use of dynamic data. Test plans will need to be developed to reflect this.

# 5 Testing of the Healthcare Scenario

In this section we propose the test plan derived applying the UIT_SD test strategy to the integration trial developed by Custodix for the healthcare scenario. In particular in section 5.1 we list the test procedures derived from the scenario representing the interaction with the Primary Care Physician Logon. As required by th UIT_SD methodology we also provide the list of setting categories and the respective choices used for test case derivation.

Note that the same flow of activities is required also when a Patient tries to logon To avoid the duplication of diagrams we differentiate the scenarios by using different values of setting categories

Due to the high number of test cases derived by combining all the choice values, in section 5.2 we only provide for each test procedure the number of test cases that have been derived and we show as example the structure of the most important test cases that have been executed. In particular we report some examples of successful test execution and the most important failures detected by a test case run.

The results of the test plan execution are discussed in section 5.3.

## 5.1.1 Primary Care Physician Logon

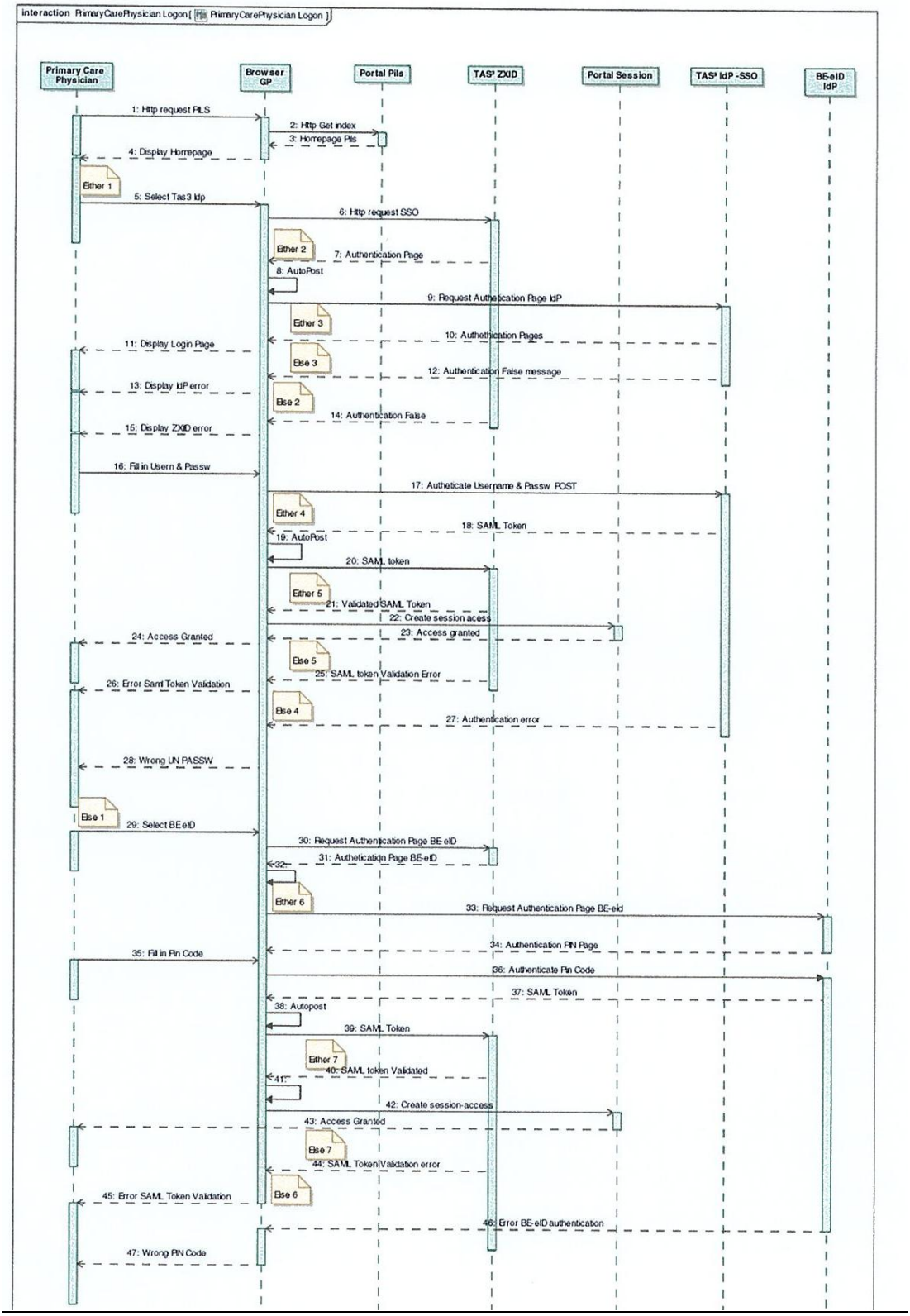


**Figure 12 Primary Care Physician Logon**

List of test procedures:

## Physician role

Test Procedure 1
*Precondition: logon as physician*
*Http request PILSt(PIL)*
*SettingCategoryValue*
*Http get indext()*
*SettingCategoryValue*
*HomepagePilst()*
*SettingCategoryValue*
*Display Homepaget()*
*SettingCategoryValue*
PIL
*SettingCategoryValue*
*Post Condition:*
*Comment and results:*

