The Four Key Differences in the RDF and XDI Graph Models—and How They Solve Four Key Problems in Distributed Data Sharing

Drummond Reed, Co-Chair, OASIS XDI Technical Committee
15 January 2015

Introduction

RDF (Resource Description Framework) is the W3C standard for the Semantic Web. It is based on a core subject-predicate-object graph model.[1] RDF and JSON are the key technologies behind Linked Data, which is gaining momentum as a standard for sharing semantic data across systems.

XDI (Extensible Data Interchange) is an emerging standard for semantic data interchange from OASIS. Since XDI starts from the same fundamental subject-predicate-object model as RDF, it can be viewed as an ontology for RDF.[2]

However, by imposing several constraints and adding several features to the RDF graph model, XDI defines an enhanced graph model that has its own JSON serialization format, exchange protocol, and ontology language[3] that are optimized for semantic data interchange.

This document describes the four key differences between the RDF and XDI graph models and explains how each of these differences is motivated by the need to solve a key problem in distributed data sharing. Each is described in its own section:

1. **Addressability:** Creating a Global Data Grid
2. **Persistent Identification:** Strongly Binging Identity to Data
3. **Abstract Identification:** Cleanly Separating Identity and Representation
4. **Context:** Modeling Structure Itself

It concludes by summarizing the benefits that these features of the XDI graph model can provide in four specific branches of the distributed data sharing problem space.

---

[1] See http://en.wikipedia.org/wiki/Triplestore

[2] An ontology is a formal, machine-readable set of rules that define the entities and relationships that exist within a domain, e.g., medicine, education, law, biology, government, sports, etc. The OASIS XDI Technical Committee has not yet tried to define XDI as a formal ontology for RDF, however it anticipates undertaking that effort once the XDI 1.0 specifications are complete.

[3] In XDI, ontologies are called *dictionaries* because of the way they encourage reuse of semantics across domains just like the words in a human language dictionary. See more about this XDI feature in the *Context* section of this document.

Addressability: Creating a Global Data Grid

The first key difference between the RDF and XDI graph models is easy to describe:

> *The RDF graph model does not require that all nodes in all RDF graphs be globally uniquely addressable. This is a requirement of the XDI graph model.*

The Problem in Distributed Data Sharing

Global addressability is as important to distributed data systems as it is to the Internet itself. Without the ability to uniquely and precisely address each node of data across a global data "grid",[4] the authority, rules, and policies applying to that node of data cannot be clearly expressed and enforced.

Why the RDF Graph Model Does Not Solve the Problem

When the RDF graph model was first developed over a decade ago, its purpose was to describe resources on the World Wide Web. Those resources already had globally unique identifiers called URIs (Uniform Resource Identifiers)—since upgraded to IRIs (Internationalized Resource Identifiers).

However unique addressability was *not* a requirement for the other nodes inside an RDF graph. For example:

1. **RDF literal nodes**—the nodes containing the actual data describing an RDF subject—do not have a unique address within an RDF graph. For example, if an RDF subject node representing a person has three literal nodes representing phone numbers, all three will be described with the same predicate IRI. So there is no means (other than using the data values themselves) to uniquely address these data nodes.[5]
2. **RDF blank nodes** are a special kind of RDF node that technically has no address outside the scope of a single RDF graph. This means RDF graphs containing blank nodes cannot be merged and still maintain the same address for each blank node.

Furthermore, until RDF 1.1, **an RDF graph itself did not have a unique address** (i.e., an IRI that identifies the graph as a whole). With RDF 1.1, this problem was solved by formally defining RDF datasets in which each RDF graph can have its own unique IRI, making it a "named graph". Note, however, that this solution exacerbates the *Abstract Identification* problem described below.

Since the RDF graph model was not designed to provide unique global addressability, it is not surprising that **RDF does not have an addressing syntax.**

[4] The term "grid" used here represents any set of federated namespaces that form a unique addressing tree. The Internet itself is a grid based on IP (Internet Protocol) addresses. The DNS (Domain Name System) is a grid based on domain names.

