The Metalist Model: A Simple and Extensible Information Model for the Future Internet

Éric Renault and Djamal Zeghlache

Institut Télécom — Télécom SudParis
RS2M Department, Évry, France
{eric.renault,djamal.zeghlache}@it-sudparis.eu

Abstract. In the scope of the Future Internet, where the network is moving from a node-centric to an information-centric organization, the way to specify the metadata associated to objects becomes crucial for scalability, performance and complexity reasons. This article presents an original information model, called the metalist model, that describes metadata in a simple, efficient and extensible way. The metadata can be provided in several ways: directly from the metalist description of the object itself, embedded into the metalist via an inclusion from another metalist and automatically translated from another metadata format. These features enable gathering metadata common to many objects into common metalists to simplify updates and synchronization and smooth harmonization with other existing object formats and descriptions.

1 Introduction

The Internet has been a strategic infrastructure with a key socio-economical role for more than a decade, leading innovation, economic growth and productivity at a world-wide scale. About 1.5 billion people are connected to the Internet today and up to 4 billion people are expected to access the Internet in very few years. This will become possible mainly with the deployment of wireless technologies that will provide a fully pervasive Internet infrastructure with anywhere and anytime connectivity. In this expected evolution, users will also become producers of content, applications and services. Combined with the emergence of communicating objects this will lead to an even wider explosion. For example, billions of components like wireless terminals, RFID tags, real and virtual world objects will become accessible, moving the network to an Internet of Things, in fact an Internet of Objects and Subjects all of which will require swift and reliable networking. New usage will appear and applications available on the Internet will be significantly different. Health care, education, proximity services, energy management, etc. will directly benefit from the expected evolutions. However, this will become feasible only if the Future Internet includes new features and services like self-configuration/organization/management, computing power on-demand, resource discovery, etc. Organizations and institutions all around the world are funding research and development projects to design a new Internet that shall meet these new needs and demands.

Most of the new services related to the Future Internet are relying on the ability of the network to provide accurate information about accessible objects. With such a requirement and assumption, the description of objects becomes a crucial element for the Future Internet. Without flexible and rich object descriptions, no efficient search is possible. Many models have been developed during the past few years in order to respond to this demand through descriptions enabling and facilitating semantic search. Some important initiatives are Dublin Core [11], EXIF [7], IIM [6], OWL [10], RDF [9] and XMP [2]. Most of these solutions are focusing on a given domain, e.g. Dublin Core is dedicated to content and intellectual property, EXIF describes digital pictures, while others adopt more generic solutions such as IPTC that deals with the description of data in general. OWL, RDF and XMP are even less restrictive by allowing users to define their own schema and leading to the development of extensions. This is the case for example with OWL2 [3], OWL-L [5], OWLS-MX [8], etc.

This article presents an original information model that describes any kind of objects and allows 1) metadata to be provided directly in a very simple manner, 2) factorization of metadata to save space and increase consistency and 3) automatic translation of metadata available in any kind of format to the proposed information model. The next section of this paper introduces the different types of objects addressed by the proposed information model which is presented in Sect. 3. Section 4 gives some examples to explain the model and illustrate its use.

2 Object Definition

Our work has been developed in the scope of the 4WARD project [1] in which objects are identified and defined [4]. To link this work with prior art, a definition of the different kinds of addressed objects is provided. Note, however, that the model presented here applies to other frameworks and is not limited to this specific context.

In order to highlight the difference between the considered types of objects, the example of a web page stored in the system is used. To illustrate the applicability of the model to larger frameworks, an example related to RFID tags is also presented.

Access to information, in the Networking of Information (NetInf) paradigm of the European project 4WARD, is based on the use of three different types of objects (as shown on Fig. 1):

– at the lowest level, *Bit-level Objects* (BO) are the binary representation of the objects, i.e. they are composed of the raw data of the object; i.e. the data stored in the storage space. Regarding the storage of a web page, the BO holds the effective content of the web page.
– on top of Bit-level Objects, *Data Objects* (DO) are used to locate the BOs associated to the object. In the case of the web page, the Data Object is the URL of the web page, i.e. its location on the storage space.
– at the highest level, *Information Objects* (IO) describe the content of objects, i.e. they contain the semantics or meaning of objects. For a web page, this is