Test Procedure 2.1
*Precondition: logon as physician*
*Select Tas3 idp (idp)*
*SettingCategoryValue*
*Http request SSO (SSO)*
*SettingCategoryValue*
*Autentication Page*
*true*
*Autopost*
*SettingCategoryValue*
*Request autenticationPage*
*SettingCategoryValue*
*Autentication Page*
*true*
*Display Login Page*
*SettingCategoryValue*
idp
*SettingCategoryValue*

SSO
*SettingCategoryValue*

*Post Condition:*
*Comment and results:*

Test Procedure 2.2
*Precondition: logon as physician*
*Select Tas3 idp*
*SettingCategoryValue*
*Http request SSO*
*SettingCategoryValue*
*Autentication False*
*false*
*DisplayZXID Errore*
*SettingCategoryValue*
idp
*SettingCategoryValue*
SSO
error

*Post Condition:*
*Comment and results:*

Test Procedure 2.3
*Precondition: logon as physician*
*Select Tas3 idp*
*SettingCategoryValue*
*Http request SSO*
*SettingCategoryValue*
*Autentication Page*
*true*
*Autopost*
*SettingCategoryValue*
*Request Autentication Page idP*
*SettingCategoryValue*
*Autentication PagesFalse message*
*false*
*Display IdP error*
*SettingCategoryValue*
idp
*false*
SSO
*SettingCategoryValue*

*Post Condition:*
*Comment and results:*

Test Procedure 3.1
*Precondition: logon as physician*
*Fit in Usern&Passw (UN,PW)*
*SettingCategoryValue*
*Autenticate Username &PasswPOST*
*SettingCategoryValue*
*SAML Token (SAML)*
*SettingCategoryValue*
*AutoPost*
*SettingCategoryValue*
*SAML Token*
*SettingCategoryValue*
*Validated SAML Token*
*SettingCategoryValue*
*Create session access*
*SettingCategoryValue*
*Access granted*
*SettingCategoryValue*
*Access granted*
*SettingCategoryValue*
UN
*SettingCategoryValue*

PW
*SettingCategoryValue*
SAML
*SettingCategoryValue*

*Post Condition:*
*Comment and results:*

Test Procedure 3.2
*Precondition: logon as physician*
*Fit in Usern&Passw (UN,PW)*
*SettingCategoryValue*
*Autenticate Username &PasswPOST*
*SettingCategoryValue*
*SAML Token*
*SettingCategoryValue*
*AutoPost*
*SettingCategoryValue*
*SAML Token*
*SettingCategoryValue*
*Validated SAML Token*
*SettingCategoryValue*
*SAML Token validation error*
*SettingCategoryValue*
*Error SAML Token Validation*
*SettingCategoryValue*
UN
*SettingCategoryValue*

PW
*SettingCategoryValue*
SAML
*error*

*Post Condition:*
*Comment and results:*

Test Procedure 3.3
*Precondition: logon as physician*
*Fit in Usern&Passw (UN,PW)*
*SettingCategoryValue*
*Autenticate Username &PasswPOST*
*SettingCategoryValue*
*Autenticaton error*
*SettingCategoryValue*
*Wrong UN PASSWORD*
*SettingCategoryValue*

*Post Condition:*
*Comment and results:*
*Post Condition:*
*Comment and results:*
UN
*Random string*

PW
*Random string*

Test Procedure 4.1
*Precondition: logon as physician*
*Select BE eID*
*SettingCategoryValue*
*Request Autentication Page BE eID*
*SettingCategoryValue*
*Autentication Page BE eID*
*SettingCategoryValue*
*Request Autentication Page BE eID*
*SettingCategoryValue*
*Autentication PIN Page*
*SettingCategoryValue*
*Post Condition:*
*Comment and results:*

Test Procedure 4.2
*Precondition: logon as physician*
*Select BE eID*
*SettingCategoryValue*
*Request Autentication Page BE eID*
*SettingCategoryValue*
*Autentication Page BE eID*
*SettingCategoryValue*
*Request Autentication Page BE eID*
*error*
*Error BE eID Autentication*
*SettingCategoryValue*
*Wrong code??*
*SettingCategoryValue*

*Post Condition:*
*Comment and results:*
*:*

Test Procedure 5.1
*Precondition: logon as physician*
*Fill in Pin Code (PIN)*
*SettingCategoryValue*
*Autentication Pin code*
*SettingCategoryValue*
*SAML Token (SAML)*
*SettingCategoryValue*
*Auto post*
*SettingCategoryValue*
*SAML Token*
*SettingCategoryValue*
*SAML Token Validated*
*SettingCategoryValue*
*Create session access*
*SettingCategoryValue*
*Access Granted*
*SettingCategoryValue*

PIN
*SettingCategoryValue*
SAML
*SettingCategoryValue*

*Post Condition:*
*Comment and results:*
*:*

Test Procedure 5.2
*Precondition: : logon as physician*
*Fill in Pin Code (PIN)*
*SettingCategoryValue*
*Autentication Pin code*
*SettingCategoryValue*
*SAML Token (SAML)*
*SettingCategoryValue*
*Auto post*
*SettingCategoryValue*
*SAML token validation error*
*SettingCategoryValue*
*Error SAML token validation*
*SettingCategoryValue*
PIN
*SettingCategoryValue*
SAML
*error*

