Designing EPCIS through Linked Data and REST principles

Stefano Turchi¹, Lucia Ciofi¹, Federica Paganelli², Franco Pirri¹, Dino Giuli¹

¹Department of Electronics and Telecommunications, University of Firenze
²CNIT Unit of Research at Università di Firenze
Via S. Marta 3, Firenze, Italy
E-mail: {stefano.turchi, lucia.ciofi, federica.paganelli, franco.pirri, dino.giuli}@unifi.it

Abstract: The EPC Information Service (EPCIS) is a set of specifications defined by the EPC Global Network and aimed at enabling heterogeneous applications to share EPC-related data, both within and across enterprises. The standard specifies the EPCIS Capture and Query Interfaces by leveraging HTTP binding and WS* specifications. In this paper we investigate how some emerging Web of Data paradigms could be exploited in order to ease the collaboration and exchange of EPC-related data across enterprises. These paradigms, namely Linked Data and the REST architectural style, are relatively recent and, while both have roots on the Web, their mutual relation is still matter of study. Here we discuss how they can complement each other to ease the collaboration and information exchange across enterprises in the EPCIS domain. For such a purpose we refer to a framework, called InterDataNet (IDN) that we conceived and developed to ease the realization of the Web of Data vision by leveraging complementary aspects of Linked Data and RESTful services. We then discuss how IDN capabilities can be exploited to actually build collaborative web-based applications based on EPCIS-related data exchange and handling.

1. INTRODUCTION

The EPC Global Network specifications define an architecture of hardware and software components and interfaces to efficiently handle logistics processes through the use of the Electronic Product Code (EPC) and RFID technologies [1].

The EPC is a universal identifier used for physical objects. It can take the form of a Uniform Resource Identifier (URI), thus enabling information systems to refer to physical objects. The EPC code is typically stored on an RFID attached to the referred object.

Main components of the EPC Global Architecture include: the RFID Tags, the Readers, the EPC Middleware, the EPC Information Services (EPCIS), the Object Naming Service (ONS) and Discovery Services. The specifications define how Readers interrogate an RFID tag. The Middleware filters and processes data gathered by Reader components. Data are then stored in EPCIS repositories and made available to external clients via the EPCIS Query Interface. The ONS offers a name resolution service that translates an EPC code into the URLs pointing to the EPCIS repositories storing data about that EPC.

The EPCIS specifications define a distributed information system enabling the exchange of EPC-related data across several enterprises [2]. EPCIS handles two types of data:

- **Event Data**: events that occur during the execution of business and logistics processes. They can be acquired via the EPCIS Capture Interface and made available to external clients via the EPCIS Query Interface. Event data represent state changes of the entities of interest (e.g., “EPC X has been detected in Location L at 13:23 on March 15, 2012”).

- **Master Data**: data that define the context needed to correctly interpret acquired event data (e.g., “Location L refers to the Warehouse located in Marconi Street 10, Firenze, Italy”). They can be made available through the EPCIS Query Interface and acquired through custom interfaces.

The EPCIS specifications define an HTTP binding for the Capture Interface, while the ECPIS Query interface is defined via WS* standards (e.g., WSDL, SOAP) and AS2 via HTTP. WS* protocols represent reference specifications for enabling data exchange across technological and organizational boundaries in heavy and complex business applications [3]. Nonetheless, the REST architectural style and Linked Data, which are two paradigms characterizing the emerging interdisciplinary field of Web Science [4], are gaining increasing interest as potential technologies for promoting data exchange and interoperability within and across enterprises.

The REST (Representational State Transfer) architectural style was proposed by Roy Fielding in his doctoral dissertation [5] as an architectural style for building large-scale distributed hypermedia systems. It is based on the principles that make the Web scalable and provides a lightweight infrastructure for building RESTful web services.

