

ECP-2006-DILI-510003

TELplus

Service description model for The European Library

Deliverable number	<i>D4.2</i>
Dissemination level	<i>Public</i>
Delivery date	<i>16 January 2009</i>
Status	<i>Version 1.0</i>
Author(s)	<i>Theo van Veen, Georg Petz</i>



eContentplus

This project is funded under the *eContentplus* programme¹,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

¹ OJ L 79, 24.3.2005, p. 1.

Executive summary

A data model has been created to describe services that might be used in combination with the TEL portal. A list of potentially useful services and a description of the mechanism to submit descriptions of such services to the TEL services registry is provided in D4.1.

The data model is presented as a machine readable XSD schema and in plain human readable text. Service descriptions enable a services infrastructure being proposed for the European Library portal as part of workpackage 4 of TELplus. The purpose of service descriptions is to provide information to The European Library portal about when and how services have to be accessed without having this hard-coded in the portal software. In this way the functionality of the TEL portal can be enhanced and personalized by means of existing services that do not have to be part of TEL.

The model has to a large extent been tested by means of a demonstrator portal that has been created for this purpose. The demonstrator portal also provides a form to help the user in creating service descriptions and submit them to TEL. After moderation they can be published in a services registry, which is also part of workpackage 4 of TELplus (see D4.1).

The XSD schema and the explanation and justification of the data model are provided in this document together with additional information.

Some information like controlled vocabularies is currently not part of the schema to allow for extensions without the need to change the basic schema. Extensions are needed when new services are offered that require extra information in the service description.

We aim at other service integrators adopting our data model. This would make the service descriptions usable for other portals and will enable users to provide the same service descriptions to different portals.

It is recommended that TEL will actually use the description model in its portal and to maintain the model and allow external parties to submit as well descriptions as proposals for improvement of the data model.

About this document

The purpose of this document is to present and explain the data model for service descriptions. It starts with a description of the rationale behind the services infrastructure. Following that it explains all the elements of the data model with an explanation of their usage. Besides that the elements are presented in a user readable table.

To be able to extend the model without violating the schema we did not yet specify controlled vocabularies. The values that are currently used are specified in separate tables with an explanation of their meaning.

The document concludes with the actual XSD schema.

1 Introduction

There is a great deal of functionality available on the web that can be used by the TEL portal by means of an appropriate method of dynamic linking and in some cases integrating the results in the TEL portal. The purpose of service descriptions in the context of the European Library is to describe this functionality and to enable applications like the TEL portal to search, select and invoke these functions and use the output in an appropriate way. These functions are referred to as services and a service is considered to be anything that can be invoked by means of a URL.

Although the main focus will be on the TEL portal it is the intention that the service descriptions are also usable by other web applications. Service descriptions are not fixed for a specific service: users may change the service description to fit their own needs.

A service description contains a number of fields that are to be used by the portal to offer a service in the right context and with the right parameters. The portal has to interpret these fields for performing the right actions. Extensibility of the data model, while retaining backwards compatibility, is therefore an issue. Applications are not supposed to validate the service description by the current schema and are supposed to safely ignore new and unknown fields.

There are already several models for service descriptions available like the IESR model, WSDL, WADL and some others. The novel manner in which we want to use the service descriptions justifies the creation of a new model.

2 Rationale

Applications may perform operations on data (for example resulting from a search) by means of algorithms that are traditionally hard coded in the application. Nowadays much functionality may be available as a service somewhere on the internet and can be used by dynamic linking. A system engineer may decide use such services and to make such a decision he has to be able to find available services together with the information on what the service does and how to invoke that service. He also has to be able to search for different services that do the same job for example translation services. Those services will behave differently in terms of quality, performance and whatsoever. A services registry will help the system engineer to find appropriate services with all the relevant details.

We want to go one step further and allow the user to select services to perform some operation on their data. In that case invoking the service is not hard-coded but configurable or even more, the user may be allowed to specify a service without the web application knowing about the existence of that service. This goes further than the browsers favourites or search bars. The data are obtained by a previous operation for example a search or a requested digital object (text, image, video etc.) and the result of the operation doesn't have to be a link: the results of the operation may be integrated in the application invoking the service.

The user has to be able to find relevant services or actually the descriptions of those services and provide the information to the portal.

As there may be very many services some of which do the same type of operation (e.g. translation) but with relevant differences (e.g. the languages that they support or their performance) there needs to be a mechanism to choose the appropriate services and to propose the relevant services to the user. Which services are relevant, will depend on the context for example the presence of specific metadata. The user probably wants to specify a list of preferred services and let the portal offer these services automatically or on request.