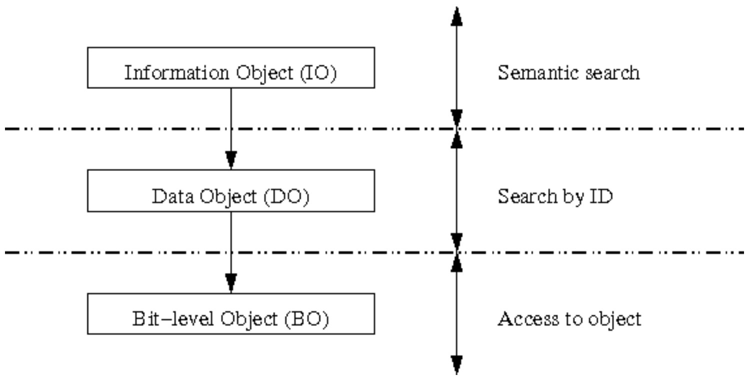

Fig. 1. Object organization in NetInf

similar to the META tag in HTML documents which are used to index web pages according to content of the pages.

Two possible search types are available. The *search by ID* is quite straightforward in the sense that it directly returns the Bit-level Object associated to the Data Object whose ID is provided. The *semantic search* relies on the information found in the metadata and gives access to the Data Objects ID. Once the object ID is obtained via this first step, a search by ID is necessary to retrieve the object.

An ID is associated to IOs and DOs, but there are no IDs for BOs. In fact, BOs are not directly accessible from outside of the NetInf architecture. BOs can only be addressed through DOs. Thus, to access an object, a user first sends a request to NetInf to get the list of objects that match a given set of keywords(those stored in the Information Objects). Associated to these information objects are Data Object IDs. After making a choice, the user sends a request to NetInf with the Data Object ID to retrieve and the Bit-level Object associated to this DO is returned to the user.

In the case of an RFID tag, the situation is very similar. In fact, the tag itself is not directly part of the Future Internet, but is accessed through a process. As a result, the running program accessing the RFID tag is the BO and any communication with the tag is performed via the process. The DO associated to this object holds the information to retrieve the location of the tag and the way to get in touch with it and the IO remains the set of meta-information that describes the characteristics, the location, the meaning, etc. of the RFID tag.

3 Information Model

As the Information Model aims at organizing Information Objects, it mainly focuses on the highest layer of Fig. 1. Several ways of providing metadata have been identified and therefore have to be taken into account:

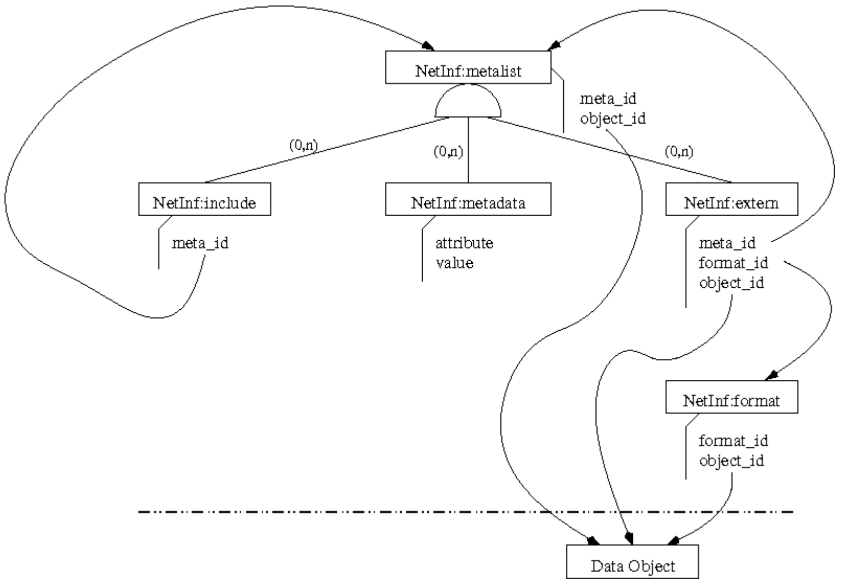

Fig. 2. Information model for the Network of Information

- the metadata can be provided directly to the Information Object and thus be embedded directly in the metalist description.
- the metadata (or part of the metadata) preexists already in another Information Object. In this case, it is better to include it using a reference rather than integrating a copy. This is useful to allow data coherence and consistency in large systems.
- the metadata preexists in another format. In this case again, it is better to include the metadata rather than inserting a copy. However, this inclusion requires a translation and this should be done automatically. Moreover, the metadata can be stored together with the object or in the object itself, or it can also be stored in a separate object. Both cases have to be taken into account.

