| | |
|---|---|
| **Grant Agreement No.** | ICT-2009-270082 |
| **Project Acronym** | PATHS |
| **Project full title** | **Personalised Access To cultural Heritage Spaces** |

# D 3.2 First Prototype and Documentation

| | |
|---|---|
| **Author:** | **Stein Runar Bergheim (AVINET);** |
| **Contributors:** | **Mark Hall (USFD)** |
| | **Eneko Agirre (UPV/EHU)** |
| | **Aitor Soroa (UPV/EHU)** |
| | **Antonis Kukurikos (iSieve)** |
| | **Kate Fernie (MDR Partners)** |
| | **Tor Gunnar Øverli (AVINET)** |

| **Project funded under FP7-ICT-2009-6 Challenge 4 – "Digital Libraries and Content"** | |
|---|---|
| **Status** | Final |
| **Distribution level** | Public |
| **Date of delivery** | 23.05.2012 |
| **Type** | Software, documentation |
| **Project website** | http://www.paths-project.eu |
| **Project Coordinator** | Dr. Mark Stevenson<br>University of Sheffield |

**Change Log**

| Version | Date | Amended by | Changes |
|---|---|---|---|
| 0.1 | 01/04/2012 | Stein Runar Bergheim | TOC |
| 0.2 | 30/04/2012 | Stein Runar Bergheim | First draft including data specification |
| 0.3 | 12/05/2012 | Stein Runar Bergheim | Added prototype web service API specification |
| 0.4 | 15/05/2012 | Stein Runar Bergheim | Added data layer documentation |
| 0.5 | 19/05/2012 | Stein Runar Bergheim | Added screenshots for section on prototype user interface and executive summary. |
| 0.6 | 22.05.2012 | Mark Stevenson, Kate Fernie, Mark Hall, Stein Runar Bergheim | Adjustments from pre-submission review by project partners. Major updates to PATHS database and PATHS Web API sections. |
| 1.0 | 23.05.2012 | Stein Runar Bergheim | Minor updates, checking. |
| | | | |
| | | | |

# Executive Summary

PATHS aims to develop a system which makes it both enjoyable and easy for users to explore cultural heritage collections held in digital libraries.

The project is taking a user centred approach to design and development. During the first year of the project user requirements were collected and analysed to inform the functional specification of the first prototype system, and the system architecture was defined. Based on this work, the first prototype of the PATHS system has been developed.

This deliverable presents the prototype system and its accompanying documentation. This prototype is designed to demonstrate the core functionality of the system and the potential of the navigation, information retrieval and content enrichment methodology proposed by the project. The prototype will be evaluated by users and, together with the laboratory trials, will inform the development of a second prototype system.

The deliverable consists of three parts:

1. A web application, the *PATHS Prototype User Interface* (UI)
2. A web service API, the *PATHS Web API*
3. A logical data model, the *PATHS Database*

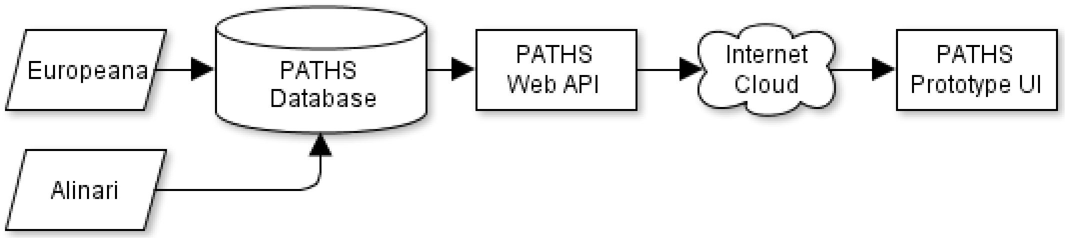


*Figure 1: The principal components of D 3.2*

The web application is based on the user requirements defined in D1.3 “Functional Specification of First Prototype” and D4.1 “Initial Prototype Interface Design”. The application itself is developed on the Python platform and performs its data I/O through web service requests to the underlying web service API.

The web service API and the logical data model correspond to the requirements defined in D3.1 “Specification of System Architecture”. The API is implemented using .NET XML web services and provides Client ports to *HttpGet*, *HttpPost*, *SOAP 1.1* and *SOAP 1.2*.

The content available through the prototype is the result of D2.1 “Processing and Representation of Content for First Prototype”. This content has been parsed into DDL statements and loaded into the PATHS logical data model which is implemented using the leading open source database, PostgreSQL.

This report provides an overview of the different parts of the system; seeks to provide with a platform for conducting system, technical and end-user testing; and to provide technical reference documentation for third parties who are interested in implementing services on the comprehensive PATHS Web API.

# 1 Introduction

The first PATHS prototype, D3.2, is a comprehensive web application infrastructure consisting of a data layer, an application layer and a client/presentation layer. The deliverable is not a stand-alone report, rather it is a combination of a web application, application code and documentation sets suitable for parties who would like to audit – or develop applications based on the PATHS API.

## 1.1 PATHS Prototype Overview

The PATHS Prototype is implemented on a web server platform. The platform runs on the Windows 2008 Server operating system and is configured for HTTP access over TCP/IP version 4 and 6.

The core web server in the platform is Internet Information Server (IIS), integrated with Apache Tomcat Servlet Container (Tomcat) version 7 in order to enable architecture components such as the search server SOLR. A specific server context has been established under IIS where requests will be forwarded to Tomcat.

All data in the system are stored and managed in the relational database management system (RDBMS) PostgreSQL version 9.3. XML based Europeana and Alinari item records (produced by the work described in D2.1) are parsed into SQL statements and loaded into PostgreSQL. All other entities such as users, paths, nodes, comments, tags and ratings are created from within the user interface itself and are “born” directly into the database.

The SOLR search server indexes items, paths and nodes. Items are static and do not need to be re-indexed, however, paths and nodes are dynamic data. Whenever a path or node is added, modified or deleted, a posting is made to the SOLR index to ensure that the search services provided by the SolrProxy Web Service returns synchronized real-time data.

Also present in the data layer of the PATHS infrastructure is a Virtuoso Triple Store. As yet, this server is not invoked by any of the web services, but is present in the infrastructure to provide resolution of sophisticated SPARQL network queries for the second PATHS prototype.

The PATHS API itself is developed as XML Web Services in ASP.NET and publishes four different bindings for Client requests:

```
    SOAP 1.1
    SOAP 1.2
    HTTP GET
    HTTP POST
```

While Web Service *requests* may be made using any of these four protocols, the Web Service *response* will always be a string of JavaScript Object Notation (JSON) data. JSON is flexible and lightweight alternative to XML for encoding and transfer of data and is supported by all mainstream web service libraries.

If the service is called via a SOAP request, the string will be encoded in a single XML string element, if the service is called via a HTTP GET or POST request, the response will be plain text.

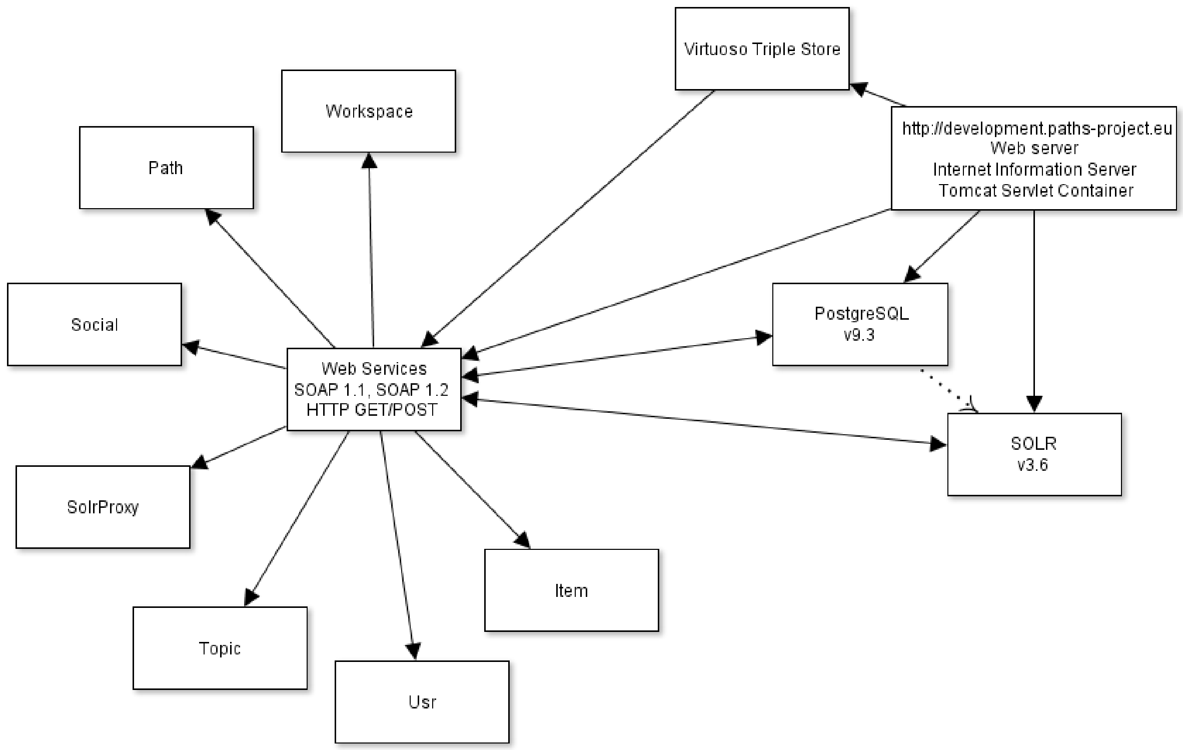


*Figure 2: The diagram shows the key service and server components of the PATHS API*

## 1.2 Relationship to other deliverables

This deliverable corresponds to four previous PATHS deliverables:

- D1.1 “User requirements analysis”
- D1.3 “Functional Specification of First Prototype”
- D3.1 “Specification of System Architecture”
- D4.1 “Initial Prototype Interface Design”

The prototype provides access to data resulting from a further PATHS deliverable:

- D2.1 “Processing and Representation of Content for First Prototype”

# 2 Paths Database

This section is an introduction to Appendix B – Paths Data layer which provides the technical documentation for the logical and physical data model of the entities and relationships in the PATHS data model.

The fundamental element of the PATHS architecture on which the prototype is based is a data layer consisting of robust and well-proven mechanisms for storing, managing and retrieving information.

There are three distinctive types of information present in the data layer:

1. Static information delivered to PATHS from content providers Europeana and Alinari.
2. Static linking information generated through semantic processing and sentiment analysis.
3. Dynamic information generated through the PATHS Prototype user interface such as users, paths, nodes, comments, ratings and tags.

The terms “static” and “dynamic” refers to when the respective information resources are created. *Static* information is loaded into the system “one-time” and may be erased and over-written during subsequent updates. *Dynamic* information is created from within the PATHS user interface.

## 2.1 Data stores

The PATHS database consists of three different data stores: (1) a PostgreSQL relational database management system server instance; (2) a SOLR search server instance and; (3) a Virtuoso triple store server instance.

For the first prototype, only the PostgreSQL and SOLR data stores are used. Virtuoso is included in the platform to cater for extended second prototype functionality as defined in D3.1.

## 2.2 Data model

The PATHS data model is described in detail in Appendix B – Paths Data layer and is an SQL based, relational data model.

Presently, graph data models expressed as RDF in triple store databases are on the rise in popularity. However, they are still outperformed by 10-20 times by traditional relational data models for regular queries and far more for queries returning structured record based information.

All data originating from Europeana and Alinari is strictly record based and the information objects defined in PATHS, paths, nodes, comments etc., all have a static structure. Out of concern for performance (D.3.1), a hybrid approach has been chosen where the infrastructure consists of both a relational database and a graph database. The latter will be used for resolving queries over graph based datasets such as thesauri and topic hierarchies. In the design of the relational model, one concept has been lent from the graph databases, namely the use of URIs as unique identifiers for information elements. In a relational model, traditionally all information is developed as tables. If you have two different tables, “paths” and “nodes”, and you would like to enable users to add comments to these tables, the database should include two additional tables “paths_comments” and “nodes_comments” each with a foreign key referencing the table to which the comments belong.

In a system like PATHS where comments, tags and ratings should be added to three different sets of objects, this would lead to an unnecessary duplication of tables and web service methods. For this reason, every record which is created for a principal PATHS information type will have its own unique URI generated. A path in the PATH table with id = "1" will get the URI "http://paths-project.eu/path/1", a user in the USR table with id= "32" will get the URI "http://paths-project.eu/usr/1".

PATHS implements tags, comments and ratings based on URIs rather than numeric foreign keys, this provides a significantly better overview in the data model – and also allows the comment, rating and tagging mechanisms to be employed on a mix of PATHS information types and external resources identifiable by a URI.

## 2.3 Data

The data loaded into the database are parsed from the XML-files resulting from D2.1. This includes: metadata records from Alinari and Europeana; background links from items to web resources, similarity links between items and; links between items and thesauri/concept hierarchies.

The work of processing and enriching Europeana and Alinari data is comprehensive, technically sophisticated and resource intensive in terms of CPU and processing time. All data are therefore pre-processed and not enriched "on-the-fly". The processes are described in detail in D2.1.

# 3 Paths Web API

This section provides an introduction to Appendix C – Paths Web API which is a programmers' reference aimed at developers implementing services based on the PATHS Web API.

Why should the PATHS system include a Web Service API? The PATHS prototype could have been implemented as a single application, hard-linked to the underlying data sources using direct database request, removing the added overhead of Web Service requests.

Such an approach would however result in a closed, "black-box" system which could not be audited and, more importantly, not be re-used by additional PATHS applications such as the second prototype and the mobile applications proposed by the PATHS project – or for that matter third party applications developed externally.

As a consequence, PATHS has chosen to implement a comprehensive Web Service API publishing all relevant data I/O methods as Web Services which can be consumed by any HTTP client capable of issuing HttpGet or HttpPost requests including all popular JavaScript frameworks and HTTP libraries from all major development languages such as .NET, Java, PHP, Python, Ruby etc.

The PATHS Web API is available for testing at the following address.

```
URI       : http://development.paths-project.eu
```

In order to test the web services, this section includes information on each web service and how it can be invoked and tested using either your web browser to point to the URL of the service end-point – or from application code. This information is relevant both for auditing the framework and for developing applications based on it.