Web applications like the portal need to be able to:

- 1) select a service based on the type of service and the context
- 2) invoke the service
- 3) use the output of the service in the most appropriate way

The concept of finding appropriate service descriptions will of course only work if there are service descriptions available in for example registries. In the context of the TELplus project a services registry will be created. If the concept proves to be successful it is expected that service providers will publish their services or that advanced users will publish service descriptions.

3 The service description model

Below we will explain step by step the fields that make up a service description.

3.1 Dublin Core

For the description of a service Dublin Core is used as starting point. The idea behind this is that it will be possible to search for all kind of objects like persons, images, services and others in an integrated way. These descriptions will follow for different types of objects a different data model but share a number of Dublin Core terms. In this way DC-aware applications are at least able to present the DC terms and smart applications might apply appropriate algorithms to process the full descriptions in a meaningful way. In case of a service description the application, in this case the portal, may propose the user to add the service to the user's preferred services.

A service will at least have:

- 1) A name (**dc:title**) of the service
- 2) An identifier (**dc:identifier**) which is the URL to invoke the service
- 3) A type (**dc:type=service**) to let applications know that the described object has to be treated as a service
- 4) A description (**dc:description**) to let the user know the relevant details for before selecting this service
- 5) The type of output (mime-type) is specified by **dc:format**.
- 6) For a service with a human interface the field **dc:language** specifies the language of the user interface

These fields should be sufficient for being interpreted by a user to search and select a service but generally not sufficient for an application to invoke a service. Other fields from the dcterms namespace may be used for describing the service but are in this context not considered very relevant for the goal we want to achieve.

The reason for using the URL as identifier is that we don't want to rely on resolution services for this purpose. When the base-URL of the service changes, this service is considered to be a new service and thus gets a new identifier. The identifier defines the service and not the description. A future option might be to add an identifier for the description of the service as soon as there is a functional need for it. Of course there is always an internal record identifier available that can be used for administrative purposes. The URL will be a base-URL and may contain additional parameters. The field "inputParameter" specifies the actual input parameter that has to be filled in by the application. For example searching in Google this is done by:

<http://google.com/search?hl=en&q=whatever>

When metadata field "subject" should trigger searching in Google the service description has to specify:

```
dc:identifier = http://google.com/search?hl=en
inputParameter = q
trigger = subject
```

In that case the portal will add the parameter "q" with the value of "subject" the identifier field to invoke a Google search. In some cases it is necessary to indicate that parts of the URL should be substituted by variable data obtained by the portal.

For example the value of dc:identifier for a Wikipedia URL may look like this:

<http://www.wikipedia.org/@subject@>

In this case the subject is not a URL parameter but a part of the URL and has to be substituted by the portal by a specific value.

3.2 Additional fields

To enable a portal to search, select and invoke a service and to use the output in an appropriate way we need additional fields. For each of these functions all necessary fields are described below.

Search:

For searching appropriate services the relevant fields are **dc:title**, **dc:type**, **dc:language**, **dc:description** but also the type of service (**serviceType**). For **serviceType** a controlled vocabulary might be desirable but is not yet established. .

Select:

For users to select a service one needs **dc:title**, **dc:language** and **dc:description**. For automatic selection by the TEL portal only the context is relevant. For example the link to a translation service should only be presented for textual parts like title and description. The field for this purpose will be the field "**trigger**". This trigger is generally a metadata field that also serves as input to the service. This trigger field may also be a generic data type like "text". In that case the portal may decide which data are to be considered as "text". In some cases there may be extra conditions, for example the number of hits being 0 or the type of object being an image. This is expressed in the field **extraCondition**.

The trigger, on most cases being the presence of a certain data field does not have to result in automatic invocation of the service. In most cases the user will actually invoke the service. The trigger field therefore does not really trigger the invocation of the service but triggers the portal to offer a link to the service. The field “invocation”, which can take the values “automatic” or “option” or “no”, specifies whether the service has to be invoked automatically. The value “no” is intended for only exchanging service descriptions. When offering the option to invoke a service this requires a text to be displayed telling the user what the service does which may be specified in the field **serviceLabel**.

The trigger field is supposed to be one of the following:

1. a metadata field possibly followed by a colon with an encoding scheme. The metadata are supposed to be Dublin Core but there is no restriction to that.
2. a data type prefixed with an underscore (e.g. `_text`)
3. a service type prefixed with an underscore (e.g. `_Translate`)
4. a global name indicating a specific condition (e.g. `_noHits`)
5. a specific service name

A web application does not have to support all these types of triggers, but the more it supports the more functionality it can offer.

