FAST AND ADVANCED STORYBOARD TOOLS

FP7-ICT-2007-1-216048

http://fast.morfeo-project.eu

Deliverable D5.2.1

Prototype Semantic Catalogue for Screen-Flow Gadgets and Back-End Services

Ismael Rivera, NUIG

Date: 20/02/2009

FAST is partially funded by the E.C. (grant code: FP7-ICT-2007-1-216048).

Version History

Rev. No.	Date	Author (Partner)	Change description
1.0	20/02/2009	Ismael Rivera (NUIG)	First version of the deliverable

Executive Summary

The present deliverable is intented to be the developer's manual of the prototype of the semantic catalogue. That said, whoever developing a component which will interact with the catalogue will find in this document a description of its architecture, the functionality provided together with the API (Application Programming Interface), data interchange formats, code errors and exceptions as well as several examples of usage.

Document Summary

Code	FP7-ICT-2007-1-216048	**Acronym**	FAST
Full title	Fast and Advanced Storyboard Tools		
URL	`http://fast.morfeo-project.eu`		
Project officer	Annalisa Bogliolo		

Deliverable	**Number**	D5.2.1	**Name**	Prototype Semantic Catalogue for Screen-Flow Gadgets and Back-End Services
Work package	**Number**	5	**Name**	Semantic catalogue of screen-flow resources and back-end Web Services

Delivery data	**Due date**	27/02/2009	**Submitted**	27/02/2009
Status	Status		final	
Dissemination Level	Public ☒ / Consortium ☐			
Short description of contents	This deliverable is the technical documentation for the prototype developed as part of the WP5. It describes the catalogue's architecture, data interchange formats, APIs and examples of how to interact with it.			
Authors	Ismael Rivera, NUIG			
Deliverable Owner (Partner)	Ismael Rivera, NUIG		**email**	ismael.rivera@deri.org
			phone	+353 91 495086
Keywords	FAST, semantic catalogue, gadget catalogue, RDF store			

Table of contents

1 Introduction

This section starts establishing the goal and scope of the present document, shows how it is structured and details the relation to others documents and work packages.

1.1 Goal and Scope

This is an introductory manual for developers who want to adopt and use the FAST semantic catalogue. It explains the main functionalities implemented so far, an overview of its architecture and a detailed *Application Programming Interface* or API of the complete set of the operations offered through a REST service.

1.2 Structure of the document

The deliverable presents both the external and internal architecture in Section 2, then in Section 3 it is detailed the Catalogue API, query formats, interchange formats, error codes and so on, and the document ends by stating some conclusions and future directions of the semantic catalogue in Section 4.

1.3 Overall approach for WP5 and relation to other Work Packages and Deliverables

The main objective of the Work Package 5, as specified in [**?**], includes the development of the FAST semantic catalogue, considered the back-end of the FAST platform, which will store and index any kind of resources such as screen-flows, screens, back-end Web services descriptions, locate appropriate gadgets, resources and Web services for users, storing and indexing user profiles, and resolve mediation problems between gadgets, back-end services and the different ontologies stored.

The deliverables stated in [**?**] for this WP are:

User manual of the FAST Catalogue This is the user manual of the catalogue, but is out of scope in this iteration of the project.

Prototype semantic catalogue for screen-flow gadgets and back-end services This deliverable is a software prototype of the semantic catalogue for experimentation purposes. Apart from the software, it will be created technical documentation and manuals for developers as the present document.

The semantic catalogue relies strongly on the FAST ontology defined in [**?**] since the ontology defines how every element within FAST will be internally stored by the catalogue, and takes into consideration specific requirements detailed in [**?**] where the functionalities of the complete FAST tool are defined.

2 Architecture

This section details the internal catalogue's architecture and which external components will interact with it. Basically three layers can be distinguished: Presentation, Business Logic and Persistence layer (see Figure 2).