## 3.1 List of Web Services

For logical clarity, all methods are not made available under one single Web Service end-point. Methods are divided into the following classes which each provide access to a set of functions dealing with logically distinctive parts of the PATHS system:

```
Functions related to users and authentication: Usr.asmx
Functions related to creation of paths and nodes: Path.asmx
Functions related to the workspace: Workspace.asmx
Functions related to user interaction and UGC: Social.asmx
Functions related to search and retrieval of items: Item.asmx
Functions related to topic hierarches: Topic.asmx
```

In addition to the native PATHS Web Services above, SOLR is used as a mechanism for information retrieval. While the PostgreSQL contains functions for full-text searching, the inverted index of SOLR outperforms that of RDBMS for complex queries and multilingual support.

The static data in the PATHS database, namely item records from Europeana and Alinari is indexed once, at the time of loading the data. The dynamic data are "posted" as documents to the SOLR web service end-point whenever a path or node is inserted, updated or deleted.

The search server SOLR provides its own set of web services but is not by default secure. The PATHS API provides a wrapper on top of the SOLR select end-point and extends it with the same type of error reporting as for the native PATHS Web Services. Whenever a service request fails, one of the status codes listed under section 3.3 below will be returned. This allows for seamless use along with the rest of the PATHS stack.

```
Functions related to information retrieval: SolrProxy.aspx
```

The technical documentation for the SOLR search server and the underlying Apache Lucene technology is not repeated here. For information on how to invoke this service, please refer to the SOLR web site: http://lucene.apache.org/solr/features.html.

## 3.2 PATHS Web API Usage Examples

This section shows examples of how the PATHS Web API may be invoked to audit its functionality and return data. This section is of a technical instructive nature.

### 3.2.1 EXAMPLE: HTTP Header of Post Request

By default, these web services will return the response JSON wrapped in an XML element named "string". The encoding will be UTF-8. To get pure JSON, the Content-Type parameter is passed as part of the HTTP/POST request:

```
Content-Type: application/json; charset=utf-8
```

Users invoking the methods of the PATHS Web API are likely to use a cross-browser AJAX/HTTP library like jQuery. Such libraries enable developers to specify the format of the return data type as shown above and in example **Error! Reference source not found.**.

### 3.2.2 EXAMPLE: jQuery.ajax request

```
.ajax({
   type: "POST",
   url: "/Usr.asmx/CreateUser",
   data: "{
       'cognitiveStyle':'1',
       'usr':'user',
       'foaf nick':'Nick Name',
       'pwd':'password’,
       'email':’user@domain.tld',
       'openid':'true'}",
   contentType: "application/json; charset=utf-8",
   dataType: ‘json’,
   success: done,
   error: cstatus
);
```

The JSON result of any web service request will be wrapped in an additional top-level object "d". Take this into account when parsing the response. This is a security feature of the .NET Framework.

On the next level of the object, the value "code" states whether the request was successful and the object data is an array of values.

### 3.2.3 EXAMPLE: Response JSON from Web Service Request

```
{
   "d":{
          "code":"2",
          "data":
[{"id":"1","fk usr id":"1","fk rel uri":"http://www.bergheim.dk","comment":"Thi
s is a third comment","isdeleted":"0","tstamp":"04/04/2012 23:56:21"}]
   }
}
```

To return the value of "fk_rel_uri" in JavaScript, you would type

```
var uri = d.data[0].fk_rel_uri;
```

When a JSON result yields more than one return item, i.e. a result set from a query, items are accessible through a zero-based Array.

### 3.2.4 EXAMPLE: Response JSON from Web Service Request yielding more than one item

```
{
   "d":{
          "code": "2",
          "data":[
{"id":"3","fk usr id":"1","fk rel uri":"http://www.bergheim.dk","comment":"A
comment","isdeleted":"0","tstamp":"04/04/2012 23:56:21"},

{"id":"2","fk usr id":"1","fk rel uri":"http://www.bergheim.dk","comment":"Anot
her comment","isdeleted":"0","tstamp":"04/04/2012 23:56:21"},

{"id":"1","fk usr id":"1","fk rel uri":"http://www.bergheim.dk","comment":"A
third comment","isdeleted":"0","tstamp":"04/04/2012 23:56:21"}
                ]
          }
}
```

An example of how to iterate through the array of comments contained in the JSON object is found below:

```
for (var i = 0; i < jsonData.d.data.length; i++) {
   var title = d.data[i].comment;
}
```

## 3.3 Service status codes

The following return codes are used for PATHS web services and can be used to validate the results.

```
NoSuchUser = -1
AuthenticationFailed = 1
OperationCompletedSuccessfully = 2
OperationFailed = 3
AuthenticationSucceeded = 4
OperationRequiresAuthentication = 5
LogoutSuccess = 6
DatabaseSQLError = 7
QueryDidNotReturnRecords = 8
FailedToCreateTemporaryUser = 9
SpecifiedObjectDoesNotExist = 10
NotImplementedYet = 99
```

Most of the service codes are self-explanatory. The latter one, 99, is only used during development of new functionality. All functions documented in the API, see 0Appendix C – Paths Web API, are fully implemented and operational.

## 3.4 Authentication

Most of the services require the user to be authenticated. Authentication is maintained between requests through a session cookie which is sent along with the HTTP-request from the Client application.

A call to the web service "Authenticate" with the credentials as parameters will set session variables letting other web services know that the user is authenticated - as well as store the usr_id for use in user profile related functions.

```
URI: http://development.paths-project.eu/Usr.asmx/Authenticate
```

Unless a cookie container is sent along with the web request, there is no mechanism to exchange session variables between requests to the Web Services; therefore, developers implementing applications on top of the API must take care to fit their HTTP requests with a cookie container.

# 4 Paths Prototype User Interface

The first PATHS prototype user interface is a Python client application which implements the functionality defined in D1.3 “Functional Specification of First Prototype” using the designs developed in D4.1 “Initial Prototype Interface Design”.

The application resolves all its data I/O operations through the PATHS API, a set of web services described in greater detail in the chapter “Paths Web API” above.

The application is available for testing from the following addresses:

```
URL                 : http://prototype.paths-project.eu/
```

## 4.1 Overview of the user interface

The prototype user interface consists of three main sections: paths; explore and search. The first section, paths, allows users to search for paths and view PATH objects. The second section allows for browsing and exploration of the information in the PATHS data model. The third section permits searching and information retrieval.

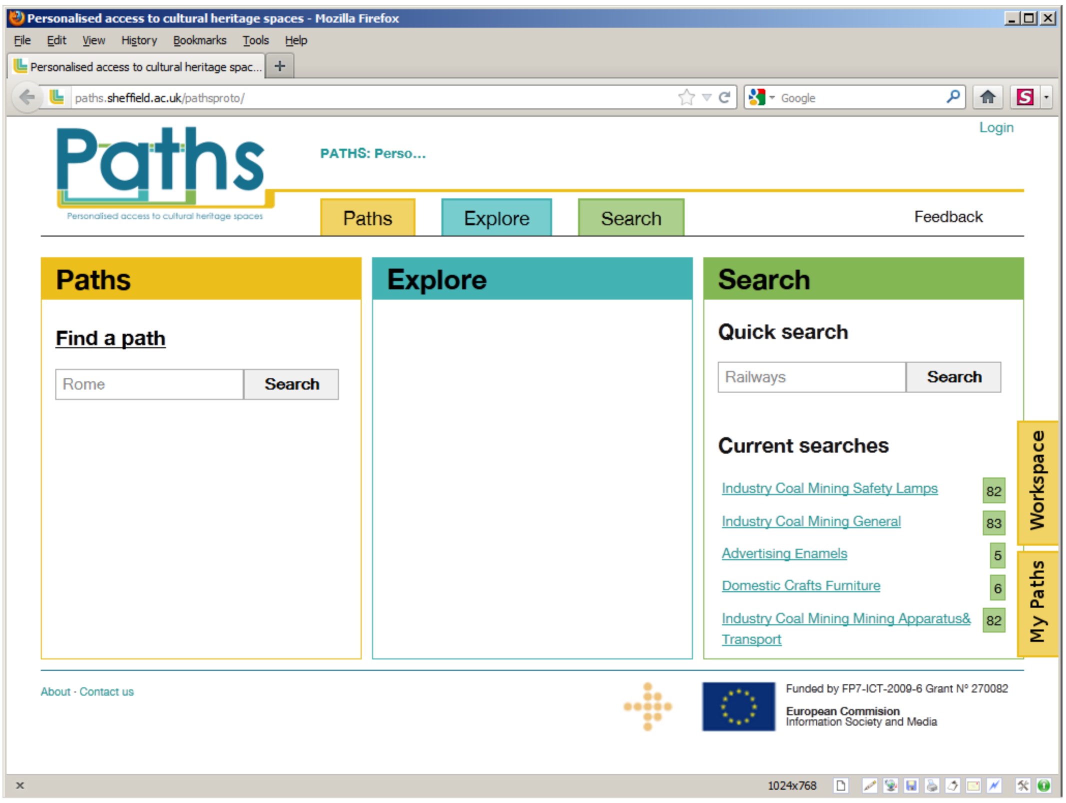

Personalised access to cultural heritage spaces - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Personalised access to cultural heritage spac...
paths.sheffield.ac.uk/pathsproto/
Google
Login
Paths
Personalised access to cultural heritage spaces
PATHS: Perso...
Paths
Explore
Search
Feedback
Paths
Find a path
Rome
Search
Explore
Search
Quick search
Railways
Search
Current searches
Industry Coal Mining Safety Lamps 82
Industry Coal Mining General 83
Advertising Enamels 5
Domestic Crafts Furniture 6
Industry Coal Mining Mining Apparatus& Transport 82
Workspace
My Paths
About · Contact us
Funded by FP7-ICT-2009-6 Grant N° 270082
European Commision
Information Society and Media
1024x768

*Figure 3: Main user interface framework for first PATHS prototype application*

Each of the three sections has a dedicated pane in the main user interface. These remain available throughout all subsequent screens, allowing for simple and recognizable navigation options. The interface allows for plenty of whitespace and uses text-links for high-level navigation in support of the WAI and WCAG guidelines.

In addition to the three main viewer functions, the right hand side of the screen has two vertical tabs, “My paths” and “Workspace”. These tabs give access to functions targeted at PATHS authors.

Below, examples screens from the prototype user interface are shown for each of the three main functions. The natural order of these three sections from the perspective of a viewer would be as displayed in the user interface. For the purpose of this introduction however, the order is shuffled to reflect the perspective of a user who would like to author PATHS:

1. Starting off by **searching** (1) and exploring for items which should form part of the PATH and adding these to the workspace
2. Editing titles and descriptions for workspace items
3. Creating **paths** (2) including title and description
4. Adding workspace items as nodes to the path
5. Publishing the path
6. **Exploring** (3) available PATHS through the user interface

## 4.2 Search functions

The first functions a user looking to build a path must use are search or exploration. In the below examples, search is assumed. From the start-up page, a simple “quick search” function is available. By clicking the tab “search” the main search screen appears.

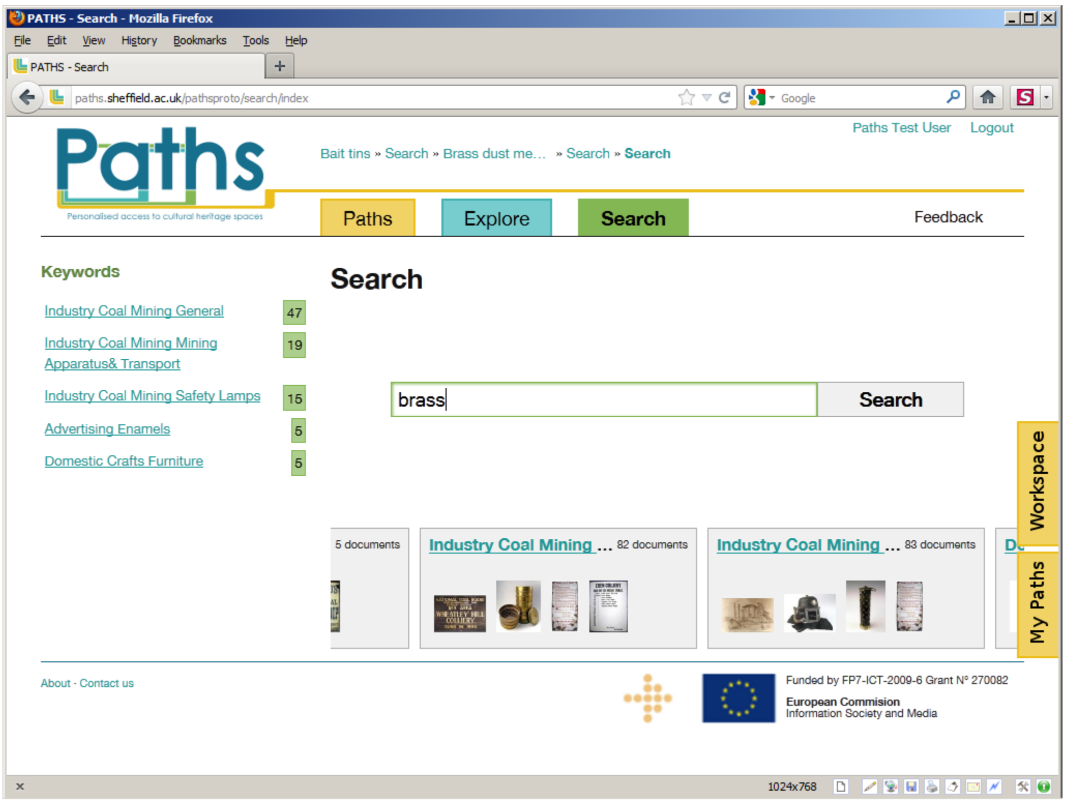
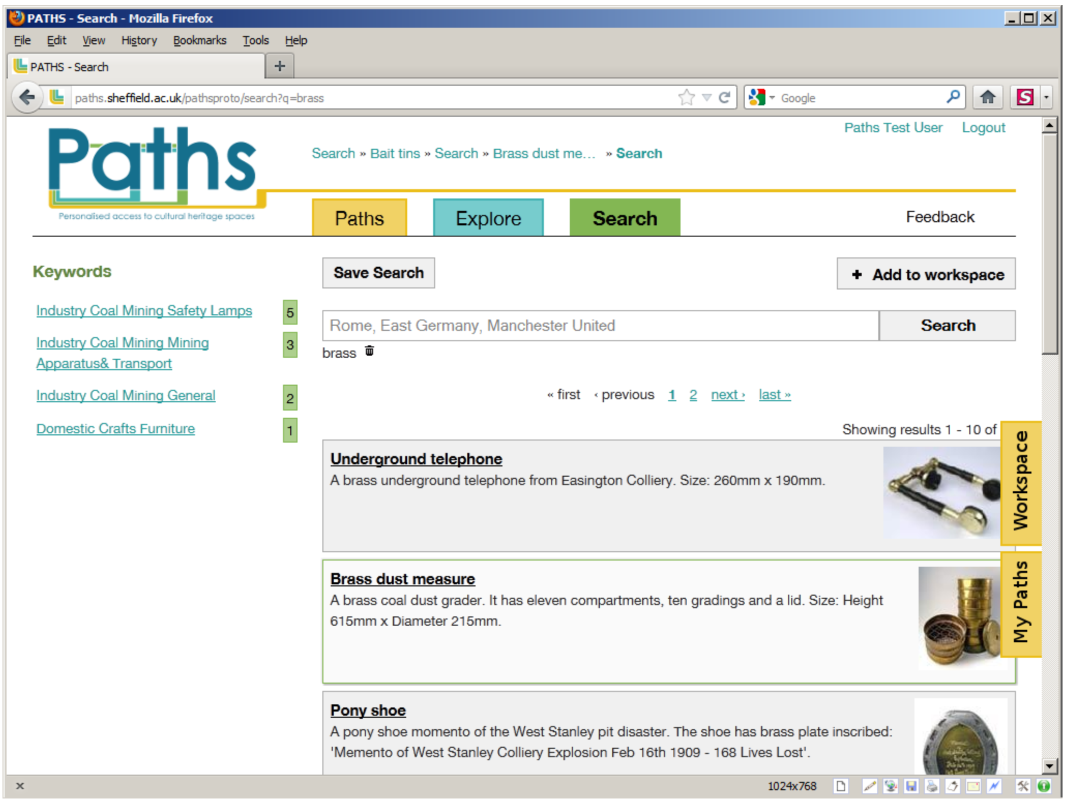