Invoke:

For invoking a service the URL and the URL-syntax is needed. This will be in the field **dc:identifier**. This field contains the complete URL except the parameter that is being supplied by the application. This parameter is specified by the field **inputParameter**. If this is not a parameter but merely a part of the URL that has to be substituted the value of **inputParameter** should be exactly the piece of text in the field **dc:identifier** that has to be replaced and surrounded by “@” for easy recognition. In most cases the data that trigger the service will also be the input for the service. This requires the name of input parameter that takes the contents of the “**trigger**” field.

Extra parameters may be a fixed part of the URL but can also be specified as fields in the service description as **fixedParameter** and will have the form “name=value”. Parameters that are supposed to be supplied by the user are specified in the field **promptParameter**.

If extra parameters are not fixed but should get the value of a metadata field, this will be specified in the field **varParameter**. This will have the form “parameter name=field name”.

There are also a number of parameters for language, geographical coordinates that will be listed separately.

A parameter may have an operation as attribute. Possible values for the operation attribute are “URLEncode”, maxlength, etc. These will be described separately.

An important variable for the portal to access a service is the protocol to be used (SRU, Z39.50, SOAP, http GET, http POST etc.). This is specified by the field “**accessType**”.

Some services require authorization. If this is needed a priori to deep link into the service the portal must invoke the authentication service first before invoking the actual service. Another way of authorization is by a proxy service that checks for the IP address. In the first case the field **authorization** will contain the address of the authentication service. In the second case the contents will be the literal “useProxy”.

When a service requires a session id this is indicated by the field “**sessionId**” being the name of the parameter that should contain the session id. The field “**sessionURL**” specifies the URL to be used to obtain this session id.

Usage:

How the output of a service is supposed depends on a number of fields in the service description and possibly some fields in the metadata. Most important are the mime type and service type. The mime type (**dc:format**) indicates how to treat the output of a service. For example when the mime-type of a service is “image” the portal has to behave differently than in the case of “HTML”. Additionally the supposed user interaction should be specified, for example whether the output should be obtained automatically or on user request. This is done in the field **invocation**. Besides that we need a field indicating what action is supposed to happen when the service is invoked. This type of action can be just a link which can be considered as the default usage or can be something special like replacing the original data by the output of the service. This field is the **typeOfUse** and a list of possible values is shown in a separate table. An example of a type of use for a translation service is to replace the input metadata field by the translated output. The field **typeOfUse** may also be accompanied by the field **fieldSpec** specifying the part of the output that has to be used. This may be an xPath-like expression or the name of a JSON variable.

Another type of usage might be that the service is a website and that after some navigation in that website some results have to be used as input to the TEL portal. This is not specified in the service description but the portal may allow the user to drag information from that website and drop this into some input box of the portal. An example of such usage is that the user finds a picture in Flickr and wants to use this image for making an annotation in TEL.

One type of specific behavior of services embedded in existing web applications is that some require running in the top window. When they are started in a frame they will automatically replace the TEL portal by the context of the web applications. The field “behavior” with value “newWindow” indicates this.

It should also be possible to pass the output of a service to another service. For example translate the output and convert the translated output to speech. For this purpose the portal should be able to link to next service. This is specified by the field **nextService**. This field specifies the type of service that is needed or meaningful to process the output of the primary service. Rather than using a type of service it is also possible specify an URL. The description of this second service may also contain a field **nextService**. It is up to the portal to invoke each subsequent next service automatically or by means of user interaction.

Another way to pass the output to another service is by providing the URL of the service to be pre-processed by another service. That service then acts as a proxy or gateway and the processed output is presented to the user. An example is a “mobilizer” service to create output suitable for mobile devices. This URL may be put in the field **preprocess**. Notice that this is different from the use of **nextService**.

For services that provide XML with a schema that is not known to the portal one might provide a reference to a stylesheet in the field **stylesheet**. This stylesheet can be applied for transforming the output. There are different ways to apply such a stylesheet: transformation by the application, by the browser or by an external application.

For protocols like SRU there is a possibility to get some information about the service, which is called the explain record. For SRU and OAI it is clear but for other type of services there might be a description containing information about the service that is not part of this service description. The field **explain** can be used to point to an additional service description.

The **logo** field is meant to contain a logo of the provider of the service. The service provider may however require this logo to be shown as a condition of use in case there is not a link to a provider's web page but only an http request to obtain the data.

In some cases one might want to make use of all functionality of the original website containing a service and at a certain moment use the results as input for TEL. An example might be navigating in an external thesaurus website. At a certain point the user might want to link back to the TEL portal with a certain term taken from the thesaurus website. However as soon as control is handed over to another website the portal doesn't have control anymore. This cannot yet be solved in an easy way by means of the service description.

The model does not imply that all elements are supported and does not imply a specific implementation.

3.3 Overview of all fields

The table shows an overview of the general fields (that is for all services) in the service description. The second column shows the type of obligation with the following meaning.

