eSciDoc Infrastructure: a Fedora-based e-Research Framework

Frank Schwichtenberg, Matthias Razum

FIZ Karlsruhe, Hermann-von-Helmholtz-Platz 1,
76344 Eggenstein-Leopoldshafen, Germany
{firstname.surname}@fiz-karlsruhe.de

Introduction

eSciDoc is the open-source e-Research environment jointly created by the German Max Planck Society and FIZ Karlsruhe. It consists of a generic set of basic services (“eSciDoc Infrastructure”) and various applications built on top of this infrastructure (“eSciDoc Solutions”). This presentation will focus on the eSciDoc Infrastructure, highlight the differences to the underlying Fedora repository, and demonstrate its powerful und application-centric programming model. In the end of 2008, we released version 1.0 of the eSciDoc Infrastructure.

Digital Repositories undergo yet again a substantial change of paradigm. While they started several years ago with a library perspective, mainly focusing on publications, they are now becoming more and more a commodity tool for the workaday life of researchers. Quite often the repository itself is just a background service, providing storage, persistent identification, preservation, and discovery of the content. It is hidden from the end-user by means of specialized applications or services. Fedora’s approach of providing a repository architecture rather than an end-user tool accommodates well to this evolution. eSciDoc, from the start of the project nearly five years ago, has emphasized this design pattern by separating backend services (eSciDoc Infrastructure) and front-end applications (eSciDoc Solutions).

Differences between Fedora and eSciDoc

Fedora provides a very generic set of functionalities, addressing the needs of various communities and use cases. On the other hand, this means that it only provides low-level functionality, requiring developers to spend time implementing high-level services. eSciDoc tries to fill that gap by adding these high-level services on top of Fedora while hiding some of the more complex aspects of Fedora, thus increasing the productivity of developers. However, this advantage contrasts with a reduction of flexibility.

1. Datastreams and Object Patterns

A Fedora Object consists of several datastreams, which contain either XML or binary content. The repository developer has to define the layout and allowed contents for each datastream – the ‘content model’. Here lies one of the challenges to start with Fedora, even for simpler use cases. Therefore, eSciDoc introduced ‘object patterns’ for basic object types: items and containers (see figure 1). For both object patterns, the layout, naming, and allowed content is predefined. eSciDoc objects are therefore less flexible, but simplify data modeling. Still, they provide flexibility where needed (e.g. storing metadata records). Our experience shows that this model accommodates for most use cases.

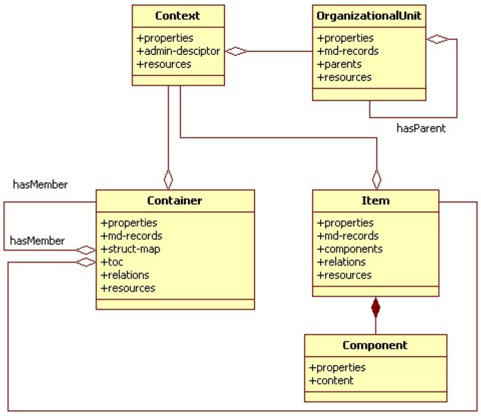

Figure 1: eSciDoc Data Model, showing the two object patterns ‘item’ and ‘container’. Items may have one or more components, which hold the different manifestations of the content.

The **item pattern** represents a single entity with possibly multiple manifestations. The **container pattern** aggregates items or other containers. Both support multiple arbitrary metadata records, object relations (within and outside of eSciDoc), and some logistic and lifecycle properties. All the information is stored in one or more Fedora objects, following the ‘atomistic’ content model paradigm. However, the user will only work with a single eSciDoc object. All the complex work is done behind the scenes by the eSciDoc Infrastructure.

2. Object Lifecycle

Both items and containers implement a basic lifecycle in the form of a simple workflow. Each eSciDoc object derived from one of the object patterns is created in status ‘pending’. Submitting the object forwards it to the quality assurance stage, from where it can be either send back to the creator for revision or be released (i.e. made publicly accessible). In rare cases, released items need to be withdrawn (e.g., because of copyright infringements), which is the last status in the lifecycle. For each status, different access rights (based on policies, roles, and scopes) may be defined. Moving objects from one status to the next is as easy as invoking a single method.