PATHS - Search - Mozilla Firefox
File Edit View History Bookmarks Tools Help
PATHS - Search
paths.sheffield.ac.uk/pathsproto/search/index
Paths Test User Logout
Paths
Personalised access to cultural heritage spaces
Bait tins » Search » Brass dust me... » Search » Search
Paths Explore Search Feedback
Keywords
Industry Coal Mining General 47
Industry Coal Mining Mining Apparatus& Transport 19
Industry Coal Mining Safety Lamps 15
Advertising Enamels 5
Domestic Crafts Furniture 5
Search
brass
Search
5 documents
Industry Coal Mining ... 82 documents
Industry Coal Mining ... 83 documents
Workspace
My Paths
About - Contact us
Funded by FP7-ICT-2009-6 Grant N° 270082
European Commision
Information Society and Media
1024x768

*Figure 4: User interface for the comprehensive search function*

The search screen includes a list of keywords which the user can select from, a single free-text search field as well as a scrolling field of sample content thumbnails representative of the contents of each of the keywords. The keywords are derived from the data processed in D2.1.


PATHS - Search - Mozilla Firefox
File Edit View History Bookmarks Tools Help
PATHS - Search
paths.sheffield.ac.uk/pathsproto/search?q=brass
Paths Test User Logout
Search » Bait tins » Search » Brass dust me… » Search
Paths Explore Search Feedback
Keywords
Save Search
+ Add to workspace
Industry Coal Mining Safety Lamps 5
Industry Coal Mining Mining Apparatus& Transport 3
Industry Coal Mining General 2
Domestic Crafts Furniture 1
Rome, East Germany, Manchester United
Search
brass
« first ‹ previous 1 2 next › last »
Showing results 1 - 10 of
Workspace
My Paths

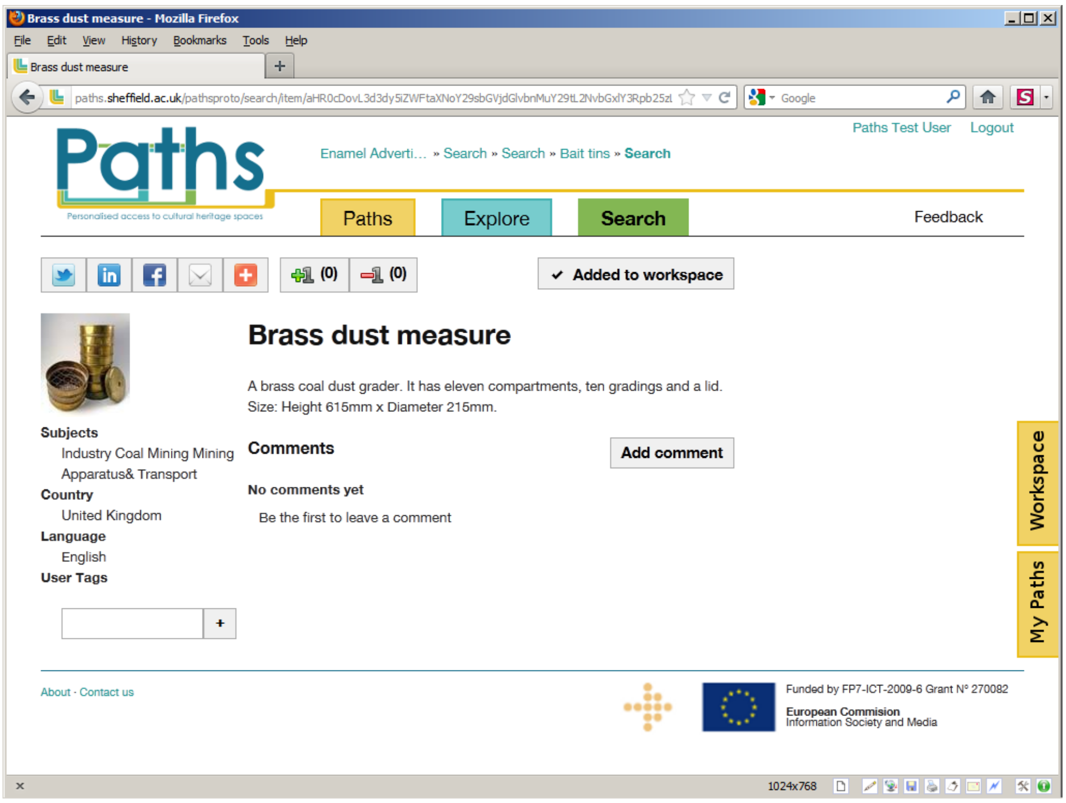
Underground telephone
A brass underground telephone from Easington Colliery. Size: 260mm x 190mm.

Brass dust measure
A brass coal dust grader. It has eleven compartments, ten gradings and a lid. Size: Height 615mm x Diameter 215mm.

Pony shoe
A pony shoe momento of the West Stanley pit disaster. The shoe has brass plate inscribed: 'Memento of West Stanley Colliery Explosion Feb 16th 1909 - 168 Lives Lost'.

*Figure 5: User interface for presentation of search results*

The search functions provide for an efficient way of retrieving information from the Alinari and Europeana collections. Results are presented with a title, a short by-line and a thumbnail (if present in the data).


Brass dust measure - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Brass dust measure
paths.sheffield.ac.uk/pathsproto/search/item/aHR0cDovL3d3dy5iZWFtaXNoY29sbGVjdGlvbnMuY29tL2NvbGxlY3Rpb25zL
Paths Test User Logout
Enamel Adverti… » Search » Search » Bait tins » Search
Paths Explore Search Feedback
(0) (0)
✓ Added to workspace
Add comment
Workspace
My Paths
About · Contact us

Brass dust measure

A brass coal dust grader. It has eleven compartments, ten gradings and a lid.
Size: Height 615mm x Diameter 215mm.

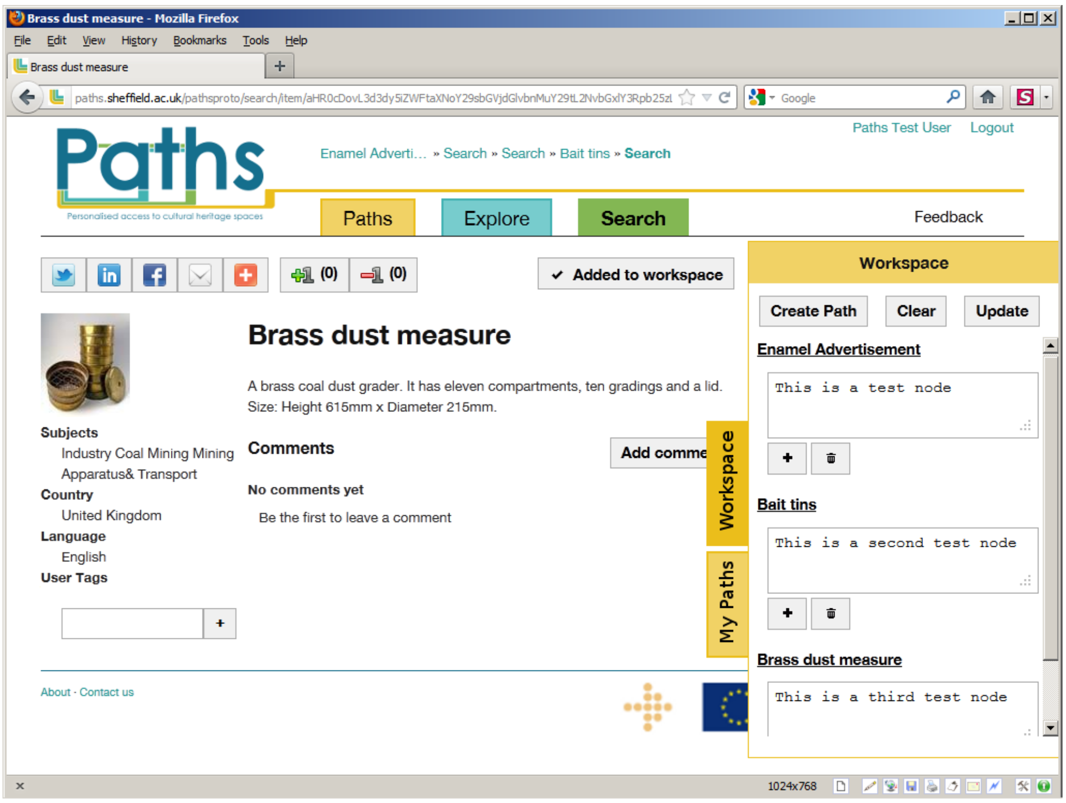
Subjects
Industry Coal Mining Mining Apparatus& Transport
Country
United Kingdom
Language
English
User Tags

Comments
No comments yet
Be the first to leave a comment

Funded by FP7-ICT-2009-6 Grant N° 270082
European Commision
Information Society and Media

*Figure 6: User interface for presentation of a single item from the Europeana and Alinari collections*

When clicking an item from the result list (or when navigating via e.g. the tag cloud) that item is shown on the screen. Interaction with social networking sites is enabled through link buttons. It is also possible to rate the content by pressing the "+1" (like) and "-1" (dislike) buttons


Brass dust measure - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Brass dust measure
paths.sheffield.ac.uk/pathsproto/search/item/aHR0cDovL3d3dy5iZWFtaXNoY29sbGVjdGlvbnMuY29tL2NvbGxlY3Rpb25zL
Google
Paths Test User Logout
Paths
Personalised access to cultural heritage spaces
Enamel Adverti... » Search » Search » Bait tins » Search
Paths
Explore
Search
Feedback
(0)
(0)
✔ Added to workspace
Workspace
Create Path
Clear
Update
Add comme
Workspace
My Paths
About - Contact us
1024x768

Brass dust measure

A brass coal dust grader. It has eleven compartments, ten gradings and a lid.
Size: Height 615mm x Diameter 215mm.

Subjects
Industry Coal Mining Mining Apparatus& Transport
Country
United Kingdom
Language
English
User Tags

Comments

No comments yet

Be the first to leave a comment

Enamel Advertisement

This is a test node

Bait tins

This is a second test node

Brass dust measure

This is a third test node

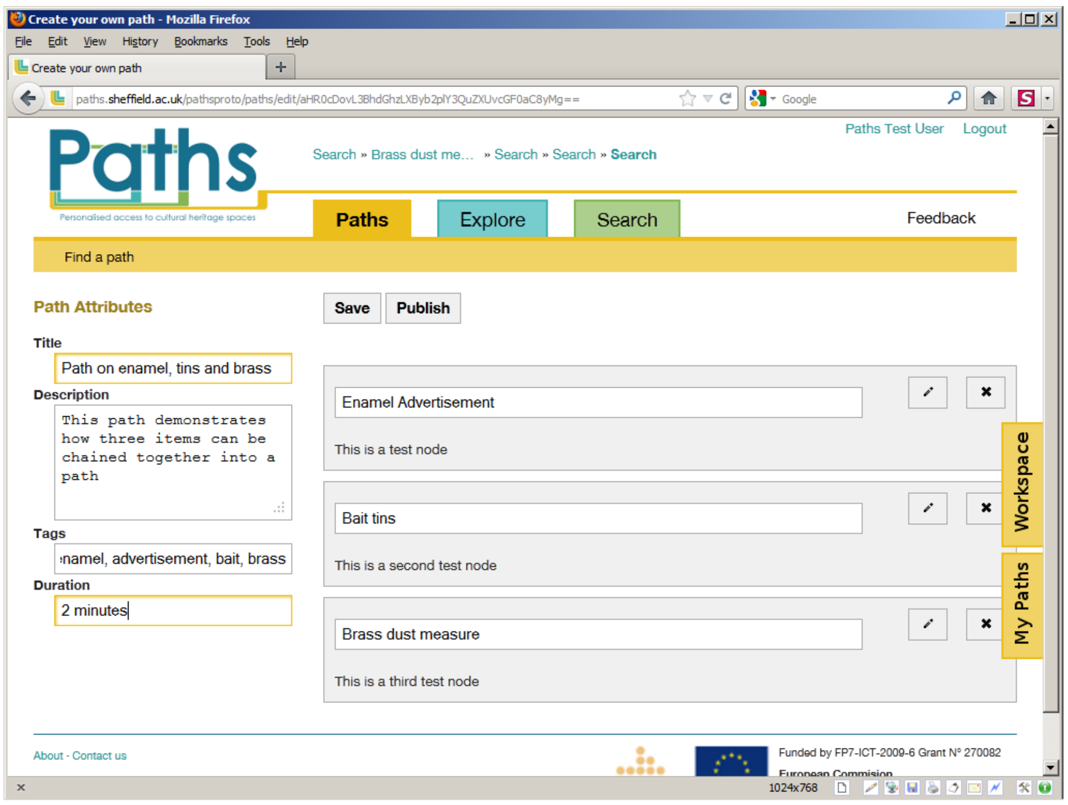
*Figure 7: User interface for adding items to workspace*

While most data creation operations require users to be authenticated, users can start to collect items which they would like to add to their PATHS even before they are logged in.

Items can be added to a temporary workspace in the current user session. This is done through clicking the "+ Add to workspace" button which is shown on all item presentation screens.

