

# A Service Infrastructure for The European Library

**Theo van Veen**, Koninklijke Bibliotheek, [theo.vanveen@kb.nl](mailto:theo.vanveen@kb.nl)

**Christian Sadilek**, Austrian Research Centers GmbH – ARC,

[christian.sadilek@arcs.ac.at](mailto:christian.sadilek@arcs.ac.at)

**Ross King**, Austrian Research Centers GmbH – ARC, [ross.king@arcs.ac.at](mailto:ross.king@arcs.ac.at)

**David Fuegi**, Eremo, [David@fuegi.info](mailto:David@fuegi.info)

**Abstract:** As part of the TELplus project<sup>1</sup>, a service infrastructure has been developed for The European Library that facilitates personalized use of functionality available on the Web. The main ingredients for this service infrastructure are a data model for describing existing Web functionality and how users want to invoke that functionality, and a registry containing these descriptions. The proposed concept can be considered as a low-barrier combination of OpenURL, Mashups and other initiatives for service composition. What is new in our approach is that users can describe the interaction of services with the portal. The registry enables users to submit these service descriptions and share them with others.

## Introduction

TELplus is a European Commission project funded under the *eContentplus* Programme and aims at strengthening and extending The European Library (TEL)<sup>2</sup>. Objectives include OCR-ing twenty-million pages of important multilingual content, improving full-text indexing, investigating automatic vocabulary mappings, adding new services and creating a modular and personalized service infrastructure. In this paper we describe a personalized service infrastructure that allows users to integrate existing functionality available on the Web with the European Library portal.

When users are searching in The European Library portal they expect not only that TEL will provide them with the information contained in the TEL libraries, but that TEL will also bring them to content in other online databases and repositories. They may want to use the search results for further navigation, for example by searching in Google or online book vendors. These types of deep links are already supported by the portal. However, users may also want to link to websites other than those that were predefined by TEL and they may want more than simple linking; for example, enrichment of metadata by means of external services. In many cases the interface of a portal is based on user surveys and the requirements of the average user. But different users may have different objectives that can also change over time. For example a user carries out research in a specific area for which she needs articles in a specific foreign language. During that research she might want to search for articles in that language and tell the portal to automatically translate the abstracts into her own language. It is not easy to anticipate this kind of specific user requirement.

Currently we are not aware of any existing portals that allow users to describe a personalized integration of existing web applications. Therefore we propose a service

---

<sup>1</sup> TELplus: A building brick in the creation of Europeana, the European digital library, museum and archive. See <http://www.theeuropeanlibrary.org/telplus/>

<sup>2</sup> <http://search.theeuropeanlibrary.org/>

infrastructure that lowers the barrier for users to integrate all kinds of existing web functionality with the European Library portal and other portals, even for functionality that is not known or controlled by the portal provider. The basis for that is a model for describing the integration of web applications. Describing services and how users want these services to be integrated is not trivial, because most services require a priori knowledge in order to invoke them.

Terminology: In this paper we use the terms “service”, “web application” and “web functionality”. Strictly speaking a web application may contain several services that provide certain functionality. Originally we used the term *service descriptions* for what we now call service integration descriptions. The schema for the descriptions of web applications or services may in the future also be referred to by the acronym SIWA (Schema for Integrating Web Applications).

## Existing models for service descriptions and mashups

Currently there are several initiatives for service descriptions like the Web Application Description Language (WADL)<sup>3</sup> for REST [6] based Web Services, Web Services Description Language (WSDL)<sup>4</sup>, and the JISC Information Environment Service Registry (IESR) [1]. These initiatives describe service invocation, but do not describe how services can be integrated within a portal. There are also initiatives for describing service composition [7] like the Web Services Business Process Execution Language (WS-BPEL) [2] and Yahoo Pipes<sup>5</sup> etc. but these present a high barrier to unskilled users, even with tool support. This was the motivation for seeking a mechanism that has a low implementation barrier for developers and is easy to be understood and used by advanced users, even without programming skills. Our approach can be seen as a follow-up or extension of the OpenURL [3] concept and was to develop a data model for descriptions of service integration that enables portals to invoke services automatically depending on context and to channel the service output in a user-defined way. Portals supporting our description model are required to have some build-in knowledge of commonly used concepts relieving the user from having to define too many details.