*Post Condition:*
*Comment and results:*

# Patient role

Test Procedure 1 bis
*Precondition: logon as patient*
*Http request PILSt(PIL)*
*SettingCategoryValue*
*Http get indext()*
*SettingCategoryValue*
*HomepagePiIst()*
*SettingCategoryValue*
*Display Homepaget()*
*SettingCategoryValue*
PIL
*SettingCategoryValue*
*Post Condition:*
*Comment and results:*

Test Procedure 2.1bis
*Precondition logon as patient*
*Select Tas3 idp (idp)*
*SettingCategoryValue*
*Http request SSO (SSO)*
*SettingCategoryValue*
*Autentication Page*
*true*
*Autopost*
*SettingCategoryValue*
*Request autenticationPage*
*SettingCategoryValue*
*Autentication Page*
*true*
*Display Login Page*
*SettingCategoryValue*
idp
*SettingCategoryValue*

SSO
*SettingCategoryValue*

*Post Condition:*
*Comment and results:*

Test Procedure 2.2 bis
*Precondition logon as patient*
*Select Tas3 idp*
*SettingCategoryValue*
*Http request SSO*
*SettingCategoryValue*
*Autentication False*
*false*
*DisplayZXID Errore*
*SettingCategoryValue*
idp
*SettingCategoryValue*
SSO
*error*

*Post Condition:*
*Comment and results:*

Test Procedure 2.3 bis
*Precondition: logon as patient*
*Select Tas3 idp*
*SettingCategoryValue*
*Http request SSO*
*SettingCategoryValue*
*Autentication Page*
*true*
*Autopost*
*SettingCategoryValue*
*Request Autentication Page idP*
*SettingCategoryValue*
*Autentication PagesFalse message*
*false*
*Display IdP error*
*SettingCategoryValue*
idp
*false*
SSO
*SettingCategoryValue*

*Post Condition:*
*Comment and results:*
*:*

Test Procedure 3.1bis
*Precondition: logon as patient*
*Fit in Usern&Passw (UN,PW)*
*SettingCategoryValue*
*Autenticate Username &PasswPOST*
*SettingCategoryValue*
*SAML Token (SAML)*
*SettingCategoryValue*
*AutoPost*
*SettingCategoryValue*
*SAML Token*
*SettingCategoryValue*
*Validated SAML Token*
*SettingCategoryValue*
*Create session access*
*SettingCategoryValue*
*Access granted*
*SettingCategoryValue*
*Access granted*
*SettingCategoryValue*
UN
*SettingCategoryValue*

PW
*SettingCategoryValue*
SAML
*SettingCategoryValue*

*Post Condition:*
*Comment and results:*

Test Procedure 3.2 bis
*Precondition: logon as patient*
*Fit in Usern&Passw (UN,PW)*
*SettingCategoryValue*
*Autenticate Username &PasswPOST*
*SettingCategoryValue*
*SAML Token*
*SettingCategoryValue*
*AutoPost*
*SettingCategoryValue*
*SAML Token*
*SettingCategoryValue*
*Validated SAML Token*
*SettingCategoryValue*
*SAML Token validation error*
*SettingCategoryValue*
*Error SAML Token Validation*
*SettingCategoryValue*
UN
*SettingCategoryValue*

PW
*SettingCategoryValue*
SAML
*error*

*Post Condition:*
*Comment and results:*
*:*

Test Procedure 3.3 bis
*Precondition: logon as patient*
*Fit in Usern&Passw (UN,PW)*
*SettingCategoryValue*
*Autenticate Username &PasswPOST*
*SettingCategoryValue*
*Autenticaton error*
*SettingCategoryValue*
*Wrong UN PASSWORD*
*SettingCategoryValue*
*Post Condition:*
*Comment and results:*
*Post Condition:*
*Comment and results:*
UN
*Random string*

PW
*Random string*

Test Procedure 4.1 bis
*Precondition: logon as patient*
*Select BE eID*
*SettingCategoryValue*
*Request Autentication Page BE eID*
*SettingCategoryValue*
*Autentication Page BE eID*
*SettingCategoryValue*
*Request Autentication Page BE eID*
*SettingCategoryValue*
*Autentication PIN Page*
*SettingCategoryValue*
*Post Condition:*
*Comment and results:*

Test Procedure 4.2 bis
*Precondition: logon as patient*
*Select BE eID*
*SettingCategoryValue*
*Request Autentication Page BE eID*
*SettingCategoryValue*
*Autentication Page BE eID*
*SettingCategoryValue*
*Request Autentication Page BE eID*
*SettingCategoryValue*
*Error BE eID Autentication*
*error*
*Wrong code??*
*SettingCategoryValue*

*Post Condition:*
*Comment and results:*

Test Procedure 5.1 bis
*Precondition: logon as patient*
*Fill in Pin Code (PIN)*
*SettingCategoryValue*
*Autentication Pin code*
*SettingCategoryValue*
*SAML Token (SAML)*
*SettingCategoryValue*
*Auto post*
*SettingCategoryValue*
*SAML Token*
*SettingCategoryValue*
*SAML Token Validated*
*SettingCategoryValue*
*Create session access*
*SettingCategoryValue*
*Access Granted*
*SettingCategoryValue*

PIN
*SettingCategoryValue*
SAML
*SettingCategoryValue*

*Post Condition:*
*Comment and results:*

Test Procedure 5.2 bis
*Precondition: : logon as patient*
*Fill in Pin Code (PIN)*
*SettingCategoryValue*
*Autentication Pin code*
*SettingCategoryValue*
*SAML Token (SAML)*
*SettingCategoryValue*
*Auto post*
*SettingCategoryValue*
*SAML token validation error*
*SettingCategoryValue*
*Error SAML token validation*
*SettingCategoryValue*
PIN
*SettingCategoryValue*
SAML
*error*