## 4.3 Paths functions

Having added a number of items to the workspace, the user can now create a PATH – for this purpose the user must be authenticated. For a screenshot and brief description of the authentication interface, please see section 4.5 below.


Create your own path - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Create your own path
paths.sheffield.ac.uk/pathsproto/paths/edit/aHR0cDovL3BhdGhzLXByb2plY3QuZXUvcGF0aC8yMg==
Google
Paths Test User Logout
Paths
Personalised access to cultural heritage spaces
Search » Brass dust me… » Search » Search » Search
Paths
Explore
Search
Feedback
Find a path
Save
Publish
Workspace
My Paths
About - Contact us
Funded by FP7-ICT-2009-6 Grant N° 270082
European Commision
1024x768

Path Attributes

Title

Path on enamel, tins and brass

Description

This path demonstrates how three items can be chained together into a path

Tags

›namel, advertisement, bait, brass

Duration

2 minutes

Enamel Advertisement

This is a test node

Bait tins

This is a second test node

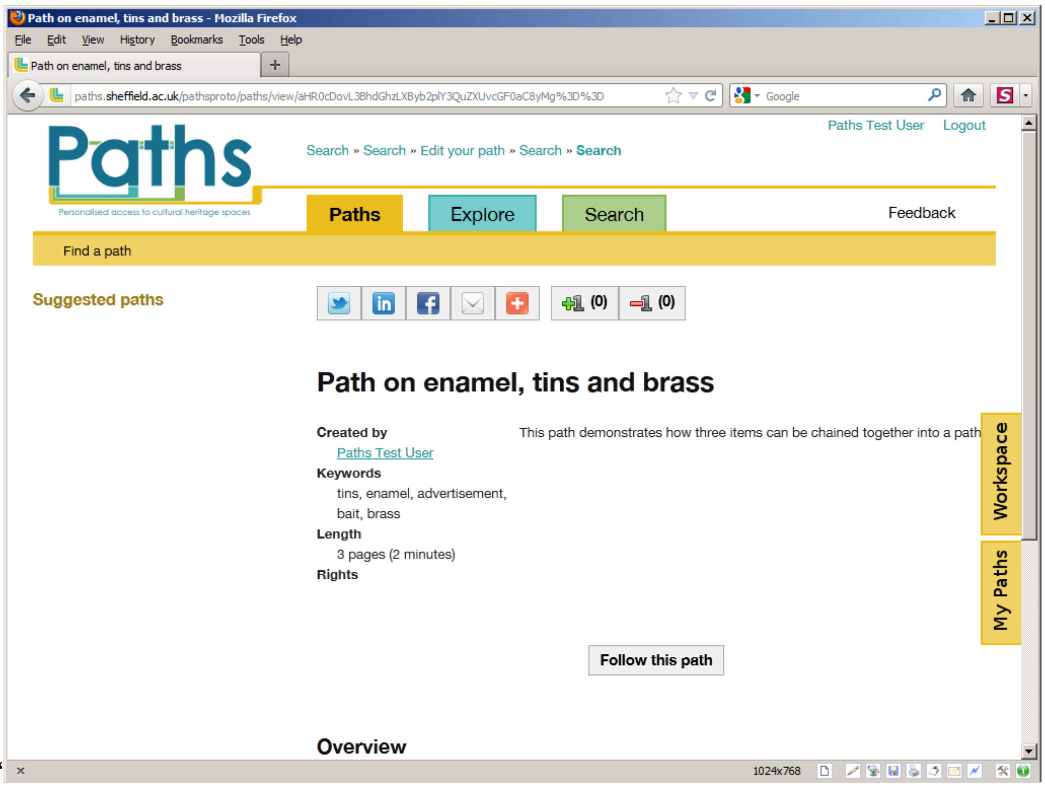
Brass dust measure

This is a third test node

*Figure 8: User interface to create a PATH*

The path creation screen includes a form on the left where information about the path such as title, description, tags and duration can be entered. On the right/center part of the screen, the nodes of the path are listed. Buttons to save drafts or publish the path for public viewing are available above the nodes.

The title of each node is by default that of the item it is based on but may be edited using the user interface. Using the "edit" pencil button which is shown to the far right of the title of each node, other node metadata can be edited.


Path on enamel, tins and brass - Mozilla Firefox
File Edit View History Bookmarks Tools Help
paths.sheffield.ac.uk/pathsproto/paths/view/aHR0cDovL3BhdGhzLXByb2plY3QuZXUvcGF0aC8yMg%3D%3D
Paths Test User Logout
Search » Search » Edit your path » Search » Search
Paths Explore Search Feedback
Find a path
Suggested paths
(0) (0)
Workspace
My Paths
Follow this path

Path on enamel, tins and brass

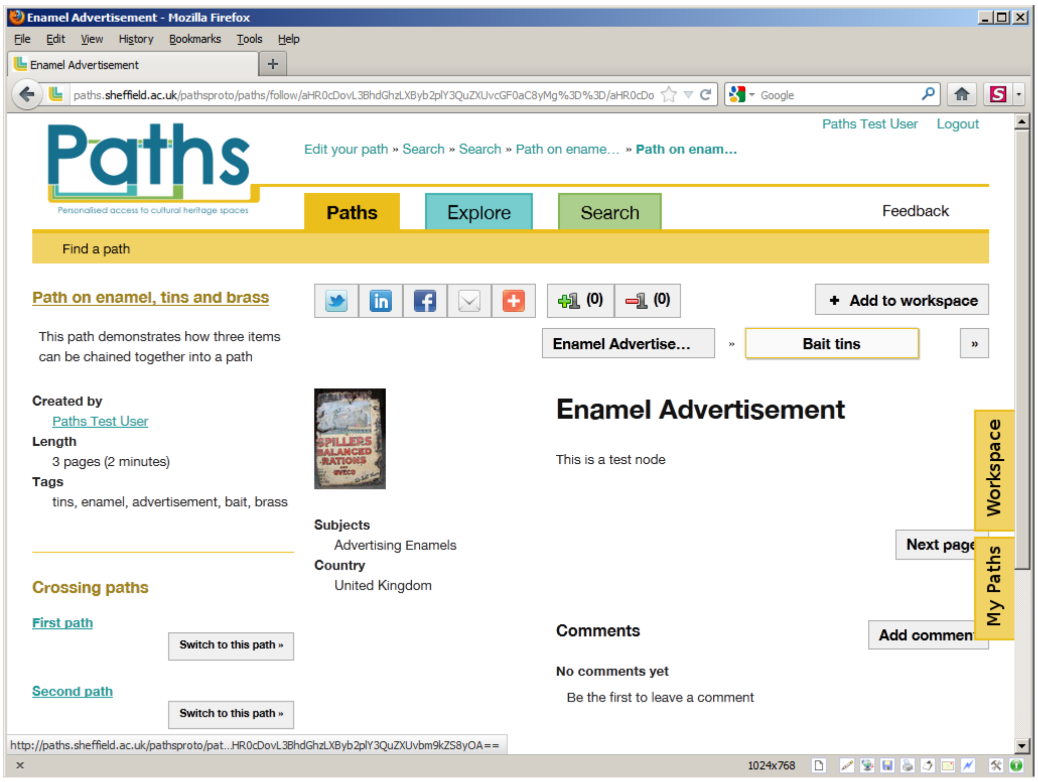
Created by
Paths Test User
Keywords
tins, enamel, advertisement, bait, brass
Length
3 pages (2 minutes)
Rights

This path demonstrates how three items can be chained together into a path

Overview

*Figure 9: User interface for viewing a single PATH*

When a path has been created it is available as a separate information element through the user interface and can be viewed just like an item. The path viewing screen includes basic metadata on the path including: suggested paths; possibility to interact with social networks; add ratings; comments or tags and; choose whether or not to follow the path.


Enamel Advertisement - Mozilla Firefox
File Edit View History Bookmarks Tools Help
paths.sheffield.ac.uk/pathsproto/paths/follow/aHR0cDovL3BhdGhzLXByb2plY3QuZXUvcGF0aC8yMg%3D%3D/aHR0cDo
Paths Test User Logout
Edit your path » Search » Search » Path on ename... » Path on enam...
Paths Explore Search Feedback
Find a path
(0) (0)
Add to workspace
Enamel Advertise... » Bait tins »
Next page
Add comment
Workspace
My Paths
Switch to this path »
Switch to this path »

Path on enamel, tins and brass

This path demonstrates how three items can be chained together into a path

Created by
Paths Test User
Length
3 pages (2 minutes)
Tags
tins, enamel, advertisement, bait, brass

Crossing paths

First path

Second path

Enamel Advertisement

This is a test node

Subjects
Advertising Enamels
Country
United Kingdom

Comments

No comments yet
Be the first to leave a comment

*Figure 10: User interface for viewing a node in a path*

If choosing to follow the path, the screen showing the node appears. The node screen includes information on the path on the left hand side of the screen including a list of crossing paths (paths which include one or more identical items). The right/center side of the screen is occupied by information about the node itself including all user specified metadata such as title, description etc.

The screen allows for user interaction including social network bookmarks, rating and comments.

A set of buttons connected by "breadcrumbs" are shown directly above the node information, providing functions to move to the next or previous node.

## 4.4 Explore functions

Users who are not looking for something particular but are merely after "edutainment" in the form of browsing the Europeana and Alinari collections through the PATHS UI, the exploration interface is the place to start. This allows for flexible navigation of paths, nodes and items through two main functions.

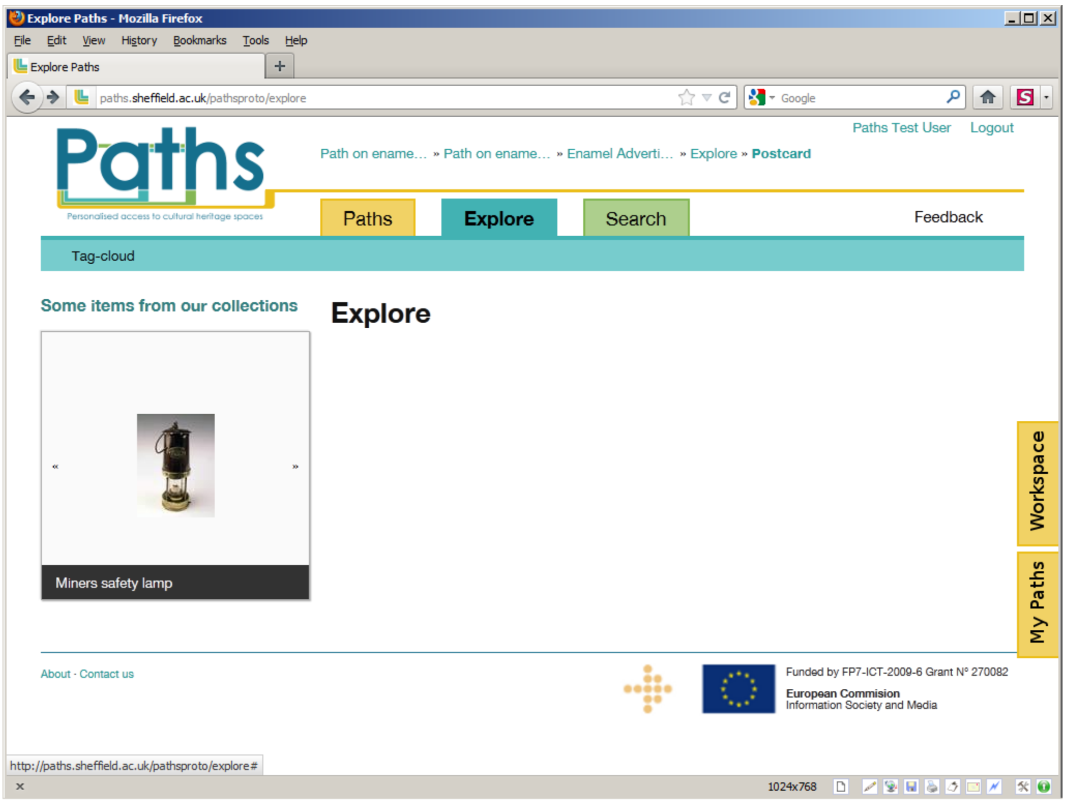
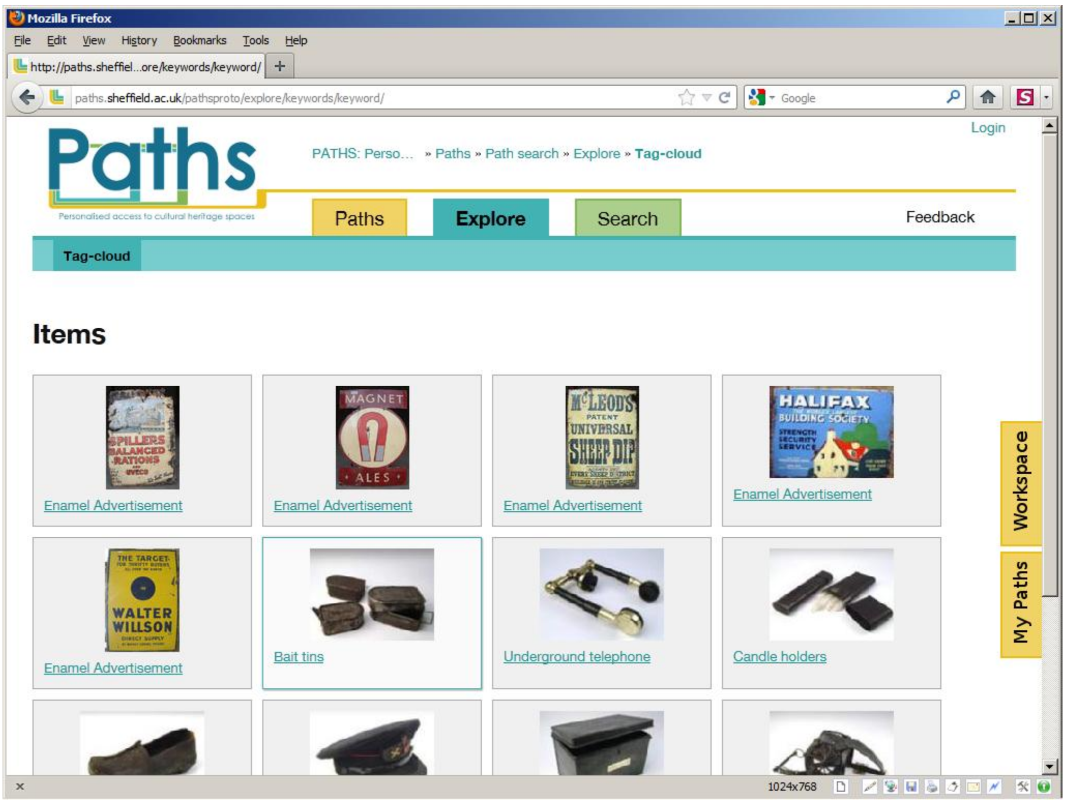

Explore Paths - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Explore Paths
paths.sheffield.ac.uk/pathsproto/explore
Google
Paths
Personalised access to cultural heritage spaces
Paths Test User Logout
Path on ename... » Path on ename... » Enamel Adverti... » Explore » Postcard
Paths
Explore
Search
Feedback
Tag-cloud
Some items from our collections
Explore
Miners safety lamp
Workspace
My Paths
About · Contact us
Funded by FP7-ICT-2009-6 Grant N° 270082
European Commision
Information Society and Media
http://paths.sheffield.ac.uk/pathsproto/explore#
1024x768

*Figure 11: User interface for exploration of selected elements from Europeana and Alinari collections*

The first function, the screen which appears when clicking the "Explore" tab shows a cycling slideshow of items and item-titles from the underlying collections, providing the user with random suggestions of content to explore.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://paths.sheffiel...ore/keywords/keyword/
paths.sheffield.ac.uk/pathsproto/explore/keywords/keyword/
Login
PATHS: Perso... » Paths » Path search » Explore » Tag-cloud
Paths
Explore
Search
Feedback
Tag-cloud
Workspace
My Paths

Items

Enamel Advertisement
Enamel Advertisement
Enamel Advertisement
Enamel Advertisement
Enamel Advertisement
Bait tins
Underground telephone
Candle holders

*Figure 12: User interface for navigating the collections of Alinari and Europeana using a visual tag-cloud*

The second function, the tag-cloud, provides an view of all the items with a thumbnail image and a title. This allows end-users to browse the collections visually. By clicking on either of the thumbnails, the corresponding item will be displayed.

