# The XDI Graph Model

To meet the design goals in the preceding section, the XDI TC developed the semantic graph model defined in this and the following sections.

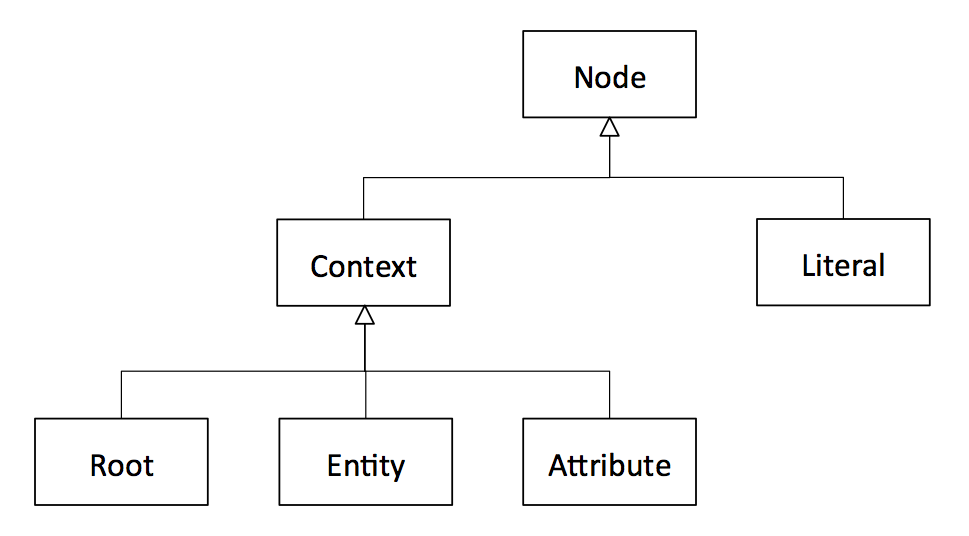
## Overview

The XDI graph model builds on the RDF *subject-predicate-object* triples model [ref]. This model in turn builds on the Entity-Attribute-Value (EAV) data model that dates back over 40 years. Note that in RDF, a graph node containing a data value is called a *literal*. So the RDF data model could also be termed an Entity-Attribute-Literal (EAL) model.

With RDF 1.1 datasets [ref], the model was expanded to *context-subject-predicate-object* quads. The fourth component—context—represents a named RDF graph. The XDI graph model also has an analogous fourth component representing the root of an XDI graph. Thus it is called the Root-Entity-Attribute-Literal (REAL) model.

## Node Types

Figure \_\_\_ shows a simple UML class diagram of the highest level node types in the XDI REAL graph model.



All graph nodes are one of two fundamental types: *literal nodes* or *context nodes*.

### Literal Nodes

As in RDF, XDI literal nodes are the terminal leaf nodes of the graph. They contain the raw data values described by all the other metadata in the graph. XDI natively supports the six data types defined by JSON [ref]:

1. Number (double-precision floating-point format in JavaScript)
2. String (double-quoted Unicode, with backslash escaping)
3. Boolean (true or false)
4. Array (an ordered, comma-separated sequence of values enclosed in square brackets; the values do not need to be of the same type)
5. Object (an unordered, comma-separated collection of key:value pairs enclosed in curly braces, with the ':' character separating the key and the value)
6. null (empty—note that this is not the equivalent of *undefined*, which is when an XDI attribute has no literal node at all)

In addition to the basic data type semantics provided by JSON, the type of a literal MAY be further described using one or more XDI type statements (see *Arc Types and Statement Types*).

### Context Nodes

All non-literal nodes in the XDI graph model are called *context nodes*. In RDF the term “context” is only used to describe the top level of semantic context available in the RDF 1.1 graph model, i.e., a named RDF graph. In addition, RDF blank nodes can be used to add a type of context to the relationship between other nodes. However, RDF does not use the term “context” for this purpose.

In XDI the term “context” is used uniformly across all levels of the REAL model to describe all forms of semantic context, including when:

* A graph root node provides context for another graph root node, an entity node, or an attribute node.
* An entity node provides context for another entity node or an attribute node.
* An attribute node provides context for another attribute node.