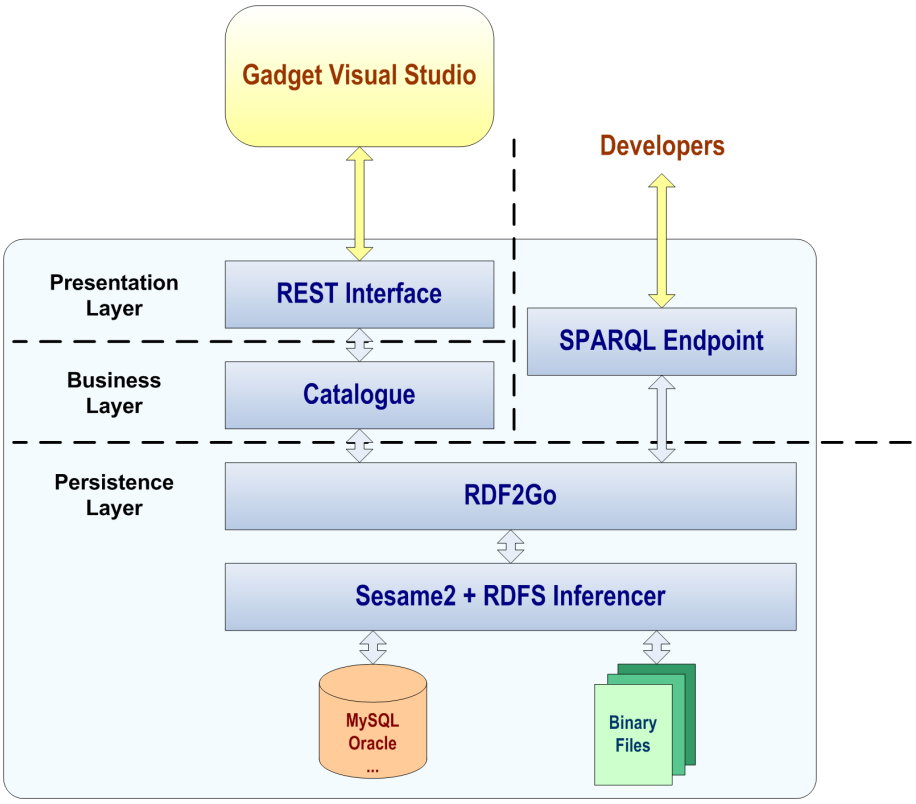

Figure 1: Catalogue's Architecture

The Presentation layer will be the public interface of the catalogue. The main purpose of this layer is to provide its functionality to other components within the Gadget Visual Studio. The service is offered in a REST (REpresentational State Transfer) style, making it easy to consume than other complex APIs. REST is an architectural style, not a toolkit or a standard. Even though, it makes use of standards like HTTP, URI, Resource Representations (XML, HTML, JSON, JPEG, etc.) or MIME types (text/xml, text/html, application/json, image/jpeg, etc.). Another characteristic of the REST style is its stateless assumption. The catalogue adopts this strategy; therefore every request needs a complete set of information in order to prepare the response.

As part of the Presentation layer, a SPARQL endpoint is offered using the SPARQL protocol service as defined in the SPROT [**?**] specification. The SPARQL endpoint is mostly offered to

enable other developers to query directly the catalogue knowledge base using SPARQL queries. This feature is supported by the Sesame RESTful HTTP interface for SPARQL Protocol for RDF.

The Business Logic layer contains all the FAST domain-specific processing. It provides functions to interact with all the elements of the domain model specified in [**?**]. It processes the input from the Presentation layer, creating specific objects modelling the business logic, and vice versa. In addition it interacts with the Persistence layer in order to persist the model.

The Persistence layer provides an API which allows the Business layer to work with a standard set of objects that read and save their state to the triple store. For this reason, on top of this layer it is used RDF2Go, an abstraction over triple (and quad) stores. The RDF2Go API allows interacting with the semantic representation of the model (triples) in a generic manner, and also brings the flexibility of choosing different triple stores to persist them. The selected implementation for the triple stored is Sesame 2. Moreover, this framework is completely extensible and configurable regarding to storage mechanisms, inferencers, RDF file formats, query result formats and query languages. For the actual catalogue prototype, the storage mechanism used is the native file storage system of Sesame, the inferencer used is a subset of the RDFS entailment rules [**?**] following a forward-chaining policy.