The term “Linked Data” refers to a set of best practices for publishing and connecting structured data on the Web [6] via standard technologies (e.g. HTTP URIs [7], RDF [8]). Thanks to links connecting data from different sources, the Web is expected to evolve towards a global graph of data that can be navigated, queried, aggregated and fused.

In this paper we discuss how REST and Linked Data paradigms can be exploited in the design of EPCIS APIs and data model, respectively.

Through the adoption of such principles, we aim at providing developers with tools easing the building of...
lightweight Web-based applications for EPC-related data exchange and consumption. Our work is to contribute to such an objective through the exploitation of a framework, called InterDataNet (IDN) [9], that we conceived and implemented to ease the development of web-based applications by exploiting REST and Linked Data principles.

2. Background

In this paragraph we introduce main principles of Linked data and REST paradigms. We then discuss related work involving the adoption of these paradigms in enterprise scenarios and motivate the contribution of our work.

2.1 Linked Data

Linked Data is a new Web concept that promotes a paradigm shift in how information is modelled and accessed. Complex information may be built aggregating simpler information units, but unlike the current Web paradigm, which conceives complex information as a whole, the information units are individually addressable and linkable.

In 2006 T.B. Lee published the Linked Data principles [6]. These principles are best practices for publishing data “in a way that all published data become part of a single global data space:

1. Use URI as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using standards (RDF [8], SPARQL [10]).
4. Include links to other URIs, so that they can discover more things” [6].

In the first principle the term “things” is used, but “resources” would be more appropriate. Indeed, Uniform Resource Identifiers (URIs) [7] are the recommended identifiers. In the second principle, a constraint is put on the URI scheme, since HTTP URIs can be dereferenced (i.e. used as reference to retrieve the resource representation) by exploiting existing Web technologies. The third principle states that the information the URI refers to has to be meaningful and useful, suggesting for this purpose the use of RDF and SPARQL standard technologies. Finally, the fourth principle specifies that the resources have to be connected to other resources to support browsing and discovery.

The emerging Linked Data paradigm leads towards a global browsable information space where data from different information sources are connected and aggregated to form new information, opening up at the same time new possibilities for domain-specific applications [6].

2.2 REST

The main concept in REST concerns the “resource”, which is an abstract information entity. Within a network, clients and servers exchange information declaring resources into concrete representations and interact with them through a uniform interface. REST architectural style is not tightly bound to HTTP [11], even though HTTP is widely adopted for its implementation. The key principles of REST are the following:

1. Use URIs to identify resources. The resources are exposed by servers using URIs the clients invoke for interaction. URIs belong to a global addressing space and so resources identified with URIs have a global scope.
2. Adopt a uniform interface. The interaction with the resource is fully expressed with four primitives: create, read, update and delete. In HTTP they are mapped on the PUT, GET, POST and DELETE verbs. PUT creates a new resource, which can be deleted using DELETE. GET retrieves the current state of a resource in some representation. POST transfers the current state onto a resource [12].
3. Adopt self-descriptive messages. Each message contains the information required for its management. Metadata is used for content negotiation (i.e. negotiate the format of the representation), errors notification, etc.
4. Adopt stateless interactions. Each request from client to server must contain all of the information necessary to understand the request. Session state is kept by the client and no client session data are stored in the server [5]. Instead, the server manages and stored the state of the resources it exposes.

The adoption of REST over HTTP has a very low cost since it leverages well-known W3C/IETF standards and the required infrastructure (HTTP client and servers) is widespread. Moreover, the statelessness property makes a RESTful web server scale well with the number of clients, and performance optimization can be achieved thanks to the possibility of choosing lightweight message formats [12].

2.3 Motivation of our work

Linked Data principles have been applied in several projects promoted by public and academic communities [6] while Linked Data for industrial applications is a matter of ongoing research [13]. Servant in [14] argues that a company’s information systems can be envisioned as a space of Linked Data. Analogously, works in [15], [16] discuss the adoption of Linked data principles and reference technologies to enable data sharing and access to internal and external repositories across enterprises boundaries.

