

PII
Deliverable D3.4

Testbed Service Description System Implementation

Editor:	Sebastian Wahle, Fraunhofer
Deliverable nature:	Prototype (P)
Dissemination level: (Confidentiality)	RE Restricted to a group specified by the consortium (including the Commission Services)
Contractual delivery date:	July 2009
Actual delivery date:	September 2009
Suggested readers:	PII consortium and community, EC, public
Version:	1.0
Total number of pages:	7
Keywords:	Teagle Implementation

Abstract

The project Pan European Laboratory Infrastructure Implementation (PII) addresses the need for large-scale testing facilities in the communications area by implementing an infrastructure for federating testbeds among innovations clusters. It builds upon the Panlab Specific Support Action which received funding by the European Commission's Sixth Framework Programme.

This document is part of the documentation on Teagle_v1.0. It describes the Teagle Portal and the Teagle Repository on a high level. Also, it provides the links to the online installations of the running prototypes as footnotes where appropriate.

Disclaimer

This document contains material, which is the copyright of certain PII consortium parties, and may not be reproduced or copied without permission.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the PII consortium as a whole, nor a certain party of the PII consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

Impressum

[Full project title] Pan-European Laboratory Infrastructure Implementation

[Short project title] PII

[Number and title of work-package] WP3, Service description, discovery and orchestration

[Document title] Testbed Service Description System Implementation

[Editor] Sebastian Wahle, Fraunhofer

[Work-package leader] Sebastian Wahle, Fraunhofer

[Estimation of PM spent on the Deliverable] 20

Copyright notice

© 2009 Participants in project PII

1 Teagle

This document summarizes the work done for the testbed service description system implementation which mostly relates to the development of the Teagle Repository and the Teagle Portal.

Teagle is the central search and composition engine of the Panlab testbed and experimental facility federation. It provides a web-based customer interface for browsing the federation’s offerings and the provisioning of Virtual Customer Testbeds (VCT).

A VCT is the sum of all resources and interconnections configured and rented by a specific customer. It is an isolated network where the customer has direct access to the resource and configurations assembled by using Teagle. Each customer operates inside its own VCT and has no access to other VCTs unless sharing is explicitly enabled. Sharing VCTs allows several users to configure and work with a VCT at the same time.

Teagle combines several functions (Figure 1) that are currently implemented in a centralized manner (although distribution is generally possible for components like the registries but this requires more complex trust hierarchy concepts):

- Registry & Repository (users, resources, configurations)
- Creation Environment (setup and configuration of VCTs, this is the VCT tool)
- Request Processor (for validating VCT configurations and trigger setup execution)
- Orchestration Engine (for generation of an executable workflow that orchestrates services form different domains to actually instantiate a VCT)
- Web Portal (for exposing search and configuration interfaces)