M: mandatory

R: optional and repeatable

A: mandatory if applicable, if not applicable it should be empty

O: optional, not repeatable

In some cases the values of the fields does require specific interpretation or substitution by the portal. In these cases the values start with a “_” as prefix or are enclosed between “@”.

Field	O	Description
dc:title	M	The name of the service
dc:type	M	The type of the described object. This is always "service"
dc:description	O	The description of the service
dc:identifier	M	The URL to invoke the service. Variables that are not a URL parameter but that have to be substituted in the URL are marked by @. This string is than obtained from the inputParameter field. For example: a wikipedia URL can be written as http://wikipedia.com/@subject@ The inputParameter then should have the value @subject@.
dc:language	O	The interface language in which the service is available. This field is not meant for the supported source and destination languages of translation services.
serviceType	M	This field indicates the type of action performed by the service like translate, search, etc. See the controlled vocabulary below. The IESR type vocabulary is adopted as starting point
trigger	M	The name of the data field that triggers the potential invocation of the service. These data usually come from metadata. If not they start with an

		underscore indicating a value from a controlled vocabulary e.g. <code>_text</code> or <code>_query</code> .
extraCondition	R	The conditions under which the web application should use the service. This applies to values of the metadata e.g. "type=image" If not the condition start with an underscore indicating a value from a controlled vocabulary e.g. <code>_nohits</code>
accessType	M	The type of access to the service. Current vocabulary: GET, POST, SRU, OpenSearch, SOAP, JavaScript and applet. In case of SOAP an additional WSDL file may be required. In case of JavaScript the field "invokeScript" will contain the JavaScript function that is needed to invoke the service. In case of an applet the identifier field represents the URL to the source and the parameters are specified in the same way as for other services.
serviceLabel	O	The text presented to the user in case the service is to be presented as a menu option. If missing the service type may be presented. This field is considered to refine the service type.
inputParameter attribute: operation	A	The name of the parameter in the URL that has to be provided or substituted. This field will get the value of the "trigger" field. The operation attribute specifies the operation to be performed on the input like URLEncoding or removing non-alphanumeric characters.
fixedParameter	R	A parameter that is always supplied with a fixed value. The contents of this field will have the form "name=value".
promptParameter	R	A parameter that is supposed to be supplied by the user when invoking the service.
varParameter	R	Additional input parameters to be supplied by the portal. This takes the form "parameter name=field name".
typeOfUse	O	Indication of how the output is to be used. Possible values are " <code>_link</code> ": just link to the service (default) or " <code>_replace</code> ": use the output of a service to replace the input (e.g. in the metadata). In many cases this is accompanied by a field called "fieldSpec" indication the part of the output to be used.
fieldSpec attribute: unitName	O	Specifies the part of the output of a service that has to be used. The may be an XPath expression or a JSON variable
CQLQuery	O	Query with CQL syntax used in combination with the CQLQuery attribute for the inputParameter. The position where the input has to be substituted is marked by "@" e.g. "@creator@". Special values are "@_input@", "@_north@" etc.
sessionURL	A	URL to initiate a session (if the service maintains a session) and to get the session identifier
sessionParameter	A	Parameter that passes the session identifier to the service.
invokeScript	A	This field is mandatory when the service can only be invoked with a JavaScript function. In this case the web page as specified in <code>dc:identifier</code> contains this JavaScript and has to be loaded prior to invoking the service.
authorization	O	Address of authentication service if authentication is needed a priori to deep linking. When the use of a local proxy service is required this field contains the literal value " <code>_useProxy</code> ". Local means located at the same server as the TEL portal.
invocation	O	Field indicating whether the service has to be invoked automatically or on user request. Possible values are "option" (a link is to be presented to the user), "automatic" (in this case it is assumed that the results can be integrated in the presentation of the web application). Default is "option".
nextService	O	URL or type of the service that will take the output of this service as input. This URL is supposed to be a service that is described in one of the service descriptions. If not, it is assumed that the output of the first service can be appended to the URL of the nextService. If a service type is specified the portal will sort out the best candidate.
preProcess	O	URL of a gateway or proxy takes the URL of current service as input. This may be a proxy, a translating service a mobilizer etc.

behavior	<input type="radio"/>	Field indicating specific behavior if the service for example that it always restarts in the top window. Current vocabulary: newWindow.
explain	<input type="radio"/>	Contains the URL to get a description of the service, like explain in SRU, identity in OAI. For SOAP this field may contain the location and WSDL file.
stylesheet	<input type="radio"/>	For services that provide XML a stylesheet transformation might be needed to convert the output to HTML.
logo	<input type="radio"/>	Logo to be presented when the service provider requires this as a condition of use.