*Post Condition:*
*Comment and results:*
:

## Setting Values for the physician

### Messages:

Select Tas3 idp():
- User has one possible IDP to choose from
- User has more than one possible IDP to choose from
- User has no possible IDP to choose from [Property error]

Fit in Usern&Passw ():
- User is not already logged in [Property error]
- User is already logged in

Select BE eID
- User has one possible BE eID to choose from
- User has more than one possible BE eID to choose from
- User has no possible BE eID to choose from [Property error]

Fill in Pin Code
- User has one possible Pin to choose from
- User has more than one possible Pin to choose from
- User has no possible Pin to choose from [Property error]

### Parameters:

UN:
- Phisician 1
- yubikey
- yubikey expired [Property error]
- Random string [Property error]
- Faked yubikey token[Property error]

PW:
- phisitian1
- random string [Property error]

SAML token
- empty [Property error],
- Person-id: empty
  Hcp-id: 11019297122
  Common Name: Elias

idp
- Correct ipd
- Wrong idp [Property error],
-

SSO
- https://idp.tas3.eu/zxididp?o=R
- Random value [Property error],

PIL
- https://tas3.portal.custodix.com
- Random value [Property error],

Pin
- correct pin code
- Random value [Property error]

## Setting Values for the patient

### Messages:

Select Tas3 idp():
- User has one possible IDP to choose from

- User has more than one possible IDP to choose from
- User has no possible IDP to choose from [Property error]

Fit in Usern&Passw ():

- User is not already logged in [Property error]
- User is already logged in

Select BE eID

- User has one possible BE eID to choose from
- User has more than one possible BE eID to choose from
- User has no possible BE eID to choose from [Property error]

Fill in Pin Code

- User has one possible Pin to choose from
- User has more than one possible Pin to choose from
- User has no possible Pin to choose from [Property error]

## Parameters:

UN:

- Homer
- yubikey
- yubikey expired [Property error]
- Random string [Property error]
- Faked yubikey token[Property error]

PW:

- homer
- random string [Property error]

SAML token

- empty [Property error],
- Person-id: 73020419964
  Hcp-id: empty
  Common Name: Elias

idp

- Correct ipd
- Wrong idp [Property error],
-

SSO

- https://idp.tas3.eu/zxididp?o=R
- Random value [Property error],

Pil

- Https://tas3-myhealthspace.custodix.com/zxididp?o=B
- Https://tas3-myhealthspace.custodix.com/zxididp?o=E
- Https://tas3-myhealthspace.custodix.com/zxididp?o=S
- Https://tas3-myhealthspace.custodix.com/zxididp?o=Q
- Https://tas3-myhealthspace.custodix.com/zxididp?o=P
- Random value [Property error],

Pin

- correct pin code
- Random value [Property error]

*Remark: The IDP only returns identification numbers within the SAML token both for the patient access and the physician access. Further attributes like name, surname, date of birth... are added afterwards within the PILS Portal application. It is the objective to extend the IDP functionality with the required attributes for the PILS Portal application to obtain further integration of the TAS³ functions. The application provides two possible IDP providers ie TAS³ ZXID Idp and the Belgian BE-ID.*