3 RESTful Catalogue API

This section provides a high-level overview of the Catalogue API. It describes the calls or operations supported, specific parameters and responses for each operation, supported interchange formats and some examples to facilitate the understanding of the API.

3.1 JSON Interchange Format

The format supported by the Catalogue API for all the request and responses is JSON[1]. JSON is a lightweight data interchange format whose simplicity has resulted in widespread use among web developers, easy to read and write and able to using any programming language because its structures map directly to data structures used in most programming languages.

Every HTTP request and response is encoded using the MIME type `application/json` and the charset `UTF-8`.

3.1.1 Example of a screen

```
{
  "uri": null,
  "label": {"en-GB": "Screen - DERI"},
  "description": {"en-GB": "Returns people working in DERI"},
  "creationDate": "2009-02-07T09:59:52+0000",
  "creator": "http://www.ismaelrivera.es/",
    "domainContext": {
      "tags": [],
      "user": null
    },
  "homepage": "http://www.deri.ie/",
  "icon": "http://www.deri.ie/icon.jpg",
```

[1] http://www.json.org/

```
  "screenshot": "http://www.deri.ie/screenshot.jpg",
  "rights": "http://creativecommons.org/",
  "version": "1.0",
  "preconditions": [],
  "postconditions": ["?person rdf:type foaf:Person .
                      ?person foaf:workplaceHomepage http://www.deri.ie/"]
}
```

3.1.2 Internationalisation I18n

In order to offers an adaptable solution to various languages and regions without major engineering changes, internationalisation is considered from an early stage in the catalogue development. Underlying representation technologies used to develop the catalogue (RDF and JSON) allow to implement this feature. This feature is implemented by the addition of a language tag to every 'string' desired. The specification of this language tag is composed by the language code and the country code, following the ISO 639[2] for languages and the ISO 3166[3] for countries. The following example illustrates how to use it properly in both formats.

```
{
  "label": {
    "en-GB": "Simple Example",
    "es-ES": "Ejemplo Sencillo",
    "de-DE": "Einfaches Beispiel"
  },
  "description": {
    "en-GB": "This is a simple example of a screen",
    "es-ES": "Esto es un ejemplo sencillo de pantalla",
    "de-DE": "Dies ist ein einfaches Beispiel für einen Bildschirm"
  },
}
```

```
<rdf:Description rdf:about="http://www.morfeoproject.eu/fast/fco#ExampleScreen">
```

[2] http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt
[3] http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

```
  <rdf:type rdf:resource="http://www.morfeoproject.eu/fast/fco#Screen"/>
  <rdfs:label xml:lang="en-GB">Simple Example</rdfs:label>
  <rdfs:label xml:lang="es-ES">Ejemplo Sencillo</rdfs:label>
  <rdfs:label xml:lang="de-DE">Einfaches Beispiel</rdfs:label>
  <description
    xmlns="http://purl.org/dc/terms/"
    xml:lang="en-GB">This is a simple example of a screen</description>
  <description
    xmlns="http://purl.org/dc/terms/"
    xml:lang="es-ES">Esto es un ejemplo sencillo de pantalla</description>
  <description
    xmlns="http://purl.org/dc/terms/"
    xml:lang="de-DE">Dies ist ein einfaches Beispiel für einen Bildschirm</description>
</rdf:Description>
```

By now it is allowed using internationalisation for the label and the description of a screen, but more attributes will be benefited by this feature.

3.2 API Calls

This section contains detailed descriptions of the interface provided by the catalogue via REST services, detailing request parameters, response elements, any special errors and examples of requests and responses. The URL format is also specified for each operation, where 'catalogueURL' has to be replaced for the real URL the service is installed (e.g. forge.morfeoproject.org/catalogue).

