The XDI RDF Model

V7, October 3, 2007

This document summarizes an XRI-addressable RDF-based data model developed by the OASIS XDI Technical Committee called XDI RDF. It evolved out of a previous model developed by the TC called the ATI (Authority/Type/Instance) model. It is proposed that the XDI RDF model become the basis for the XDI 1.0 specifications once the OASIS XRI Technical Committee finalizew the XRI Resolution 2.0 specification (expected in October 2007).

Motivations

The motivations for the XDI RDF data model are:

- *Full fidelity with the RDF graph model.* With the XDI RDF model, the object of a subject/predicate assertion may be either an XRI reference to another XDI subject, or a data element representing a literal value, but not both. This is the same limitation imposed by the RDF graph model, but was not imposed by the ATI (Authority/Type/Instance) model, which permits data nodes to be associated with subject, predicate, and object nodes.
- *Clean separation of RDF predicates and RDF objects.* In the XDI RDF model, all XDI addresses represent RDF statements. Statements about concrete data objects consist of two XRI segments (an XRI representing the RDF subject and an XRI representing the RDF predicate). Statements about other XDI resources consist of three XRI segments (the third segment being an XRI identifying the target XDI subject or object). In the ATI model, the third XRI segment did not clearly correspond to an RDF object, as it could contain both an XRI and concrete data. This can lead to ambiguity about how to model different data structures.
- *Explicit representation of references and links*. In the ATI model, refs and links were not fully representable as XRIs. In the XDI RDF model, refs and links are now fully expressed as XRIs, so each ref and link has its own explicit XDI address, and all XDI operations can be performed on refs and links just like all other XDI RDF statements.
- *Simpler and more expressive dictionary definitions.* The XDI RDF model uses four root predicate XRIs ($is, $has, isa, hasa) and one root type XRI ($type) to support the construction of XDI dictionary definitions with self-describing schemas and datatypes. Higher-level constraints (cardinality, sequence, etc.) can be added by extending these XRIs.
- *Simplied link contract structure.* The XDI RDF model simplifies and generalizes the model for how any XDI authority can control access, exchange, and rights to any XDI data for which they are authoritative.
- *Simplified human understanding.* With the XDI RDF model, the XDI universal graph can be represented in the same (or simpler) notational formats as other RDF graphs, including an X3 notation that corresponds closely to RDF N3 notation. It is also easy to represent XDI graphs using a four-column table (where for each row there is an entry in either the third or fourth column but not both).

The XDI RDF Model

In the XDI RDF model, all data is both addressed and described using RDF subject/predicate/object statements encoded as XRIs. Each XRI representing an XDI RDF statement consists of up to three segments, each of which is itself either an absolute or relative XRI.

First segment (RDF Subject)	Second segment (RDF Predicate)	Third segment (RDF Object)	Addresses & Describes
XRI	Absent	Absent	The XDI graph (set of all XDI RDF statements) rooted in an XDI subject.
XRI	XRI	Absent	The XDI graph rooted in the target of the predicate, which is either: a) a set of XRIs referencing other XDI subjects, or b) a literal.
XRI	XRI	XRI	The XDI RDF statement whose object is another XDI subject.

XDI RDF statements can be joined together in a pattern that looks like *subject/predicate/subject/predicate*…. In this pattern, the presence of an empty segment indicates that any further addressing extends inside the value of the <xdi:data> element at that node of the XDI graph. See *Nesting XDI Documents* below.

About XRI 3.0 Syntax

The XRI addressing using in the XDI RDF model is based on the proposed ABNF for XRI Syntax 3.0. This adds two additional features to XRI 2.0 syntax (specified in the OASIS XRI Syntax 2.0 Committee Specification) as described in this section.

Global Context Symbols

The proposed ABNF for XRI Syntax 3.0 reduces the set of XRI global context symbols—single characters expressing the global context of an identifiers—from the 5 specified in XRI 2.0 to the four shown below. (In XRI Syntax 2.0, the ! symbol was used as both a global context symbol and a local context symbol. The XRI TC has reached concensus to deprecate its use as a global context symbol beginning in XRI 3.0.)

GCS Char	Applies To	Description
$	Classes (dictionaries)	A specified context. $ is the root of the XRI dictionary specified by the OASIS XRI Technical Committee and XDI upper ontology specified by the OASIS XDI Technical Committee.
+	Classes (dictionaries)	The generic context.
=	Instances (registries)	A personal context (i.e., the XDI authority is an individual person).
@	Instances (registries)	An organizational context (i.e., the XDI authority is a groups or organization).