3.4 Service related elements

In most cases the portal can substitute values from for example the metadata without having a notion of the meaning of those metadata. In some cases however extra service related data are to be supplied by the portal. For example a translation service takes the direction of translation as parameter. Because the direction depends on the language of the source and the language of the user the portal has to be aware of the meaning of this parameter to find the right value for the direction parameter.

New services will sometimes require extensions to the data model. For these types of extensions a container element "serviceTypeSpecificParameters" has been introduced. When new services are introduced requiring specific elements these elements can be added in the service description without violating the XML schema. Web applications and portals do not have support all the service type specific elements and are supposed to ignore service descriptions containing unsupported elements.

The distinction between elements as specified in the schema and the unspecified service type specific elements is the following. Applications are supposed to be aware of the elements in the schema and take the appropriate actions for elements that are not supported. Unknown elements outside the service type specific container should be just ignored without making the service description as invalid. For the service type specific elements applications should check for elements that it supports and mark the complete description as being not supported when it contains unknown elements. These elements might be incorporated in new versions of the data model. How to deal with new versions of the data model retaining backwards compatibility is not yet specified.

The TEL demonstrator portal currently or in the near future supports the following service type specific elements.

northParameter eastParameter southParameter westParameter	<input type="radio"/>	Parameters used for geographical coordinates. For points the parameters for latitude and longitude are specified by northParameter and eastParameter.
supportedLanguages directionParameter	<input type="radio"/>	Translation related parameters.
scaleParameter winWidthParameter winHeightParameter widthParameter heightParameter xParameter yParameter	<input type="radio"/>	Parameter used for a zoom service and other image related services.

If in the long term the extensions prove to be of general use, the model will be appropriately extended.

3.5 Overview of service types

It would be convenient to make use of an existing vocabulary for service types. The JISC Information Environment Services Registry (IESR) vocabulary is used here as starting point. See: <http://iesr.ac.uk/>. This list is shown below. In the demonstrator portal the service types are not really used to determine the behavior of the portal. Currently they serve mainly as a global classification for searching services.

Service type	Description
Alert	
Annotate	Annotation service to add annotations or links to object metadata
Archive	
Ask	
Authenticate	
Authorise	
Contribute	
Find	Search and retrieval of objects
Harvest	
Info	Added by TEL for generic use if the other service types do not apply
Lend	
Locate	
Map	Show a location on a map
Monitor	
OpenURL	Added for linking to OpenURL resolvers; this is different from "Resolve"
Pay	
Personalise	
Rate	
Register	
Request	Request item
Reserve	
Resolve	Resolve identifier to a URL
Save	
Test	Added by TEL for testing purposes
Relate	Added by TEL for search and retrieval or related terms and name variants etc.
Thumbnail	Added for retrieving thumbnails (e.g. front pages of books) to add in the presentation of metadata.
Supply	
Translate	Translate text
Validate	
Zoom	Added by TEL for enlarging images

3.6 Vocabularies

The table below gives an overview of the currently anticipated or supported values for elements and attributes. They are mentioned in the schema but not all are yet part of a controlled vocabulary because of extendibility without violating the schema.

Element/attribute	Possible value	Meaning
accessType	GET	Service invoked with GET
	POST	Service invoked with POST
	SRU	Service uses SRU protocol
	SOAP	Service uses SOAP
	JSON	Service is access via a <script> tag
	JavaScript	Service is invoked by calling a script in the HTML
	applet	Service is an applet
typeOfUse	alertOccurrences	Give an alert when a specified field occurs in the output
	alertIfResultCount	Give an alert if the result count > 0
	createSearchList	Create a list of values that can be used for further navigation
	alertWhenText:<text>	Give an alert if <text> is present
	replaceText	Replace a metadata field by text in the output. It is up to the portal to mark and recognize the text
	replaceField	Replace a metadata field by a specific field in the output
	metasearch	Use service in a metasearch (SRU only)
	linkOnly	Just link to the service and present as web page (default)
	search	
	operation	URLEncode
makeAlphaNumeric		Remove all non alphanumeric characters
maxLength=<value>		Clip the length of the input field
point		Input exists of two parameters containing geographical coordinates
area		Input exists of four parameters containing geographical coordinates
CQLQuery		Input has to comply to CQL language
behavior	newWindow	Service will put open itself a new window

3.7 Special values

In some cases values do not correspond to metadata elements but are to be substituted by the portal. These special (global) values are not considered part of the data model but are used analogue to metadata elements.