---

<sup>3</sup> <https://wadl.dev.java.net/>

<sup>4</sup> <http://www.w3.org/TR/wsdl>

<sup>5</sup> <http://pipes.yahoo.com>

## The services infrastructure

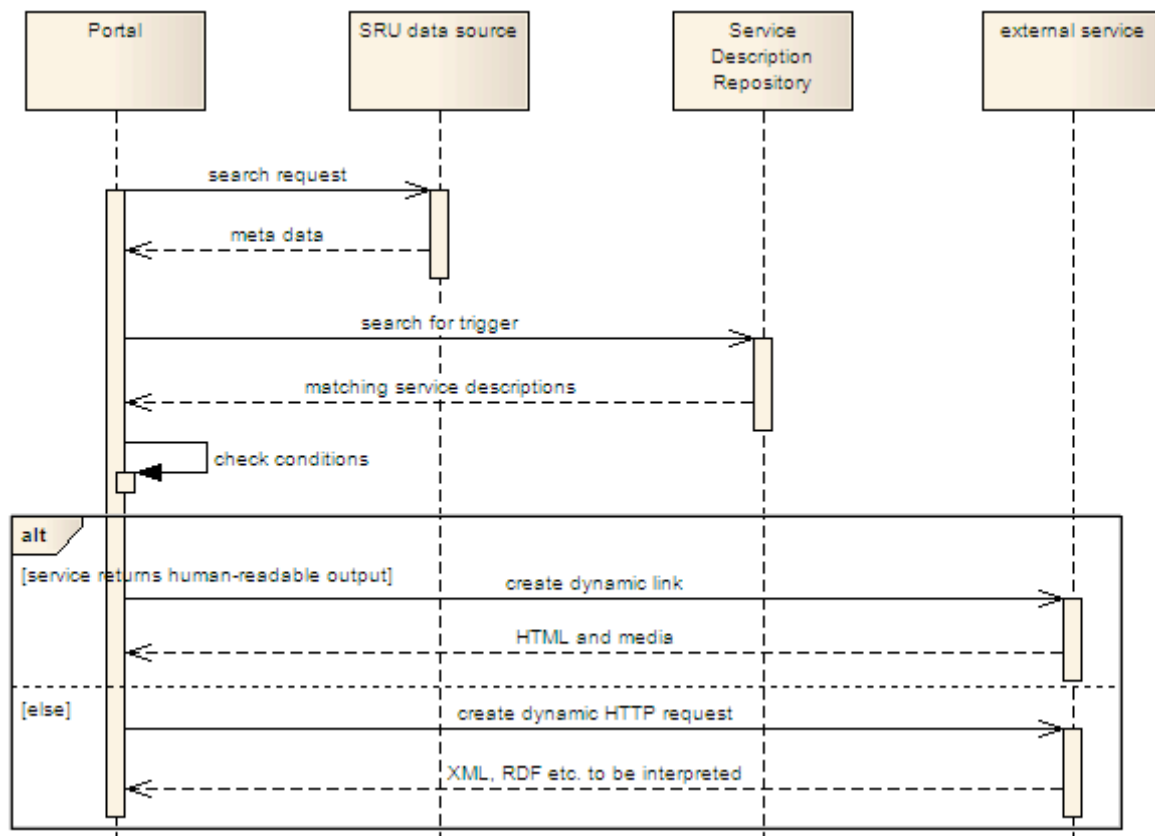


Figure 1: Service Integration Sequence Diagram

The concept is shown in Figure 1. The portal reads the service integration descriptions as provided by the European Library or provided by the user. After metadata have been received by the portal, for example as a result of a search, the portal checks if there are fields (triggers) for which a description specifies that a given service must be invoked, either automatically or on user request. This description also specifies the syntax of the URL to invoke the service, which URL parameters receive metadata as input, how the service must be accessed and how the output should be used. The first step in realizing such an infrastructure is developing an initial version of a data model that covers a wide spectrum of known services. Next steps will cover more complicated situations that arise when we are actually using the infrastructure. For developing the data model a test portal was used. The model is however not tied to a specific implementation. We also allow users to submit service integration descriptions to a service registry. In this way advanced users can share these descriptions with others..

## The Data Model

The purpose of the service integration descriptions is to enable users to search, and invoke services. The model is based on Dublin Core (title, identifier, description, format and type) and access is provided via SRU, as this presents a low barrier for searching and presenting the descriptions. The intention was to develop a light-

weight schema<sup>6</sup> that optimally fits our goals. The data model is meant as a standard for exchanging this information between users and portals and hopefully service providers. The main fields of the data model are described below. For an extensive description please refer to an article describing the model [4].