3.2.1 CRUD operations for Screens

A screen is a resource which can be created, retrieved, modified or deleted since the catalogue provides CRUD support for screens. These features are provided following the REST philosophy, using the HTTP methods. Two concepts have to be defined: *collection*, as a set of screens which

access URI is http://catalogueURL/screens/ where catalogueURL is the specific URL where the catalogue server is installed, and *member* of the collection, in other words, the screen itself, which access URI is http://catalogueURL/screens/<id> where the <id> has to be replaced by the identifier of a screen. The details of the operations and which HTTP verb has to be used can be found in the Table 1.

Table 1: CRUD operations for Screens

Resource	HTTP method	HTTP body	Description
Collection URI	GET	N/A	**List** the members of the collection.
Collection URI	PUT	N/A	Not used.
Collection URI	POST	JSON representation of the screen	**Create** a new entry in the collection where the URI is assigned automatically by the collection. The URI created is returned by this operation.
Collection URI	DELETE	N/A	Not used.
Member URI	GET	N/A	**Retrieve** the addressed member of the collection.
Member URI	PUT	JSON representation of the screen	**Update** the addressed member of the collection or create it with a defined URI.
Member URI	POST	N/A	Not used.
Member URI	DELETE	N/A	**Delete** the addressed member of the collection.

Here are shown various examples of request and responses using the catalogue server. To create a new screen, a POST request is send to the catalogue server, including the JSON representation of the screen as the body of the request:

```
{
  "uri": null,
  "label": {"en-GB": "Screen - DERI"},
  "description": {"en-GB": "Returns people working in DERI"},
  "creationDate": "2009-02-07T09:59:52+0000",
  "creator": "http://www.ismaelrivera.es/",
    "domainContext": {
```

```
      "tags": [],
      "user": null
    },
  "homepage": "http://www.deri.ie/",
  "icon": "http://www.deri.ie/icon.jpg",
  "screenshot": "http://www.deri.ie/screenshot.jpg",
  "rights": "http://creativecommons.org/",
  "version": "1.0",
  "preconditions": [],
  "postconditions": ["?person rdf:type foaf:Person .
                      ?person foaf:workplaceHomepage http://www.deri.ie/"]
}
```


The URI will be not taken into account since the catalogue will generate a unique URI for each resource created. So, the response for this operation is:


```
{
  "creationDate": "2009-02-07T09:59:52+0000",
  "creator": "http://www.ismaelrivera.es/",
  "description": {"en-GB": "Show information about a person working in DERI"},
  "domainContext": {
    "tags": [],
    "user": "null"
  },
  "homepage": "http://www.deri.ie/",
  "icon": "http://www.deri.ie/icon.jpg",
  "label": {"en-GB": "Screen - DERI"},
  "postconditions": [],
  "preconditions": ["?person rdf:type foaf:Person .
                     ?person foaf:workplaceHomepage http://www.deri.ie/"],
  "rights": "http://creativecommons.org/",
  "screenshot": "http://www.deri.ie/screenshot.jpg",
  "uri": "http://www.morfeoproject.eu/fast/fco#Screen1234552194269",
  "version": "1.0"
```

```
}
```