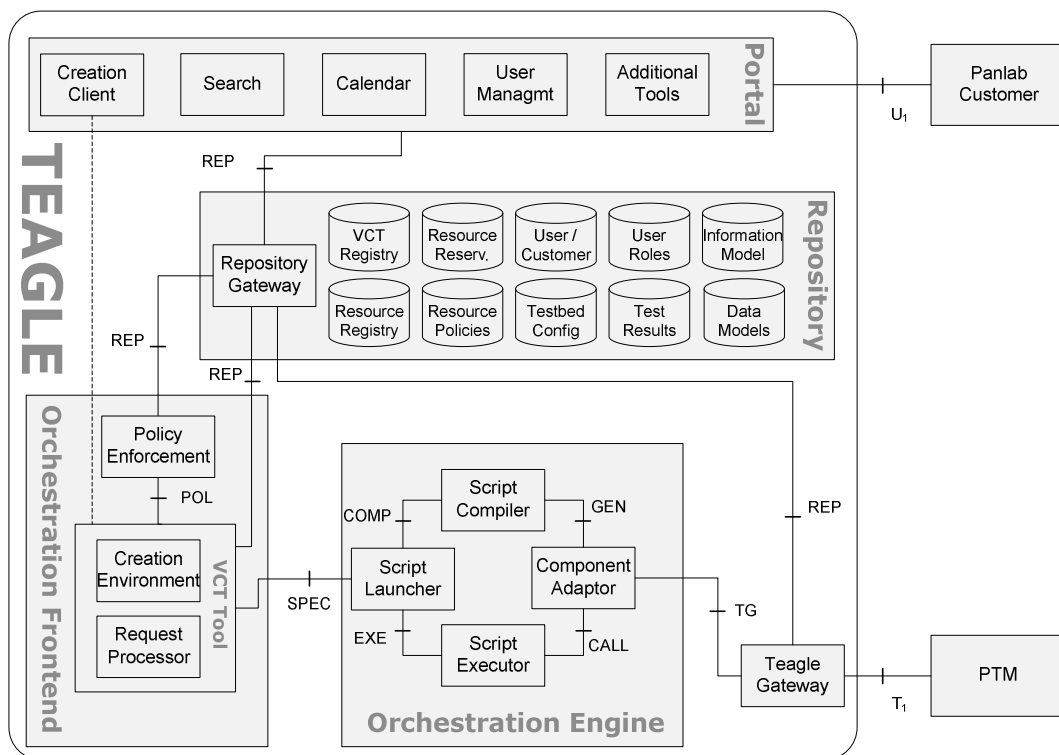


Figure 1: Teagle architecture overview

2 The Teagle Portal

The TEAGLE Portal¹ is the main customer interface and provides information on Teagle and the offered services. Furthermore, it is the entry point for testbed composition as the Java Web Start application that allows resource configuration and deployment (the VCT tool, see D3.5) is loaded from this page. Also, via the Teagle portal the customer may use the Teagle Repository, browse offered resources, manage his VCTs, etc. The Teagle Portal is also used by Panlab providers to manage and add its own resources, register PTMs (domains), etc.

teagle
A Pan-European Laboratory Project

Home
News
Testbed Repository
Results Repository
Search
Members Area
Links

Info
You are not logged in.
[Create account](#) or [login](#)

Welcome to the Teagle Portal. This is a project related to the [Panlab concept](#). The Teagle Portal provides information about our partner testbeds and allows you to manage your Private Virtual Test Lab.

What is a Private Virtual Test Lab?
A Private Virtual Test Lab is a testbed that is composed by a number of distributed software and hardware components situated in existing testbeds across Europe. It provides you with a testing and prototyping environment that supports your very specific requirements.
Example: You are looking for a NGN testbed that allows you to test an application with multiple users from different countries in Europe?
No Problem, we have that!

What is Teagle?
Teagle is the central coordination instance that holds together all our partner test labs that are needed to provide you with the maximum range of testing and prototyping possibilities. All our partner labs form a federation of testbeds.
Via Teagle you can:

- Search for specific functionality offered by any of our partner labs
- Browse partner labs
- Setup and configure your own Private Virtual Test Lab
- Browse and compare previous test results
- Publish your own test results
- Find help and contact a responsible person

hosted by

Fraunhofer Institute for Open Communication Systems

News
2009-06-16
EXFFI 2009
ICUMT Workshop on Experimental Facilities for the Future Internet (EXFFI 2009)
[more](#)
2009-05-18
Draft Tridentcom2010 CFP available
Draft Tridentcom2010 CFP available [more](#)

RSS Feed
 [Teagle RSS Feed](#)

[How to get started!](#)
Browse our [Testbed Repository](#) to search for what you need or [create an account](#) to participate and contribute.

[Linking Policy](#) | [Publishing Notes](#) | [Data Protection](#)

Figure 2: Teagle Portal start page

The Portal uses a role-based architecture and different roles (Admin, Panlab Customer, Panlab Provider, etc.) to display the appropriate content for the accessing user. For example, Panlab Customers can access their rented resources and VCTs but are not allowed to administer resource parameters. This is only allowed for resource providers. Resource providers are only allowed to edit the resources that they have added.

More information on the Portal is available online. You may need to create an account to access all available features of the Teagle Portal (e.g. using the VCT tool).

¹ <http://www.fire-teagle.org/>

3 The Teagle Repository (currently running prototype from Fraunhofer FOKUS)

The Fraunhofer repository prototype has basically been developed to speed up the Teagle Portal and VCT Tool development (both at Fraunhofer). With the very specific requirements coming from the VCT Tool and Teagle Portal developers, the prototyping of a lightweight XML database (and associated Web Services) serving as the Teagle Repository supporting specific use cases was possible within a short time frame. Having the Teagle Repository, Portal, and VCT Tool developers in one geographical location was speeding up the development as lengthy synchronization processes were reduced to a minimum.