Direct Concatenation

The second key feature of XRI 3.0 necessary for XDI RDF is called *direct concatenation*. It is the ability for any two valid XRI 3.0 absolute XRIs (excluding fragments) to be directly concatenated to create another valid XRI 3.0 XRI. This feature is fundamental to the XDI RDF addressing algorithm.

For example, following are three different XRIs representing an organization, a tag, and a person, respectively:

```
@example.company        +human.resources        =example.person.name
```

Using direct concatenation, these three XRIs can be joined into a single XRI that functions as a single structured identifier.

```
@example.company+human.resources=example.person.name
```

In this structured identifier, each component XRI appears in the context of its predecessor. Like XML, such an identifier has a machine- and human-understandable structure that supports introspection, discovery, mapping, and other benefits not available from opaque identifiers.

Cross-References

A third key XRI feature has been part of XRI syntax since XRI 1.0. As a language for structured identifiers, XRI has always needed the ability to encapsulate and describe other identifiers from other identifier syntaxes and namespaces very much the same way XML can encapsulate and "tag" data from different native data sources. This feature, called *cross-references*, uses parentheses to syntactically encapsulate the cross-referenced identifier. It is critical for XDI RDF because it enables all any URI to be included in an XRI expressing an XDI RDF statement. This enables conventional RDF documents to be expressed and addressed in XDI RDF.

For example, following is an N3 relationship expressed using URIs:

```
<http://equalsdrummond.name> <http://dc.org/tag/author> <http://example.com/doc,html>
```

Following is the same statement rendered as an XRI using XDI RDF addressing syntax:

```
=(http://equalsdrummond.name)/+(http://dc.org/tag/author)/=(http://example.com/doc.html)
```

The XDI RDF Root Predicates

The XDI RDF model uses four predicates defined in the XRI $ dictionary space to express fundamental relationships between XDI subjects. Each of these four predicates also has an inverse that expresses the same relationship in the opposite direction, i.e., the same RDF arc but pointing in the opposite direction.

Predicate	Inverse Predicate	Relationship Type	RDF/RDFS/OWL Analogs
$is	$has$is	Equivalence	No direct equivalent for XDI RDF subject equivalence. `owl:equivalentClass` for XDI RDF object equivalents.
isa	$has$is$a	Inheritance	`owl:class` for for XDI RDF subjects. `rdf:type` for XDI RDF objects except those rooted in the XDI RDF **$type** space, which would be `rdf:parseType`
$has	ishas	Aggregation	`exns:hasXXX`, where `exns` is the specified namespace and `XXX` is the property name
hasa	ishas$a	Composition	`exns:hasxxx`, where `exns` is the specified namespace and `xxx` is the property name

Following are example XDI RDF statements expressed as XRIs using these predicates:

Predicate	Statement	Inverse Statement
$is	+car/$is/+auto	+auto/$has$is/+car
	=drummond/$is/=drummond.reed	=drummond.reed/$has$is/=drummond
isa	+sedan/isa/+car	+car/$has$is$a/+sedan
	=drummond/isa/+person	+person/$has$is$a/=drummond
$has	+car/$has/+expensive	+expensive/ishas/+car
	=drummond/$has/+car	+car/ishas/=drummond
hasa	+car/hasa/+engine	+engine/ishas$a/+car
	=drummond/hasa/+age	+age/ishas$a/=drummond

XDI RDF Addressing Rules

A fundamental feature of XDI RDF is that:

- Every XDI RDF statement is an XRI, and
- Every XDI XRI functions as a unique address into the XDI RDF graph.

This means *100% of the XDI RDF graph*—from the largest XDI documents down to the smallest atomic data elements—is addressable using XRIs. The XDI RDF addressing rules are based on the four root predicates as summarized in the following table:

XDI RDF Predicate	Applies to	Addressing Rule
$has	First segment	Represented by subsegment delegation within the first segment of an XDI XRI (the subject segment) to express an aggregation relationship.
hasa	Second segment	Represented by subsegment delegation wthin the second segment of an XDI XRI (the predicate segment) to express a composition relationship.
$is	Second segment	Appears literally as an XDI RDF predicate value to assert an equivalence relationship.
isa	Second segment	Appears literally as an XDI RDF predicate value to assert an inheritance relationship.

See the sections below for examples.