To obtain all the screens stored in the catalogue a HTTP GET request is sent to the Collection URI and the response may be something like this:

```
[
  {
    "creationDate": "2009-02-07T09:59:52+0000",
    "creator": "http://www.ismaelrivera.es/",
    "description": {"en-gb": "Returns people working in DERI"},
    "domainContext": {
      "tags": [],
      "user": null
    },
    "homepage": "http://www.deri.ie/",
    "icon": "http://www.deri.ie/icon.jpg",
    "label": {"en-gb": "Screen - DERI"},
    "postconditions": ["?person rdf:type foaf:Person .
                        ?person foaf:workplaceHomepage http://www.deri.ie/"],
    "preconditions": [],
    "rights": "http://creativecommons.org/",
    "screenshot": "http://www.deri.ie/screenshot.jpg",
    "uri": "http://www.morfeoproject.eu/fast/fco#Screen1234552164472",
    "version": "1.0"
  },
  {
    "creationDate": "2009-02-07T09:59:52+0000",
    "creator": "http://www.ismaelrivera.es/",
    "description": {"en-gb": "Show information about a person working in DERI"},
    "domainContext": {
      "tags": [],
      "user": null
    },
    "homepage": "http://www.deri.ie/",
```

```
    "icon": "http://www.deri.ie/icon.jpg",
    "label": {"en-gb": "Screen - DERI"},
    "postconditions": [],
    "preconditions": ["?person rdf:type foaf:Person .
                       ?person foaf:workplaceHomepage http://www.deri.ie/"],
    "rights": "http://creativecommons.org/",
    "screenshot": "http://www.deri.ie/screenshot.jpg",
    "uri": "http://www.morfeoproject.eu/fast/fco#Screen1234552194269",
    "version": "1.0"
  },
  {
    "creationDate": "2009-02-07T09:59:52+0000",
    "creator": "http://www.ismaelrivera.es/",
    "description": {"en-gb": "Show information about a person"},
    "domainContext": {
      "tags": [],
      "user": null
    },
    "homepage": "http://www.deri.ie/",
    "icon": "http://www.deri.ie/icon.jpg",
    "label": {"en-gb": "Screen - DERI"},
    "postconditions": [],
    "preconditions": ["?person rdf:type foaf:Person"],
    "rights": "http://creativecommons.org/",
    "screenshot": "http://www.deri.ie/screenshot.jpg",
    "uri": "http://www.morfeoproject.eu/fast/fco#Screen1234552373865",
    "version": "1.0"
  }
]
```

3.2.2 CRUD operations for Screen-Flows

A screenflow is a resource which can be created, retrieved, modified or deleted since the catalogue provides CRUD support for screenflows. These features are provided following the REST philosophy, using the HTTP methods. As in the previous section the concepts *collection* and *member* are used including their URLs. Table 2 details the operations offered for screen-flows.

Table 2: CRUD operations for Screen-Flows

Resource	HTTP method	HTTP body	Description
Collection URI	GET	N/A	**List** the members of the collection.
Collection URI	PUT	N/A	Not used.
Collection URI	POST	JSON representation of the screen-flow	**Create** a new entry in the collection where the URI is assigned automatically by the collection. The URI created is returned by this operation.
Collection URI	DELETE	N/A	Not used.
Member URI	GET	N/A	**Retrieve** the addressed member of the collection.
Member URI	PUT	JSON representation of the screen-flow	**Update** the addressed member of the collection or create it with a defined URI.
Member URI	POST	N/A	Not used.
Member URI	DELETE	N/A	**Delete** the addressed member of the collection.

The following is an example of the request and response while creating a screen-flow:

```
{
  "uri": null,
  "label": {"en-GB": "My First ScreenFlow"},
  "description": {"en-GB": "A example of screenflow"},
  "creationDate": "2009-02-05T13:11:41+0000",
  "creator": "http://www.ismaelrivera.es/",
  "domainContext": {
```

```
    "tags": ["people"],
    "user": null
  },
  "homepage": "http://www.screenflow.com/",
  "icon": "http://www.screenflow.com/icon.jpg",
  "screenshot": "http://www.screenflow.com/screenshot.jpg",
  "rights": "http://creativecommons.org/",
  "version": "1.0",
  "preconditions": [],
  "postconditions": [],
  "screens": [
    "http://www.morfeoproject.eu/fast/fco#Screen1233750841383",
    "http://www.morfeoproject.eu/fast/fco#Screen1233750828836",
    "http://www.morfeoproject.eu/fast/fco#Screen1233750820914"
  ]
}

{
  "creationDate": "2009-02-05T13:11:41+0000",
  "creator": "http://www.ismaelrivera.es/",
  "description": {"en-GB": "A example of screenflow"},
  "domainContext": {
    "tags": ["people"],
    "user": null
  },
  "homepage": "http://www.screenflow.com/",
  "icon": "http://www.screenflow.com/icon.jpg",
  "label": {"en-GB": "My First ScreenFlow"},
  "postconditions": [],
  "preconditions": [],
  "rights": "http://creativecommons.org/",
  "screens": [
    "http://www.morfeoproject.eu/fast/fco#Screen1233750841383",
    "http://www.morfeoproject.eu/fast/fco#Screen1233750828836",
```

```
    "http://www.morfeoproject.eu/fast/fco#Screen1233750820914"
  ],
  "screenshot": "http://www.screenflow.com/screenshot.jpg",
  "uri": "http://www.morfeoproject.eu/fast/fco#ScreenFlow1233752391898",
  "version": "1.0"
}
```

