
Franco Pirri, Michela Paolucci, Davide Chini, Maria Chiara Pettenati, Samuele Innocenti
Electronics and Telecommunications Department
University of Florence
franco.pirri@unifi.it, paolucci@dsi.unifi.it, davide.chini@unifi.it,
mariachiara.pettenati@unifi.it, samuele.innocenti@unifi.it

Abstract

At present e-Government procedures for creation and management of official documents pose specific problems with collaborative information authoring, data distribution and storage, use of a unique name for resource addressing, attribution of the entity responsible for the documents.

Many benefits can be obtained through the informatization of the procedures. To this end we propose InterDataNet (IDN) as a distributed framework that can solve many of the issues present in e-Government contexts through collaborative information management, data distribution and storage, workflow processes.

To evaluate the IDN approach validity, in this paper we describe the hypothetical scenario related to the management of Italian administrative acts.

1. Introduction

The effective and efficient use of telematic systems is a major concern for the development of e-Government services. Systems designed to this end, must account for operational processes involved and for technical, organizational or juridical constraints.

In fact Public Administrations (P.A.) have to follow specific rules such as the observance of law regarding data sensitivity, transparency obligation, etc. Moreover P.A. offices are widely distributed and have to work with many different legacy systems.

In this context, we propose InterDataNet (IDN) as an infrastructure oriented to support inter-organization interoperability and collaborative work.

The IDN fundamental idea is to view every document as composed of many pieces of information, and to give to each piece of information a personal independence from the whole document. So IDN-documents are defined through a set of information units and links interconnecting such units.

In spite of the independence of each piece of information (named “primitive information unit” in IDN) an immediate advantage of InterDataNet is the information reuse. Each primitive information unit can be managed by applications independently of its physical location. Therefore IDN can just-in-time map a data, stored in a legacy system, to a primitive information unit. This enables a native integration and reuse of the resources which are actually managed through legacy systems. In this paper we describe how InterDataNet can improve the creation, management, and adoption of Italian administrative acts.

However this is just an example that is used to test the suitability of the IDN approach which can be used in other e-Government contexts. IDN can solve a wide set of problems unlike other studies that concern only specific problems (e.g. [1], [2] are respectively related to the interoperability and to the document flow management) and it is more scalable than centralized solutions (like the scenario described in [3] that suggests a front-end service decentralization realized through a centralized back-office). More specifically, IDN is a distributed system which manages the documents versioning and workflow, allows the association between a document and the related entity responsible for it (that can be different from the author), provides Access-Control-lists management, etc.

This paper is organized as follows: section 2 describes some current approaches used in management of administrative documents; section 3 details the Italian acts requirements, taken as example while section 4 deals with general requirements for e-Gov documents; section 5 describes the IDN model framework; in section 6 the IDN Information Model is
described; section 7 expounds the IDN Service Architecture; section 8 contains the IDN APIs definition; section 9 applies the IDN approach to the managing of administrative acts.

2. Current approaches in administrative document management

At present, when a P.A. office has to create a new administrative act (Act yyy in Figure 1), it consults an old act (Act xxx), copies some pieces of information (inf 2), then rewrites and archives the new act. In this way even if the two documents contain the same piece of information (inf 2) there aren’t connections between them and the common information is repeated and stored twice. Managing the information shared by the two acts is not currently a straightforward task.

![Figure 1: Information reuse in absence of interoperability solutions.](image)

If we suppose the IDN adoption in the same scenario (see Figure 2), a P.A. officer can consult the IDN-old-act (IDN-Act xxx) to create the new ID-new-act (IDN-Act yyy). Now the officer can directly use the piece of information contained in the IDN-old-act (inf 2) and can create the new act as an IDN-document composed of some new information units (inf 3) and of a link to the old piece of information (inf 2). In this way “inf 2” is stored only one time and it is possible to manage and take advantage of the two acts “intersection”.

