Thessaloniki, 16/3/2012
Europass team

ANNEX X1
TECHNICAL DESCRIPTION OF THE EUROPASS WEB-BASED RESOURCES

1. THE EUROPASS PLATFORM

1.1. Introduction

The Europass multilingual website (http://europass.cedefop.europa.eu) was launched in February 2005; it is now available in 26 languages. It is the main resource for the implementation of the Europass initiative and gives citizens:

- direct access to two Europass instruments, the Europass CV and the Europass Language Passport which can be filled in by citizens, either offline or online using an user-friendly interface; completed CVs or ELP can be saved in various formats (Microsoft Word/OpenOffice/PDF, HTML and XML) and later on uploaded (in XML format) for update;

- indirect access to the other three Europass instruments: the Europass Mobility, (see also below "The Europass Mobility management tool"), the Certificate Supplement and the Diploma Supplement.

A full custom Content Management System (CMS) has also been developed by Cedefop, through which all content in the Europass platform and the Europass Mobility Management Tool is managed (accessed, edited, added, manipulated). The CMS also serves the purpose of the batch creation of a set of Excel files with all content, in a special format facilitating an easy translation, and also the extraction of any part of the content, in any language, in XML format.

The Europass platform is developed, maintained and hosted by Cedefop in Thessaloniki.

1.2. Technical architecture

From the user perspective (user experience) Europass is presented as a homogeneous Thin Client application, i.e. HTML & JavaScript, and it can be accessed by a common Internet browser. The language of presentation of the content has to be any of the official languages of EU, and the site gives the ability to a user (who browses through its contents) to change the presentation language at any point of the site. Thus, Europass is multilingual (UTF-8) in all interfaces presented to the user or administrator. Also its internal data transfer and storage mechanisms employ UTF-8.

From the functional perspective, Europass can be divided in the following three subsystems:

(1) **The Europass Site**, which contains all the static and informative content of the site. This content is managed by the Content Management System (CMS).

(2) **Instruments on-line applications**. These are viewed as separate applications with main purpose to help a user fill them in, and generate a document in the appropriate format and language to be downloaded by the user. CMS is used on this subsystem to manage the static textual content (help, tooltips, labels & instructions) fully multilingual as it does for the rest of Europass site. The Instruments On line applications are the following:

- Europass CV
- Europass LP (Language Portfolio)

In order to allow the multiple editing of an instrument (without database storage), each instrument can be serialized and unserialized in an XML file. This XML file holds all the data of the instrument, along with user preferences specifying part of the appearance.

The Europass documents are generated by the direct, programmatic synthesis of the data supplied in the on-line HTML forms, to the XML-based OpenOffice (sxw) format. After the SXW file is created, if requested, it is transformed to the appropriate format (PDF, Microsoft Word), through the OpenOffice software running in server mode.

(3) **Instruments Web Services**: This subsystem is a standard SOAP Access point to the instruments. It relies on the Instruments APIs to provide online generation of documents to external systems.

1.3. The Europass Mobility management tool

1.3.1. Introduction

The Europass Mobility is a standard document to record any organised period of time (called Europass Mobility Experience) that a person spends in another European country for the purpose of learning or training. This includes for example a work placement in a company, an academic term as part of an exchange programme or a voluntary placement in an NGO.

The mobility experience is monitored by two partner organisations, the first in the country of origin and the second in the host country. Both partners agree on the purpose, content and duration of the experience; a mentor is identified in the host country. The partners may be universities, schools, training centres, companies, NGOs, etc.

The Europass Mobility is completed by the home and host organisations involved in the mobility project in a language agreed between both organisations and the person concerned.

Cedefop has developed a distributed, Internet-based tool for the management of the Europass Mobility by National Europass Centres and partner organisations. The system will gradually be deployed to all European countries by the end of the year 2009.

The Europass Mobility management tool is developed and maintained by Cedefop in Thessaloniki and is hosted partly by Cedefop and partly by the NECs.

1.3.2. Technical Architecture

The Mobility System is a distributed system that provides means of information exchange between National Europass Centers (NECs). Each NEC has its own server that can communicate with remote NEC systems and that can provide partners (Companies, Universities etc.) a way of formulating and exchanging Mobility data. From the user perspective (user experience) each partner is connected to an application that it is presented as a Thin Client application, i.e. HTML & JavaScript, and it can be accessed by a common Internet browser.