The XDI RDF "REST" Operations

Following the REST model, the XDI protocol supports four atomic operations on the XDI RDF graph itself and one abstract operation on XDI data described by the graph. The atomic operations all operate on the logical "table" formed by the set of XDI RDF statements in any XDI RDF document (see the following sections for examples).

These REST operations are summarized in the following table:

XDI RDF Graph Operation	CRUD Equivalent	Description
$get	read	Read one or more rows from the XDI RDF table.
$add	create	Write one or more new rows to the XDI RDF table.
$mod	update	Modify a data element value in a row of the XDI RDF table.
$del	delete	Delete one or more rows from the XDI RDF table.
$do	N/A	Perform an operation on XDI data elements whose API is defined by XDI subject rooted on this XRI.

Declaring XRIs for these explicit XDI protocol operations establishes the basis for permissioning in XDI link contracts (see examples below).

Example XDI RDF Tables

Because the XDI RDF model enables all XDI resources to be both addressed and described using XRIs of up to three segments, the content of XDI documents can be represented with 100% fidelity using a four-column table.[1] The first column represents the XDI RDF subject; the second column the XDI RDF predicate. The third and fourth columns represent the two different options for an XDI RDF object:

- If the object is an XRI referencing another XDI RDF subject, it goes in the third column and is called an XDI *reference*.
- If the object is a literal, it goes in the fourth column and is called XDI *data*. Note that one form of XDI data is another XDI document. See *Nesting XDI Documents* below.

It is trivial to transform an XDI RDF table format into a valid XDI XML document (or another XDI serialization format) and vice versa.

The following sections are example XDI RDF tables. Notes about these examples:

- / as the first character of an XRI cross-reference means the cross-reference is relative to the current XDI subject.
- Blank shaded rows are inserted between different XDI subjects only for improved readability – they have no meaning.
- XDI subjects and predicates are not repeated in the table when they are identical to the previous subject or predicate. This is similar to X3 syntax (described below).

XDI Dictionary Definitions

The following XDI RDF table provides an example of XDI dictionary definitions, where all subjects are in the $ or + space and all predicates are rooted on one of the four root XDI RDF predicates. The vast majority of XDI RDF statements in an XDI dictionary are references because the express relationships between XDI subjects in the dictionary, i.e., an ontology. However for human-readablity, a typical XDI dictionary may also include XDI predicates such as **$type$mime$text** (representing that value of the data element is has mime type `text`). Rows using this predicate would contain a human-readable text description of the dictionary entry.

Note that, if needed, a single XDI document can transmit both XDI instances (i.e., XDI subjects rooted in the = and @ spaces) together with the XDI dictionary definitions relevant to these instances (XDI subjects rooted in the $ or + space). In the examples below, this would simply be a matter of combining some or all of the first XDI RDF table with the subsequent XDI RDF tables.

The following example is limited to simple attributes of a person. The scope of XDI dictionaries is, like human-language dictionaries, bounded only by the context of the dictionary.

[1] When XDI documents are nested, as described later in this document, XDI RDF tables also need to nested.

Subject	Predicate	Reference	Data
+	$has	person	
+person	hasa	$uri	
		+home	
		+work	
		+friend	
		+spouse	
		+name	
		+email	
		+phone	
+home	isa	$has+person	
+work	isa	$has+person	
+friend	isa	$has+person	
+name	isa	$type$xsi$string	
+person+name	isa	$type$xsi$string	
	hasa	+first	
		+last	
		+legal	
		+preferred	
		+middle	
		+nickname	
+email	isa	$type$xsi$string	
	hasa+person	+home	
		+work	
		+primary	
		+alt	
		+preferred	
+phone	isa	$type$xsi$string	
	hasa	+country.code	
		+area.code	
		+number	
		+extension	
	hasa+person	+home	
		+work	
		+primary	
		+alt	

		+preferred	
phone+country.code	isa	$type$xsi$short	
	$type$xsi$enumeration	1	
		20	
		30	
		…	
phone+area.code	isa	$type$xsi$short	
Phone+number	isa	$type$xsi $postiveinteger	
phone+extension	isa	$type$xsi$short	

XDI Cards

The following XDI RDF tables represent typical XDI documents that might be shared between individuals as "XDI business cards". They might also be expressed using a data sharing user interface metaphor developed by the Higgins Project[2] called an *i-card*: a set of claims about a digital subject[3] shared in a specific context. XDI RDF supports this metaphor by enabling claims to be expressed as XDI RDF statements and allowing them to be packaged to describe an XDI RDF subject in a precise context.