[5] Trying to use the data values themselves to provide unique addressability of literal nodes has many other problems—for example the data values may be encrypted for security reasons.

While an IRI may be used to identify an RDF graph, and an IRI may be used to identify a subject node within that graph, there is no defined syntax for:

1. Addressing a particular subject node inside a particular RDF graph.
2. Addressing a literal node connected to an RDF subject node (e.g., a phone number for a person).
3. Addressing a blank node.

Lastly, because the RDF graph model does include the concept of context, there is no syntax for addressing contextual relationships between nodes. See the *Context* section below.

How the XDI Graph Model Solves the Problem

Since XDI started in the distributed data sharing problem space, it has always had the requirement of global unique addressability of every graph node. This requirement has driven the evolution of the XDI graph model since the founding of the XDI TC at OASIS in 2004. It is the foundation for several key features of XDI:

1. **Discoverability**—the ability to traverse a set of XDI graphs to discover the location of other XDI graphs.
2. **Portability**—the ability to move an XDI graph from one location on a network to another without changing the semantics.
3. **Authorization**—the ability for an XDI link contract[6] to precisely identify the data it controls.

To provide global unique addressability, the XDI graph model includes a standard addressing syntax for all types of XDI graph nodes:[7]

1. **Root nodes** that identify an entire XDI graph.
2. **Entity nodes**—the equivalent of RDF subjects.
3. **Attribute nodes**—the equivalent of RDF predicates that describe a literal node.
4. **Literal nodes**—the equivalent of RDF literal nodes.

This addressing syntax also supports addressing contextual relationships between these nodes (covered in the *Context* section, below). Lastly the XDI graph model has **no blank nodes**—to avoid the addressability and persistence issues they raise.

[6] A link contract is created between two XDI authorities just like a real-world contract is created between two parties. As the primary control structure in an XDI graph, a link contract grants permissions from the *authorizing authority* (the party controling the data) to the *requesting authority* (the party requesting access to the data). Permissions can be specific operations (read, write, delete); can be subject to specific policies, and can apply to specific data in the graph.
[7] The acronym for these four node types is REAL.

Persistent Identification: Strongly Binding Identity to Data

The second key difference between the RDF and XDI graph models is:

> *The RDF graph model does not distinguish persistent identifiers for graph nodes. This is a requirement of the XDI graph model.*

The Problem in Distributed Data Sharing

In a globally distributed data sharing system—one that supports portability of semantic data graphs between hosts—both the authorities for the data graphs and the nodes in the data graphs must have persistent identifiers (identifiers that will never change and never be reassigned) or else the security and privacy model has a glaring weakness for attackers.

This weakness is easy to describe: if the address for a data authority is re-assignable (for example, if the identifier for Alice can be reassigned to Bob), then the new assignee can take over the identity and data graphs controlled by the original authority.[8] Similarly, if the address of one data node can be reassigned to a different data node, then permissions assigned to the original data node using that address will now apply to the new data node.

Why the RDF Graph Model Does Not Solve the Problem

Since the RDF graph model uses IRIs to identify resources, it is possible for RDF graphs to use IRI schemes, such as the **uuid:** scheme for universally unique identifiers, that support persistent identification.[9] However the use of such identifiers is opaque to normal RDF graph processing. So there is no standard way in RDF to signal if the IRI assigned to a particular resource is persistent.

In addition, RDF blank nodes are **by definition not persistently identifiable** across multiple RDF graphs.

How the XDI Graph Model Solves the Problem

First, as discussed above, the XDI graph model does not have blank nodes, so all XDI graph nodes are 100% addressable.

Secondly, XDI addressing syntax includes explicit syntax for persistent identifiers. For example, the following XDI address for a phone number includes a persistent address (a UUID) for a person (the part beginning with `[=]!`) and a persistent address (also a UUID) for the phone number (the part beginning with `[#tel]!`).