Logistics and supply chain scenarios that exploit RFID and EPC technologies pose challenging requirements for data exchange and interoperability. In this domain, Guinard et al.
[18] extend the existing EPCIS repository with a module offering a RESTful interface on top of standard WS* Query Interfaces. REST has also been applied in several works in the domain of the “Internet of Things” [17].

Our work is similar to [18], in that we aim at exploiting Web-derived paradigms for enabling EPCIS data sharing across enterprises. While [18] focuses on REST principles, our contribution aims at exploiting both REST and Linked Data principles. REST and Linked Data are two recent paradigms and while both have roots on the Web their mutual relation is a matter of study.

Though REST and Linked Data show similarities (e.g., the resource abstraction, the use of URLs for resource identification, and the need of representing relations between resources), they have different scopes: REST defines an API for programming applications based on the HATEOAS constraint, while Linked Data focuses on the definition of a distributed data model. We agree with the analysis carried out by Page et al. [19], who highlights how these scopes could complement each other in the design of domain-driven applications: on one side Linked Data proposes principles for defining a shared representation of data in a given application domain, on the other side REST defines a lightweight API for accessing, modifying and publishing such data according to the given application domain purposes.

In the following paragraph we introduce the InterDataNet Framework, which has been conceived by exploiting Linked Data and REST principles.

3 InterDataNet

InterDataNet (IDN) [9] is an open source framework offering capabilities for representing and managing information units and their structural and semantic relations on the Web, in a RESTful way. For the sake of conciseness, in this paragraph we provide a brief introduction of the IDN framework. Further details can be found in [9].

The main goal of IDN is to enable the easy exploitation and reuse of globally web-addressable information units to support collaboration around data. For this purpose, IDN considers documents as first class entities. A document can be defined as “a set of information pertaining to a topic, structured for human comprehension, represented by a variety of symbols…” [20]. Humans are thus very comfortable to conceive data aggregation in documental form. Therefore, in IDN, documents represent a structured aggregation of data that conveys some meaningful (and shared) information in a given application domain. In the following, we refer to a document in IDN as IDN-Document.

The IDN-Information Model defines the rules for data organization while building an IDN-Document. More precisely, an IDN-Document is a graph G = (V, E) where V is the set of vertices and E is the set of edges. The elements of V and E are the nodes containing the granular information (IDN-Nodes) and the relations between IDN-Nodes, respectively. IDN supports different types of relations between IDN-Nodes such as containment, reference, etc. Let D be an IDN-Document modeled as a graph G. Hence the topology of G expresses the IDN-IM used to represent D.

Through the IDN-IM it is possible to define an IDN-Document as an aggregation of data provided by different information sources. Indeed, an IDN-Node can be referred to by more than one IDN-Document, thus favoring reuse of information across different applications. This is possible since each IDN-Node is associated to the information provider that is authoritative for the information the IDN-Node refers to. Hence, gathering information from the proper sources enforces an appropriate responsibility distribution across the information providers, who are responsible for the quality of the provided information. In addition, IDN-IM can be extended with metadata enforcing privacy, licensing, security, provenance, consistency, versioning and availability properties attached to IDN-Nodes and affecting IDN-Documents [9]. Such features are crucial to support effective and trusted collaboration on real world scenarios.

IDN-Documents are exposed as resources through the IDN-Service Architecture (IDN-SA) API. The IDN-SA API is a set of generic REST interfaces for addressing, resolving and handling IDN-Documents and IDN-Nodes.