The first section below represents the XDI RDF subject **=drummond** and includes many literal data values. The second section represents the **=drummond+home** context and consists mostly of cross-references to the data values in the **=drummond** context. If the **=drummond+home** context was being shared, the XDI document would include only the **=drummond+home** rows plus the **=drummond** rows referenced by the **=drummond+home** cross-references (the XRIs in the Reference column enclosed in parentheses). The same would be true if only the **=drummond+work** context was being shared.

Note that this XDI content uses the XDI dictionary definitions established in the previous section.

Subject	Predicate	Reference	Data
=drummond	$is	=drummond.reed	
	isa	+person	
	$has	+home	
		+work	
	$uri		http://equals drummond.name
	+blog	(/$uri)	
	+person+name		Drummond Reed
	+person+name+first		Drummond
	+person+name+middle		Shattuck
	+person+name+last		Reed
	+person+name+legal		Drummond Shattuck Reed
	+person+name +preferred	(/+person+name)	
	+person+name +nickname		Drum
	+email	(/+email$v*2)	
	+email$v*1		drummond@ example.com

[2] http://www.eclipse.org/higgins/
[3] http://idgang.idcommons.net/moin.cgi/Lexicon

	+email$v*2		drummond.reed@example.net
	+email+home	(/+email)	
	+email+work		drummond.reed@cordance.net
=drummond+home	isa	+person	
	+person+name	(=drummond/+person+name)	
	+person+name+first	(=drummond/+person+name+first)	
	+person+name+last	(=drummond/+person+name+last)	
	+person+name+nickname	(/+person+name+nickname*1)	
		(/+person+name+nickname*2)	
	+person+name+nickname*1	(=drummond/+person+name+nickname)	
	+person+name+nickname*2		Shad
	+email	(=drummond/+email)	
	+email+home	(=drummond/+email+home)	
=drummond+work	isa	+person	
	+person+name	(=drummond/+person+name)	
	+person+name+first	(=drummond/+person+name+first)	
	+person+name+last	(=drummond/+person+name+last)	
	+email	(=drummond/+email+work)	
	+email+work	(=drummond/+email+work)	

XDI Link Contracts

A particularly attractive feature of the XDI RDF model is the simplicity of XDI link contracts. Link contracts are themselves fully addressable XDI documents that control access, authorization, and usage rights to other XDI documents. In essence a link contract is directly analogous to: a) a real-world contract, for example a legal contract between companies (such as a non-disclosure agreement) or a "social contract" between people, and b) an access control list (ACL), such as those used by directories and other networked applications to control access to resources.

The basic structure of a link contract is a **hasa** relationship called **$contract** that describes a relationship between two XDI subjects. This **$contract** can be further refined to subtypes, such as **$contract+person** and **$contract+org**. The OASIS XDI TC will define only the basic **$contract** container structure for link contracts; the human-readable text and machine-readable attributes of specific link contracts will be defined by standards bodies, governments, industry consortia, or other communities of usage such as the Identity Commons Identity Rights Agreements Working Group (http://wiki.idcommons.net/moin.cgi/IdentityRightsAgreementsCharter).

Following are several examples of how link contracts can be applied to personal data sharing relationships. They all reference an instance of a hypothetical personal data sharing link contract defined by Identity Commons (**@idcommons/$contract+person**). Note that this reference to an externally-defined link contract only governs the non-machine-readable data sharing policies. The machine-processible access and authorization policies are expressed using XDI RDF statements about the XDI operations described above.

Note also how **$has** relationships are used to create groups such as **+friend**, **+friend+soccer**, and **+spouse**. Typically link contracts are then associate with these groups. Each group member then inherits the permissions granted to that group.

Subject	Predicate	Reference	Data
=drummond	$has	+friend	
		+friend+soccer	
		+spouse	
	=andy.dale	=drummond+friend	
	=les	=drummond+friend	
		=drummond+friend +soccer	
	=rainyday*bob	=drummond+friend +soccer	
	=hip.hop	=drummond+friend +soccer	
	=mvr	=drummond+spouse	
=drummond+friend	isa	+friend	
	$has	=andy.dale	

		=les	
	$contract	(@idcommons/ $contract+person)	
	$get	(=drummond +home/*)	
		(=drummond +work/*)	
=drummond +friend+soccer	isa	=drummond+friend	
	$has	=les	
		=rainyday*bob	
		=hip.hop	
	$contract	(@idcommons/ $contract+person)	
	$get	(=drummond +home/+email)	
=drummond+spouse	isa	+spouse	
	$has	=mvr	
	$contract	(@idcommons/ $contract+person)	
	$get	(=drummond*/*)	
	$add	(=drummond*/*)	

X3 Notation

XDI TC member Bill Barnhill propsed thatXDI RDF tables can be represented in plain text using a notation very similar to the N3 (Notation 3) format used with RDF. This "X3" format would simply express each of the XDI RDF statements as XRIs.