> [=]!:uuid:f81d4fae-7dec-11d0-a765-00a0c91e0001[#tel]!:uuid:9ce739f0-7665-11e2-bcfd-0800200c0002/&/"+1-206-555-1212"

This explicit syntax enables an XDI processor to verify that every XDI address used within a security context (such as an XDI link contract) consists of only persistent

[8] In the OpenID community, this problem became known as "OpenID recycling". See https://developer.yahoo.com/openid/faq.html.
[9] A UUID can be generated and assigned without the use of any central registration authority.

identifiers (and issue a warning if this is not the case). Note that the actual persistence of such identifiers may only be verified by conformance to operational policies, which is out-of-scope for the XDI technical specifications (but in-scope for other mechanisms such as digital trust frameworks).

Abstract Identification: Cleanly Separating Identity and Representation

The third key difference between the RDF and XDI graph models is their approach to abstract identification, i.e., how to identify the entities described in a data graph.

> *In the RDF graph model, entities are identified with IRIs, which causes semantic ambiguity between the abstract identity of a resource and the address of a concrete representation on the Web. In the XDI graph model, entities are identified with XDI addresses, which are always the abstract identity of a resource and never the address of a concrete representation on the Web.*

The Problem in Distributed Data Sharing

While this might seem the most obtuse of the four problems, it is closely entwined with the other three. The problem arises in the need to distinguish between the *identity* of an entity being described by a semantic data graph, such as a person or a company, and a *representation* of that entity that exists itself as a resource on the Web, such as a web page or an image or a PDF file.

A simple example is an IRI describing a person. For example:

> *http://forestgump.com/me.gif*

The question is: does this IRI identify Forest Gump, the person? Or does it identify a picture of Forest Gump? Forest Gump, the person, cannot be "retrieved" over the Web. But the image of Forest Gump can be retrieved and viewed in a browser.

This distinction is important for three reasons:

1. **Both the abstract entity and the representation of the entity may need their own semantic descriptions.** If it is not clear whether the IRI above represents a person or a picture of a person, then how do you describe the birthdate of the person vs. the date that the picture was posted to the Web?
2. **Applications need to know the difference between an abstract entity that cannot be retrieved and a representation that can be retrieved.** There is no use in attempting to retrieve a resource that is not actually represented on the Web.
3. **Abstract identification can be portable across multiple locations; concrete representations cannot**. This is why *uniform abstraction*—abstract identification that works the same way across all systems—is as crucial to identity and data portability as persistent identification.

Why the RDF Graph Model Does Not Solve the Problem

In the RDF graph model, all entities being described are identified with IRIs. Because a IRI *may* be the address of a retrievable resource on the Web, it is semantically ambiguous as to whether the IRI identifies an abstract entity (such as a person, organization, or concept) or whether it identifies a retrievable resource (such as a web page or RDF graph).

This problem is so well-known in RDF circles that it has its own name—the *HTTPRange-14 problem*—with its own Wikipedia page.[10]

The solutions proposed by the W3C RDF Working Group fall into two categories:

1. **Use a special convention for IRI addresses**: end them with a # fragment identifier if the IRI identifies an abstract resource, otherwise end them with a forward slash.
2. **Attempt to retrieve the resource at the IRI** and determine whether it is abstract or not based on the HTTP response code returned.

The problem with the first solution is: a) it breaks the opacity of RDF IRIs, and b) a fragment identifier may in fact represent an actual fragment of a Web resource, such as a portion of an HTML document.

The problem with the second solution is that: a) it places semantics outside the RDF graph itself, and b) it requires external processing (and network availability and bandwidth) that may not be available to an inference engine or other graph processor.

How the XDI Graph Model Solves the Problem

The XDI graph model has a very simple solution to this problem: all XDI graph nodes are identified with XDI addresses that are **always** abstract and **never** the address of a retrievable resource on the Web. If the entity being described has one or more representations available for retrieval on the Web—and if the IRIs for those representation(s) are stored in that XDI graph—then those IRIs may be discovered by making an XDI query.[11] The process of looking up the IRI(s) associated with an XDI resource is known as *XDI discovery*.[12]

By making all XDI addresses abstract, XDI provides a clean distinction between identification and representation that does not depend on IRI syntax or information available outside an XDI graph. And this uniform abstraction means XDI graphs can be fully portable and capable of describing long-lived persistent resources.