However, the Teagle Repository solution currently developed at WIT as described in the following section, will offer a number of additional features compared to the currently running Fraunhofer prototype. Once the WIT development has entered a mature state, the depending Teagle components (such as Portal, VCTtool, etc.) will need to switch to the services offered by the WIT repository. In the following we describe the Fraunhofer solution.

The repository is one of the very central entities enabling the entire federation. Running experiments don't involve interaction with the repository, but setting up the experimentation environment does. The purpose of the central repository is allowing customers to find desired components and keep track of their past activity through the means of a search engine and log files. The search process is backed up by a database with resources (a resource registry), testbeds, and further identities (customers, partners, etc.). During operation, after the initial resource location stage has ended and the test phase is initiated, customer interaction with the testbeds is unhindered by the repository.

The main point of interest and the one driving our whole design for the repository is a Web Service component in its position of a central hub. We have grouped all the functions for remote export by means of a Web Service within a single entity, the repository manager. As its name implies, the public methods of the object interface repository operations. By design, it doesn't need to know details about the different types of repositories and it has no direct control on their workings.

Model

Each collection of information with use to the project is persisted inside the database. The software components responsible for storing data in an orderly fashion will be referred here as registries.

Obviously, different resources have different attributes, not only by value but here also by attribute name. When searching the repository, a user may be interested only in a resource type satisfying a specific condition for a property only shared by a subset of all the resources. This behaviour is tedious to implement by using a classic approach to data storage, such as a relational database management system (RDBMS) but comes naturally for a native XML database. In the context of XML storage, a change from the standard RDBMS scheme is in place. Instead of describing data stores with a static structure, the table column names from the RDBMS universe, here no predefined model is required. Any programmer-defined structured type can be used.

For implementing the whole solution we have chosen the Java programming language. By deriving from a set of common ancestors, the classes used for resource description implement specific behaviours and are correct at least from a structural perspective. Thus, the job of validating and auditing models has been automatically delegated upon the Java compiler and libraries. Conversion between the Java class model and the XML internal representation is handled conveniently by the XStream open source library.

Repository

Testbed resource types from the owners of the laboratories within the federation are stored in the most important XML database modelled by the resource registry. The operations provided allow adding and removing resource type descriptions and searching them by different criteria. The purpose of the resource repository is providing expressive search capabilities for the descriptions of the resources provided by the laboratory owners involved with the project. All the descriptions follow a predefined pattern – the part where they represent functionality by following a coherent model. Retrieval/search is the essential available operation. Considering the fact that different devices may function in

completely unrelated ways and the federation is as technology agnostic as possible, it is impossible to foresee while constructing the software component of the repository the precise usage scenarios it will be put through during exploitation. Hence, the resources model must be self-describing and a format that is easy to manipulate from the machine side, as XML is, IT presents itself as the best solution.

Of course, each instance of a resource type consumes memory, port ranges, maybe even security certificates and CPU cycles from the computer it is running on. Therefore it's foreseeable that the laboratory owners may only allow a finite quantity of resource instances being deployed at one time. This is managed with the help of a resource instance registry which keeps track of all the deployments of a resource.

Searches within the XML documents storing resource descriptions are realized with the help of XPath expressions. With XPath, nodes can be accessed or searched (read-only operations) in an expressive manner governed by the structure of the XML file. Fig. 3 exemplifies XPath application on an XML structure. Derived from XPath, for updates within the XML database we are using XUpdate, which is supported by the eXist database which is maintaining the repository.