We will illustrate the basic mechanism by presenting an extremely simplified description for using Google images.

The URL that invokes an image search for a person looks like:

```
http://images.google.com?q=<name of person>
```

A user might want to provide a search in Google images available as link for each creator (that is, each person identified as a creator in a metadata record). So the input parameter “q” must be assigned the value of the metadata field “creator”. In this case, the description will look like:

```
<service>
<title>Google images</title>
<identifier>http://images.google.com?</identifier>
<trigger>creator</trigger>
<inputParameter>q</inputParameter>
</service>
```

The “trigger” element tells the portal to offer a link to Google images if a value for the metadata field “creator” is present and that this value should be used as input to the service invocation. The element “inputParameter” defines the URL parameter, in this case “q”, that will get the input. So, if the metadata contain a field with “creator=shakespeare“, then the portal will create a link with “http://images.google.com?q=shakespeare”.

Some additional fields in the description model are briefly explained below:

**identifier:**

The identifier field is used to specify the base-URL of the service, including the fixed parameters. When the base-URL or a fixed parameter is changed it is considered to be another service. There is no identifier for the actual service and the description should also not be considered persistent. This choice may be criticized but it prevents false confidence.

**accessType:**

The manner in which a service is accessed. Currently supported values are linkOnly, GET, POST, JSON, SRU. For GET, POST and SRU the output of the service is read via an HTTP request. For other values than linkOnly the portal is assumed to operate on the output or a part of it, which is specified in typeOfUse. The intention is that in future versions more standard protocols will be supported.

**typeOfUse:**

Defines how to use or present the output of the service; for example, replace the original input data, create a list of data items that can be used for new queries (query expansion or query reformulation) or alert the user if there is a non-zero result count.

---

<sup>6</sup> TELplus service description model  
<http://dev.theeuropeanlibrary.org/tpportal/model/>

**invocation:**

Defines whether the service is to be invoked automatically or on user request for example as a menu option or a button.

**extraCondition:**

This field specifies an extra condition for presenting or invoking the service, additional to trigger. In most cases an extra condition will be the presence or the value of a specific metadata field.

**varParameter:**

Defines an extra parameter to obtain input from metadata fields.

**serviceLabel:**

Defines the text to be presented to the user, for example in a menu of services that are available for a certain context.

**fieldSpec:**

Defines which part of the output should be used. This may be a JSON variable or an XPath expression.

**callbackParameter:**

For JSON requests this field specifies which URL parameter is used to define the JavaScript function that will process the JSON response.

**sessionParameter:**

This field specifies the URL parameter that contains a session identifier

**sessionURL:**

The URL to be used in advance before accessing the service for obtaining a session identifier.

**authorization:**

This parameter specifies the URL of the authorization service that has to be accessed before obtaining access to the actual service.

Some elements require additional attributes and some elements require a specific syntax. The model does not yet include service type specific elements. In order to allow extension of the service descriptions without being forced to change the model, it is possible to add service-type-specific elements in a special container element in such a way that the description remains valid with respect to the XML schema<sup>7</sup>. The portal is supposed to support all other elements in the model. Specification of service-type-specific elements will be included in a follow-up of the current model.

We consider this version of the schema as a first step in the development of a more refined schema. A new version will be based on our experience with the current model, feedback from users, service providers, and service integrators and especially with service descriptions submitted by users. The main challenge will be to extend the model for integrating services that cannot yet be described within the current model.

## The Service Registry

The number of potential useful services might become too large to present them all in the portal. It is expected that users may want to specify their own preferred services. This can be accomplished in various ways, for example by allowing the user to

---

<sup>7</sup> <http://www.w3.org/XML/Schema>

specify a URL of a service description file. The services registry can be used to allow users to select services they wish to add to their personalized service descriptions. The TEL service registry is accessible as any other TEL collection via SRU. Because the descriptions are based on Dublin Core (with extensions), most SRU-compliant applications will at least be able to present the Dublin Core fields. Another way of using the registry is by allowing the user to select a service on the fly for ad hoc usage, for example when the user is interested in an abstract to be translated but the translation service is not in his list of preferred services. The portal might offer the option to search for additional services in the registry and temporarily add the service to the user's list of services. Service descriptions will in general be made by advanced users only. Therefore users are offered the option to submit service descriptions to TEL. After moderation the service descriptions will be added to the registry and become available for all users. We are also developing a mechanism for the automatic creation of service descriptions for Web pages that are suited for this purpose.