See *Contextual Arcs and Contextual Statements*, below.

All context nodes MUST have:

1. Exactly one *context type* identified in XDI syntax by a single *context symbol*.
2. One or more *context roles* identified in XDI syntax by zero or more pairs of *context brackets*.

### Context Types and Symbols

The XDI REAL model defines six global context types in three groups:

1. **Classes** represent entity and attribute types.
2. **Instances** represent entity and attribute individuals.
3. **Authorities** represent the two specific types of entities that are ultimately responsible for control of XDI graphs.

The context symbols for each type are shown in the table below:

|  |  |  |
| --- | --- | --- |
| **Group** | **Context Type** | **Symbol** |
| Classes | Reserved | **$** |
| Unreserved | **#** |
| Instances | Ordered | **@** |
| Unordered | **\*** |
| Authorities | Personal | **=** |
| Legal | **+** |

A definition of each context type is provided in the Entity section below.

Note that XDI syntax also uses two other single-character symbols:

1. **&** (ampersand) to identify a literal arc (see *Literal Arcs and Literal Statements*, below).
2. **!** (exclamation mark) to indicate an immutable XDI identifier (see *Mutable and Immutable Identifiers*).

### Context Roles and Brackets

The XDI REAL model defines six context roles in two groups:

1. **Primary roles:** every context node MUST have exactly one primary role.
2. **Secondary roles:** depending on the context, a context node MAY have one or more secondary roles.

The context brackets for each role are shown in the table below:

|  |  |  |
| --- | --- | --- |
| **Group** | **Context Role** | **Brackets** |
| Primary | Entity | none |
| Attribute | **< >** |
| Root | **( )** |
| Secondary | Collection | **[ ]** |
| Definition | **| |** |
| Variable | **{ }** |

Each context role is defined in its own section below.

## Arc Types and Statement Types

An RDF graph is a directed graph in which every predicate represents a directed arc from a subject node to an object node. Each unique RDF subject/predicate/object *statement* represents one such arc.

The same is true of the XDI graph model, however in XDI, an arc MUST be one of three types:

1. **Literal arcs** describe the relationship between a context node and a literal node.
2. **Contextual arcs** define the identity, type, and role of one context node in the context of another context node.
3. **Relational arcs** describe any other relationship between two context nodes.

Each type of arc is expressed using a specific type of XDI statement as defined in this section.

### Literal Arcs and Literal Statements

In the XDI REAL model, a literal node MUST be the object of exactly one *literal arc* expressed by exactly one *literal statement*. The subject of a literal arc MUST be an XDI attribute context node. An XDI attribute node MUST have no more than one literal arc.

There are two key differences between XDI literal arcs and RDF predicates whose object is a literal node:

1. **In RDF, the semantic meaning of a literal is expressed by its predicate arc.** In XDI, the semantic meaning of a literal is expressed by the sequence of XDI attribute node(s) that preceed the literal arc.
2. **In RDF, a literal may have its own datatype and language attributes.** In XDI, a literal node is always an atomic leaf node. Any other semantic description of a literal node MUST be expressed using XDI type statements (statements whose object is the parent XDI attribute node).

Because of the first difference above, an XDI literal arc is the semantic equivalent of the rdfs:value property in RDF [ref]. Thus in XDI, all literal arcs MUST have the same XDI identifier: the ampersand character &. This is called the *literal symbol*. All XDI literal statements MUST use the literal symbol as the predicate. Examples:

|  |  |  |
| --- | --- | --- |
| **Subject** | **Predicate** | **Object** |
| =example<#email> | & | "foo@example.com" |
| +example<#main><#tel> | & | "+44-2222-888888" |
| \*!1234[<#event>]<@78><$t> | & | "2010-09-20T10:11:12Z" |

In XDI JSON serialization format:

{

"=example<#email>": {

"&": "foo@example.com"

},

"+example<#main><#tel>": {

"&": "+44-2222-888888"