![Figure 2: Reuse with InterDataNet.](image)

However, the administrative act creation is only the final phase of the administrative process that includes many different operations. As we can see in Figure 3, a citizen (or a P.A. member) can request the opening of a new administrative process to the entitled office. The office receives the request and defines the set of documents to be created. The main documents which are usually created are: the administrative act proposal, which is a draft of the future administrative act, together with the administrative file, that contains the documents needed for the preliminary investigations (e.g. norms, fact, old acts, documents owned by the P.A. or by others external organizations, etc.) and from which the P.A. officers will start to produce the final act.

![Figure 3: Administrative Process.](image)

During the administrative process these three documents have to be handled by a group of authorized subjects (such as the citizen who has begun the process, one of the P.A. officers, an expert, etc.) and many different document versions are created. If P.A. offices will use IDN to store and manage this kind of process, the information reuse will not be the unique advantage.

In fact, in order to re-engineer such processes, an appropriate telematic infrastructure such as InterDataNet is needed.

3. Italian administrative acts

The Italian administrative acts are the final result of the administrative process. This process involves a series of both different entities (citizens, public officers, experts, etc.) operating in Public Administration and many documents (administrative act, administrative act proposal, administrative file).

The working phases, provided in the Italian administrative processes, are: creation and changing of the administrative act proposal, creation and changing of the administrative file, submission of these two objects to the competent authorities, act adoption (see Figure 3).

Once the administrative act proposal is approved by the authority, it becomes the administrative act. At this stage the act must comply with the following specific constraints [4] (act proposal must comply with the same constraints except with that of the “signature”):
4. Requirements for e-Gov documents

The administrative process must often guarantee the respect of a series of constraints. In order to introduce the specific problems, we hereafter detail some of the main requirements which must be addressed for the management of Italian e-Government documents [5]:

1. **indexing and cataloguing:** often Public Administrations must manage complex documentary flows indexing and cataloguing the information treated during the possible multiple steps of a given process;
2. **traceability:** in many cases the set of operations and entities involved in a resource lifecycle, need to be traced;
3. **responsibility:** correct management of various responsible entities (physical or juridical) involved in the procedures must often be guaranteed;
4. **persistent storage:** documents must be maintained in a persistent and safe way for given periods of time;

5. **appropriate management of sensitive data:** some information types (e.g. data related to users) must often be managed according to specific security and privacy laws;
6. **content, formal and structural constraints:** the creation and management of specific documents must be performed according to the characteristics of the documents themselves. This approach concerns both the contents and the formal and structural aspects;
7. **digital signature management:** the information management of official documents requires the association between the document and a unique identifier and the use of digital signature techniques to guarantee legal validity to the documents;
8. **heterogeneity and distribution:** the entities involved in e-Government processes are heterogeneous from different points of view. Many methodologies are adopted in the offices for the document management and the data storage, besides the public offices are often widely distributed.

In the following sections (5, 6, 7, 8) we will detail on the IDN architecture and functions. The use of the IDN conceptual framework in e-Government contexts provides a support for the creation and management of well-formed administrative documents.

5. IDN Framework

InterDataNet (IDN) is an architecture enabling the NETwork of INTERoperable DATA through a layered, service-oriented framework. IDN is expected to provide the models, the languages and the architectural solutions to comply with the following objectives:

1. providing a shared Information Model to enable Data interoperability;
2. providing a reference Service Architecture to handle the Information Model so as to: support global Data addressability and provide a set of collaboration-oriented services for distributed Data management to IDN-compliant applications.

At a general level, to tackle these objectives, we use the following set of conceptual and technological design patterns:

- the design of a layered framework following the Service Oriented Architecture (SOA) approach, [6]; this will allow the development of loosely coupled and interoperable services which can be combined into more complex systems. Layering allows the break-down of
system complexity exploiting information hiding and separation of concern [7], [8];
• the use of REST-style interfaces (Representational State Transfer) [9], [10] to enable resource addressability for performance, scalability and resource abstraction in distributed systems [11].

The IDN framework complies with these two goals through the ensemble of concepts, models and technologies pertaining to the following three entities (see Figure 4).

**IDN-IM (InterDataNet Information Model).** It is the shared information model representing a generic document model which is independent from specific contexts and technologies. It defines the requirements, desirable properties, principles and structure of the document to be managed by IDN. In the IDN-IM, information is highly structured and endowed of specific collaboration-oriented metadata.