Note that, for readability, subjects, predicates, and objects are all indented on different lines, and repeated subjects and predicates are not shown (just as in the XDI RDF tables above). Also, when literals are shown, they appear after the three slashes separating the three parts of an XDI address in quotes. Note that *they are not themselves part of the address*, rather they are the target of the address.

The following examples are X3 notation for the XDI RDF tables in the main body of the document.

```
=drummond
      /$is
            /=drummond.reed
      /$is$a
            /+person
      /$has
            /+home
            /+work
            /+friend
            /+friend+soccer
            /+spouse
      /$uri
            //"http://equalsdrummond.name"
      /+blog
            /(/$uri)
      /+person+name
            //"Drummond Reed"
      /+person+name+first
            //"Drummond"
      /+person+name+last
            //"Reed"
      /+person+name+middle
            //"Shattuck"
      /+person+name+legal
            //"Drummond Shattuck Reed"
      /+person+name+preferred
            /(/+person+name)
      /+person+name+nickname
            //"Drum"
      /+email
            /(/+email$v*2)
      /+email$v*1
            //"drummond@example.com"
      /+email$v*2
            //"drummond.reed@example.net"
      /+email+home
            /(/email)
      /+email+work
            //"drummond.reed@cordance.net"
```

```
      /=andy.dale
            /=drummond+friend
      /=les
            /=drummond+friend
            /=drummond+friend+soccer
      /=rainyday*bob
            /=drummond+friend+soccer
      /=hip.hop
            /=drummond+friend+soccer
      /=mvr
            /=drummond+spouse
=drummond+home
      /$is$a
            /+person
      /+person+name
            /(=drummond/+person+name)
      /+person+name+first
            /(=drummond/+person+name+first)
      /+person+name+last
            /(=drummond/+person+name+last)
      /+person+name+nickname
            /(=drummond/+person+name+nickname*1)
            /(=drummond/+person+name+nickname*2)
      /+person+name+nickname*1
            /(=drummond/+person+name+nickname)
      /+person+name+nickname*2
            //"Shad"
      /+email
            /(=drummond/+email)
      /+email+home
            /(=drummond/+email+home)
=drummond+work
      /$is$a
            /+person
      /+person+name
            /(=drummond/+person+name)
      /+person+name+first
            /(=drummond/+person+name+first)
      /+person+name+last
            /(=drummond/+person+name+last)
      /+email
            /(=drummond/+email+work)
      /+email+work
            /(=drummond/+email+work)
=drummond+friend
      /$is$a
            /+friend
      /$has
            /=andy.dale
            /=les
      /$contract
            /(@idcommons/$contract+person)
      /$get
            /(=drummond+home/*)
            /(=drummond+work/*)
=drummond+friend+soccer
      /$is$a
```

```
            /=drummond+friend
      /$has
            /=les
            /=rainyday*bob
            /=hip.hop
      /$contract
            /(@idcommons/$contract+person)
      /$get
            /(=drummond+home/+email)
=drummond+spouse
      /$is$a
            /+spouse
      /$has
            /=mvr
      /$contract
            /(@idcommons/$contract+person)
      /$get
            /(=drummond*/*)
      /$add
            /(=drummond*/*)
```

Nesting XDI Documents

An XDI data element may contain data of any type, from a simple string or number to binary data to other structured XML documents. One special case is when an XDI data element contains another XDI document, i.e., nesting. Nesting XDI documents has many uses:

- It is a simple, clean mechanism for logging XDI operations on an XDI document.
- It provides a simple way to store and manage XDI document versions.
- It can provide an XDI-based caching mechanism.
- It is the proposed structure for XDI messaging: the outer XDI document is the XDI message "envelope"; the inner document is the message, and message attachments are nested at lower levels.

Note that XDI addressing rules can extend directly into nested XDI documents, permitting composition of even richer XDI RDF statements.