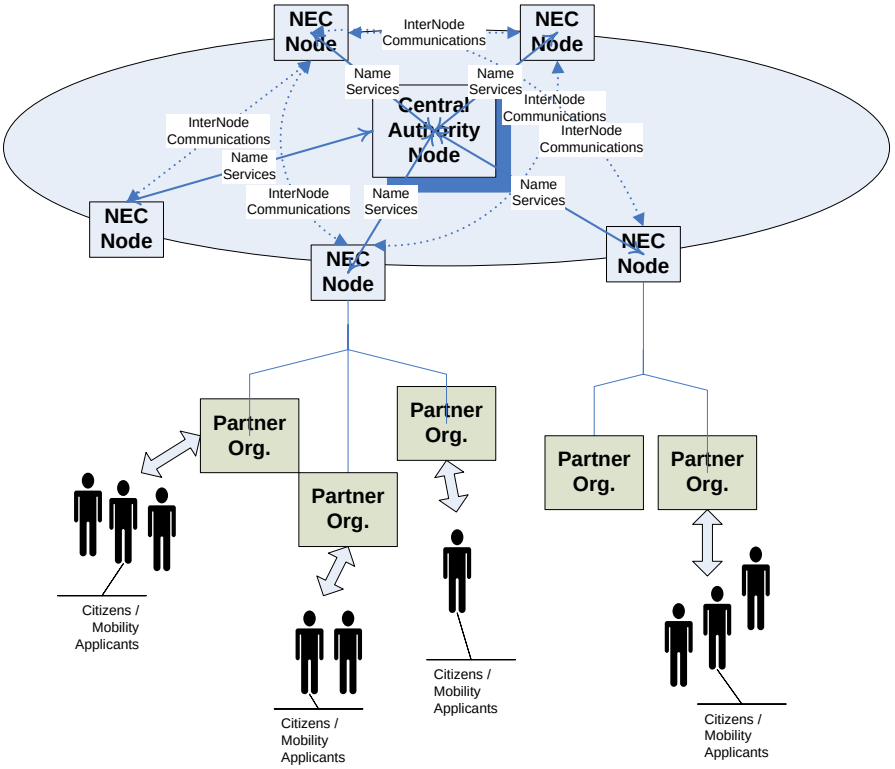

Figure 1- Europass Mobility System Architecture

From the functional perspective, Europass Mobility can be divided in the following subsystems:

1. **The Naming Web Service,** which is responsible for distributing information for all available NECs on the system, playing the role of a central repository. This is a standard SOAP Access Point.
2. **The Naming Application,** which is an administrative application for administering the central NEC repository, and collecting statistics.
3. **The NEC Application,** which is the application that each NEC is

setting up, so that NEC partners can connect to it, and produce Mobility Records (Experience data) to be exchanged between different partners, allocated on remote NEC systems.

4. **The NEC Web Service** that is responsible for accepting experience data from remote NECs, and that can provide other information to remote NEC systems. This is a standard SOAP Access Point.

The first two subsystems make the Central Authority System, that is only installed and working on a central location/institution, such as Cedefop.

The other two subsystems provide the NEC functionality, and are the components that each NEC should operate in order to facilitate information exchange.

All data transfers are using the HTTPS protocol. Certificates are provided for the time being from a central Open CA Server (Certificate Server).

XML files that are exchanged and that contain sensitive data are always signed. Also all web services are secure and client certificates are used to provide authentication during the invocation of web services.

1.3.3. Further information

For further information, you should consult the Europass Mobility Paper, titled "A Distributed System for Issuing Europass Mobility Documents" available in several European languages at: http://europass.cedefop.europa.eu/europass/home/vernav/InformationOn/EuropassMobility/Restricted/MobilityDownload/MobPaper/navigate.action (username: mobility, password: mobility), as well as access the system directly at http://nec-cedefop.mobility.cedefop.europa.eu. (request for username and password through the interface)

1.4. Technologies

The development is platform neutral. It does not require the existence of a particular Database server or Web server. For the implementation of this project, the following are used:

- Web Server: Apache 2.0.50+
- Servlet Container: Apache Tomcat 5.0.28+
- Database Server: MS SQL Server 2000, HSQL DB 1.8

The architecture is database independent and the possibility to use other available databases (such as Oracle) is given.

1.4.1. J2EE

J2EE framework is used for the development of the application logic, i.e. Content Management System and Instrument specific applications in this project. The J2EE platform uses a multi-tiered distributed application model. This means application logic is divided into components according to function, and the various application components that make up a J2EE application are installed on different machines depending on which tier in the multi-tiered J2EE environment the application component belongs.