trigger	_query	The input for the service is the value of the last query
	_dropped	The input of the service is a part of a web page dropped as input for the portal. The input is complete unknown and has to be analyzed and processed to extract relevant information.
	_text	Text as output from a service can be used as input for another service
	_XML	XML as output from a service can be used as input for another service
extraCondition	_noHits	The extraCondition is fulfilled when there are no hits.
	_contentType	The contentType taken from the response header of a service is considered as field name
CQLQuery	_north, _east, _south, _west	In a CQLQuery metadata from the last response may be substituted. The special values specify substitution of geographical coordinates
varParameter	_destLang, _srcLang	The parameters take the value of the language of the data source or the destination language, being the preferred language of the user

4 Example of a service description

Below a simplified service description of Google images is shown to explain the basics. The field trigger indicates that the service has to be invoked for the metadata field “creator”. The field “inputParameter” specifies that “q” is the URL-parameter that will get the value of “creator” and this will be appended to the field “identifier”.

```

<sd:service>
<dc:title>Google images</dc:title>
<dc:identifier>http://images.google.com/images?svnum=10&amp;hl=en&amp;btnG=Search&amp;</dc:
identifier>
<dc:type>service</dc:type>
<sd:serviceType>Info</sd:serviceType>
<sd:serviceLabel>Search pictures of the creator in Google</sd:serviceLabel>
<sd:triggers>
<sd:trigger>creator</sd:trigger>
</sd:triggers>
<sd:inputParameter>q</sd:inputParameter>
<dc:format>HTML</dc:format>
<sd:accessType>GET</sd:accessType>
<sd:invocation>option</sd:invocation>
</sd:service>

```

When the input parameter is not a “name=value” parameter, its position in the URL needs to be marked, and this marker will be considered to be the name of that input parameter. For example, in the URL <http://myhost/xyz/@par1@> the input parameter is marked by “@par1@”. Using the field identifier in this way might look “ugly” but is very effective, and we didn’t find practical reasons not to do it this way.

5 Recommendations

The data model for service descriptions is only a requirement when service descriptions are exchanged between users and portals as part of the services infrastructure as described earlier in this document. The impact becomes bigger when the use is not restricted to TEL but also adopted by others.

It is recommended that the TEL portal be modified to use these service descriptions. Even if users will not be allowed to integrate their preferred services with the TEL portal, these service descriptions can still be used for configuration and administrative purposes. The use of service descriptions is the alternative for hard-coding TEL selected services in the portal.

The schema as being developed here should be considered as one step toward a larger development. As soon as other parties adopt the services philosophy the impact of the data model will grow but also the need to adapt it to new requirements will grow. Therefore it is recommended that TEL will host the schema and will facilitate others to provide input to improve the data model and publish new versions. After the publication but during the TELplus project relevant feedback might be obtained from others working in this area and TEL should allow for improvements to allow other parties to participate in the services infrastructure.

Further development of the data model goes hand in hand with new services published in the TEL services registry (see D4.1). Submission of service descriptions for the TEL services registry should therefore also allow for proposing improvements of the data model.

6 Schema

Below the XML schema for service descriptions is presented.

```

<?xml version="1.0" encoding="UTF-8" ?>
<: <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:sd="http://www.theeuropeanlibrary.org/servicedescription"
  xmlns:dcterms="http://purl.org/dc/terms/"
  targetNamespace="http://www.theeuropeanlibrary.org/servicedescription"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://purl.org/dc/elements/1.1/"
    schemaLocation="http://dublincore.org/schemas/xmls/qdc/2008/02/11/dc.xsd" />
  <xs:import namespace="http://purl.org/dc/terms/"
    schemaLocation="http://dublincore.org/schemas/xmls/qdc/2008/02/11/dcterms.xsd" />
<: <xs:element name="services">
<: <xs:complexType>
<: <xs:sequence>
<: <xs:element name="service" maxOccurs="unbounded">
<: <xs:complexType>
<: <xs:all>
<: <xs:element ref="dc:title">
<: <xs:annotation>
  <xs:documentation>The name of the service.</xs:documentation>
  </xs:annotation>
  </xs:element>
<: <xs:element ref="dc:type">
<: <xs:annotation>
  <xs:documentation>The type of object (=service).</xs:documentation>
  </xs:annotation>
  </xs:element>