## 5.1.2 Service discovery

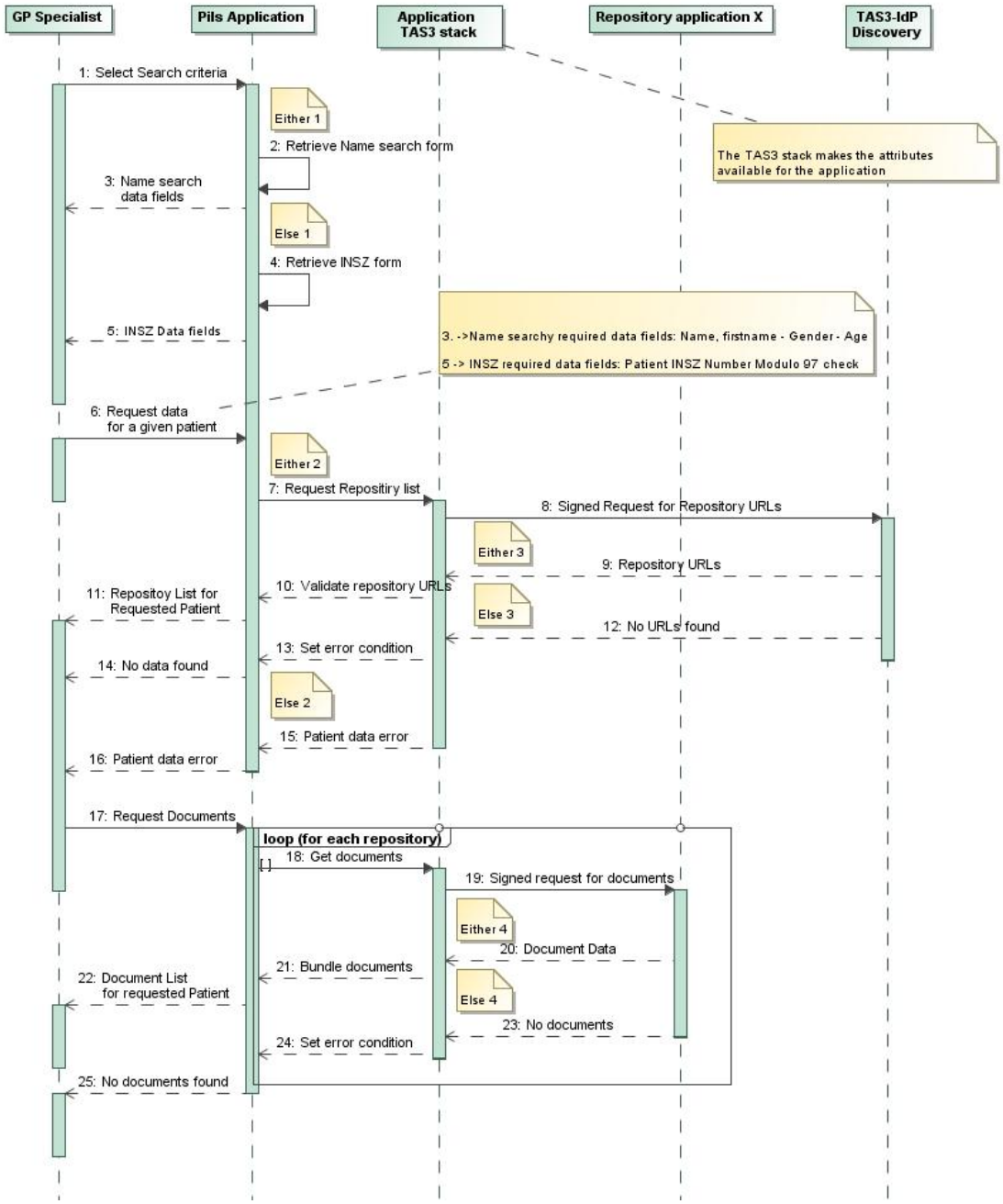


**Figure 13 Service discovery SD**

List of test procedures:

Test Procedure 1.1

*Precondition: General Practitionar or Specialist logged in through SSO and obtained the SAML attributes*
*User has two options to chose from*
Select Search criteria
*SettingCategoryValue*
Retrieve Name search form
*SettingCategoryValue*
Name Search Data fields
*SettingCategoryValue*

*Post Condition:Patient ID or physician ID has been entered and the data has been submitted to the Pils Application*
*Comment and results*:

Test Procedure 1.2

*Precondition: General Practitionar or Specialist logged in through SSO and obtained the SAML attributes*
*User has two options to chose from*
Select Search criteria
*SettingCategoryValue*
Retrieve INSZ form
*SettingCategoryValue*
INSZ Data fields
*SettingCategoryValue*
*Post Condition:Patient ID or physician ID has been entered and the data has been submitted to the Pils Application*
*Comment and results*:

Test Procedure 2.1

*Precondition:*
Request Data for a given patient (First Name, Last Name, Age, Sex)
*SettingCategoryValue*
Request Repository list
*SettingCategoryValue*
Signed Request for Repository list
*SettingCategoryValue*
Respository URLs
*SettingCategoryValue*
Validate Repository URLs
*SettingCategoryValue*
Repository List for requested patient
*SettingCategoryValue*
First Name
*SettingCategoryValue*
Last Name
*SettingCategoryValue*
Sex
*SettingCategoryValue*
Age:
*SettingCategoryValue*

*Post Condition List of hospital repositories containing data for the patient is shown.*
*Comment and results*:

Test Procedure 2.1 bis

*Precondition:*
Request Data for a given patient (INSZ)
*SettingCategoryValue*
Request Repository list
*SettingCategoryValue*
Signed Request for Repository list
*SettingCategoryValue*
Respository URLs
*SettingCategoryValue*
Validate Repository URLs
*SettingCategoryValue*
Repository List for requested patient
*SettingCategoryValue*
INSZ
*SettingCategoryValue*
*Post Condition List of hospital repositories containing data for the patient is shown.*
*Comment and results*:

Test Procedure 2.2

*Precondition:*
Request Data for a given patient (First Name, Last Name, Age, Sex)
*SettingCategoryValue*
Request Repository list
*SettingCategoryValue*
Patient data error
*SettingCategoryValue*
Patient data error
*SettingCategoryValue*
First Name
*SettingCategoryValue*
Last Name
*SettingCategoryValue*
Sex
*SettingCategoryValue*
Age:
*SettingCategoryValue*

*Post Condition The patient data is erroneous*
*Comment and result: The Idp could not find the patient based on the data submitted*

Test Procedure 2.2 bis

*Precondition:*
Request Data for a given patient (INSZ)
*SettingCategoryValue*
Request Repository list
*SettingCategoryValue*
Patient data error
*SettingCategoryValue*
Patient data error
*SettingCategoryValue*
INSZ
*SettingCategoryValue*
*Post Condition The patient data is erroneous*
*Comment and results The Idp could not find the patient INSZ number based on the data submitted or the modulo 97 check returned an error*

Test Procedure 2.3

*Precondition: Repository contains no data*
Request Data for a given patient (First Name, Last Name, Age, Sex)
*SettingCategoryValue*
Request Repository list
*SettingCategoryValue*
Signed Request for Repository list
*SettingCategoryValue*
Set error condition
*SettingCategoryValue*
No data found
*SettingCategoryValue*