3.2.3 Find

The find operation search inside the catalogue for any screen which could be somehow related to the gadget the user is creating. It provides a recommended set of screens depending on the domain context, the canvas, etc. The specific URL to access this operation is http://catalogueURL/find using HTTP POST method. From now on, the method 'find' only considers the domain context, being a list of tags and the pre/postconditions of the screens. It will try to satisfy all the unsatisfied preconditions of a given list of screens also known as canvas. The results will be all the screens stored in the catalogue which fulfil some of these preconditions, and are tagged with the tags specified in the domain context. The request parameters are shown in Table 3 and the response parameters are shown in Table 4.

Table 3: Find Request Parameters

Name	Description	Type
Canvas	The canvas is composed by a list of screens. Only the URI is needed.	Optional
Domain Context	The domain context contains a list of tags and a user.	Optional
Elements	It is a list of resources, previously stored in the catalogue. For example, the list of screens recommended last execution. Only accepts screens.	Optional

Following is shown an example of usage of this operation considering a canvas with the screen `http://www.morfeoproject.eu/fast/fco#Screen1234552373865`.

```
{
  canvas: [
```

Table 4: Find Response Parameters

Name	Description	Type
N/A	A list of recommended screens is returned.	Required


```
    {
      uri: "http://www.morfeoproject.eu/fast/fco#Screen1234552373865"
    }
  ],
  domainContext: {
    tags: [],
  user: null
  },
  elements: []
}
```


After the execution of the recommendation algorithm, the response given by the catalogue is:


```
[{
  "creationDate": "2009-02-07T09:59:52+0000",
  "creator": "http://www.ismaelrivera.es/",
  "description": {"en-gb": "Returns people working in DERI"},
  "domainContext": {
    "tags": [],
    "user": null
  },
  "homepage": "http://www.deri.ie/",
  "icon": "http://www.deri.ie/icon.jpg",
  "label": {"en-gb": "Screen - DERI"},
  "postconditions": ["?person rdf:type foaf:Person .
                      ?person foaf:workplaceHomepage http://www.deri.ie/"],
  "preconditions": [],
  "rights": "http://creativecommons.org/",
  "screenshot": "http://www.deri.ie/screenshot.jpg",
```

```
  "uri": "http://www.morfeoproject.eu/fast/fco#Screen1234552164472",
  "version": "1.0"
}]
```

3.2.4 Check

This operation verifies the state of certain list of screens depending on a specific criterion. The only criterion accepted in this stage of the prototype is 'reachability'. A screen will be reachable if it all its preconditions are satisfied by postconditions of the screens the canvas contain. The specific URL to access this operation is http://catalogueURL/check using HTTP POST method.

Table 5: Check Request Parameters

Name	Description	Type
Canvas	The canvas is composed by a list of screens. Only the URI is needed.	Optional
Domain Context	The domain context contains a list of tags and a user.	Optional
Elements	It is a list of resources, previously stored in the catalogue. For example, the list of screens recommended last execution. Only accepts screens.	Optional
Criterion	The criterion specifies what it has to be check. The only possible supported now is 'reachability' in order to check if the preconditions of a screen are satisfied.	Required

Table 6: Check Response Parameters

Name	Description	Type
Canvas	The canvas indicating which screens satisfy the critetion. More aditional information may be included, for example, a satisfaction attribute for every precondition for reachability.	Optional
Domain Context	The domain context contains a list of tags and a user.	Optional
Elements	The elements indicating which satisfy the critetion. More aditional information may be included as in the Canvas.	Optional