---

[10] The name "HTTPRange-14" comes from the number assigned to this issue by the W3C Technical Architecture Group (TAG).

[11] This assumes that the discoverable IRI data is public. By default all XDI resources are private, and access is granted via one or more XDI link contracts. This form of access control can also be applied to XDI discovery.

[12] The XDI Discovery protocol is one of the specifications in the XDI 1.0 specification suite.

Context: Modeling Structure Itself

The fourth key difference between the RDF and XDI graph models may ultimately be the most important:

> *The RDF graph model does not provide a standard way to model contextual relationships. This is a requirement of the XDI graph model, and deeply affects every aspect of XDI graph structure and ontology construction.*

The Problem in Distributed Data Sharing

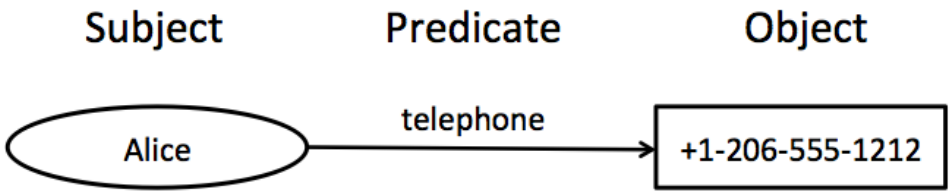
The term "context" in English has a very general meaning:

> The surroundings, circumstances, environment, background or settings that determine, specify, or clarify the meaning of a word, phrase, or event.

In distributed data sharing—and more precisely in semantic graph models—we can give "context" a much more precise meaning:

> The set of graph nodes (metadata) that determine, specify, or clarify the meaning of another graph node (data or metadata).

In fact, the concept of context is inherent in the very definition of "metadata" as "data that describes other data". The purpose of metadata—the description—is to provide context for understanding the data. For example, the following RDF subject-predicate-object triple[13] uses the subject "Alice" and the predicate "telephone" to provide the context for understanding the raw data "+1-206-555-1212"



The concept of context actually applies at an even more basic level. It is the essence of any directed graph, the fundamental structure underlying both RDF and XDI graphs. The subject-predicate-object model itself includes the concept of context:

- Every subject provides the context for a predicate.
- Every predicate provides the context for an object.

In fact contextual relationships are the atomic building blocks for describing structure of any kind—hierarchy, polyarchy, composition, aggregation, etc. And structure is the essence of structured data. So it is vital that a distributed data sharing model be able to model context in a standard way.

[13] Note that for purposes of illustration, the identifiers used in these examples are English language words instead of IRIs.

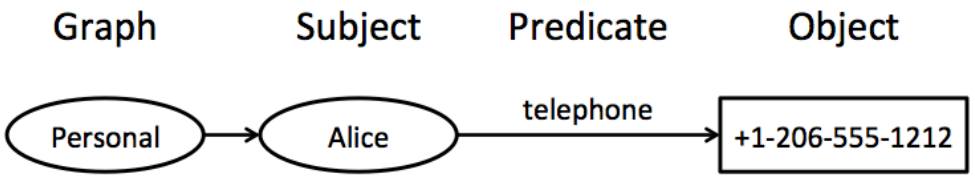
Why the RDF Graph Model Does Not Solve the Problem

The RDF subject-predicate-object triples model standardizes *two levels of context*—a subject and a predicate—for describing a literal data node. This core model is the basis for the entire RDF and OWL family of standards for the Semantic Web.

Eventually the RDF community realized it needed at least one more level of context to express *the context of an RDF graph itself*. In other words, if an RDF graph is a collection of triples (called *statements*), then how do you distinguish the statements in one RDF graph from the statements in another RDF graph?[14] And how can you reference an entire RDF graph as a resource on the Web?