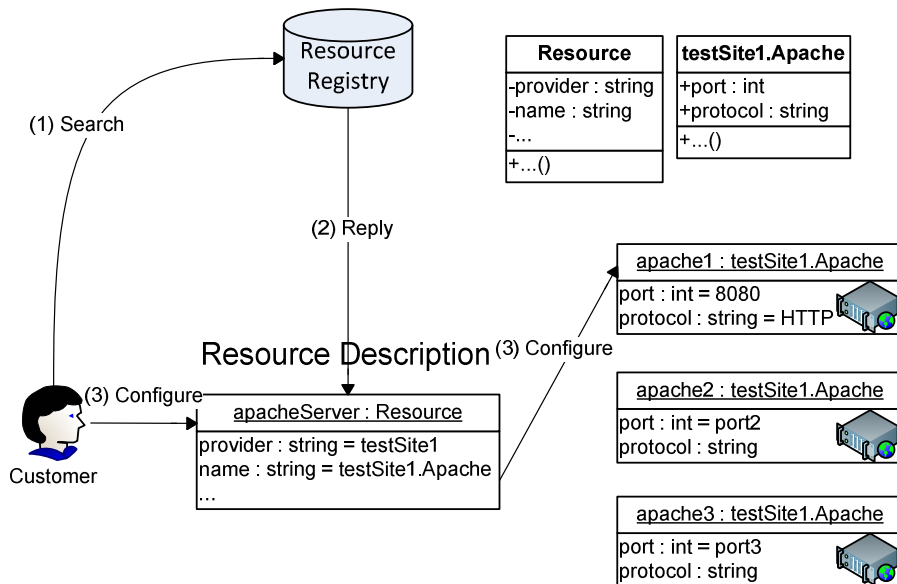


Figure 3: Resource configuration

4 The Teagle Repository (upcoming prototype from WIT)

The Teagle Repository is designed as an information persistence system for Teagle components. These include the Teagle Portal and the VCT tool. It is required to store XML documents to relevant data objects for future use by Teagle components. The structure of this information should follow a generally accepted information model to promote interoperability in Teagle components and other PII systems.

The Teagle Repository uses a RESTful interface to expose representations of underlying resources. These come in the form of XML documents which are used by the other Teagle components. As the VCT tool considers the repository the Model in an MVC pattern, the resources behave very much like models. They simply persist model data and associations.

The model resources defined in the repository are used to support the Teagle Portal and the VCT Tool. There is a VCT model for collected VCT information from the VCT Tool. Also present, PTM model for storing information on registered PTMs from the Portal. There is a Resource Model for identifying the testbed resources. The Configuration Resource provides basic support for defining the structure of each resource configuration. The Person and Organisation models for allow the definition of ownership of VCTs, PTMs and Resources. Finally, a Reservation model stores Resource Reservation data which can be used by the Request Processor and for asserting Resource availability.

An information model is necessary in order to manage the large number of heterogeneous testbed resources and services that PII will offer. The information model provides structure and a framework that will allow the Teagle to catalogue, search, reserve, connect and configure resources to function within a custom testbed setup. A major issue that a management system of heterogeneous resources needs to tackle is interoperability. The resources despite being of the same type may have different attributes, methods, configuration parameters, dependencies and functionalities thus making it difficult for Teagle to automatically orchestrate a testbed setup. Using an information model enables both the testbed resources and their relationships to be defined precisely in a common manner thus alleviating the interoperability issues of the heterogeneous resources. The Information model can also be employed to manage the legal and operational aspects of PII. These include aspects such as SLAs and the federation's management policies versus the local testbed provider's policies.

The DEN-ng information model was chosen as using this model meant that the work could build upon the general concepts within DEN-ng. Although DEN-ng does not model testbed resources explicitly, it could be extended and had many other useful concepts modelled such as configuration and policies that could easily be adapted to suit a federated testbed's requirements.

The three initial PII domains that were developed were a VCT model, a Configuration model and a ManagementApplication model. The VCT model defined the entities and relationships between the testbed resources and services required to form the VCT. The Configuration model defines how the resources within the VCT can be configured and the ManagementApplication model represents the PII concepts of a PTM (Panlab Testbed Manager) and RA (Resource Adaptor).

Once the initial PII information model was defined, a process to translate this information model to data models that can be stored in a repository was developed. This involved using a Model Driven Architecture approach provided by the Eclipse tool Acceleo. Acceleo allowed template generators to be written that outputted Domain Classes for the Repository from the PII information model.

As the interface provided to client components is RESTful, this decouples the architecture of the repository from the architecture of the clients involved. However, the architecture behind this is relatively flexible to account for evolving requirements as the project progresses. It uses the Glassfish Application Server as the basis for application deployment and management. The Repository itself is composed of a Grails Framework front-end (for marshalling the requests and responses from other Teagle components). The data objects are derived from the information model as described above. This uses Hibernate as an Object-to-Relational Mapper (ORM). In prototype development, the lightweight HSQL database is used. Due to the use of the ORM, this can be transparently replaced should the need arise.