3. Versioning

eSciDoc extends Fedora's versioning approach. Because of the atomistic content model, a single eSciDoc object may in fact be a graph of Fedora objects. However, the user conceives eSciDoc objects as one entity and expects what we call 'whole-object versioning'. Even for actions that involve modifications to several datastreams or even several Fedora objects, the eSciDoc Infrastructure will only generate a single intellectual version of the whole Fedora object graph. Instead of timestamps, versions are numbered, which conforms better to user expectations.

Not all method invocations create new versions, e.g. forwarding an object to the next state in its lifecycle. Therefore, eSciDoc maintains an additional event log that tracks all actions using PREMIS[1].

4. Application-oriented Representation

The fact that an eSciDoc object actually is a graph of Fedora objects with multiple datastreams is completely transparent to the user. The eSciDoc Infrastructure exposes its contents as XML representations, which contain all relevant information for typical application scenarios, version information, metadata, and references to other objects or parts within the same object. Making object relations explicit by means of XLink href's allow for easy navigation through the object graph. Seldomly requested parts of an eSciDoc object, like the event log, are not part of the standard representation, but can easily be retrieved by means of 'virtual resources'.

5. Authentication and Authorization

All authentication and authorization functionality is encapsulated in the eSciDoc Infrastructure, therefore the application programmer has not to consider the complexity of the implantation – this is especially important when several applications run on the same infrastructure, eventually sharing data. Every request to the infrastructure has to pass the included authentication and authorization layer to reach the actually business logic.

The eSciDoc authorization relies on roles, scopes and policies. A role represents a set of policies defining the privileges for performing actions in accordance to defined conditions, e. g. in accordance to object states. By granting these roles to users, their privileges are defined. Additionally, for each eSciDoc role a scope is defined that specifies for which resource objects the role has been defined. The policies of a role are only evaluated if the object is within the scope of the role.

[1] http://www.loc.gov/standards/premis/

Due to the distributed nature of the eSciDoc authentication, the system does not come with a user administration. All users are maintained in local identity management systems (either LDAP or Shibboleth). The infrastructure never stores user credentials, like passwords. However, in order to be able to associate users with roles, the infrastructure creates and maintains proxy objects for users that have accessed at least once an eSciDoc Solution.

Application-oriented Programming Model

The eSciDoc e-Research environment is built as a service-oriented architecture (SOA). The infrastructure consists of several independent services. Each service implements both a REST and a SOAP API. The APIs support simple CRUD and task-oriented methods. These APIs together with object patterns, their XML representations, versioning, and the powerful authentication and authorization form eSciDoc's application-oriented programming model, which focuses on the mindset of application developers and hides the technical details of the implementation as much as possible.

In our presentation, we will show how easy it is to create a minimal, yet useful application on top of the eSciDoc Infrastructure. It will integrate the *Schema Driven Metadata Editor for eResearch* developed by Archer and MAENAD[2] in order to allow comfortable editing of metadata records of eSciDoc Objects. It also exposes the ability to view image content via the integrated DigiLib. We will introduce the idea of a simple client based on XSLT transformations of object representations delivered by the eSciDoc Infrastructure. Including manipulation facilities, the views of such a client may be seen as pluggable *mini-solutions*.

The CRUD-based HTTP programming interface can easily be called from existing applications or even websites. Every call is not only secured by the authorization component, but leads in case of insufficient authentication to a Shibboleth login site or an LDAP backed form. Because of the build-in simple workflow, versioning, and authentication/ authorization, already the thinnest possible client supports basic repository features. So the eSciDoc Infrastructure is a *ready-to-integrate*, fast prototyping, developer-friendly application framework.

Conclusion and Outlook

The eSciDoc Infrastructure encapsulates Fedora as its core component, but adds a wide range of higher-level services and its application-oriented programming model. It allows to build various types of solutions, from light-weight Javascript hacks to fully-blown Java applications. It fulfills the vision of creating an efficient, flexible, programmer-friendly e-Research framework supporting web-based publication, collaboration and communication for research environments assembled with the repository capabilities of Fedora Digital Repository.

[2] http://metadata.net/sfprojects/mde.html