Figure 2 presents a modeling of data that includes all three cases presented previously and described below. At the top of the diagram, the elements are related to the Information Objects while the lower part of the figure corresponds to Data Objects. Since the only requirement here is that Data Objects are all associated an ID to identify them uniquely, the rest of the paper will not provide any in depth analysis of Data Objects but instead focus on the IOs.

The name of all tags (or entity) are preceded by the “NetInf:” prefix as a reference to the *Network of Information*. This reduces name pollution and puts the focus on a single name space. The use of a prefix is especially interesting when using a modeling language like XML. Associated properties are not subject to this prefix as they are implicitly included inside the name space.

The basic element for the description of metadata is *NetInf:metadata*. It specifies metadata for an object. A NetInf:metadata is associated two properties:

- *value* is any byte stream with a semantic meaning that is helpful to describe the object. There is no real limit on what can be stored in the value. It can be as short as an empty string as this can be associated a meaning for a given attribute (see the attribute property below), and it can be as long as needed for any given use. On one hand, it can be flat or have no structure; on the other hand, it can include a hierarchy of data, e.g. with nested tags as available with XML.
- *attribute* is a string of characters that tags the value of the metadata. Attributes may be flat or organized into hierarchies. Both possibilities are offered to the user.

None of these two properties are mandatory. However, at least one of them must be specified. Typically, this means that one can specify: a value without attribute which means that the value is the metadata and it is not tagged; a value and an attribute which means that the attribute is associated the value; an attribute without value which means that the value associated to the attribute is empty. However, one cannot specify a metadata with no value and no attribute at the same time.

Metadata cannot be provided as is. They must be embedded inside a *NetInf:metalist*, i.e. a list of metadata. There are two properties associated to a NetInf:metalist:

- *meta_id* is the unique identifier of the metalist. This property is mandatory. However, its use is interesting at least for two main reasons: the first one is that it allows updating the metalist later and the second one is it allows including the list of metadata into another list (see below).
- *object_id* is the unique ID of the object the list of metadata is semantically describing. This ID is no mandatory in the metalist. If the object_id is provided, it means that the metalist describes the corresponding object. If the object_id is not provided, this means that the metalist is virtual (not associated to a specific object) and it is likely to be included into another metalist. Note that this does not mean that a metalist associated to an object via an object_id cannot be included into another metalist.

The *NetInf:include* element is used to include a list of metadata into another one. It is associated one and only one property:

- *meta_id* is the identifier of the metalist to include into the current one. This property is mandatory.

Allowing a metalist to include the list of metadata already associated to another one is a very interesting feature as it avoids redundancy and enhances coherence and consistency.

The NetInf:include element as described above allows the inclusion of metadata that are already in the metalist format. However, there may be other metadata available for an object that are described in another format. A typical example is the EXIF file for JPEG pictures taken with any modern digital camera. Almost any JPEG image includes an EXIF file that contains some extra information about the way the picture was taken (camera model, date and time, image resolution, aperture value, focal length, etc.). These metadata may be very useful for many photographers. Another typical example is the set of information associated to a file in a file system. Under the Unix file system, each file is associated a name, a size, a last modification date, a set of access rights, etc. These are also metadata that may be of interest to some users.

The *NetInf:extern* is the element that allows the inclusion of metadata provided in any other recognized format (see the description of NetInf:format below). It is associated up to three properties:

– *format_id* is a reference to the format entity (a script, a process, etc. – see below) that is able to translate the metadata from the format they are currently stored in to the metalist format. This property is mandatory.
– *object_id* is the ID of the Data Object where the metadata to translate are stored.
– *meta_id* is the ID of the Information Object which holds the ID of the Data Object where the metadata to translate are stored. Offering the possibility to access extern metadata using a meta_id is motivated by the fact that what is considered a metadata for a given object may represent or mean data for another one. With the EXIF file example, information in such a file may be data for an application (with an associated Information Object) while it can also be considered as metadata for another one (for an image typically with an Information Object associated with the image).

The last two properties are exclusive as external metadata cannot be located at two different locations. If this may occur, two extern inclusions have to be specified. If neither an object_id nor a meta_id is specified, the metadata in the original format are supposed to be located in the Data Object associated with the inner most object_id (as a NetInf:extern element may be set inside an included metalist and NetInf:include may be nested).