In the diagram, we show the main layers of the IDN-SA framework. The IDN-IM layer is the architecture that implements the services needed to enforce the IDN-Document’s properties and capabilities. IDN-SA has been designed with separation of concern and information hiding principles in mind and it is organized according to a layered architectural pattern. IDN-SA has four main layers, from the top to the bottom: Virtual Resource (VR), which provides RESTful APIs for accessing, creating, and modifying IDN-Documents; Information History (IH), which implements information versioning capabilities; Replica Management (RM), which guarantees resources availability through resource replication across distributed hosts; Storage Interface (SI), which offers persistence capabilities. In addition, we defined a set of horizontal services, including names resolution and security and privacy management.

By exploiting IDN-SA APIs (namely the Virtual Resource upper interface), it is possible to develop web-based
applications (called IDN-Compliant Applications, IDN-CA) that browse and handle graphs of distributed information units (i.e. IDN-Documents), through a uniform interface.

4 EPCIS service over IDN

In this paragraph we describe the use of IDN for implementing EPCIS specifications. We consider the following reference scenario: a product tagged with an EPC code is ready for the delivery in stock. When the product crosses a gate to start the delivery stage, an EPCIS ObjectEvent is generated. According to the EPCIS vocabulary, this phase is described as "shipping". A similar event is generated when the product is loaded onto a mean of transportation ("loading"), unloaded ("unloading") and delivered to the recipient ("receiving"). A logistic provider could supply his customers with a web application able to track the product through the different stages of transport (e.g. by showing both the planned and the tracked route on a map). Hence, the web application has to be able to access EPCIS Event and Master Data handled by different players involved in the process, such as the manufacturer, transport and terminal operators.

The solution hereafter proposed leverages two different elements:

- An IDN-IM designed to shape EPC related information (Event Data and Master Data) in the IDN infrastructure as an IDN-Document.
- An IDN-CA which offers both the RESTful Query and Capture interfaces and can be used as a basis for the development of the web application in the above mentioned reference scenario.

Hereafter we describe the design of an IDN-Document representing an ObjectEvent.

We consider an ObjectEvent as a complex resource made up of Event, EPC, Reader and Location resources. The ObjectEvent resource maps to the IDN-Document aggregating the EPC, Reader and Location IDN-Nodes as shown in Fig. 2. The bold edges directed from the Event IDN-Node to the Epc, Reader and Location IDN-Nodes represent a containment relation while the reversed red edges represent a reference relation.

The IDN-Document is identified by an HTTP URI and the same is for every IDN-Node. The URI templates for the resources URI design are listed hereafter:

- URI template for Event resources:
  \[ \text{http://authority/event/\{event_id\}} \]
- URI template for Reader resources:
  \[ \text{http://authority/reader/\{reader_id\}} \]
- URI template for Location resources:
  \[ \text{http://authority/location/\{location_id\}} \]
- URI template for EPC resources:
  \[ \text{http://authority/epc/\{epc_id\}} \]

where authority refers to the authoritative VR service for the IDN-Node addressed in the URI. Each player in the reference scenario can choose its own trusted VR service to be authoritative over the data produced within its domain.

By exploiting the capabilities offered by IDN to handle the IDN-Documents modeling EPCIS data, the IDN-CA can expose RESTful services compliant with the EPCIS Capture and Query interfaces. More specifically, the IDN-CA can handle IDN-Documents through the generic IDN-SA APIs exposed by the VR layer.

![Figure 2 – IDN-Document modeling an EPCIS Object Event](image)

4.1 RESTful Capture Interface

The standard EPCIS Capture Interface acquires EPCIS events to populate an Event Repository. The proposed RESTful Capture Interface accepts a POST request whose payload is the current standard XML representation of an ObjectEvent. This request triggers the creation of the corresponding IDN-Document in the IDN infrastructure. The POST response returns an HTTP URI referencing the newly created resource.