Nested XDI documents are easily represented in X3 notation. Following is an example of an XDI change log in X3. It shows that on a specified date, **=drummond** added two XDI RDF statements to the XDI subject **=drummond+home**:

```
=drummond+home$v*3
      /$d
            /$d*2007-09-06T14:27:34Z
      /=
            /=drummond
      /$add
            //
                  =drummond+home
                        /+phone+mobile
                              /"+1 206.618.8530"
                        /+phone+emergency
                              /(/+phone+mobile)
```

Appendix A: The XDI RDF Schema

Note that this schema is intentionally extremely simple: although it adapts the XDI RDF graph model to be serialized in XML, all the semantics are in the XRIs.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Proposed XDI RDF schema v4, posted 2007-09-06 -->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xdi="http://xdi.oasis-open.org" targetNamespace="http://xdi.oasis-open.org"
elementFormDefault="qualified">
	<element name="xdi">
		<complexType>
			<sequence>
				<element ref="xdi:s" maxOccurs="unbounded"/>
			</sequence>
		</complexType>
	</element>
	<element name="s">
		<complexType>
			<sequence>
				<element ref="xdi:p" minOccurs="0"
maxOccurs="unbounded"/>
			</sequence>
			<attribute name="xri" type="string" use="required"/>
		</complexType>
	</element>
	<element name="p">
		<complexType>
			<choice>
				<element ref="xdi:ref" maxOccurs="unbounded"/>
				<element ref="xdi:data"/>
			</choice>
			<attribute name="xri" type="string" use="required"/>
		</complexType>
	</element>
	<element name="ref">
		<complexType>
			<attribute name="xri" type="string" use="required"/>
		</complexType>
	</element>
	<element name="data" type="anyType"/>
</schema>
```

Appendix B: An Example XDI Document

Following is an example XDI document intended to illustrate a small set of XDI personal cards and associated link contracts taken from the example XDI RDF tables above. Note that XDI/XML is only one potential serialization format; other formats optimized for certain uses (such as JSON) may also be defined. See also the X3 format in Appendix C.

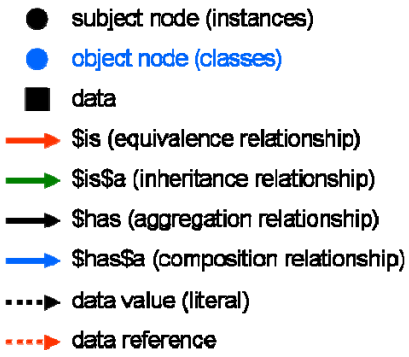
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Example XDI instance document using XDI RDF schema v4, posted 2007-09-06
-->
<xdi:xdi xmlns:xdi="http://xdi.oasis-open.org"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <xdi:s xri="=">
                <xdi:p xri="$has">
                        <xdi:ref xri="drummond"/>
                </xdi:p>
        </xdi:s>
        <xdi:s xri="=drummond">
                <xdi:p xri="$is">
                        <xdi:ref xri="=drummond.reed"/>
                </xdi:p>
                <xdi:p xri="$is$a">
                        <xdi:ref xri="+person"/>
                </xdi:p>
                <xdi:p xri="$has">
                        <xdi:ref xri="+home"/>
                        <xdi:ref xri="+work"/>
                        <xdi:ref xri="+friend"/>
                        <xdi:ref xri="+spouse"/>
                </xdi:p>
                <xdi:p xri="+email">
                        <xdi:ref xri="(/+email+home)"/>
                        <xdi:ref xri="(/+email+work)"/>
                </xdi:p>
                <xdi:p xri="+email+home">
                        <xdi:data>drummond@example.com</xdi:data>
                </xdi:p>
                <xdi:p xri="+email+work">
                        <xdi:data>drummond.reed@cordance.net</xdi:data>
                </xdi:p>
                <xdi:p xri="=andy">
                        <xdi:ref xri="=drummond+friend"/>
                </xdi:p>
                <xdi:p xri="=les">
                        <xdi:ref xri="=drummond+friend"/>
                        <xdi:ref xri="=drummond+friend+soccer"/>
```

```
            </xdi:p>
            <xdi:p xri="=rainyday*bob">
                  <xdi:ref xri="=drummond+friend"/>
            </xdi:p>
            <xdi:p xri="=hip*hop">
                  <xdi:ref xri="=drummond+friend"/>
            </xdi:p>
            <xdi:p xri="=mvr">
                  <xdi:ref xri="=drummond+spouse"/>
            </xdi:p>
      </xdi:s>
      <xdi:s xri="=drummond+home">
            <xdi:p xri="+email">
                  <xdi:ref xri="(=drummond/+email+home)"/>
            </xdi:p>
      </xdi:s>
      <xdi:s xri="=drummond+work">
            <xdi:p xri="+email">
                  <xdi:ref xri="(=drummond/+email+work)"/>
            </xdi:p>
      </xdi:s>
      <xdi:s xri="=drummond+friend">
            <xdi:p xri="$is$a">
                  <xdi:ref xri="+friend"/>
            </xdi:p>
            <xdi:p xri="$has">
                  <xdi:ref xri="+soccer"/>
            </xdi:p>
            <xdi:p xri="=andy">
                  <xdi:ref xri="=drummond+friend"/>
            </xdi:p>
            <xdi:p xri="=les">
                  <xdi:ref xri="=drummond+friend"/>
            </xdi:p>
            <xdi:p xri="+contract">
                  <xdi:ref xri="(@idcommons/+contract+person"/>
            </xdi:p>
            <xdi:p xri="$get">
                  <xdi:ref xri="(=drummond+home/*)"/>
            </xdi:p>
      </xdi:s>
      <xdi:s xri="=drummond+friend+soccer">
            <xdi:p xri="$is$a">
                  <xdi:ref xri="+friend+soccer"/>
            </xdi:p>
            <xdi:p xri="=les">
```