## 4.5 Other functions

The selection of user interface screens above is not exhaustive and many other utility functions are available throughout the user interface. The one associated with authentication is shown below.

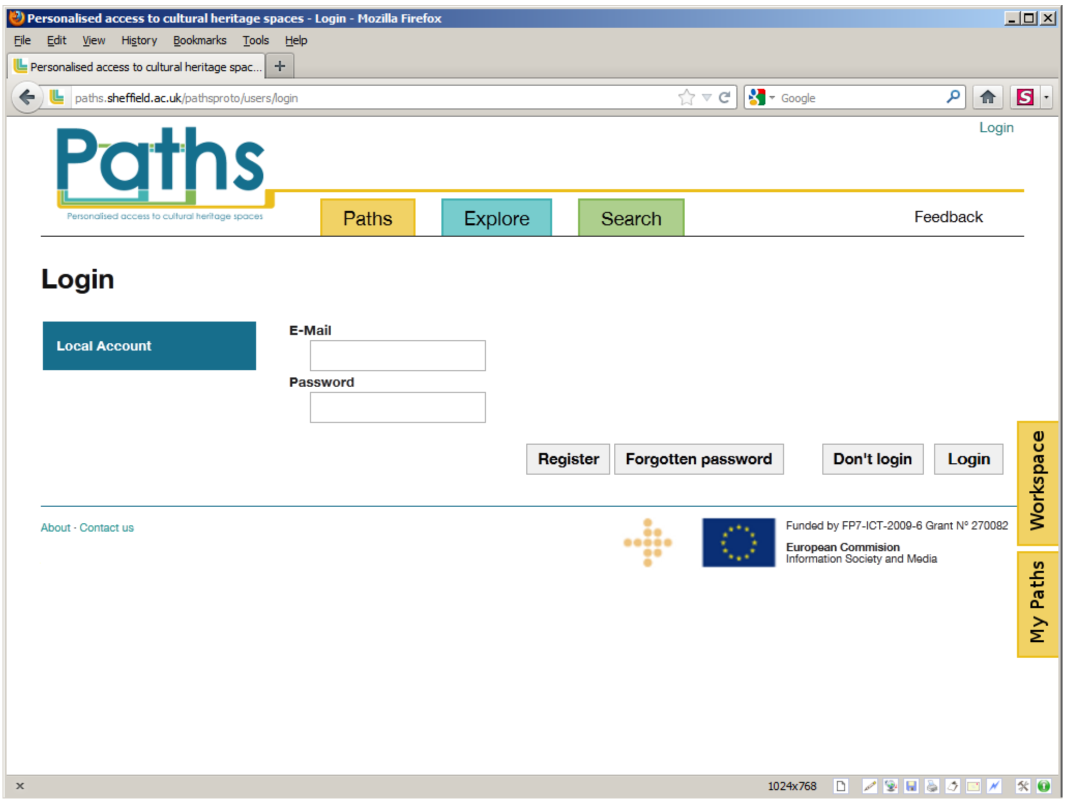

Personalised access to cultural heritage spaces - Login - Mozilla Firefox
File Edit View History Bookmarks Tools Help
paths.sheffield.ac.uk/pathsproto/users/login
Login
Paths
Personalised access to cultural heritage spaces
Paths
Explore
Search
Feedback
Login
Local Account
E-Mail
Password
Register
Forgotten password
Don't login
Login
Workspace
My Paths
About · Contact us
Funded by FP7-ICT-2009-6 Grant N° 270082
European Commision
Information Society and Media

Figure 13: User interface for end-user authentication

The authentication screen provides access to login, register or receive a password reminder by e-mail if the user has forgotten the password chosen at the time of registration.

# 5 Appendices

## Appendix A – Acronym List and Glossary

| Term | Description |
|---|---|
| **API** | Application Programming Interface |
| **HTML** | Hyper-Text Mark-up Language |
| **HTTP** | Hyper-Text Transfer Protocol |
| **IP** | Internet Protocol |
| **JavaScript** | See: ECMA Script |
| **JDBC** | Java DataBase Connectivity |
| **JSON** | JavaScript Object Notation |
| **KML** | Keyhole Mark-up Language |
| **ODBC** | Open DataBase Connectivity |
| **OGC** | Open Geospatial Consortium |
| **OMG** | Object Modelling Group |
| **RDBMS** | Relational Database Management System |
| **REST** | REpresentational State Transfer |
| **SDLC** | System Development Life Cycle |
| **SMB** | Server Message Block. A protocol for file sharing on Windows and Unix based systems |
| **SOA** | Service-Oriented Architecture |
| **SPARQL** | Simple Protocol And RDF Query Language |
| **SQL** | Structured Query Language |
| **TCP** | Transmission Control Protocol |
| **UML** | Unified Modelling Language |
| **WFS** | Web Feature Server. A protocol for on-the-fly generation of map images using http requests. |
| **WMS** | Web Map Server. A protocol for query and download of vector maps using http requests. |
| **WP** | Work Package |
| **WS** | Web Service |
| **WSDL** | Web Service Description Language |
| **XML** | eXtensible Mark-up Language |

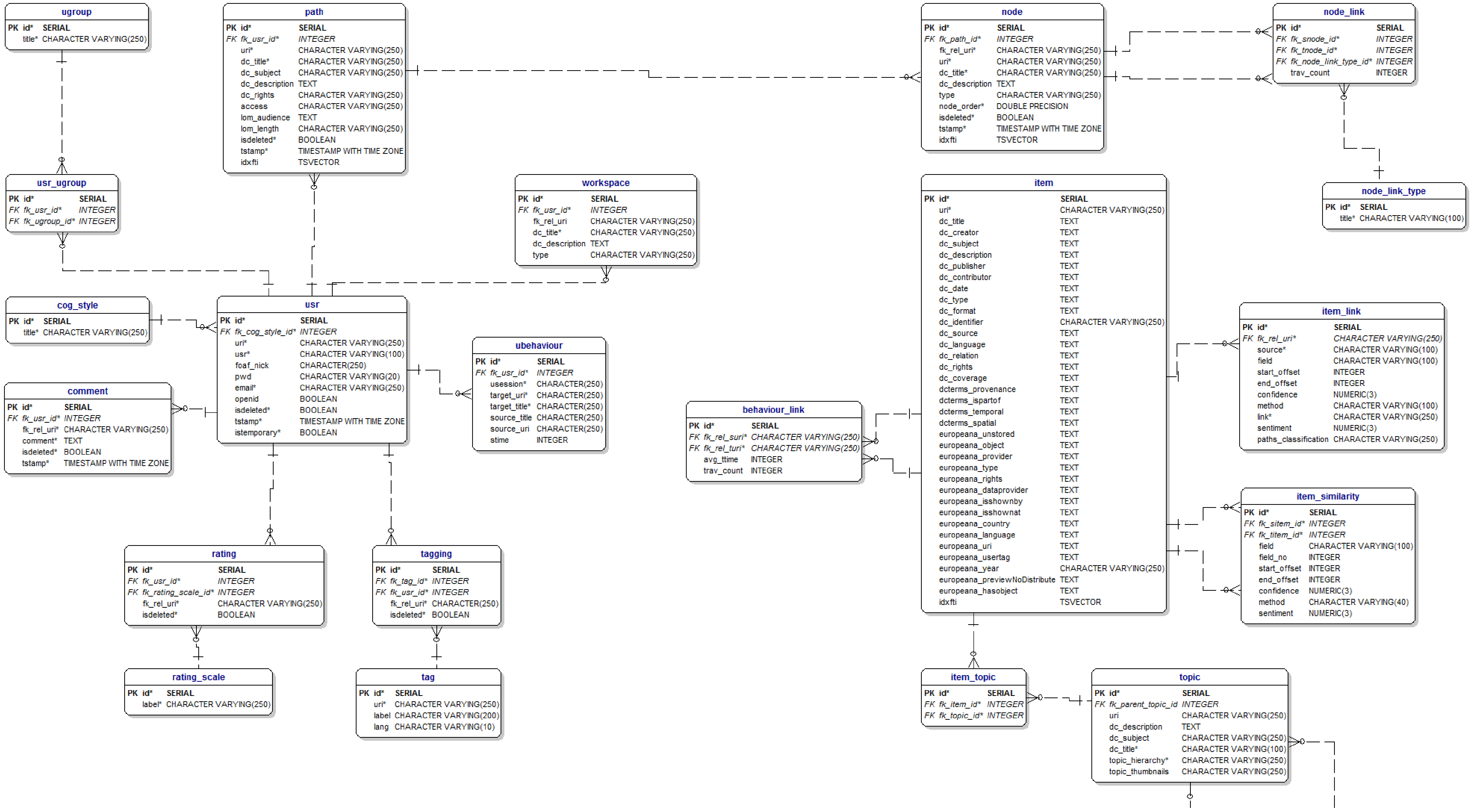
| | |
|---|---|
| **SFS** | Simple Features Specification |
| **CVS** | Concurrent Versioning System |
| **WAI** | Web Accessibility Initiative |
| **WCAG** | Web Content Accessibility Guidelines |
| **JSON** | JavaScript Object Notation |

# Appendix B – Paths Data layer

Logical data model report

| | |
|---|---|
| Author | (Stein) Runar Bergheim, Asplan Viak Internet A/S (Ed.) |
| Copyright | ICT-2009-270082 - PATHS - Personalised Access To cultural Heritage Spaces |
| Target DBMS | PostgreSQL 9.3 |
| Created | 2012-03-11 |
| Modified | 2012-05-15 |

## Entity Relationship diagram



## List of entities

| Entity name | Primary key attributes | # Attributes | Description |
|---|---|---|---|
| behaviour_link | id | 5 | Information on which Items a user has traversed between. |
| cog_style | id | 2 | Codelist of different cognitive styles. A user may have one cognitive style. |
| comment | id | 6 | Comments added to objects identifiable by a URI |
| Item | id | 37 | Information on resources imported from Alinari and Europeana, corresponding to the Europeana Semantic Elements specification. |
| item_link | id | 11 | Links between Items and external background resources (e.g. Wikipedia) as derived from semantic processing. |
| item_similarity | id | 10 | Information on similarity between Items as derived from semantic processing. |
| item_topic | id | 3 | Many to many table between item and topic. One topic may have many items, one item may have many topics. |
| Node | id | 11 | Information about path nodes such as title, description, node_order etc. |
| node_link | id | 5 | Links two nodes together and allows information and attributes to be attached to the relationship between two nodes. |
| node_link_type | id | 2 | Type of relationship between two nodes. |
| Path | id | 13 | Information about paths such as title, subject, description etc. |
| rating | id | 5 | Assigns a rating to any resource identifiable by a URI. Rating is linked to a rating scale and a user. A user is only allowed to rate a URI resource once. |

| | | | |
|---|---|---|---|
| rating_scale | id | 2 | Rating scale for paths and other resources identifiable by a URI. 1 = dislikes, 2 = likes. |
| Tag | id | 4 | Tags: keywords and keyphrases assigned to URI resources. Tags may be language specific and are identifiable by a URI. |
| tagging | id | 5 | Association between tags, users and resources identifiable by a URI. A user can only add the same keyword to a resource once. |
| Topic | id | 8 | Information about topic hierarchies |
| ubehaviour | id | 8 | Information on the way users navigate through information in the PATHS database. |
| ugroup | id | 2 | Codelist of user groups used to distinguish what privieges each user has in the PATHS system. New users by default are members of the 'user' group (id=1). Administrator users are members of the 'admin' group (id=2). New groups may be added to further differentiate privileges. |
| Usr | id | 11 | Information about users such as username, password, nickname etc. |
| usr_ugroup | id | 3 | Many-to-many relationship table between users and user groups. |
| workspace | id | 6 | Temporary storage table for half-baked nodes and items that a user wants to add to PATHS at a later stage after working on them. |

## Entity details

### Entity: behaviour_link

Entity details:

| | |
|---|---|
| Description | Information on which Items a user has traversed between. |
| Primary key constraint name | PK_behaviour_link |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | Unique identifier |
| FK | fk_rel_suri | CHARACTER VARYING | Yes | Source URI resource (the URI of the resource the user came from) |
| FK | fk_rel_turi | CHARACTER VARYING | Yes | Target URI resource (the URI of the resource the user browsed to) |
| | avg_ttime | INTEGER | No | Average time at target URI in seconds |
| | trav_count | INTEGER | No | Number of times the link has been traversed. |

### Entity: cog_style

Entity details:

| | |
|---|---|
| Description | Codelist of different cognitive styles. A user may have one cognitive style. |
| Primary key constraint name | PK_cog_style |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | Unique identifier |
| | title | CHARACTER VARYING | Yes | Name of cognitive style |