**IDN-SA (InterDataNet Service Architecture).** It is the architectural model handling IDN-IM documents. The IDN-SA defines the reference functionalities, subsystems, protocols and interfaces for IDN document collaborative management. The IDN-SA exposes an IDN-API (Application Programming Interface) on top of which IDN-compliant Applications can be developed.

**IDN Applications.** IDN-compliant applications implement business logic using the IDN-API to exploit the IDN-SA services. IDN applications use the shared IDN-IM to represent and handle their pieces of information and documents. In section 9, we describe a possible IDN-Application in e-Government contexts.

### 6. IDN-IM

An Information Model (IM) can be defined as an abstraction and universal representation of the entities in a managed environment - their properties, operations and relationships. It is independent of any specific repository, application, protocol or platform [12]. According to this definition, the IDN Information Model is not concerned with details but it aims at capturing abstractions and fundamental requirements of the entities to be modeled. The adoption of an Information Model thus imply the capability to support many concrete Data Models. This capability enables scalability and adaptability of the model in different contexts.

#### 6.1. IDN-IM: basic concepts

Generic information modeled in IDN is formalized as an aggregation of elementary data units, named “primitive information units” (see Figure 5). Each primitive information unit contains generic data and metadata; at an abstract level, a primitive information unit is a node in a connected directed acyclic graph (DAG), [13]. It is worth recalling that a tree structure is a specific case of a DAG in which each node has at most one parent. DAG structures are therefore capable of modeling all tree-structured resources. As shown in Figure 5, on the left side, a primitive information unit contains a set of data and metadata. All data and metadata are handled by the Service Architecture. Among those, some may be visible also to applications (“application properties”) while others are exclusively handled by the Service Architecture (“SA-properties”). Each node is addressed by one or more URIs related to one or more names, as it will be explained in section 6.3. DAG nodes are related to each other through directed links connecting two nodes. As a result of the ordering and the absence of cycles, if there is a path between two nodes one is the predecessor and the other one is the successor. In the special case of two directly connected nodes, one of them is the parent while the other one is the child. An IDN-document structures information units and it is composed by nodes related to each other. IDN-documents can be inter-linked, so two types of links are needed and defined in the Information Model:

- **aggregation links:** to express relations among nodes inside the IDN-document;
- **reference links:** to express relations between distinct IDN-documents.

A generic IDN-document has its root in one node and it is composed of a set of nodes linked to each other through “aggregation links”. IDN-compliant applications can extend the Information Model defining a third type of link named **IDN-Application-level links.** Using such links Applications can handle

---

1 Compared to Information Models, Data Models define managed objects at a lower level of abstraction. They include implementation- and protocol-specific details, e.g. rules that explain how to map managed objects onto lower-level protocol constructs [12].
cross references between documents and can introduce cycles in the graph, if necessary. Such link types are also stored into nodes and handled by IDN-SA in the same way as other IDN Application defined properties.

Figure 5: Example of IDN primitive information units and IDN documents.

In the right-side of Figure 5 two IDN-documents are displayed. “Doc1” contains four nodes and four aggregation links (“1”, “2”, “3” and “4”). “Doc2” contains three nodes and two aggregation links (both named “6”). “Doc1” is defined starting from its root (“Doc 1 root”) following the links named “1” and “2” and then, “3” and “4”. “Doc2” is defined starting from its root (“Doc 2 root”) following the links named “6”. “Doc1” contains a reference link to “Doc2” named “5”. Each node has several identifiers. Each identifier depends on the position of the node in the IDN-DAG. Each node contains a property named “nodeType”. The “nodeType” property embeds the primitive information unit schema defining the application-level properties contained in the node. Aggregation and reference links properties are also defined through nodeTypes, which also concur to define the IDN DAG structure. For example, in Figure 5, nodeType “A” constrains the node to have two children with nodeType “B” and “C” respectively referred to by aggregation links.