The solution the W3C RDF Working Group incorporated into RDF 1.1 is called *named graphs*. A named graph is a RDF graph (a collection of RDF statements) that has its own IRI. A collection of named graphs is called an *RDF dataset*.

A named graph provides *exactly one more level of context* to uniquely describe an entire RDF graph, so that each triple in the RDF graph now becomes a *quad*.[15]



However because there is no formal RDF predicate defined for the relationship between the graph name ("Personal") and the subject(s) within the graph (e.g., "Alice"), there is still considerable debate within the RDF community about the semantic meaning of this new level of context.[16]

This begs a larger question: why should context—as the most fundamental type of relationship defining structure—be limited to just the relationships between graphs and subjects, subjects and predicates, and predicates and objects? What about contextual relationships between graphs? Between subjects? Between predicates?

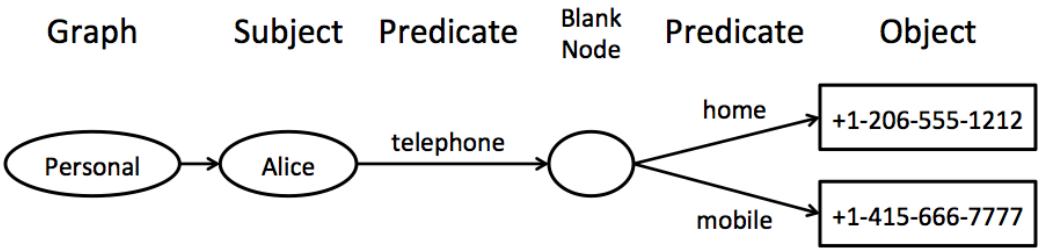
In fact the RDF graph model does have a mechanism for modeling context between RDF predicates: blank nodes. A blank node is an RDF graph node that has no IRI. Following is an illustration of how a blank node can be used to add context between a "telephone" predicate and two more predicates "home" and "mobile" representing types of a telephone number:

---

[14] This is particularly important for *data provenance*, i.e., determining the source and authority for a particular set of data. See https://en.wikipedia.org/wiki/Provenance#Computers_and_law

[15] The fourth component is the IRI that identifies the named graph, so the quad is <graphname> <subject> <predicate> <object>. See https://en.wikipedia.org/wiki/Named_graph.

[16] See *RDF 1.1: On Semantics of RDF Datasets*, https://dvcs.w3.org/hg/rdf/raw-file/default/rdf-dataset/index.html



This example shows how blank nodes can be used to add context between RDF predicates. However, this is only a partial solution because:

1. As mentioned earlier, blank nodes are not addressable outside the context of a single RDF graph (and thus not persistently addressable).
2. Blank nodes only enable context to be modeled between RDF predicates. They do not model context between RDF subjects or RDF graphs.

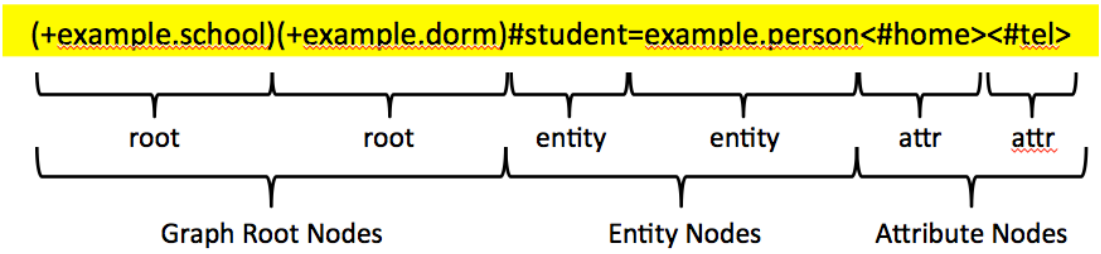
How the XDI Graph Model Solves the Problem

The XDI graph model solves the problem by defining context as *a fundamental type of relationship in the graph model itself*. This means contextual relationships can be expressed between all types of XDI graph nodes:

1. XDI graph root nodes can provide context for other XDI graph root nodes.
2. XDI entity nodes can provide context for other XDI entity nodes.
3. XDI attribute nodes can provide context for other XDI attribute nodes.[17]

Contextual relationships are in fact so central to the XDI graph model that all XDI graph nodes except data literals are called **context nodes**. And all context nodes are globally addressable (unlike RDF blank nodes) and can be nested to any depth.