### Entity: comment

Entity details:

| | |
|---|---|
| Description | Comments added to objects identifiable by a URI |
| Primary key constraint name | PK_comment |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | Unique identifier |
| FK | fk_usr_id | INTEGER | Yes | Id of user creating comment |
| | fk_rel_uri | CHARACTER VARYING | Yes | URI of resource which comment is assigned to |
| | comment | TEXT | Yes | Comment text |
| | isdeleted | BOOLEAN | Yes | Flag indicating |

| | | | | |
|---|---|---|---|---|
| | | | | whether the entry is deleted (true=deleted) |
| | tstamp | TIMESTAMP WITH TIME ZONE | Yes | Timestamp for the time of creation of the record |

**Entity: item**

Entity details:

| | |
|---|---|
| Description | Information on resources imported from Alinari and Europeana, corresponding to the Europeana Semantic Elements specification. |
| Primary key constraint name | PK_item |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| | uri | CHARACTER VARYING | Yes | Automatically generated uri at the time of creating a new record |
| | dc_title | TEXT | No | |
| | dc_creator | TEXT | No | |
| | dc_subject | TEXT | No | |
| | dc_description | TEXT | No | |
| | dc_publisher | TEXT | No | |
| | dc_contributor | TEXT | No | |
| | dc_date | TEXT | No | |
| | dc_type | TEXT | No | |
| | dc_format | TEXT | No | |
| | dc_identifier | CHARACTER VARYING | No | |
| | dc_source | TEXT | No | |
| | dc_language | TEXT | No | |
| | dc_relation | TEXT | No | |
| | dc_rights | TEXT | No | |
| | dc_coverage | TEXT | No | |
| | dcterms_provenance | TEXT | No | |
| | dcterms_ispartof | TEXT | No | |
| | dcterms_temporal | TEXT | No | |
| | dcterms_spatial | TEXT | No | |
| | europeana_unstored | TEXT | No | |
| | europeana_object | TEXT | No | |
| | europeana_provider | TEXT | No | |
| | europeana_type | TEXT | No | |
| | europeana_rights | TEXT | No | |
| | europeana_dataprovider | TEXT | No | |
| | europeana_isshownby | TEXT | No | |
| | europeana_isshownat | TEXT | No | |
| | europeana_country | TEXT | No | |
| | europeana_language | TEXT | No | |
| | europeana_uri | TEXT | No | |
| | europeana_usertag | TEXT | No | |
| | europeana_year | CHARACTER VARYING | No | |

| | | | | |
|---|---|---|---|---|
| | europeana_previewNoDistribute | TEXT | No | |
| | europeana_hasobject | TEXT | No | |
| | idxfti | TSVECTOR | No | An index field including keyword information from main metadata fields to be used by PostgreSQLs internal full-text search functions |

**Entity: item_link**

Entity details:

| | |
|---|---|
| Description | Links between Items and external background resources (e.g. Wikipedia) as derived from semantic processing. |
| Primary key constraint name | PK_item_link |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| FK | fk_rel_uri | CHARACTER VARYING | Yes | |
| | source | CHARACTER VARYING | Yes | |
| | field | CHARACTER VARYING | No | |
| | start_offset | INTEGER | No | |
| | end_offset | INTEGER | No | |
| | confidence | NUMERIC | No | |
| | method | CHARACTER VARYING | No | |
| | link | CHARACTER VARYING | Yes | |
| | sentiment | NUMERIC | No | |
| | paths_classification | CHARACTER VARYING | No | |

**Entity: item_similarity**

Entity details:

| | |
|---|---|
| Description | Information on similarity between Items as derived from semantic processing. |
| Primary key constraint name | PK_item_similarity |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| FK | fk_sitem_id | INTEGER | Yes | |
| FK | fk_titem_id | INTEGER | Yes | |
| | field | CHARACTER VARYING | No | |
| | field_no | INTEGER | No | |
| | start_offset | INTEGER | No | |
| | end_offset | INTEGER | No | |
| | confidence | NUMERIC | No | |
| | method | CHARACTER | No | |

| | | | | |
|---|---|---|---|---|
| | | VARYING | | |
| | sentiment | NUMERIC | No | |

**Entity: item_topic**

Entity details:

| | |
|---|---|
| Description | Many to many table between item and topic. One topic may have many items, one item may have many topics. |
| Primary key constraint name | PK_item_topic |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| FK | fk_item_id | INTEGER | Yes | |
| FK | fk_topic_id | INTEGER | Yes | |

**Entity: node**

Entity details:

| | |
|---|---|
| Description | Information about path nodes such as title, description, node_order etc. |
| Primary key constraint name | PK_node |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| FK | fk_path_id | INTEGER | Yes | |
| | fk_rel_uri | CHARACTER VARYING | Yes | |
| | uri | CHARACTER VARYING | Yes | Automatically generated uri at the time of creating a new record |
| | dc_title | CHARACTER VARYING | Yes | |
| | dc_description | TEXT | No | |
| | type | CHARACTER VARYING | No | |
| | node_order | DOUBLE PRECISION | Yes | |
| | isdeleted | BOOLEAN | Yes | |
| | tstamp | TIMESTAMP WITH TIME ZONE | Yes | |
| | idxfti | TSVECTOR | No | An index field including keyword information from main metadata fields to be used by PostgreSQLs internal full-text search functions |

**Entity: node_link**

Entity details:

| | |
|---|---|
| Description | Links two nodes together and allows information and attributes to be attached to the relationship between two nodes. |

| | |
|---|---|
| Primary key constraint name | PK_node_link |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| FK | fk_snode_id | INTEGER | Yes | |
| FK | fk_tnode_id | INTEGER | Yes | |
| FK | fk_node_link_type_id | INTEGER | Yes | |
| | trav_count | INTEGER | No | |

**Entity: node_link_type**

Entity details:

| | |
|---|---|
| Description | Type of relationship between two nodes. |
| Primary key constraint name | PK_node_link_type |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| | title | CHARACTER VARYING | Yes | |

**Entity: path**

Entity details:

| | |
|---|---|
| Description | Information about paths such as title, subject, description etc. |
| Primary key constraint name | PK_path |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | Primary key and unique identifier |
| FK | fk_usr_id | INTEGER | Yes | Id of user who created path |
| | uri | CHARACTER VARYING | Yes | Automatically generated uri at the time of creating a new record |
| | dc_title | CHARACTER VARYING | Yes | Title of path, taken from Dublin Core namespace |
| | dc_subject | CHARACTER VARYING | No | Subject of path, taken from Dublin Core namespace. Multiple values are separated by semi-colon ";" |
| | dc_description | TEXT | No | Description of path, taken from Dublin Core namespace. |
| | dc_rights | CHARACTER VARYING | No | Rights to use path, taken from Dublin Core namespace. |
| | access | CHARACTER VARYING | No | Any access restrictions associated with path |

| | | | | |
|---|---|---|---|---|
| | lom_audience | TEXT | No | Intended audience, taken from Learning Object Model namespace |
| | lom_length | CHARACTER VARYING | No | Approximate time required/duration of path, taken from Learning Object Model namespace. |
| | isdeleted | BOOLEAN | Yes | A boolean value indicating whether the resource has been marked for deletion or not. |
| | tstamp | TIMESTAMP WITH TIME ZONE | Yes | An automatically created timestamp at the time of creating a new record |
| | idxfti | TSVECTOR | No | An index field including keyword information from main metadata fields to be used by PostgreSQLs internal full-text search functions |

**Entity: rating**

Entity details:

| | |
|---|---|
| Description | Assigns a rating to any resource identifiable by a URI. Rating is linked to a rating scale and a user. A user is only allowed to rate a URI resource once. |
| Primary key constraint name | PK_rating |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| FK | fk_usr_id | INTEGER | Yes | |
| FK | fk_rating_scale_id | INTEGER | Yes | |
| | fk_rel_uri | CHARACTER VARYING | Yes | |
| | isdeleted | BOOLEAN | Yes | |

**Entity: rating_scale**

Entity details:

| | |
|---|---|
| Description | Rating scale for paths and other resources identifiable by a URI. 1 = dislikes, 2 = likes. |
| Primary key constraint name | PK_rating_scale |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| | label | CHARACTER VARYING | Yes | |

**Entity: tag**

Entity details:

| | |
|---|---|
| Description | Tags: keywords and keyphrases assigned to URI resources. Tags may be language specific and are identifiable by a URI. |
| Primary key constraint name | PK_tag |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| | uri | CHARACTER VARYING | Yes | Automatically generated uri at the time of creating a new record |
| | label | CHARACTER VARYING | No | |
| | lang | CHARACTER VARYING | No | |

**Entity: tagging**

Entity details:

| | |
|---|---|
| Description | Association between tags, users and resources identifiable by a URI. A user can only add the same keyword to a resource once. |
| Primary key constraint name | PK_tagging |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| FK | fk_tag_id | INTEGER | Yes | |
| FK | fk_usr_id | INTEGER | Yes | |
| | fk_rel_uri | CHARACTER | Yes | |
| | isdeleted | BOOLEAN | Yes | |

**Entity: topic**

Entity details:

| | |
|---|---|
| Description | Information about topic hierarchies |
| Primary key constraint name | PK_topic |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| FK | fk_parent_topic_id | INTEGER | No | |
| | uri | CHARACTER VARYING | No | Automatically generated uri at the time of creating a new record |
| | dc_description | TEXT | No | |
| | dc_subject | CHARACTER VARYING | No | |
| | dc_title | CHARACTER VARYING | Yes | |
| | topic_hierarchy | CHARACTER VARYING | Yes | |
| | topic_thumbnails | CHARACTER VARYING | No | |

**Entity: ubehaviour**

Entity details:

| | |
|---|---|
| Description | Information on the way users navigate through information in the PATHS database. |
| Primary key constraint name | PK_ubehaviour |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| FK | fk_usr_id | INTEGER | Yes | |
| | usession | CHARACTER | Yes | |
| | target_uri | CHARACTER | Yes | URI of object user is navigating to |
| | target_title | CHARACTER | Yes | |
| | source_title | CHARACTER | No | |
| | source_uri | CHARACTER | No | URI of object user is navigating from |
| | stime | INTEGER | No | Time spent at source in seconds |

**Entity: ugroup**

Entity details:

| | |
|---|---|
| Description | Codelist of user groups used to distinguish what privieges each user has in the PATHS system. New users by default are members of the 'user' group (id=1). Administrator users are members of the 'admin' group (id=2). New groups may be added to further differentiate privileges. |
| Primary key constraint name | PK_user_group |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| | title | CHARACTER VARYING | Yes | |

**Entity: usr**

Entity details:

| | |
|---|---|
| Description | Information about users such as username, password, nickname etc. |
| Primary key constraint name | PK_usr |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| FK | fk_cog_style_id | INTEGER | Yes | |
| | uri | CHARACTER VARYING | Yes | Automatically generated uri at the time of creating a new record |
| | usr | CHARACTER VARYING | Yes | |
| | foaf_nick | CHARACTER | No | |
| | pwd | CHARACTER VARYING | No | |
| | email | CHARACTER VARYING | Yes | |
| | openid | BOOLEAN | No | |
| | isdeleted | BOOLEAN | Yes | |
| | tstamp | TIMESTAMP WITH TIME ZONE | Yes | |
| | istemporary | BOOLEAN | Yes | |

**Entity: usr_ugroup**

Entity details:

| | |
|---|---|
| Description | Many-to-many relationship table between users and user groups. |
| Primary key constraint name | PK_usr_ugroup |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| FK | fk_usr_id | INTEGER | Yes | |
| FK | fk_ugroup_id | INTEGER | Yes | |

**Entity: workspace**

Entity details:

| | |
|---|---|
| Description | Temporary storage table for half-baked nodes and items that a user wants to add to PATHS at a later stage after working on them. |
| Primary key constraint name | PK_workspace |

Attributes:

| Key | Attribute name | Data type | Not null | Description |
|---|---|---|---|---|
| PK | id | SERIAL | Yes | |
| FK | fk_usr_id | INTEGER | Yes | |
| | fk_rel_uri | CHARACTER VARYING | No | |
| | dc_title | CHARACTER VARYING | Yes | |
| | dc_description | TEXT | No | |
| | Type | CHARACTER VARYING | No | |

# Appendix C – Paths Web API

**Web Service: Usr**

**Summary:** The Usr web service contains methods for authenticating users, creating and modifying users, logging user behavior and issuing reminder e-mails upon forgetting passwords. The service is fundamental to web services which require authentication.

## Web Method: Authenticate

**Summary:** Perform authentication of user

**Authenticate Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| Usr | s:string | User name |
| Pwd | s:string | Password |

Authenticate Response

| Data type | Description |
|---|---|
| s:string (JSON) | AuthenticationSucceeded (code=4) on success, AuthenticationFailed (code=1) on wrong username/password, OperationFailed (code=3) on error. |

**Example of Authenticate HttpGet Request**

**Request:**
http://development.paths-project.eu/Usr.asmx/Authenticate?usr=s:string&pwd=s:string

**Response:**
```
{
   "code": 4,
   "msg": "Authentication succeeded, user authenticated",
   "extmsg": "1"
}
```

### Web Method: Logout

**Summary:** Logs the current user out of the system by erasing user information from the session

**Logout Request Parameters**

**N/A** (this web method does not accept any calling parameters)

**Logout Response**

| Data type | Description |
|---|---|
| s:string (JSON) | Always returns LogoutSuccess (code=6) |

**Example of Logout HttpGet Request**

**Request:**
http://development.paths-project.eu/Usr.asmx/Logout?

**Response:**
```
{
   "code": 6,
   "msg": "User logged out",
   "extmsg": "User logged out"
}
```

### Web Method: CreateUser

**Summary:** Create a new user

**CreateUser Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| fk_cog_style_id | s:int | Integer, the primary key id of the cognitive style associated with the user |
| Usr | s:string | Username |
| foaf_nick | s:string | Nickname/display name |
| Pwd | s:string | Password |
| Email | s:string | E-mail address |
| Opened | s:Boolean | Whether or not the user account is an OpenID account (Boolean, true/false) |

**CreateUser Response**

| Data type | Description |
|---|---|

| | |
|---|---|
| s:string (JSON) | Returns OperationCompletedSuccessfully (code=2) and the user data for the created user |

**Example of CreateUser HttpGet Request**

**Request:**
http://development.paths-project.eu/Usr.asmx/CreateUser?fk_cog_style_id=s:int&usr=s:string&foaf_nick=s:string&pwd=s:string&email=s:string&openid=s:boolean

**Response:**
```
{
   "code": 2,
   "data": {
      "id": "80",
      "fk_cog_style_id": "1",
      "uri": "http://paths-project.eu/usr/80",
      "usr": "Paths-Test-User",
      "foaf_nick": "Anonymous",
      "pwd": "test",
      "email": "user@paths-project.eu",
      "openid": "0",
      "isdeleted": "0",
      "tstamp": "08.05.2012 10:11:51 PM",
      "istemporary": "0"
   }
}
```