4.2 RESTful Query Interface

The standard EPCIS Query Interface allows querying an EPCIS Repository. Thus, the proposed RESTful Query Interface allows to easily query the IDN-based EPCIS repository. The query is submitted via a GET request on an HTTP-URI compliant with specific URI templates. Several URI templates have been designed to express EPCIS Simple Event Queries. Examples of queries that can be handled are listed in the following:

- which events are related to a given location (location_id) at time \( t \)?
  \[ \text{http://authority/location/location_id/event?time=t} \]
- which events are related to a given reader point (reader_id) at time \( t \)?
  \[ \text{http://authority/reader/reader_id/event?time=t} \]
- which events are related to a given EPC (epc_id) at time \( t \)?
  \[ \text{http://authority/epc/epc_id} \]
where authority refers to the IDN-CA. In brief the Query Interface allows browsing ObjectEvent IDN-Documents to solve a specific query. The result is an XML response message whose payload contains a list of EPCIS events instances that satisfy the query.

4.3 Use Case

To clear the things up, an example is presented. The ACME company is a neon lights producer willing to take advantage of the EPCIS system. When a goods item is produced it is tagged with an EPC code and then it is entered in the EPC global system. To do so, the ACME company can use an IDN-CA exposing the aforementioned Capture Interface and Query Interface and implementing the IDN-IM shown in Fig. 2. This ACME IDN-CA needs to be configured with the name of the one (or more) VR service(s) authoritative on the ACME EPCs (for instance, www.acme.com/VR). Finally the document is created on the IDN side through the HTTP request: PUT www.acme.com/VR/epc having as a payload the list of the EPCs currently available for the ACME company. Once the resource is created, every single EPC will be accessible at www.acme.com/VR/epc/{epc}. The insertion of a new EPC in the system will be performed by submitting the new “epc/{epc}” document to IDN and accordingly modifying the “epc” document. The PUT request is handled by the VR layer, which properly manages and processes the information graph and forwards processed data to the IH layer. The processing flow proceeds top-down until data are persisted. When the ACME IDN-CA asks IDN to retrieve the information, it leverages the interface exposed by IDN for graphs (documents) management. To serve the ACME IDN-CA request the flow through the IDN layers follows the inverse path.

Since Events, EPCs, Readers and locations are potentially managed by different players, the document will be multi-domains spread. Adding and interlinking new data produced by different providers can be handled through the reference and aggregation links provided by the IM.

For example, the insertion of an ObjectEvent representing the detection of an EPC in a given location can be handled by creating a new ObjectEvent resource which contains proper links to Location and EPC resources. The operation is performed in a globally-addressable resource space, thus easing the maintenance of system coherence.

4.4 Implementation

Current implementation stage consists of both a prototype implementation of IDN-SA and an in progress development of the IDN-CA. At present, the prototype implementation of IDN-SA includes the VR and SI layers. Both layers are exposed as independent RESTful services. The implementation is based on the Java language, the Spring Framework [21], and the Apache Tomcat container [22]. As mentioned in the use case, an IDN-CA has to know the names of the VR services which are authoritative for the IDN-Nodes handled by the application logic. The ONS service, as specified by EPC Global, returns the network endpoints that are authoritative for a given EPC. However, it does not handle other types of query parameters (e.g. location, event and reader identifiers). We thus implemented an “enhanced” discovery service by exploiting the Apache Solr search server [23].

5. CONCLUSIONS

In this paper we proposed an implementation of the EPCIS specifications based on the adoption of Linked Data and REST principles.

As a matter of fact, expected benefits of such principles include the development of scalable applications and easy sharing of data and models across the Web. We discussed the adoption of such principles in the EPCIS domain, with the objective of improving information sharing across organizational boundaries and easing the development of web-based applications exploiting data and events made available through the EPC Network.

These principles have been put into practice through the use of the InterDataNet (IDN) Framework. Coherently with the IDN design methodology, we discussed the design of an Information Model and RESTful services compliant with the EPCIS data model and interface specifications.

In the near future, we are extending the Information Model to represent Master Data and further types of EPCIS events and we are extending the prototype implementation to include subscription services. Finally, future work will include the design and development of a set of demo applications for validation purposes.

REFERENCES