The scenario for the following example is composed by a screen in the canvas which precondition is a foaf:Person which foaf:workplaceHomepage is http://www.deri.ie/, and a screen in the elements list which precondition is a foaf:Person. The state to check in this case is the reachability and satisfaction of the screens and preconditions.


```
{
  canvas: [
    {
      uri: "http://www.morfeoproject.eu/fast/fco#Screen1234552194269"
    }
  ],
  domainContext: [],
  elements: [
    {
      uri: "http://www.morfeoproject.eu/fast/fco#Screen1234552373865"
    }
  ],
  criterion: "reachability"
}
```


As you can see in the response, no screen is reachable at the moment, and no precondition is satisfied.


```
{
  "canvas": [{
    "preconditions": [{
      "expression": "?person rdf:type foaf:Person .
                     ?person foaf:workplaceHomepage http://www.deri.ie/",
      "satisfied": false
    }],
    "reachability": false,
    "uri": "http://www.morfeoproject.eu/fast/fco#Screen1234552194269"
  }],
  "elements": [{
    "preconditions": [{
```

```
      "expression": "?person rdf:type foaf:Person",
      "satisfied": false
    }],
    "reachability": false,
    "uri": "http://www.morfeoproject.eu/fast/fco#Screen1234552373865"
  }]
}
```


3.2.5 GetMetadata

Although the information about a resource can be obtained by its specific retrieval operation supported by the CRUD interface, sometimes is needed to retrieve information which would imply a request per resource, hence this operation allow to get the metadata of a list of resources in only one request. This prototype supports this operation for screenflows and screens. The specific URL to access this operation is http://catalogueURL/getmetadata using HTTP POST method. Table 7 shows the parameters needed for the invocation and Table 8 details the different parameters may contain a response.

Table 7: GetMetadata Request Parameters

Name	Description	Type
N/A	A list of URIs.	Required

Table 8: GetMetadata Response Parameters

Name	Description	Type
Screenflows	A set of screenflows with all the metadata associated to them..	Optional
Screens	A set of screens with all the metadata associated to them.	Optional

The following example send two URIs of screens and a URI of a screen-flow to the GetMetadata operation:


```
[
```

```
  "http://www.morfeoproject.eu/fast/fco#Screen1234552164472",
  "http://www.morfeoproject.eu/fast/fco#Screen1234552194269",
  "http://www.morfeoproject.eu/fast/fco#ScreenFlow1233752391898"
]
```


The response obtained are two lists, one containing all the metadata regarding to the screen-flows and another one with the information about the screens:


```
{
  "screenflows": [{
    "creationDate": "2009-02-05T13:11:41+0000",
    "creator": "http://www.ismaelrivera.es/",
    "description": {"en-gb": "A example of screenflow"},
    "domainContext": {
      "tags": ["people"],
      "user": null
    },
    "homepage": "http://www.screenflow.com/",
    "icon": "http://www.screenflow.com/icon.jpg",
    "label": {"en-gb": "My First ScreenFlow"},
    "postconditions": [],
    "preconditions": [],
    "rights": "http://creativecommons.org/",
    "screens": [
      "http://www.morfeoproject.eu/fast/fco#Screen1233750820914",
      "http://www.morfeoproject.eu/fast/fco#Screen1233750828836",
      "http://www.morfeoproject.eu/fast/fco#Screen1233750841383"
    ],
    "screenshot": "http://www.screenflow.com/screenshot.jpg",
    "uri": "http://www.morfeoproject.eu/fast/fco#ScreenFlow1233752391898",
    "version": "1.0"
  }],
  "screens": [
    {
      "creationDate": "2009-02-07T09:59:52+0000",
```

```
    "creator": "http://www.ismaelrivera.es/",
    "description": {"en-gb": "Returns people working in DERI"},
    "domainContext": {
      "tags": [],
      "user": null
    },
    "homepage": "http://www.deri.ie/",
    "icon": "http://www.deri.ie/icon.jpg",
    "label": {"en-gb": "Screen - DERI"},
    "postconditions": ["?person rdf:type foaf:Person .
                        ?person foaf:workplaceHomepage http://www.deri.ie/"],
    "preconditions": [],
    "rights": "http://creativecommons.org/",
    "screenshot": "http://www.deri.ie/screenshot.jpg",
    "uri": "http://www.morfeoproject.eu/fast/fco#Screen1234552164472",
    "version": "1.0"
  },
  {
    "creationDate": "2009-02-07T09:59:52+0000",
    "creator": "http://www.ismaelrivera.es/",
    "description": {"en-gb": "Show information about a person working in DERI"},
    "domainContext": {
      "tags": [],
      "user": null
    },
    "homepage": "http://www.deri.ie/",
    "icon": "http://www.deri.ie/icon.jpg",
    "label": {"en-gb": "Screen - DERI"},
    "postconditions": [],
    "preconditions": ["?person rdf:type foaf:Person .
                       ?person foaf:workplaceHomepage http://www.deri.ie/"],
    "rights": "http://creativecommons.org/",
    "screenshot": "http://www.deri.ie/screenshot.jpg",
```

```
      "uri": "http://www.morfeoproject.eu/fast/fco#Screen1234552194269",
      "version": "1.0"
    }
  ]
}
```