The last element to describe for our model is the *NetInf:format* that is used to specify the Data Object to be used to perform an automatic translation from a given metadata format to the metalist format. It is associated two properties:

– *format_id* is the unique ID associated to the format to allow future references from the metalists.
– *object_id* is the ID of the Data Object that is effectively performing the translation. This Data Object may be of any type and may be developed using any language. The only requirement is that it must conform to the specifications associated to automatic translators for the Network of Information, e.g. the Data Object containing the metadata to translate are provided on the standard input and the metadata in the metalist format provided by the automatic translator is generated on the standard output.

Note that there is no limitation on the number of NetInf:include, NetInf:metadata and NetInf:extern elements that can be provided in a NetInf:metalist and on the number of nested NetInf:include elements.

Our proposed model makes use of different identifiers and it is very important to make sure they are unique in the system. There are two possibilities to ensure their uniqueness: the first one consists in leaving the generation of IDs to the Network of Information; the second one consists in leaving the management of IDs to the users and checking the uniqueness every time an ID is provided (if not, the request is rejected). There is no real impact of the choice here on the model. At use, it may be interesting to leave the management of IDs to users with a control from NetInf as this would allow users to use more comprehensible IDs than what the Network of Information would generate automatically.

4 Some Examples

After the above formal description of the Metalist model, this section aims at providing some examples to highlight the use of the different elements. In this section, examples are presented using the XML language. This language was chosen as it is both widely used and based on a very simple hierarchical structure. However, any description language could be used.

Tag elements and properties used in the XML examples are exactly following the names presented in the formal description, especially regarding tag names that are all included inside the *NetInf* namespace.

For the first example, assume a user has gone to China and has taken pictures (s)he wants to publish on the Future Internet. The first picture to publish represents the Great Wall located close to Beijing. In order to ensure the picture is well referenced, the user sets the metadata as presented in Example 1.

```
<NetInf:metalist meta_id="mid1" object_id="oid1">
    <NetInf:metadata attribute="content">Holidays in China</NetInf:metadata>
    <NetInf:metadata attribute="content">From June 2nd to June 10th</NetInf:metadata>
    <NetInf:metadata attribute="content">The Great Wall</NetInf:metadata>
</NetInf:metalist>
```

Example 1. Simple “object description” using the Information Model

All three NetInf:metadata are provided as is inside a NetInf:metalist structure. In this example, all metadata are tagged with an attribute named *Content*. The ID associated to the metalist (*mid1*) is provided by property meta_id and the Data Object ID this Information Object is semantically describing is *oid1* provided by property object_id. The ID of the metalist (*mid1*) can then be used for later references.

It is very unlikely that visiting China the user takes only one picture. For example, (s)he may be willing to publish two pictures, the first one representing the Great Wall and the second one showing the Forbidden City. In this case, both pictures were taken during the same journey (from June 2nd to June 10th) and at the same occasion (during holiday in China). As a result, it is better to factorize this information, in order to avoid having it twice in the Network of Information.

```
<NetInf:metalist meta_id="mid1">
    <NetInf:metadata attribute="content">Holidays in China</NetInf:metadata>
    <NetInf:metadata attribute="content">From June 2nd to June 10th</NetInf:metadata>
</NetInf:metalist>

<NetInf:metalist meta_id="mid2" object_id="oid1">
    <NetInf:include meta_id="mid1" />
    <NetInf:metadata attribute="content">The Great Wall</NetInf:metadata>
</NetInf:metalist>

<NetInf:metalist meta_id="mid3" object_id="oid2">
    <NetInf:include meta_id="mid1" />
    <NetInf:metadata attribute="content">The Forbidden City</NetInf:metadata>
</NetInf:metalist>
```

Example 2. An example of inclusion

Example 2 presents a possible representation of the metadata in this case. First, common metadata are grouped into a metalist with ID equal to *mid1*. As no specific Data Object is associated to this metalist, no object_id is provided for this metalist. Then, each picture is associated a specific metalist with its specific metadata, i.e. *The Great Wall* on one hand and *The Forbidden City* on the other hand, and a reference to the common list of metadata is added using the NetInf:include element with property meta_id equal to *mid1*. Note that for both *mid2* and *mid3* metalist, an Data Object ID is provided.