## Web Method: ModifyUser

**Summary:** Modifies information about a user identified by its URI

**ModifyUser Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| usr_uri | s:string | URI of the user to be modified |
| fk_cog_style_id | s:string | The id of the users cognitive style (optional) |
| Usr | s:string | Username (optional) |
| foaf_nick | s:string | Nickname/display name (optional) |
| Pwd | s:string | Password (optional) |
| Email | s:string | E-mail (optional) |
| Opened | s:string | Whether the user is an OpenID user (Boolean, optional) |

**ModifyUser Response**

| Data type | Description |
|---|---|
| s:string (JSON) | User data object for modified user |

**Example of ModifyUser HttpGet Request**

**Request:**
http://development.paths-project.eu/Usr.asmx/ModifyUser?usr_uri=s:string&fk_cog_style_id=s:string&usr=s:string&foaf_nick=s:string&pwd=s:string&email=s:string&openid=s:string

**Response:**
```
{
   "code": 2,
   "data": [{
      "uri": "http://paths-project.eu/usr/80",
      "fk_cog_style_id": "1",
      "usr": "Renamed-User",
      "foaf_nick": "Anonymous II",
      "email": "changed@email.com",
      "openid": "0",
      "istemporary": "0",
      "tstamp": "08.05.2012 10:11:51 PM"
   }]
}
```

### Web Method: DeleteUser

**Summary:** Deletes a user from PATHS

**Remark:** Method requires authentication

#### DeleteUser Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| usr_uri | s:string | URI of the user to be deleted |

#### DeleteUser Response

| Data type | Description |
|---|---|
| s:string (JSON) | OperationCompletedSuccessfully (code=2) if the user was either marked as deleted or did not exist, DatabaseSQLError (code=7) on error. |

#### Example of DeleteUser HttpGet Request

**Request:**
http://development.paths-project.eu/Usr.asmx/DeleteUser?usr_uri=s:string

**Response:**
```
{
   "code": 2,
   "msg": "Operation completed successfully",
   "extmsg": "User successfully marked for deletion"
}
```

### Web Method: ForgotPassword

**Summary:** Sends an e-mail with the password of the user corresponding

#### ForgotPassword Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| pUsr | s:string | The username of the user to whom the password reminder should be sent |

#### ForgotPassword Response

| Data type | Description |
|---|---|
| s:string (JSON) | Always returns OperationCompletedSuccessfully (code=2). If the username is found, an e-mail with the corresponding password is |

| | |
|---|---|
| | sent to the users e-mail address. |

**Example of ForgotPassword HttpGet Request**

**Request:**
http://development.paths-project.eu/Usr.asmx/ForgotPassword?pUsr=s:string

### Web Method: GetCurrentUser

**Summary:** Gets information about the currently authenticated or temporary user.

**GetCurrentUser Request Parameters**

**N/A** (this web method does not accept any calling parameters)

**GetCurrentUser Response**

| Data type | Description |
|---|---|
| s:string (JSON) | User data object for current user |

**Example of GetCurrentUser HttpGet Request**

**Request:**
http://development.paths-project.eu/Usr.asmx/GetCurrentUser?

### Web Method: GetUserByUri

**Summary:** Returns information about the user identified by the specified URI

**GetUserByUri Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| usr_uri | s:string | URI of user |

**GetUserByUri Response**

| Data type | Description |
|---|---|
| s:string (JSON) | User data object |

**Example of GetUserByUri HttpGet Request**

**Request:**
```
http://development.paths-project.eu/Usr.asmx/GetUserByUri?usr_uri=s:string
```

**Response:**
```
{
   "code": 2,
   "data": {
      "uri": "http://paths-project.eu/usr/80",
      "fk_cog_style_id": "1",
      "usr": "Renamed-User",
      "foaf_nick": "Anonymous II",
      "email": "runarbe@gmail.com",
      "istemporary": "0",
      "tstamp": "08.05.2012 10:11:51 PM",
      "paths_ugroup": []
   }
}
```

### Web Method: LogPageView

**Summary:** Logs a URI to the browsing history of the user and returns the five last pages visited during the session.

#### LogPageView Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| myTargetTitle | s:string | Title of web page to log |
| myTargetUri | s:string | URI of web page to log |

#### LogPageView Response

| Data type | Description |
|---|---|
| s:string (JSON) | List of five most recent logged page view objects for current session |

#### Example of LogPageView HttpGet Request

**Request:**
```
http://development.paths-
project.eu/Usr.asmx/LogPageView?myTargetTitle=s:string&myTargetUri=s:string
```

**Response:**
```
{
   "code": 2,
   "data": [{
      "id": "1",
        "usession": "2qxv0t55eqlut455rt1xls55",
        "target_uri": "http://paths-project.eu/",
        "target_title": "Page Title",
        "source_uri": "http://paths-project.eu/",
        "source_title": "Page Title",
        "stime": ""
    }, {
        "id": "54",
        "usession": "2qxv0t55eqlut455rt1xls55",
        "target_uri": "http://paths-project.eu/",
        "target_title": "Page Title",
        "source_uri": "",
        "source_title": "",
        "stime": ""
    }, {
        "id": "53",
        "usession": "o0ai45yi01rkni45pk5kdjeb",
        "target_uri": "http://paths-project.eu/",
        "target_title": "Page Title",
        "source_uri": "http://paths-project.eu/",
        "source_title": "Page Title",
        "stime": ""
    }, {
        "id": "52",
        "usession": "o0ai45yi01rkni45pk5kdjeb",
        "target_uri": "http://paths-project.eu/",
        "target_title": "Page Title",
        "source_uri": "http://paths-project.eu/",
        "source_title": "Page Title",
        "stime": ""
    }, {
        "id": "51",
        "usession": "o0ai45yi01rkni45pk5kdjeb",
        "target_uri": "http://paths-project.eu/",
        "target_title": "Page Title",
        "source_uri": "http://paths-project.eu/",
        "source_title": "Page Title",
        "stime": ""
    }, {
        "id": "50",
        "usession": "o0ai45yi01rkni45pk5kdjeb",
        "target_uri": "http://paths-project.eu/",
        "target_title": "Page Title",
        "source_uri": "http://paths-project.eu/",
        "source_title": "Page Title",
        "stime": ""
    }]
}
```

## Web Service: Item

**Summary:** The Item web service contains methods for querying and retrieving information about items. PATHS items are information derived from Europeana and Alinari and includes most attributes defined by the Europeana Semantic Elements. Items have been enriched with (1) background links, (2) topic links and (3) item similarity links.

### Web Method: Search

**Summary:** Experimental function to enable full-text search without using SOLR

#### Search Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| myQuery | s:string | Query expression |
| myLang | s:string | One of english, spanish or leave empty |
| myLength | s:string | How many result records to retrieve |
| myOffset | s:string | Where to start retrieving in a result set (paging) |

#### Search Response

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: List of items |

#### Example of Search HttpGet Request

**Request:**
http://development.paths-project.eu/Item.asmx/Search?myQuery=s:string&myLang=s:string&myLength=s:string&myOffset=s:string

**Response:**

```
{
    "code": 2,
    "data": {
        "id": "1",
        "uri": "http://www.beamishcollections.com/collections/display.asp?ItemID=1",
        "dc_title": "Enamel Advertisement",
        "dc_creator": "",
        "dc_subject": ["Advertising Enamels"],
        "dc_description": "Enamel Advertisement \"Spillers Balanced Rations and UVECO\"/
\"For Cattle, Sheep, Pigs& Poultry\"/ \"We Sell Them\" Height: 1280mm x 795mm.",
        "dc_publisher": "",
        "dc_contributor": "",
        "dc_date": "",
        "dc_type": "Image",
        "dc_format": "",
        "dc_identifier": "http://www.beamishcollections.com/collections/display.asp?ItemID=1",
        "dc_source": "Beamish Treasures",
        "dc_language": "",
        "dc_relation": "",
        "dc_rights": "",
        "dc_coverage": "",
        "dcterms_provenance": "",
        "dcterms_ispartof": "Beamish Treasures",
        "dcterms_temporal": "",
        "dcterms_spatial": "",
        "europeana_unstored": "",
        "europeana_object":
"http://www.peoplesnetwork.gov.uk/dpp/resource/2060233/stream/thumbnail_image_jpeg",
        "europeana_provider": "CultureGrid",
        "europeana_type": "IMAGE",
        "europeana_rights": "",
        "europeana_dataprovider": "",
        "europeana_isshownby": "",
        "europeana_isshownat":
"http://www.beamishcollections.com/collections/display.asp?ItemID=1",
        "europeana_country": "uk",
        "europeana_language": "en",
        "europeana_uri":
"http://www.europeana.eu/resolve/record/09405/8BBFE1B9EC70EEA34651852DD27A3C0F
2532624C",
        "europeana_usertag": "",
        "europeana_year": "",
        "europeana_previewnodistribute": "",
        "europeana_hasobject": "true",
        "paths_topic": [{
            "uri": "http://paths-project.eu/topic/1",
            "dc_description": "Description",
            "dc_subject": ["Subject"],
            "dc_title": "Title",
            "topic_hierarchy": "PATHS",
            "topic_thumbnails": "thumb.gif"
        }, {
            "uri": "http://paths-project.eu/topic/2",
            "dc_description": "Description 2",
            "dc_subject": ["Subject 2"],
            "dc_title": "Title 2",
            "topic_hierarchy": "PATHS",
            "topic_thumbnails": "thumb2.gif"
        }, {
            "uri": "http://paths-project.eu/topic/3",
            "dc_description": "Description 3",
            "dc_subject": ["Subject 3"],
            "dc_title": "Title 3",
            "topic_hierarchy": "PATHS",
            "topic_thumbnails": "thumb3.gif"
        }],
        "paths_rating": [{
            "likes": "0",
            "dislikes": "0"
        }]
    }
}
```

## Web Method: GetItemsForTopic

**Summary:** Get all items associated with a specific topic.

**GetItemsForTopic Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| topic_uri | s:string | URI of topic |
| myLimit | s:string | Number of results to retrieve |
| myStart | s:string | Where to start retrieving in a result set (paging) |

**GetItemsForTopic Response**

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: List of items |

**Example of GetItemsForTopic HttpGet Request**

**Request:**
http://development.paths-project.eu/Item.asmx/GetItemsForTopic?topic_uri=s:string&myLimit=s:string&myStart=s:string

**Response:**
```
{
    "code": 2,
    "data": {
        "count": 1,
        "items": [
            // Items go here
            ]
    }
}
```

## Web Method: GetItemByUri

**Summary:** Get a single item by its URI

**GetItemByUri Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| item_uri | s:string | URI of item |

**GetItemByUri Response**

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: Single item information |

**Example of GetItemByUri HttpGet Request**

**Request:**
http://development.paths-project.eu/Item.asmx/GetItemByUri?item_uri=s:string

**Response:**

```
{
   "code": 2,
   "data": {
      "id": "1",
      "uri": "http://www.beamishcollections.com/collections/display.asp?ItemID=1",
      "dc_title": "Enamel Advertisement",
      "dc_creator": "",
      "dc_subject": ["Advertising Enamels"],
      "dc_description": "Enamel Advertisement \"Spillers Balanced Rations and UVECO\"/
\"For Cattle, Sheep, Pigs& Poultry\"/ \"We Sell Them\" Height: 1280mm x 795mm.",
      "dc_publisher": "",
      "dc_contributor": "",
      "dc_date": "",
      "dc_type": "Image",
      "dc_format": "",
      "dc_identifier": "http://www.beamishcollections.com/collections/display.asp?ItemID=1",
      "dc_source": "Beamish Treasures",
      "dc_language": "",
      "dc_relation": "",
      "dc_rights": "",
      "dc_coverage": "",
      "dcterms_provenance": "",
      "dcterms_ispartof": "Beamish Treasures",
      "dcterms_temporal": "",
      "dcterms_spatial": "",
      "europeana_unstored": "",
      "europeana_object":
"http://www.peoplesnetwork.gov.uk/dpp/resource/2060233/stream/thumbnail_image_jpeg",
      "europeana_provider": "CultureGrid",
      "europeana_type": "IMAGE",
      "europeana_rights": "",
      "europeana_dataprovider": "",
      "europeana_isshownby": "",
      "europeana_isshownat":
"http://www.beamishcollections.com/collections/display.asp?ItemID=1",
        "europeana_country": "uk",
        "europeana_language": "en",
        "europeana_uri":
"http://www.europeana.eu/resolve/record/09405/8BBFE1B9EC70EEA34651852DD27A3C0F
2532624C",
        "europeana_usertag": "",
        "europeana_year": "",
        "europeana_previewnodistribute": "",
        "europeana_hasobject": "true",
        "paths_topic": [{
            "uri": "http://paths-project.eu/topic/1",
            "dc_description": "Description",
            "dc_subject": ["Subject"],
            "dc_title": "Title",
            "topic_hierarchy": "PATHS",
            "topic_thumbnails": "thumb.gif"
        }, {
            "uri": "http://paths-project.eu/topic/2",
            "dc_description": "Description 2",
            "dc_subject": ["Subject 2"],
            "dc_title": "Title 2",
            "topic_hierarchy": "PATHS",
            "topic_thumbnails": "thumb2.gif"
        }, {
            "uri": "http://paths-project.eu/topic/3",
            "dc_description": "Description 3",
            "dc_subject": ["Subject 3"],
            "dc_title": "Title 3",
            "topic_hierarchy": "PATHS",
            "topic_thumbnails": "thumb3.gif"
        }],
        "paths_rating": [{
            "likes": "0",
            "dislikes": "0"
        }]
    }
}
```

### Web Method: GetItemByID

**Summary:** Get a single item by its ID

**GetItemByID Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| ItemID | s:string | ID of item |

**GetItemByID Response**

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: Single item information |

**Example of GetItemByID HttpGet Request**

**Request:**
http://development.paths-project.eu/Item.asmx/GetItemByID?ItemID=s:string

## Web Service: Topic

### Web Method: GetTopicHierarchy

**Summary:** Returns the parent hierarchy of a topic by its ID

**GetTopicHierarchy Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| topic_id | s:string | Unique database identifier of topic |

**GetTopicHierarchy Response**

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: Topic hierarchy |

**Example of GetTopicHierarchy HttpGet Request**

**Request:**
http://development.paths-project.eu/Topic.asmx/GetTopicHierarchy?topic_id=s:string

### Web Method: GetTopicByUri

**Summary:** Get parent topic hierarchy of topic by its URI

#### GetTopicByUri Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| topic_uri | s:string | URI of topic |

#### GetTopicByUri Response

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: Topic hierarchy |

#### Example of GetTopicByUri HttpGet Request

**Request:**
http://development.paths-project.eu/Topic.asmx/GetTopicByUri?topic_uri=s:string

### Web Method: GetTopicById

**Summary:** Get a topic by its ID

#### GetTopicById Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| topic_id | s:int | Unique database identifier of topic |

#### GetTopicById Response

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: Single topic information |

#### Example of GetTopicById HttpGet Request

**Request:**
http://development.paths-project.eu/Topic.asmx/GetTopicById?topic_id=s:int

## Web Service: Workspace

**Summary:** The Workspace web service contains methods for creating, managing, querying and deleting workspace items. A workspace item can be considered a node which has not yet been completed and/or assigned ot a Path. Workspace items can refer to any object identifiable by a URI and most commonly references records from the Items table.