## **Demonstrator Portal**

A test portal<sup>8</sup> has been developed using JavaScript. The intention is to use this demonstrator for collaborative further development of the service infrastructure. It focuses only on the service infrastructure without the burden of layout and additional functionality, and more options and possible services are implemented than will initially be available in the TEL production portal.

## **Services Created in TELplus**

A number of services have been integrated with the portal using the described infrastructure. Examples include annotation services, a service to highlight search terms in digitized text images, a service to present historical maps with a timeline on Google maps, a text analyser and more. Although there is not yet an established policy for using TEL services by others outside the scope of TEL, it is technically possible for users to do this. If TEL enables the use of external services it seems only fair to allow others to use TEL services. In order to prevent overloading our servers, we have build in a so called fair-use checker in some services. In some cases TEL might benefit from users using the TEL services loosely coupled with other portals for example to learn from user behaviour. A good example is the annotation service [5]. By allowing users to annotate objects that are not part of TEL, it increases the impact of the annotation database because users will navigate in a much richer annotation environment. Another advantage of sharing services is that it facilitates the standardization of access to services of a specific type. In the same way SRU has become the main standard for search and retrieval, we hope to set the standard for other types of services.

---

<sup>8</sup> <http://dev.theeuropeanlibrary.org/tpportal>

## Open Issues

One open issue is the cross-domain security restrictions of browsers. This makes it difficult to access services from other domains by an HTTP request. Another issue is the legal aspect of using services from other providers. Of course TEL is allowed to describe services of other parties but it will not always be allowed to integrate these services with the TEL portal without permission. To avoid the cross-domain restrictions a proxy has been developed. To use this proxy, a user must confirm a legal statement about responsibility because TEL cannot control whether a user has permission to integrate a service with the TEL portal. Another issue is the restricted number of simultaneous connections with a Web server and thereby the proxy. This is solved by introducing a simple protocol for accessing the proxy. When the portal invokes a request via the proxy the proxy returns an acknowledge message in case there is already a pending request. When the response of the first responding target is received the portal initiates a new request to wait for the response of the next target. In this way the portal is completely AJAX<sup>9</sup> compatible and allows the user to continue even if there are more pending requests. This is extremely useful for services that need more time to provide a response or for chaining multiple services. The portal must of course alert the user for delayed responses and requires additional intelligence to decide whether a delayed response is useful for the user.

## Conclusions

At the time of writing of this paper, support for the integration descriptions is only available in the test portal. There are services for which invocation and usage can not easily be described by the current data model; access for these services must in general be hard coded. The next step in the development of the data model is to attempt to manage these situations with extensions to the model. Nevertheless, we can draw the conclusion that the concept is quite promising because we can enrich the functionality of portals that support the use of service descriptions without coding or with minimal additional coding.

While developing the service integration data model, our intention was to describe the world as it is and not how we want it to be, because many services do not follow a standard protocol or use standard formats. As demand for service integration rises, however, we expect more standardisation and uniformity in this area. This should help to keep the model simple and to maintain a low barrier to adaption of our service integration approach.

## Acknowledgments

This work has been partially supported by the TELplus Targeted Project for digital libraries, as part of the eContentplus Program of the European Commission.

---

<sup>9</sup> AJAX: Asynchronous JavaScript and XML  
[http://nl.wikipedia.org/wiki/Asynchronous JavaScript and XML](http://nl.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML)

## References

1. Apps, A.: The JISC Information Environment Service Registry. *ASSIGNation* 22(3) pp 9–11 (2005)
2. OASIS Standard: Web Services Business Process Execution Language Version 2.0 Organization for the Advancement of Structured Information Standards (2007)  
<http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
3. ANSI/NISO Standard Z39.88: The OpenURL Framework for Context-Sensitive Services (OpenURL). National Information Standards Organization (2004)
4. van Veen, T., Petz, G., Sadilek, C., Koppelaar, M.: Sharing Functionality on the Web - A Proposed Services Infrastructure for The European Library. *D-Lib Magazine*, Volume 15 Number 1/2 (2009).  
<http://www.dlib.org/dlib/january09/vanveen/01vanveen.html>
5. Haslhofer, B., Jochum, W., King, R., Sadilek, C., Schellner, K.: The LEMO annotation framework: weaving multimedia annotations with the web. *International Journal on Digital Libraries*, 10.1007/s00799-009-0050-8. Springer, Heidelberg (2009)
6. R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.
7. C. Peltz, "Web Services Orchestration and Choreography," *Computer*, vol. 36, no. 10, pp. 46–52, 2003.