6.2. IDN-IM: advanced features

IDN-IM advanced features are supported by engineering the IDN-IM basic concepts illustrated in section 6.1. Advanced features provide services such as: management of the entity responsible for primitive-information-units, ACL (Access Control List), versioning and workflow. Each primitive information unit contains specific metadata that are defined to associate the author to such unit. In specific contexts, such as e-Government applications, an entity responsible for a primitive information unit must be handled. In IDN-IM this entity is a physical or juridical person that is entitled to guarantee the correctness of node contents as well as the validity of structural relations between nodes. The entity responsible for a primitive information unit is not necessarily its author. Access rights are granted through ACL (Access Control List) which manage user access rights to a particular node. Each primitive information unit can contain specific data that are used to check if the user has the rights to perform the requested operation (e.g. get a primitive information unit, update a primitive information unit, etc.). In order to guarantee information unit changes traceability, IDN-IM has a built-in versioning system. It is based on UEVM (Unified Extensional Versioning Model, [14]) and allows changes traceability of both information unit contents and IDN-DAG structure. Moreover IDN-IM provides a way to manage primitive information units and IDN-document lifecycles. IDN applications can exploit such methods to implement application-dependent IDN document workflow management.

6.3. IDN-IM: primitive information unit names

Welcoming Ayers’ perspective on “viewing the link as a unit of data” ([15]), we hereafter detail the Information Model as regards the naming conventions we affect to IDN document nodes. At the Application level, IDN-IM node names are hierarchical and each name reflects the IDN-DAG structure. For example in Figure 6, node “path/0/1/3” (it is the node with type “D” in “Doc1”) has this name because it is a child of node “path/0/1” through a link named “3”. For the same reason it has the name “path/0/2/4” because it is also reachable from node “path/0/2/4” through the link named “4”. In other words each name has the following structure:

“node_parent_name” + “/” + “link_name”

Where:
- “node_parent_name” is the node parent name;
- “/” is the concatenation operator between strings;
- “link_name” is the link name.

“Special nodes” are also defined in the IDN-DAG. These nodes are directly addressed regardless of the DAG hierarchical structure through a name referred to as “namespace”, [16]. Such names are a “local root” for the naming system: starting from these nodes the names of their successors are expressed. For example (see Figure 6) if we consider the node “path/0/0” (it is the node with type “A” in “Doc1”) as a local root, its successors have the following names: “/1”, “/2/1”, “1/3” and “2/4”, for nodes in “Doc1”; “/5/”, “/5/6[0]”, “/5/6[1]” for nodes in “Doc2”.

So, each node name must be related to the root from which the node name itself is expressed. In other words, the absolute node name (that allows
unambiguous node identification) is composed of two parts: the “namespace” and the “relative path” in the DAG hierarchical structure.

![Diagram](image1.png)

**Figure 6: Example of IDN-IM names.**

As we can see in Figure 6, same name siblings are allowed in this model and can be addressed using the convention defined in XPath, [17]: they are distinguished through “[integer_index]” at the end of the name (in the last example same name siblings are identified as: “/5/6[0]”, “/5/6[1]”).

So far, we have described logical names which are used at the IDN application level to identify the IDN documents and the primitive information units.

When it comes to implementation, starting from these logical names, methods to physically locate and access the specific resource are needed.

To this end, IDN architecture envisages a three layers naming space (see Figure 7):

- **Human Friendly Names (HFNs),** [18], are flexible and descriptive names easily understandable by people. These are the logical names described in section 6.3;
- **Uniform Resource Names (URNs),** [19], [20], [21], are defined in the IDN framework and are used to unambiguously, univocally and persistently identify the resources independently of their physical locations;
- **Uniform Resource Locators (URLs),** [22], are used to identify resource replicas as well as to access them.

![Diagram](image2.png)

**Figure 7: Three layers IDN naming system.**

The IDN framework automatically manages URNs and URLs and hides them to IDN applications. Since any resource can be replicated in several locations, each URN can correspond to many URLs (see Figure 7). Resolution processes are required to access a resource starting from its HFN. As all names are subclasses of the URI name category, they are hierarchical and their direct and inverse resolution is possible using DNS-like (Domain Name System) systems, [23], [24].