3.2.6 FindAndCheck

Not supported yet.

3.3 API Error Codes

There are two types of error codes, client and server.

- Client error codes are generally caused by the client and might be an invalid domain or an invalid request parameter. These errors are accompanied by a 4xx HTTP response code. For example: ResourceNotFound.
- Server error codes are generally caused by a server-side issue and should be reported. These errors are accompanied by a 5xx HTTP response code. For example: ServerUnavailable.

The following table lists all the error codes.

Table 9: API Error Codes

Error	Description	HTTP Status Code
ResourceNotFound	The resource <resourceURI> has not been found.	404 Not Found
AccessFailure	Access to the resource <resourceName> is denied.	403 Forbidden
InternalError	Request could not be executed due to an internal service error.	500 Internal Server Error
InvalidAction	The action <actionName> is not valid for this web service.	404 Bad Request
InvalidHttpRequest	The HTTP request is invalid. Reason: <reason>.	400 Bad Request
InvalidLiteral	Illegal literal in the filter expression.	400 Bad Request
InvalidParameterValue	The specified parameter value is not valid.	400 Bad Request
InvalidURI	The URI <requestURI> is not valid.	400 Bad Request
MissingAction	No action was supplied with this request.	400 Bad Request
MissingParameter	The request must contain the specified missing parameter.	400 Bad Request
NotYetImplemented	Feature <feature> is not yet available.	401 Unauthorized
ServiceUnavailable	The service is currently unavailable. Please try again later.	
UnsupportedHttpVerb	The requested HTTP verb is not supported: <verb>.	400 Bad Request
URITooLong	The URI exceeded the maximum limit of <maxLength>.	400 Bad Request

4 Conclusions and Future Work

As a result of the work done for the back-end catalogue, any one who read this document should be able to understand the capabilities implemented and how to use them. Summarising, this prototype allows the storage of screens and the composition of screen-flows. Future work will involve the storage of the internal components of a screen, such as connectors, filters, forms and back-end services. However, the most interesting work will be done in the optimization of the resources recommendation feature, in other words, it will be investigate new mechanisms to improve the results of the find such as weighting of answers, planning, etc. Last of all, more complicated queries will be taken into account, improving the matching algorithm between pre and postconditions.

References

[DoW, 2007] (2007). Fast description of work.

[Grant et al., 2008] Grant, K., Feigenbaum, L., and Torres, E. (2008). Sparql protocol for rdf. Recommendation, World Wide Web Consortium (W3C).

[Hayes, 2004] Hayes, P. (2004). Rdf semantics. Recommendation, World Wide Web Consortium (W3C).

[Möller, 2009] Möller, K. (2009). Ontology and conceptual model for the semantic characterisation of complex gadgets. Deliverable 2.2.1, FAST Project (FP7-ICT-2007-1-216048).

[Villoslada, 2009] Villoslada, E. (2009). Fast requirements specification. Deliverable 2.3.1, FAST Project (FP7-ICT-2007-1-216048).

Appendix A (Lists of Tables and Figures)

List of Tables

List of Figures