```

```

_ <xs:element ref="dc:description" minOccurs="0">
_ <xs:annotation>
  <xs:documentation>The description of the service.</xs:documentation>
</xs:annotation>
</xs:element>
_ <xs:element ref="dc:identifier">
_ <xs:annotation>
  <xs:documentation>The URL to invoke the service. Fixed parameters may be part of the dc:identifier.
  The parameter that is filled in by the portal should be left out. Variables that have to be substituted
  in the URL are marked as such in dc:identifier. The name of the parameter to be filled in by the
  portal is obtained from the trigger field. Remark: There is no field to uniquely identify the service or
  the service description itself.</xs:documentation>
</xs:annotation>
</xs:element>
_ <xs:element ref="dc:format" minOccurs="0">
_ <xs:annotation>
  <xs:documentation>The mime-type of the output of a service. Currently supported values are HTML,
  XML, image, video, sound, text.</xs:documentation>
</xs:annotation>
</xs:element>
_ <xs:element ref="dc:language" minOccurs="0">
_ <xs:annotation>
  <xs:documentation>The languages in which the service is available. This field is not meant for the
  supported source and destination languages of translation services.</xs:documentation>
</xs:annotation>
</xs:element>
_ <xs:element name="serviceType" type="xs:string" minOccurs="0">
_ <xs:annotation>
  <xs:documentation>This field indicates the type of action performed by the service like translate, search
  etc. The possible service types are initially taken from the IESR model with some
  extensions.</xs:documentation>
</xs:annotation>
</xs:element>
_ <xs:element name="triggers">
_ <xs:complexType>
_ <xs:sequence>
_ <xs:element name="trigger" type="xs:string" minOccurs="0" maxOccurs="unbounded">
_ <xs:annotation>
  <xs:documentation>The actual data that trigger the potential invocation of the service. This can be the
  name of a metadata field, a service type or one of some predefined values. A predefined value is for
  example "_query", meaning that the original query is used as input for the
  service.</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
_ <xs:element name="extraConditions" minOccurs="0">
_ <xs:complexType>
_ <xs:sequence>
_ <xs:element name="extraCondition" maxOccurs="unbounded">
_ <xs:annotation>
  <xs:documentation>The conditions under which the web application should use the service. This applies
  to values of the metadata, for example "type=image" or predefined values like
  "_nohits".</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
_ <xs:extension base="xs:string">
  <xs:attribute name="field" type="xs:string" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

```

```

    </xs:element>
  <xs:element name="accessType" type="xs:string">
  <xs:annotation>
    <xs:documentation>The type of access to the service. Current vocabulary: GET, POST, SRU, OpenSearch, SOAP, Z3950 and JavaScript. In case of SOAP an additional WSDL file may be required. In case of JavaScript the field "invokeScript" will contain the name of the JavaScript that needs to be invoked after the service page is loaded.</xs:documentation>
  </xs:annotation>
  </xs:element>
  <xs:element name="serviceLabel" type="xs:string" minOccurs="0">
  <xs:annotation>
    <xs:documentation>The text presented to the user in case the service is to be presented as a menu option. If missing, the service type or name may be presented. This field is considered to refine the service type.</xs:documentation>
  </xs:annotation>
  </xs:element>
  <xs:element name="inputParameter">
  <xs:annotation>
    <xs:documentation>The name of the parameter in the URL that has to be provided or substituted. This field will get the value of the "trigger" field. The operation attribute specifies the operation to be performed on the input like URL-encoding, creating a CQL query, setting a maximum length, removing dashes etc.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
  <xs:simpleContent>
  <xs:extension base="xs:string">
    <xs:attribute name="operation" type="xs:string" />
  </xs:extension>
  </xs:simpleContent>
  </xs:complexType>
  </xs:element>
  <xs:element name="fixedParameters" minOccurs="0">
  <xs:complexType>
  <xs:sequence>
  <xs:element name="fixedParameter" type="xs:string" maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>A parameter that is always supplied with a fixed value. The contents of this field will have the form "name=value".</xs:documentation>
  </xs:annotation>
  </xs:element>
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  <xs:element name="promptParameters" minOccurs="0">
  <xs:complexType>
  <xs:sequence>
  <xs:element name="promptParameter" maxOccurs="unbounded" type="xs:string">
  <xs:annotation>
    <xs:documentation>A parameter that is supposed to be supplied by the user when invoking the service.</xs:documentation>
  </xs:annotation>
  </xs:element>
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  <xs:element name="varParameters" minOccurs="0">
  <xs:complexType>
  <xs:sequence>
  <xs:element name="varParameter" maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>Additional input parameters to be supplied by the portal taking the form "parameter name=field name"</xs:documentation>
  </xs:annotation>
  </xs:complexType />
  </xs:element>
  </xs:sequence>