For this project the following J2EE artefacts are used:

- The Tomcat Server used supports Servlet 2.4 and JSP 2.0 specifications
- Model View Controller: Apache Struts 1.2.4+

1.4.2. OpenOffice

The OpenOffice platform was chosen for the development of those subsystems of the Instruments On-line applications requiring the generation of files in formats that are able to be processed by Office suites (MS DOC & RTF), or printed in a local to the user printer (PDF). The OpenOffice server is a part of the OpenOffice suite and can be deployed simply by installing the OpenOffice suite. The recommended type of installation, is Network Installation, that comprises of the following parts:

- A centralized installation, keeping the necessary program files.
- Multiple workstation installations on users that exist on the machine. This installation installs only setting and configuration files.

This installation schema favours the usage of the Open Office server in a multiple listener mode, one listener for each user, enabling load-balancing features.

The OpenOffice version used is 1.1.3+ (but not 2.0).

1.4.3. Java 3rd Party libraries

All the libraries in this project are Open Source libraries, licensed under the Apache or Apache like License. The following are of major importance for the project:

- The well known ORM Library Hibernate 2.1 is used for the implementation of the Data Abstraction Layer. Hibernate belongs now to the JBoss Open Source Project.
- The Open Symphony OSCache library is used for the implementation of the site caching layer. The version used is 2.0.2.
- The Apache Lucene 1.4 library is used for the implementation of the multilingual site indexing layer.
- The iTEXT 1.3.6 library is used for PDF creation and manipulation. Each PDF produced is signed, and with attachments.
- Apache Java XML-Security 1.3 is used for XML signing.
- A lot of libraries from the Jakarta Commons project are used to provide utility methods for file processing, encoding, collection processing, etc.

1.4.4. Apache Axis

For the implementation of the WEB Services, Apache Axis 1.3 is used. This is an implementation of the SOAP protocol, written in Java that is application Server independent. It supports:

- W3C SOAP [WWW] v1.1 and [WWW] v1.2 Candidate Recommendation
- W3C Web Service Description Language (WSDL) v1.1
- Sun SOAP with Attachments API for Java (SAAJ) v1.1

Also, the Axis code has successfully passed all of the JAX-RPC and SAAJ TCK (Technology Compatibility Kit) tests.

1.5. Subsystems

From the implementation perspective, Europass is composed by the following subsystems:

- **Site caching layer.** This layer is based on Open Symphony OSCache component, an open source, widely used, high performance J2EE caching framework. It is used to improve speed on the site, by caching information, that are not changing continuously, such as the site's layout, menus etc. By using site caching, the need for page compilations on the application server, and actual database access is minimized. With a large set of capabilities, OSCache allows the caching not only of entire pages, but also of 'tagged' parts of pages.
- **Multilingual Indexing Layer.** This layer is based on the Apache Lucene library, along with the Snowball project which supports indexing for most of the European languages. This layer is composed of a batch indexing engine which crawls all the contents of the site in order to produce index data, and a search renderer which allows the users to search for specific content and navigate to it through the results.
- **Content Rendering system**
  The Content Rendering system is responsible for the actual site presentation to the site visitor. It gets information stored in the CMS database, and depending on their type, by using the appropriate renderer, constructs the actual page, and presents the results to the visitor of the site. There can exist different type of renderers.
- **Application logic**, including the Instrument online Applications & the Content Management back office.

  The **content management system (CMS**) enables authorized persons to develop a web – site and to publish information to their web – sites. The CMS holds all necessary information, in order to construct a site, such as:

  - Documents, data, and their possible translations
  - Relationships between categories, contents, and content types.

  CMS contains the following subsystems:

  - Document management (Insert, edit, approve, publish)
  - User rights management
  - Statistics
  - Mass Translation tool
- **Office document generator (Open Office server)**. This is a separate server running standalone as a daemon process, and accepts requests through its API. These requests concern the generation of files in the following formats: MS Word, PDF, SXW (OpenOffice.org).
- **Data abstraction layer.** This layer is based on the Open source project called "Hibernate" and it controls the access of data. It is used to provide database server independence. All access to the database are implemented through the API this layer provides for specific databases. This API provides specific Java methods to the application programmer, which can be implemented to access the stored data. On the other side (the connection to the database) this layer has implemented the appropriate connections and data management and retrieval

function for any specific database is used. Thus the application can be ported to use any other database server, without having to change the code that access data.

- **Database Schema.** The database stores all the site data (content) in a multilingual format. A meta model specifies the available types of contents that can be stored in the database and their actual values are stored separately using a locale (language / country association) based repository.