The user can also be a professional photographer willing to provide his/her digital camera settings when taking the picture. These information are provided by most digital cameras and added to JPEG images using the EXIF format. As a result, data in EXIF format stored in the JPEG picture are also metadata for the picture and these are the metadata the user wants to associate to the picture.

```
<NetInf:format format_id="exif-2.2" object_id="exif-2.2tometalist" />

<NetInf:metalist meta_id="mid1" object_id="oid1">
    <NetInf:metadata attribute="content">Holidays in China</NetInf:metadata>
    <NetInf:metadata attribute="content">From June 2nd to June 10th</NetInf:metadata>
    <NetInf:metadata attribute="content">The Great Wall</NetInf:metadata>
    <NetInf:extern format_id="exif-2.2" />
</NetInf:metalist>
```

Example 3. Importing external metadata

Example 3 shows how this is made possible using the metalist model. First, the translator (the process, the script, the program, etc.) in charge of performing the translation from the EXIF version 2.2 format to the metalist format has to be declared. As long as this is public, this declaration has to be performed once in the Network of Information to be used by any user. For the sake of readability, it is assumed that the Data Object ID associated to the translator is *exif-2.2tometalist* (but this Data Object ID could also have been generated automatically by the storage system) and it is associated format_id *exif-2.2*. Then, every time a translation from the EXIF version 2.2 format has to be performed to generate a metalist, it just requires to be referenced using this format_id in the NetInf:extern tag. Note that no object_id property is provided with this tag. As a result, the inner most object_id in the set of inclusion has to be used to locate the EXIF metadata. This leads to object_id *oid1* which is the Data Object ID of the picture being semantically described. The other metadata set in this metalist are provided as a reference in order to show that NetInf:metadata and NetInf:extern can coexist in the same metalist, just like NetInf:metadata and NetInf:include do (see Example 2).

All examples presented above include very simple metadata, typically a single string of characters. However, these metadata may be far more complex. For example, the value of the metadata can be hierarchically structured using the XML language and/or using any other description language like RDF or OWL. It is then the responsibility of the user to ensure both the syntax and the semantic of the metadata is acceptable.

5 Conclusion and Future Works

This article described the Metalist model, a simple and flexible Information Model for the Future Internet. As presented above, the Metalist model includes three main features: 1) the ability to provide metadata in a simple and straightforward way; 2) the ability to include a metalist into another one, which allows to save memory space and ensures data consistency; 3) the ability to automatically import metadata available in another format to the metalist. All these features have been illustrated with short examples along a realistic use case.

Several functions and operations on information objects can be achieved via the Metalist model, including the ability to check if the metadata provided in a metalist comply with some given schema, the development of a security infrastructure for the management of Information Objects and Data Objects, the management of mobility, the improvement of search engines.

6 Disclaimer

This work has been supported by the IST 7th Framework Programme Integrated Project 4WARD, which is partially funded by the Commission of the European Union. The views expressed in this paper are solely those of the authors and do not necessarily represent the views of Institut Télécom (ex. GET-INT) or the respective projects and sponsors.

References

1. The FP7 4WARD Project, `http://www.4ward-project.eu/`
2. XMP Specification. Adobe Systems Incorporated (September 2005)
3. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: the Next Step for OWL. Semantic Web Challenge 6(4), 309–322 (2008)
4. Dannewitz, C., Pentikousis, K., Rembarz, R., Renault, E., Strandberg, O., Ubillos, J.: Scenarios and Research Issues for a Network of Information. In: MobiMedia 2008, Oulu, Finland (July 2008)
5. Hsu, I.-C., Tzeng, Y.K., Huang, D.C.: OWL-L: an OWL-Based Language for Web Resources Links. Computer Standards & Interfaces 31(4), 846–855 (2009)
6. IPTC-NAA Information Interchange Model, version 4.1. International Press and Telecommunications Council (July 1999)
7. Exchangeable Image File Format for Digital Still Cameras: Exif Version 2.2. Standard of Japan Electronics and Information Technology Industries Association (April 2002)
8. Klusch, M., Fries, B., Sycara, K.: OWLS-MX: a Hybrid Semantic Web Service Matchmaker for OWL-S Services. In: Web Semantics: Science, Services and Agents on the World Wide Web. Elsevier Science Publishers, Amsterdam (2008)
9. Manola, F., Miller, E.: RDF Primer. W3C Recommendation (February 2004)
10. Dean, M., Schreiber, G.: OWL Web Ontology Language Reference. W3C Recommendation (February 2004)
11. Weibel, S., Kunze, J., Lagoze, C., Wolf, M.: Dublin Core Metadata for Resource Discovery. RFC 2413 (September 1998)