In IDN architecture a Logical Domain Name System (LDNS) is envisaged to resolve HFN names into a URN name. A Localization Service (LS) is instead devoted to the resolution of a URN name into URL addresses (see Figure 7). The DNS-like approach to resolution allows scalability of the system as well as hierarchical and local delegation of responsibility in the collaborative creation management and access to data thus enabling the global addressability of primitive information units.

7. **IDN-SA**

   The IDN-Service Architecture (IDN-SA) provides for an effective and efficient IDN-IM implementation. IDN-SA is a layered service oriented architecture and it is composed of four layers (see Figure 8, from bottom to top): Storage Interface Layer; Replica Management Layer; Information History Layer; Virtual Repository Layer.

   IDN-Compliant Applications are built on top of the Virtual Repository Layer exposing the IDN APIs. IDN-SA layers functions are hereafter briefly specified, starting from the bottom of the stack.

   ![Diagram](image3.png)

   **Figure 8: IDN-SA layers.**

   **Storage Interface Layer (SI);** it provides the services related to the permanent and distributed storage of the data. Storage systems often depend on the specific (legacy) platform (hardware and software)
for which a reference standard or model is not a-priori defined. The SI layer provides a REST-like uniform view over distributed data independently from their location and physical storage platform. This layer provides physical addressability to resources through URLs addresses. Basic SI functions provided to the upper layer are: access, creation, modification, deletion of URL-addressed resources.

Replica Management Layer (RM); this layer provides a delocalized view of the resources to the upper layer offering URN to URL address resolution (see Figure 7) through the LS (Localization Service). This has two consequences: one is that the upper layer can handle only univocal and persistent resource identifiers (URNs); the other is that it allows the association of several physical resources to the same identifier. This layer is charged with treating this set of physical resources as replicas of the same logical information providing replica updating and synchronization. In principle the creation of a replica can be done with any criteria that can be defined through IDN-SA APIs by IDN compliant applications.

Information History Layer (IH); this layer manages primitive information unit history. Navigation through the versions of a non-structured information unit is consequently allowed as a result of the services provided by this layer. IH can also maintain multiple development branches simultaneously; branching and merging functionalities are therefore exposed to the upper layer. This layer is also devoted to manage locking functions on the information units which are collaboratively accessed and edited.

Virtual Repository Layer (VR); it exposes the IDN APIs (described in section 8) to the IDN-compliant Applications exploiting lower layers services. VR provides the maximum abstraction of structured information to the application. Indeed, this layer is seen by the application as the container-repository of all primitive information units. The resolution of logical resources names (HFNs) into unique identifiers (URNs) (see also Figure 7) is achieved at this level through LDNS (Logical Domain Name System).

8. About the IDN APIs

The IDN-IM Information Model allows the definition of a set of operations that are used by the Applications to interact with the IDN-documents and their primitive information units. Such conceptual operations are achieved through the definition of the IDN-SA APIs (Application Programming Interface) and its consequent implementation. IDN-IM API functions include:

- **basic functions**: access, create, modify and remove nodes and documents; add/remove types to node, define node types, check types constraints;
- **identity management functions**: user login/logout, add/remove users, access rights management;
- **naming and navigation functions**: add/remove names, browse and search nodes;
- **replication functions**: enable, disable, configure primitive information units replication policies;
- **versioning management functions**: branch, merge and history browsing;
- **workflow management functions**: define node states and lifecycle, define state transitions, execute transitions;
- **notification management functions**: notify to registered applications events related to changes in nodes properties and/or states;
- **concurrency management functions**: locking and unlocking nodes.

9. Applying IDN approach to manage administrative processes

This section describes how the issues introduced in section 4 can be handled using an IDN-Application (section 5) and how the administrative process can be modeled as an IDN-document. The IDN-Application treats the administrative process and the related entities as IDN-documents. This eases the complex management of workflow transitions and can solve the problems of indexing and cataloguing as well as the problem of traceability. Each document or phase involved in the administrative process can be treated as a IDN-document (a single primitive information unit or a structured set of them). So the ACL management and the possibility of assigning an entity responsible for the units (see section 6.2) provide a solution to the problem of responsibility and the problem of managing sensitive data. As stated in section 6.1, the IDN Information Model, which is used to define the IDN-documents, is a DAG (directed acyclic graph). Recalling that a tree structure is a specific case of a DAG, if an IDN-document is a tree (such as the example in Figure 9) then it can be described with an XML-Schema, [25]. To realize an XML-Schema starting from an IDN-document it is necessary to know that: IDN-Types correspond to the XML-Schema Types (simple or
IDN-link names correspond to XML-Schema element names.