```

```

    </xs:complexType>
  </xs:element>
  <xs:element name="sessionParameter" type="xs:string" minOccurs="0">
  <xs:annotation>
    <xs:documentation>Parameter that passes the session identifier to the service.</xs:documentation>
  </xs:annotation>
  </xs:element>
  <xs:element name="sessionURL" type="xs:string" minOccurs="0">
  <xs:annotation>
    <xs:documentation>URL to initiate a session (if the service maintains a session) and to get the session identifier</xs:documentation>
  </xs:annotation>
  </xs:element>
  <xs:element name="serviceTypeSpecificParameters" minOccurs="0">
  <xs:annotation>
    <xs:documentation>These parameters have a special meaning and the portal has to be aware of this meaning and are specified in the XML schema. Examples are language, geographical coordinates, image dimensions etc.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
  <xs:sequence>
    <xs:any namespace="http://www.theeuropeanlibrary.org/servicedescription" processContents="lax"
      minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  <xs:element name="callbackParameter" type="xs:string" minOccurs="0">
  <xs:annotation>
    <xs:documentation>Name of the parameter containing the callback function for Json output</xs:documentation>
  </xs:annotation>
  </xs:element>
  <xs:element name="typeOfUse" type="xs:string" minOccurs="0">
  <xs:annotation>
    <xs:documentation>Indication of how the output is to be used. The default type of use is just link to a service. Other types of use are 'replace' meaning that the original metadata field is replaced by the output of the service, or 'alertOccurrences' meaning to indicate the alerting of the occurrence of specific data. This should be combined with the fieldSpec field to indicate which part of the output is relevant. See below for a more detailed discussion on the type of use.</xs:documentation>
  </xs:annotation>
  </xs:element>
  <xs:element name="CQLQuery" type="xs:string" minOccurs="0">
  <xs:annotation>
    <xs:documentation>Query with CQL syntax used in combination with the operation="CQLQuery" attribute for the inputParameter. The position where the input has to be substituted is marked by "@" e.g. "@creator@. Special values are "@_input@", "@_north@" etc.</xs:documentation>
  </xs:annotation>
  </xs:element>
  <xs:element name="fieldSpec" minOccurs="0">
  <xs:annotation>
    <xs:documentation>Field specifying the data field that is to be used by the portal.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="unitName" type="xs:string" />
    </xs:extension>
  </xs:simpleContent>
  </xs:complexType>
  </xs:element>
  <xs:element name="invokeScript" type="xs:string" minOccurs="0">
  <xs:annotation>
    <xs:documentation>This field is used to indicate the name of the Javascript function that triggers the actual service. This is the case for forms that do not make use of a submit button.</xs:documentation>
  </xs:annotation>
  </xs:element>

```

```

</xs:element>
<xs:element name="authorization" type="xs:string" minOccurs="0">
<xs:annotation>
<xs:documentation>Address of authentication service if authentication is needed prior to deeplinking. When the use of a local proxy service is required this field contains the literal value "useProxy". Local means located at the same server as the TEL portal.</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="invocation" default="option" minOccurs="0">
<xs:annotation>
<xs:documentation>Field indicating whether the service has to be invoked automatically or on user request. Possible values are "option" (a link is to be presented to the user), "automatic" (in this case it is assumed that the results can be integrated in the presentation of the web application). Default is "option".</xs:documentation>
</xs:annotation>
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="option" />
<xs:enumeration value="automatic" />
<xs:enumeration value="no" />
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="nextService" type="xs:string" minOccurs="0">
<xs:annotation>
<xs:documentation>URL or type of the service that will take the output of this service as input. This URL can be a service that is described in one of the service descriptions. If not, it is assumed that the output of the first service can be appended to the URL of the nextService. This field may contain a service type rather than a service URL. It is up to the portal to select one.</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="preProcess" type="xs:anyURI" minOccurs="0">
<xs:annotation>
<xs:documentation>URL of a gateway or proxy that takes the URL of current service as input for example a service that makes output displayable on a mobile device.</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="behavior" type="xs:string" minOccurs="0">
<xs:annotation>
<xs:documentation>Field indicating specific behavior of the service, for example, that it always restarts in the top window. Current vocabulary only contains "newWindow".</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="explain" type="xs:anyURI" minOccurs="0">
<xs:annotation>
<xs:documentation>Contains the URL to get an additional description of the service, like explain in SRU, identify in OAI. For SOAP this field may contain the location of the WSDL file.</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="stylesheet" type="xs:anyURI" minOccurs="0">
<xs:annotation>
<xs:documentation>Reference to an external stylesheet for services that provide XML that needs to be converted to, for example, HTML.</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="logo" type="xs:string" minOccurs="0">
<xs:annotation>
<xs:documentation>Logo to be presented when the service provider requires this as a condition of use.</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="extensibleElements" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:any namespace="##other" processContents="lax" minOccurs="unbounded" />
</xs:sequence>

```



```
</xs:complexType>  
</xs:element>  
</xs:all>  
</xs:complexType>  
</xs:element>  
</xs:sequence>  
</xs:complexType>  
</xs:element>  
</xs:schema>
```