Following is an example XDI address that shows two levels of context between graph root nodes, two levels of context between entity nodes, and two levels of context between attribute nodes.



[17] In XDI, an RDF predicate (graph arc) that describes a literal data node is represented as a graph node called an *XDI attribute* for two reasons: a) addressability, and b) contextualization. The actual literal node containing the data value of an XDI attribue is identified with the XDI predicate "&".

Like all XDI addresses, this example reads left-to-right as a sequence of component XDI addresses that identify each XDI graph node in the context of the previous graph node.

1. **(+example.school)** appears in parentheses because it is the root node of an XDI graph. As the first node in this XDI address, it the starting context for XDI discovery. It can be queried to find the IRI for the graph **(+example.dorm)**.
2. **(+example.dorm)** is the XDI address of a second XDI graph root node. This graph contains the rest of the XDI address.
3. **#student** is an entity (subject) inside the **(+example.dorm)** graph. # is the XDI context symbol for an unreserved class, so **#student** indicates there is a set of entities in this graph who are students.
4. **=example.person** identifies a specific person as an entity in the **#student** context. (In English, nodes #3 and #4 are like saying, "student John Doe".)
5. **<#home>** is the first attribute context. It can only be followed by other attribute contexts.
6. **<#tel>** is the final attribute context, which indicates that the literal value of this attribute is a telephone number.

In sum: this is the XDI address of a student's home telephone number that can be discovered through his/her dorm at his/her university.

This is a simple example of how contextual relationships are integral to XDI usage. XDI contexts are also fundamental to more advanced features of XDI, including:

- **Versioning** of any branch of any XDI graph—all versions are subcontexts of the versioned node.
- **Link contracts** are a standard subcontext within an XDI graph and also depend on persistent references to other standard subcontexts within the graph to provide interoperable, portable authorization.
- **Policy expression** within XDI messages and link contracts depend on subcontexts to both define policies and to structure a Boolean logic tree of policy evaluation.
- **Dictionaries** use subcontexts to both define XDI terms and to specialize those terms—for example the term #ski can be specialized by defining it in the contexts #snow, #water, and #ski. This produces the terms #snow#ski, #water#ski, and #jet#ski.

Summary: The Benefits of the XDI Graph Model

To summarize, the four key enhancements that the XDI graph model brings to the RDF graph model are:

1. 100% globally unique addressability of all nodes in all graphs.
2. Clear syntax and semantics for persistent identifiers so identity can be strongly bound to data.
3. Clear separation between the abstract identity of a resource described in a graph and any IRI-addressable representation of that resource.
4. The ability to model context to any depth at any of the three semantic levels of a triples-based graph model: graphs, subjects, and predicates.

The combination of these features—and especially the ability to model the "extra dimension" of context—has been the motivation for the development of XDI as a semantic data interchange format and protocol.

We will conclude by explaining how these features translate into specific benefits in four branches of the distributed data sharing problem space:

1. Internet identity management
2. Data portability
3. Privacy and security control
4. Semantic data sharing

Internet Identity Management

First, because of its explicit support for both persistent and abstract identification, together with global addressability and discoverability, XDI is particularly well suited to distributed, federated identity management at Internet scale. This includes identity for people, identity for organizational entities of any kind, and identity for independent objects, such as required for the Internet of Things.

This is one reason XDI is one of the federated identity standards (along with OpenID, OAuth, SAML, and others) that has been a focus of the Internet Identity Workshop since its inception in 2005.