First Name
*SettingCategoryValue*
Last Name
*SettingCategoryValue*
Sex
*SettingCategoryValue*
Age:
*SettingCategoryValue*

*Post Condition The Application ends with no data found message*
*Comment and results: It is not necessarily an error condition, because it is possible that no data is available about this patient in the queried repositories*

Test Procedure 2.3 bis

*Precondition: Repository contains no data*
Request Data for a given patient (INSZ)
*SettingCategoryValue*
Request Repository list
*SettingCategoryValue*
Signed Request for Repository list
*SettingCategoryValue*
*SettingCategoryValue*
Set error condition
*SettingCategoryValue*
No data found
*SettingCategoryValue*
INSZ
*SettingCategoryValue*
*Post Condition The Application ends with no data found message*
*Comment and results: It is not necessarily an error condition, because it is possible that no data is available about this patient in the queried repositories*

Test Procedure 3.1

*Precondition:The Physician wants to query for medical data of the patient and has selected the repositories from the list*

Requests documents
*SettingCategoryValue*
Get documents ()
*SettingCategoryValue*
Signed request for documents
*SettingCategoryValue*
Document Data
*SettingCategoryValue*
Bundle documents
*SettingCategoryValue*
Document List for requested Patient
*SettingCategoryValue*

*Post Condition: The Physician can now select a document and view the content*

*Comment and results*: *The Document request is executed for each repository of the repository list*

Test Procedure 3.2

*Precondition:The Physician wants to query for medical data of the patient and has selected the repositories from the list*

Requests documents
*SettingCategoryValue*
Get documents ()
*SettingCategoryValue*
Signed request for documents
*SettingCategoryValue*
Document Data
*SettingCategoryValue*
Bundle documents
*SettingCategoryValue*
Get documents ()
*SettingCategoryValue*
Signed request for documents
*SettingCategoryValue*
Document Data
*SettingCategoryValue*
Bundle documents
*SettingCategoryValue*

Document List for requested Patient
*SettingCategoryValue*

*Post Condition: The Physician can now select a document and view the content*

*Comment and results*: *The Document request is executed for each repository of the repository list*

Test Procedure 3.3

*Precondition:*

Requests documents
*SettingCategoryValue*
No documents found
*SettingCategoryValue*
Set error condition
*SettingCategoryValue*
No documents found
*SettingCategoryValue*

*Post Condition:*

*Comment and results*: *The Document request is executed for each repository of the repository list.*

## Setting Values

### Messages:

Select Search Criteria ()

- User has one possible Search Criterium to choose from
- User has more than one possible Search Criterium to choose from
- User has no possible Search Criterium to choose from [Property error]

Request data for a given patient ():

- User enters data for name search
- User enters data for INSZ number
- User does not enter data at all (property error)

Request Documents

- User has one possible repository to choose from
- User has more than one possible repository to choose from
- User has no possible repository to choose from [Property error]

### Parameters:

Patient First Name

- Homer
- Marge
- Bart
- Lisa
- Maggie
- GP1
- GP2
- yubikey
- yubikey expired [Property error]
- Random string [Property error]
- Faked yubikey token[Property error]

Patient Last Name

- Simpson
- Bouvier
- random string [Property error]

Sex

- male(default)
- female

Age:

- 35
- 36

- 10
- 8
- 1
- Random alpha characters [Property error]

INSZ Number

- 73020419964
- 73070728421
- 99040117163
- 01051022218
- 09081418824
- 76020800111
- 52081266625
- Random characters string (Property error)

## 5.2 Test case derived

We report in the table below the number of test cases derived by each test procedure combining the setting values provided. Moreover we provide some examples of the test cases that have been executed. In particular we report some examples of successful test execution and the most important failures detected by a test case run.

Note that for reporting the test result we exploited the structure of the test case itself. Thus once executed in the field "comment and results" we report the pass or fail verdict and in this last case the corrective action executed so to fix the observed failure.

| Name of the SD | Test procedure | Valid test cases | Invalid test cases |
|---|---|---|---|
| Primary Care Physician Logon | 1 | 1 | 1 |
| | 2.1 | 2 | 3 |
| | 2.2 | 2 | 2 |
| | 2.3 | 2 | 2 |
| | 3.1 | 2 | 6 |
| | 3.2 | 2 | 5 |
| | 3.3 | 1 | 1 |
| | 4.1 | 2 | 1 |
| | 4.2 | 2 | 1 |
| | 5.1 | 2 | 3 |
| | 5.2 | 2 | 2 |
| Primary Care Patient Logon | 1 bis | 5 | 1 |
| | 2.1 bis | 2 | 3 |
| | 2.2 bis | 2 | 2 |
| | 2.3 bis | 2 | 2 |
| | 3.1 bis | 2 | 6 |
| | 3.2 bis | 2 | 5 |

| | | | |
|---|---|---|---|
| | 3.3 bis | 1 | 1 |
| | 4.1 bis | 2 | 1 |
| | 4.2 bis | 2 | 1 |
| | 5.1 bis | 2 | 3 |
| | 5.2 bis | 2 | 2 |
| Service discovery | 1.1 | 2 | 1 |
| | 1.2 | 2 | 1 |
| | 2.1 | 160 | 5 |
| | 2.1 bis | 7 | 2 |
| | 2.2 | 160 | 5 |
| | 2.2 bis | 7 | 2 |
| | 2.3 | 160 | 5 |
| | 2. 3 bis | 7 | 2 |
| | 3.1 | 2 | 1 |
| | 3.2 | 2 | 1 |
| | 3.3 | 2 | 1 |
| Total | | 555 | 80 |