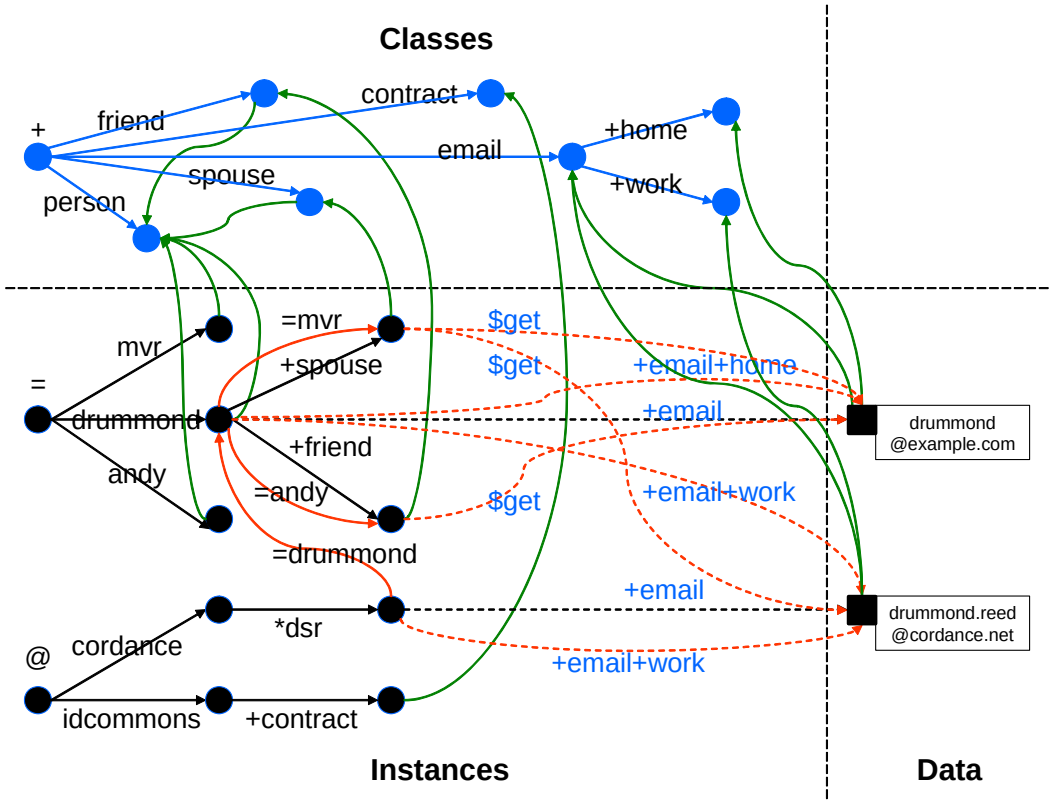
```
                <xdi:ref xri="=drummond+friend+soccer"/>
            </xdi:p>
            <xdi:p xri="=rainyday*bob">
                <xdi:ref xri="=drummond+friend+soccer"/>
            </xdi:p>
            <xdi:p xri="=hip.hop">
                <xdi:ref xri="=drummond+friend+soccer"/>
            </xdi:p>
            <xdi:p xri="+contract">
                <xdi:ref xri="(@idcommons/+contract+person"/>
            </xdi:p>
            <xdi:p xri="$get">
                <xdi:ref xri="(=drummond/+email+home)"/>
            </xdi:p>
        </xdi:s>
        <xdi:s xri="=drummond+spouse">
            <xdi:p xri="$is$a">
                <xdi:ref xri="+spouse"/>
            </xdi:p>
            <xdi:p xri="=mvr">
                <xdi:ref xri="=drummond+spouse"/>
            </xdi:p>
            <xdi:p xri="+contract">
                <xdi:ref xri="(@idcommons/+contract+person"/>
            </xdi:p>
            <xdi:p xri="$get">
                <xdi:ref xri="(=drummond*/*)"/>
            </xdi:p>
            <xdi:p xri="$add">
                <xdi:ref xri="(=drummond*/*)"/>
            </xdi:p>
        </xdi:s>
</xdi:xdi>
```