Data Portability

Data portability refers to the ability for data authorities, such as individuals or companies, to be able to "take their data with them" and reuse it with different applications, services, social networks, and other systems without being "locked in" to a single app, database, vendor or silo. This is an essential feature of personal clouds and other services where data lock-in would effectively prohibit adoption.

All four of the key features of the XDI graph model are needed for data portability:

1. Portable data must continue to be **consistently addressable** in its new context no matter what where it moves on the network.
2. Portable data requires **persistent identifiers** so that the identity bound to the data will never change when it changes locations on the network.
3. Portable data requires **abstract identification** for both the data authority and the data so neither are tied to a specific location on the network.
4. By definition, data portability requires that the entire **context** of an XDI data graph (or any portion of it) be able to change without changing the semantics of the data within that graph.

Privacy and Security Control

With our increasing dependence on the Internet, maintaining privacy and security in our transactions and relationships online has become a major social, political, an economic issue. It has also become an essential requirement of compliance with privacy and security legislation in many countries, such as the U.S. HIPAA and FERPA regulations.

XDI infrastructure can assist with this problem in at least three ways:

1. **Enabling the Internet identity management features** described above. This is a foundational element of building trust on the Internet as highlighted by the U.S. NSTIC (National Strategy for Trusted Identities in Cyberspace) initiative, the U.K. IDAP (Identity Assurance Program), and other national cybersecurity programs.
2. **Providing the distributed public key management infrastructure** required to implement usable, interoperable encryption at Internet scale. Such a Web of Trust has long been envisioned by Phil Zimmermann, Tim Berners-Lee and others, but it has not been feasible to implement without strong identity and trusted addressability/discoverability of public keys.
3. **Implementing link contracts** to bring a new level of privacy and security control to the exchange of any type of sensitive non-public data (personal, corporate, medical, educational, governmental, etc.).

Semantic Data Sharing

Today's popular data sharing networks—email, file sharing, document sharing, calendar sharing—use non-semantic message, file, and markup formats. The partial exception is social networks that have started to add semantics via proprietary protocols such as the Facebook Open Graph protocol.

The next evolutionary step is *semantic data sharing*: networks that provide interoperable semantic data sharing based on open standards. In addition to the benefits already described above, XDI can enable these networks to provide hallmark new features such as:

1. **Semantic dictionary services**. XDI dictionaries already support sharing semantic "words" across contexts. But with the XDI protocol, XDI dictionaries can facilitate semantic reuse on a second dimension: collaborative development of dictionary definitions. Sharing dictionary services across an XDI network permit members of a community (and software agents running on their behalf) to dynamically establish the shared semantics they need to solve their interoperability challenges—and to evolve these semantics as their needs change.
2. **Semantic data integration.** Mapping the semantics of one system to another is the heart of data integration. But if an integrator develops an XDI connector for a system, the semantic map to one or more XDI dictionaries only has to be built once. From then on, data from that system can be shared with any other XDI-connected system that has a mapping to the same dictionaries. If different dictionaries are used, a cross-mapping between those dictionaries only needs to be built once.
3. **Semantic relationship management.** Data is shared by authorities with identities. This sharing is subject to contracts between those authorities and policies from those authorities. With different systems managing data, identities, contracts, and policies, is it any wonder data sharing is so hard today? With XDI semantic identity management, link contracts, and policy expressions, we finally have the prospect of true interoperable digital relationship management—being able to do for machine-readable data sharing what the Web did for human-readable content sharing. [18]

For More Information

For more information about XDI, please contact us the OASIS XDI Technical Committee.

Drummond Reed, Co-Chair
drummond.reed@xdi.org

Markus Sabadello, Co-Chair
markus.sabadello@xdi.org

[18] This new form of relationship management will not stop at data integration. It will revolutionize CRM (Customer Relationship Management) and lay the groundwork for VRM (Vendor Relationship Management). See https://en.wikipedia.org/wiki/Vendor_relationship_management