### Web Method: AddWorkspaceItem

**Summary:** Adds an item to the present users workspace.

#### AddWorkspaceItem Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| fk_rel_uri | s:string | Any URI, but commonly a reference to the URI of a PATHS Item |
| dc_title | s:string | Title of workspace item |
| dc_description | s:string | Description of workspace item (optional) |
| type | s:string | Type of workspace item (optional, used?) |

**AddWorkspaceItem Response**

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: Workspace item |

**Example of AddWorkspaceItem HttpGet Request**

```
Request:
http://development.paths-
project.eu/Workspace.asmx/AddWorkspaceItem?fk_rel_uri=s:string&dc_title=s:string&dc_de
scription=s:string&type=s:string
```

### Web Method: DeleteWorkspaceItem

**Summary:** Deletes an item from the workspace.

#### DeleteWorkspaceItem Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| workspace_id | s:string | Unique datbase identifier of workspace item to be deleted |

#### DeleteWorkspaceItem Response

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: OperationCompletedSuccessfully (code=2) on success, DatabaseSQLError (code=7) on error. |

#### Example of DeleteWorkspaceItem HttpGet Request

**Request:**
http://development.paths-project.eu/Workspace.asmx/DeleteWorkspaceItem?workspace_id=s:string

### Web Method: UpdateWorkspaceItem

**Summary:** Updates the information about an item in the users workspace

#### UpdateWorkspaceItem Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| workspace_id | s:int | Unique database identifier of the workspace item to be updated. |
| fk_rel_uri | s:string | URI of referenced object |
| dc_title | s:string | Title of workspace item |
| dc_description | s:string | Description of workspace item (optional) |
| type | s:string | Type of workspace item (optional) |

#### UpdateWorkspaceItem Response

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: Single workspace item information |

#### Example of UpdateWorkspaceItem HttpGet Request

**Request:**
http://development.paths-

project.eu/Workspace.asmx/UpdateWorkspaceItem?workspace_id=s:int&fk_rel_uri=s:string&dc_title=s:string&dc_description=s:string&type=s:string

### Web Method: GetWorkspaceItem

**Summary:** Get a workspace item by its ID

**GetWorkspaceItem Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| workspace_id | s:string | Unique database identifier of workspace item to be retrieved. |

**GetWorkspaceItem Response**

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: Single workspace item information |

**Example of GetWorkspaceItem HttpGet Request**

**Request:**
http://development.paths-project.eu/Workspace.asmx/GetWorkspaceItem?workspace_id=s:string

### Web Method: GetWorkspaceItems

**Summary:** Get all workspace items for the current authenticated or temporary user.

**GetWorkspaceItems Request Parameters**

**N/A** (this web method does not accept any calling parameters)

**GetWorkspaceItems Response**

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: List of workspace items information |

**Example of GetWorkspaceItems HttpGet Request**

**Request:**
http://development.paths-project.eu/Workspace.asmx/GetWorkspaceItems?

## Web Service: Path

**Summary:** The Path web service contains methods for creation, editing and deletion of paths and path nodes. Furthermore, it has functions to transfer work space items to nodes in a path and to qury paths and nodes. Paths and nodes are the core dynamic objects in the PATHS Web Service API. A path consist of one or more nodes, a node references an item (or another object) via a URI.

### Web Method: DeletePathNode

**Summary:** Delete a node identified by its URI

**Remark:** Method requires authentication

#### DeletePathNode Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| node_uri | s:string | URI of node to be deleted |

#### DeletePathNode Response

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: Single node information |

#### Example of DeletePathNode HttpGet Request

**Request:**
http://development.paths-project.eu/Path.asmx/DeletePathNode?node_uri=s:string

### Web Method: DeletePath

**Summary:** Delete a node identified by its URI

**Remark:** Method requires authentication

#### DeletePath Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| path_uri | s:string | URI of path to be deleted |

#### DeletePath Response

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: Single node information |

#### Example of DeletePath HttpGet Request

**Request:**
http://development.paths-project.eu/Path.asmx/DeletePath?path_uri=s:string

## Web Method: UpdatePathNode

**Summary:** Update information of a node identified by its URI

**Remark:** Method requires authentication

### UpdatePathNode Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| node_uri | s:string | URI of node to be updated |
| fk_path_id | s:string | Unique database identifier of path node should be assigned to (Integer, optional) |
| fk_rel_uri | s:string | URI of object referenced by node. Often an item but can be any object identifiable by a URI (URI, optional) |
| dc_title | s:string | Title of node (optional) |
| dc_description | s:string | Description of node (optional) |
| type | s:string | Type of node (optional, used?) |
| node_order | s:string | Number indicating the position of the node within a path (Double, optional) |

### UpdatePathNode Response

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: OperationCompletedSuccessfully (code=2) on success |

### Example of UpdatePathNode HttpGet Request

**Request:**
http://development.paths-project.eu/Path.asmx/UpdatePathNode?node_uri=s:string&fk_path_id=s:string&fk_rel_uri=s:string&dc_title=s:string&dc_description=s:string&type=s:string&node_order=s:string

## Web Method: AddNodeFromWorkspaceToPath

**Summary:** Add a workspace item from the users workspace to a path as a node.

**Remark:** Metod requires a user to be authenticated

### AddNodeFromWorkspaceToPath Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| path_uri | s:string | URI of path to which node should be added |
| workspace_id | s:string | Unique database identifier of workspace item |
| node_order | s:string | Number indicating the position of the node within the path, defaults to the highest number + 1 (Double, optional) |

### AddNodeFromWorkspaceToPath Response

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: Single node information |

### Example of AddNodeFromWorkspaceToPath HttpGet Request

**Request:**
http://development.paths-project.eu/Path.asmx/AddNodeFromWorkspaceToPath?path_uri=s:string&workspace_id=s:string&node_order=s:string

## Web Method: UpdatePath

**Summary:** Update information of a node identified by its URI

**Remark:** Method requires authentication

### UpdatePath Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| path_uri | s:string | URI of path to be modified |
| dc_title | s:string | Title of node (optional) |
| dc_subject | s:string | Modified subject of path (optional) separater multiple entries by a semicolon ";" |
| dc_description | s:string | Description of node (optional) |
| dc_rights | s:string | Modified rights statement of path (optional) |
| access | s:string | Modified access information for path (optional) |
| lom_audience | s:string | Modified audience for path (optional) |
| lom_length | s:string | Modified length/duration of path (optional) |

### UpdatePath Response

| Data type | Description |
|---|---|
| s:string (JSON) | JSON String: OperationCompletedSuccessfully (code=2) on success |

### Example of UpdatePath HttpGet Request

**Request:**
http://development.paths-project.eu/Path.asmx/UpdatePath?path_uri=s:string&dc_title=s:string&dc_subject=s:string&dc_description=s:string&dc_rights=s:string&access=s:string&lom_audience=s:string&lom_length=s:string

## Web Method: CreatePath

**Summary:** Create a new path

**Remark:** Methods requires a user to be authenticated

### CreatePath Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| dc_title | s:string | Title of path |
| dc_subject | s:string | Subject of path, separate multiple values by a semicolon ";" |
| dc_description | s:string | Description of path |
| dc_rights | s:string | Rights statement for path |
| access | s:string | Access information for path |
| lom_audience | s:string | Audience for path |
| lom_length | s:string | Length/duration of path |

### CreatePath Response

| Data type | Description |
|---|---|
| s:string (JSON) | Path data object for created path |

### Example of CreatePath HttpGet Request

**Request:**
http://development.paths-project.eu/Path.asmx/CreatePath?dc_title=s:string&dc_subject=s:string&dc_description=s:string&dc_rights=s:string&access=s:string&lom_audience=s:string&lom_length=s:string

## Web Method: GetPath

**Summary:** Get a single path identified by its URI

### GetPath Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| path_uri | s:string | URI of path to be retrieved |

### GetPath Response

| Data type | Description |
|---|---|
| s:string (JSON) | Path data object |

**Example of GetPath HttpGet Request**

**Request:**
http://development.paths-project.eu/Path.asmx/GetPath?path_uri=s:string

### Web Method: GetCurrentUserPaths

**Summary:** Get list of paths created by current authenticated user

**Remark:** Method requires a user to be authenticated

**GetCurrentUserPaths Request Parameters**

**N/A** (this web method does not accept any calling parameters)

**GetCurrentUserPaths Response**

| Data type | Description |
|---|---|
| s:string (JSON) | OperationCompletedSuccessfully (code=2) + list of path data objects on success; or QueryDidNotReturnRecords (code=8) if current user has no paths |

**Example of GetCurrentUserPaths HttpGet Request**

**Request:**
http://development.paths-project.eu/Path.asmx/GetCurrentUserPaths?

### Web Method: GetPathsForItem

**Summary:** Get paths associated with a specific item

**GetPathsForItem Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| item_uri | s:string | URI of item for which associated paths should be returned |

#### GetPathsForItem Response

| Data type | Description |
|---|---|
| s:string (JSON) | OperationCompletedSuccessfully (code=2) + list of path data objects on success. |

#### Example of GetPathsForItem HttpGet Request

**Request:**
http://development.paths-project.eu/Path.asmx/GetPathsForItem?item_uri=s:string

## Web Service: Social

**Summary:** The web service Social contains all functionality associated with user generated content which may be attached to paths, nodes and items. UGC elements are associated with resources via a URI and may in principle be attached to any web resource. This reduces the amount of tables required for the connections and simplifies the data management.

### Web Method: GetCommentsForUri

**Summary:** Get comments for a web resource with specified URI

#### GetCommentsForUri Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| fk_rel_uri | s:string | URI of web resource for which comments should be retrieved. |

#### GetCommentsForUri Response

| Data type | Description |
|---|---|
| s:string (JSON) | OperationCompletedSuccessfully (code=2) + list of comment data objects on success. |

#### Example of GetCommentsForUri HttpGet Request

**Request:**
http://development.paths-project.eu/Social.asmx/GetCommentsForUri?fk_rel_uri=s:string

### Web Method: AddComment

**Summary:** Add new comment to web resource identified by URI

**Remark:** Web method requires user to be authenticated

**AddComment Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| fk_rel_uri | s:string | URI of web resource to be commented upon |
| comment | s:string | Comment text |

**AddComment Response**

| Data type | Description |
|---|---|
| s:string (JSON) | OperationCompleteSuccessfully (code=2) + single comment data object |

**Example of AddComment HttpGet Request**

**Request:**
http://development.paths-project.eu/Social.asmx/AddComment?fk_rel_uri=s:string&comment=s:string

## Web Method: DeleteComment

**Summary:** Deletes comment with specified identifier

**Remark:** Method requires a user to be authenticated.

**DeleteComment Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| comment_id | s:int | Unique database identifier of comment to be deleted |

**DeleteComment Response**

| Data type | Description |
|---|---|
| s:string (JSON) | OperationCompletedSuccessfully (code=2) on success. |

**Example of DeleteComment HttpGet Request**

**Request:**
http://development.paths-project.eu/Social.asmx/DeleteComment?comment_id=s:int

## Web Method: AddTag

**Summary:** Adds a tag (keyword) to a resource identified by a URI

**Remark:** Method requires a user to be authenticated

**AddTag Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| fk_rel_uri | s:string | URI of resource which tag should be added to |
| tag | s:string | Any keyword or keyphrase to be used as tag |

**AddTag Response**

| Data type | Description |
|---|---|
| s:string (JSON) | Tag data object and OperationCompletedSuccessfully (code=2) on success |

**Example of AddTag HttpGet Request**

**Request:**
http://development.paths-project.eu/Social.asmx/AddTag?fk_rel_uri=s:string&tag=s:string

## Web Method: DeleteTag

**Summary:** Delete tag with specified URI

**Remark:** Method requires a user to be authenticated

**DeleteTag Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| tag_uri | s:string | URI of the tag to be deleted |

**DeleteTag Response**

| Data type | Description |
|---|---|
| s:string (JSON) | OperationCompletedSuccessfully (code=2) on success |

**Example of DeleteTag HttpGet Request**

**Request:**
http://development.paths-project.eu/Social.asmx/DeleteTag?tag_uri=s:string

## Web Method: GetTagsForUri

**Summary:** Get list of tags associated with a specific resource identified by its URI

**GetTagsForUri Request Parameters**

| Parameter | Data type | Description |
|---|---|---|
| fk_rel_uri | s:string | URI of resource for which tags should be retrieved |

**GetTagsForUri Response**

| Data type | Description |
|---|---|
| s:string (JSON) | QueryDidNotReturnRecords (code=8) if no tags are found, OperationCompletedSuccessfully (code=2) and list of tag data objects on success |

**Example of GetTagsForUri HttpGet Request**

**Request:**
http://development.paths-project.eu/Social.asmx/GetTagsForUri?fk_rel_uri=s:string

### Web Method: AddRating

**Summary:** Add rating to a resource identified by its URI

**Remark:** Requires an authenticated or temporary user session

#### AddRating Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| fk_rating_scale_id | s:int | Unique database identifier for rating_scale table. 1 = dislikes, 2=likes |
| fk_rel_uri | s:string | URI of resource which rating should be added to |

#### AddRating Response

| Data type | Description |
|---|---|
| s:string (JSON) | QueryDidNotReturnRecords (code=8) if no rating values exist; OperationCompletedSuccessfully (code=2) and count of ratings |

#### Example of AddRating HttpGet Request

**Request:**
http://development.paths-project.eu/Social.asmx/AddRating?fk_rating_scale_id=s:int&fk_rel_uri=s:string

### Web Method: DeleteRatingForUri

**Summary:**

#### DeleteRatingForUri Request Parameters

| Parameter | Data type | Description |
|---|---|---|
| fk_rel_uri | s:string | |

#### DeleteRatingForUri Response

| Data type | Description |
|---|---|
| s:string (JSON) | |

**Example of DeleteRatingForUri HttpGet Request**

```
Request:
http://development.paths-project.eu/Social.asmx/DeleteRatingForUri?fk_rel_uri=s:string
```