Appendix C: An Example XDI Graph

Like RDF, XDI RDF can also be represented as a visual graph. The following example uses this key:



This is an example graph of a typical XDI document containing a subset of the XDI RDF statements from the preceeding XDI RDF tables. Note that the upper left-hand quadrant represents the XDI dictionary space, i.e., it illustrates XDI RDF object definitions or "classes". The lower left-hand quadrant represents the XDI RDF instance space, and the green arrows up to the XDI dictionary space represent the class/instance relationships. The lower right-hand quadrant represents the literal data space, with green arrows up to the dictionary space representing isa relationships to the simple data types defined there (note that further details of these definitions are not shown).



Appendix D: Revision History

V7 – 2007-10-03

- Added inverse relations for **$is**, **$is$a**, **$has**, **hasa**.
- Removed the placeholder cardinality syntax (this will become part of the XDI upper ontology).
- Updated **$type** children to use the **$** space.
- Corrected XDI RDF table examples for "group membership" (added $has predicate).

V6 – 2007-09-06

- General rewrite of introductory sections based on feedback from XDI RDF presentations and implementations.
- Added new section on XRI 3.0 syntax.
- Added new section on nested XDI documents.
- Revision to XDI RDF v4 schema in Appendix A to rename **<xdi:o>** to **<xdi:ref>** and change the type of **<xdi:data>** to *anyType*.
- Revised example XDI document in Appendix B to conform XDI RDF v4 schema.

V5 – 2007-05-21

- Clarified references to Higgins and Higgins contexts in the XDI Cards section.
- Updated references to XRI Syntax 2.1 to XRI Syntax 3.0.
- Miscellaneous editing throughout for readability.
- Moved revision history to Appendix.

V4 – 2007-03-26

- Updated terminology to use "ATI" instead of "3L".
- Simplified table listing options for XDI RDF model.
- Adjusted text and examples to use proposed XRI 3.0 syntax.
- Updated changed terminology in graph examples.

V3 – 2007-02-14

- New terminology.
- Updated the XDI RDF model section to specify XDI RDF statement forms that can be expressed with XRIs.
- Replaced the core XDI RDF predicate names with their simpler English equivalents (suggested by Bill Barnhill, Paul Trevithick, and Laurie Rae, among others).
- Changed **$data** to **$type** (per suggestion from Bill Barnhill) and removed it from the core predicate table (because it is only used as an object).
- Added information about the RDF/RDFS/OWL equivalents for the four core XDI RDF predicates.
- Updated the link contract examples to show a simpler way to express group membership.
- Added section show X3 syntax.

V2 – 2007-02-06

- Shortened the <subject>, <predicate>, and <object> element names in the XDI RDF schema to make instance documents easier to read.
- Revised the XDI RDF diagram to illustrate link contracts instead of dictionaries.
- Provided a more typical example of an XDI instance document including examples of link contracts.
- Added a phone number example to the XDI RDF tables in the XDI Dictionary Definitions section to show a complex object definition.

To-Dos for Future Versions

- Show separate example XDI RDF documents for dictionary entries and instances of those entries.
- Show the exact representation of several example XDI RDF documents in all four formats (XML, X3, table, and graph). (The current document shows slightly different data sets in all four formats.)