## 5.2.1 Some of the most meaningful test cases

### 5.2.1.1 Successful retrieval of medical documents for a given patient from different hospital repositories

SD Primary Care Physician Logon Test Procedure 1
*Precondition: logon as physician*
*Http request PILSt(PIL)*
*https://tas3-portal.custodix.com*
*Http get indext()*
*https://tas3-portal.custodix.com/Welcome.xhtml*
*Display Homepaget()*
*Welcome Page*
*Post Condition:*
*Comment and results: Pils Portal Welcome Page is displayed.*

SD Primary Care Physician Logon Test Procedure 2.1
*Precondition: logon as physician*
*Select Tas3 idp (idp)*
*ZXID*
*Http request SSO (SSO)*
*https://idp.tas3.eu/zxididp?o=R*
*Autentication Page*
*true*
idp
*Correct IdP*
SSO
https://idp.tas3.eu/zxididp?o=R

*Post Condition:*
*Comment and results: User successfully selected ZXID-SSO Authentication Page*

SD Primary Care Physician Logon Test Procedure 3.1

*Precondition: logon as physician*
*Fit in Usern&Passw (UN,PW)*
*Yubikey*
*Autenticate Username &PasswPOST*
*true*
*Validated SAML Token*
Hcp-id: 11019297122
Common Name: Elias
*Create session access*
SSO-session-ID= MSESQD9Sr6fJmOeK3PZH1vZ3rBAA
*Access granted*
Pils-Session-ID= FBB39886534E779DC027C4E79CE14E58
*Access granted*
*true*
UN
*yubikey*

PW
*yubikey*

*Post Condition:*
*Comment and results: The physician successfully signed on using TAS³ ZXID and obtained access to the Pils portal application*

SD Service discovery Test Procedure 1.1

*Precondition: Physician logged in through SSO and obtained the SAML attributes*
Select Search criteria
Search by Name selected
Retrieve Name search form
Name search form displayed

*Post Condition: The Physician is requested to fill in the required fields in the Name search form*
*Comment and results: the display of the Name search form indicates a successful access to the PILS Portal application.*

SD Service discovery Test Procedure 2.1.

*Precondition: Physician is granted access to the PILS application*
*Physician fills in First Name :Homer*
*Physician fills in Last Name: Simpson*
*Physician selects Sex: Male*
*Physician fills in Age: 40 years*
*User requests Repository list for patient Homer Simpson by hitting the "Search" button*
PILS
*Patient INSZ number is included in the repository request*

TAS3 Stack
*The request is completed with the SAML attributes*
*Signature of the requester is added*
*Signed request is submitted to the TAS3 –Discovery service*

TAS3-Idp Discovery
*Known URLs of Hospital repositories are challenged with the request*

TAS3 Stack
Validated Repository URLs

PILS
*Hospital names are retrieved and included in the form*
*Marvin Monroe Memorial Hospital*
*Springfield General Hospital*

*Post Condition:* List of Repositories comprising documents for Homer Simpson with tick boxes
*Comment and results:* Documents for Homer Simpson have been found at the repositories within these hospitals. The user is invited to select the hospitals to retrieve the documents via the tick boxes.In the background the full list of repositories queried is also available.

SD Physician Test Procedure 3.1.

*Precondition: The Physician obtained a list of repositories where documents are found for Homer Simpson*

*Physician selects all repositories*
*Physician submits the request for documents by hitting the "View Documents" button.*

PILS
*Generate request for documents by repository (SOAP)*
TAS³ Stack
*Generate Signed request incl. SAML attributes*
Repository
*Retrieve documents for Homer Simpson from Marvin Monroe Memorial Hospital and Springfield General Hospital*

PILS
*Bundle documents by Hospital i!n the document browser form*

*Post Condition: The documents are listed by header and by hospital*

*Comment and results:*
*The physician can view the content, download the document, delete de document from the list, or import the document in his Patient Record software.*

### 5.2.1.2 No data for a given patient found in the hospital repositories

SD Primary Care Physician Logon Test Procedure 1
*Precondition: logon as physician*
*Http request PILSt(PIL)*
*https://tas3-portal.custodix.com*
*Http get indext()*
*https://tas3-portal.custodix.com/Welcome.xhtml*
*Display Homepaget()*
*Welcome Page*
*Post Condition:*
*Comment and results: Pils Portal Welcome Page is displayed.*

SD Primary Care Physician Logon Test Procedure 2.1
*Precondition: logon as physician*
*Select Tas3 idp (idp)*
*ZXID*
*Http request SSO (SSO)*
*https://idp.tas3.eu/zxididp?o=R*
*Autentication Page*
*true*
idp
*Correct IdP*
SSO
https://idp.tas3.eu/zxididp?o=R

*Post Condition:*
*Comment and results: User successfully selected ZXID-SSO Authentication Page*

SD Primary Care Physician Logon Test Procedure 3.1
*Precondition: logon as physician*
*Fit in Usern&Passw (UN,PW)*
*Yubikey*
*Autenticate Username &PasswPOST*
*true*
*Validated SAML Token*
Hcp-id: 11019297122
Common Name: Elias
*Create session access*
SSO-session-ID= MSESQD9Sr6fJmOeK3PZH1vZ3rBAA
*Access granted*
Pils-Session-ID= FBB39886534E779DC027C4E79CE14E58
*Access granted*
*true*
UN
*yubikey*