Versioning management is another important service exposed by IDN framework and allows us to model the document changes during the administrative process.

In order to explain the IDN versioning criteria, in Figure 10 we can see that a change in the primitive information unit called “path/admin_act_proposal/preamble/phases[1]/law_element” creates new versions (v2) of the following primitive information units:

- “path/admin_act_proposal/preamble/phases[1]/law_element” (the directly changed information unit);
- “path/admin_act_proposal/preamble/phases[1]”;
- “path/admin_act_proposal/preamble”;
- “path/admin_act_proposal”.

The new versions are propagated from a child to its parents (or to its parent if the structure is a tree) according to UEVM (Unified Extensional Versioning Model, [14]).

InterDataNet also manages the change of the administrative act proposal to the administrative act.

In IDN when the authority notifies the system to subscribe the act, the system creates a new IDN-document named “path/admin_act” (see Figure 9) which represents the administrative act and has the following features:

- it is linked to the last version of the administrative act proposal;
- it is connected to each information unit contained into the proposal;
- it is linked to a new primitive information unit containing the digital signature of the authority who has subscribed the act and can't be modified, [5].

Moreover IDN provides a three layer naming system and a unique interface (section 6.3 and section 8) for heterogeneous and distributed systems, thus resolving the heterogeneity and distribution problem which is critical in the P.A. offices. These offices are usually widely distributed. As a result of specific replication policies and the three level naming system presence (see section 6.3) each document can be archived and stored according to its specific requirements. This solves the persistent storage problem. As for the administrative process and the three documents created during it (administrative act, administrative act proposal and administrative file), they are treated in IDN as structured documents. In fact, as we can see in Figure 9, each of them has a related Type (the nodeType defined in section 6.1). Moreover, each link has a name on it which represents the primitive information unit role. For example, the administrative act proposal and the administrative act are created through the same IDN-Type (Admin. Act) because they have to respect the same requirements described in section 3 but their names are not the same (admin_act_proposal; admin_act) to indicate their different role into the process.

So an information unit name (at the IDN application level, see section 6.3) is to indicate the semantic...
meaning (role) of the unit itself. This enables information reuse. In Figure 9 the primitive information unit with Type “T: Document” (located in the lower right side of the figure) has two names:

- “path/admin_act_proposal/preamble/phases[1]/law_element”;
- “path/admin_file/documents”.

These two names are present because the same information unit (stored in IDN only once) is used two times in different contexts.

This solves the problem of digital signature management thus assigning juridical value to documents.

10. Conclusion

In this paper we describe how InterDataNet (IDN) can improve the creation, management and adoption of Italian administrative acts. However this is just an example which is used to test the suitability of the IDN approach. IDN can handle Public Administrations (P.A.) official documents to match requirements for interoperability and collaboration functions in a distributed environment. Each P.A. office works on multiple pieces of information, many of which are owned by the office itself, others are owned by other P.A. offices and the remaining resources are derived from other external organizations. Using IDN (sharing the same Information Model and exploiting IDN Service Architecture services), each P.A. office can work as if the resources (information units) were owned by the office itself. This collaborative approach permits information reuse. If P.A. offices will use IDN for official document management, the information reuse will not be the unique advantage. In fact IDN satisfies other P.A. requirements such as information cataloguing, information traceability, information responsible entity management, structured information management, digital signature use. These requirements are matched exploiting IDN features as a result of both ACL, workflow, versioning, responsible entity management and the fact that IDN is a distributed system. At present the IDN framework is being implemented as a prototype. Other case studies related to IDN framework concern the use of IDN as a data repository for semantic web applications and for collaborative infrastructure in a multimedia content production and sharing platform.

11. References