PW
*yubikey*

*Post Condition:*
*Comment and results: The physician successfully signed on using TAS³ ZXID and obtained access to the Pils portal application*

SD Service discovery Test Procedure 1.2
*Precondition: Physician logged in through SSO and obtained the SAML attributes*
Select Search criteria
Search by INSZ selected
Retrieve INSZ form
INSZ search form displayed

*Post Condition: The Physician is requested to fill in the required fields in the INSZ search form*
*Comment and results: the display of the INSZ search form indicates a successful access to the PILS Portal application.*

SD Service discovery Test Procedure 2.3.
*Precondition: Physician is granted access to the PILS application*
*Physician fills in INSZ Number / 99040117163*

*User requests Repository list for patient 99040117163by hitting the "Search" button*

PILS
*Patient INSZ number is OK*

TAS3 Stack
*The request is completed with the SAML attributes*
*Signature of the requester is added*
*Signed request is submitted to the TAS3 –Discovery service*

TAS3-Idp Discovery
*Known URLs of Hospital repositories are challenged with the request*

TAS3 Stack
Validated Repository URLs

PILS
*"Your query returned no results"*

*Post Condition:* No repositories containing data for the given patient were found.
*Comment and results:*.In the background the full list of repositories queried is available.
This is not a program error but an indication that no data was found for the patient. The patient is known by the system.

## 5.3 Discussion and lessons learned

The Healthcare integration trial was build upon an existing application PILS (Patient Information Locator Services). This application has been tested and is operational since 2 years. The existing authentication, authorisation and repository discovery services were replaced by the appropriate TAS3 components. Much less focus was put on the features and functions of the existing PILS application. The integration (interfaces & information exchanges) between TAS3 components and the existing application were challenged with the test plan execution.

A dummy test set (the Simpson Family) was created to avoid any use of real patient data during the test execution. The test strategy resulted to be very helpful to solve bugs in the error reporting and condition setting between the TAS3 components and the PILS application modules.

It learned developers that more important modifications were required within the application than originally was estimated. As it was a first iteration in using TAS3 modules in an existing application room for improvement was recognised.

# 6 Conclusion and Future work

This document constitutes the first iteration of the pilot evaluation process.
Firstly we focused on the employability scenario because it was the most comprehensive from a testing point of view. For this scenario we proposed two different test strategies specific for integration testing one based on the sequence diagram specification and another based on the XACML policy even if only the former has been used for test plan definition.
The Healthcare scenario test execution focused on the integration and interfaces with the TAS3 components and much less on the application functionality.
The test plan execution lead to the discovery of some critical bugs in both integration trials and to the possibility of discussing and applying immediate corrective actions were possible.
A posterior analysis of test results evidences the necessity of a an accurate step of components verification and evaluation avoiding to postpone some type of bugs at integration testing level.
Time constraints and the initial use of different diagramming techniques forced the testers to redo some work, which could be avoided. The interpretation of the diagrams by the test execution suite required some manual intervention to correctly present the data as input to the test execution.
As a future work we plan to continue the application of UIT_SD to the other pilots scenario. To that end we find it necessary to organize training sessions for the testers to correctly prepare the scenarios to avoid unnecessary manual intervention at testing time. Moreover as PDP modules and PEP modules become more mature and populated with appropriate policies we will start to use the test strategy based on XACML policy to better evaluate the performance of the next iteration of pilots applications.

# 7 REFERENCES

[1] M. Balcer, W. Hasling, and T. Ostrand, "Automatic Generation of Test Scripts from FormalTest Specifications", Proceedings of ACM SIGSOFT'89 - Third Symposium on Software Testing, Verification, and Analysis (TAVS-3), ACM Press, pp. 257-71, June 1990.

[2] F. Basanieri, A.. Bertolino, and E.Marchetti, "The Cow_Suite Ap-proach to Planning and Deriving Test Suites in UML Projects", Proc. 5th Int. Conf. UML 2002, Dresden, Germany, LNCS 2460, pp. 383--397, 2002.

[3] B. Beizer, Software Testing Techniques 2nd Edition, International Thomson Computer Press, 1990.

[4] A. Bertolino, "Knowledge Area Description of Software Testing", Chapter 5 of SWEBOK: The Guide to the Software Engineering Body of Knowledge. Joint IEEE-ACM Software Engineering Coordination Committee. 2001. http://www.swebok.org/.

[5] P. C Jorgensen, Software Testing a Craftsman's Approach. CRC Press, 1995.

[6] J.C. Laprie, "Dependability - Its Attributes, Impairments and-Means",Predictably Dependable Computing Systems, B. Randell, J.C. Laprie, H. Kopetz, B. Littlewood, eds.:, Springer , 1995

[7] T.J. Ostrand, and M.J Balcer, M.J, "The Category-Partition Method for Specifying and Generating Functional Tests", ACM Comm, vol. 31, no. 6, pp. 676—686, 1988.

[8] S.L. Pfleeger, Software Engineering Theory and Practice, Prentice Hall, 2001.

[9] Rational Unified Process version 2000.02.10. Rational Software Corporation. On-line at http://www.rational.com/products/rup