},

"\*!1234[<#event>]<@78><$t>": {

"&": "2010-09-20T10:11:12Z"

}

}

Because an XDI attribute node may only contain one literal node, that literal node is uniquely addressable by appending the literal symbol & to the XDI address of the attribute node. Examples:

|  |  |
| --- | --- |
| **XDI Address of Attribute Node** | **XDI Address of Literal Node** |
| =example<#email> | =example<#email>& |
| +example<#main><#tel> | +example<#main><#tel>& |
| \*!1234[<#event>]<@78><$t> | \*!1234[<#event>]<@78><$t>& |

### Contextual Arcs and Contextual Statements

In the RDF graph model, a blank node exists to provide context for other nodes, however a blank node does not have a URI. It can only be identified relative to the RDF graph in which it exists. [ref]

In the XDI graph model, all context nodes can provide context for other context nodes, and all context nodes are uniquely addressable. With the exception of the common root node, a context node MUST be the object of exactly one *contextual arc* expressed by a *contextual statement*. The subject of a contextual statement MUST be another context node, called the *parent node*. Only the common root node has no parent. The predicate of a contextual statement MUST be empty. The object of a contextual statement MUST have an XDI identifier that is unique in that context.

The result of these requirements is that XDI context nodes form a directed rooted graph, called a *semantic tree*, in which every node is uniquely addressable and every node has a semantic meaning. The absolute XDI address of a context node is the concatenation of the XDI identifiers for the sequence of contexts from the common root node to the target context node.

If the common root node of an XDI graph is itself assigned a URI, all nodes in the graph become globally addressable in the universal URI addressing space as recommended by [WebArch]. See the *XDI Addressing* section for details.

Following is an example of three contextual statements (each with the empty predicate) that establish the context for the final literal statement. In this example, =example and #car are XDI entities; <#interior> and <#color> are XDI attributes.

|  |  |  |
| --- | --- | --- |
| **Subject** | **Predicate** | **Object** |
| =example |  | #car |
| =example#car |  | <#interior> |
| =example#car<#interior> |  | <#color> |
| =example#car<#interior><#color> | & | “black” |

Figure \_\_\_ shows the same set of statements in XDI JSON serialization format. Note that when serialized the empty predicate in a contextual statement is represented by two forward slashes:

{

"=example": {

"//": [

"#car"

]

},

"=example#car": {

"//": [

"<#interior>"

],

"<#interior>": {

"//": [

"<#color>"

]

},

"<#interior><#color>": {

"&": "black"

}

}

}

Contextual statements are inherent in the XDI addresses of the subjects and objects of literal or relational statements. Therefore contextual statements are not included in the JSON serialization by default. They are only added if they are explicitly requested using the \_\_\_\_\_\_\_\_\_\_\_\_\_ parameter (see the *Serialization* section for details). Figure \_\_\_ shows the same example graph without the contextual statements:

{

"=example#car": {

"<#interior><#color>": {

"&": "black"

}

}

}

### Relational Arcs and Relational Statements

Any relationship between two XDI context nodes that is not described by a contextual arc MUST be described by a *relational arc* expressed by a *relational statement*. Both the subject and the object of a relational statement MUST be context nodes. The predicate of a relational statement MUST be a sequence of one or more XDI entities.

XDI relational arcs are the equivalent of RDF predicates that describe the relationship between two RDF resource nodes. Examples:

|  |  |  |
| --- | --- | --- |
| **Subject** | **Predicate** | **Object** |
| =person-1 | #friend | =person-2 |
| =person-1 | #friend | =person-3 |
| =person-1 | #best#friend | =person-3 |
| =person-1 | #employer | +example.company |
| [#device]\*!:uuid:1234 | #owner | =person-1 |

In the XDI JSON serialization, the predicate of a relational statement is prefixed with a forward slash character:

{

"=person-1": {

"/#friend": [

"=person-2",

"=person-3"

],

"/#best#friend": [

"=person-3"

],

"/#employer": [

"+example.company"

]

},

"[#device]\*!:uuid:1234": {

"/#owner": [

"=person-1"

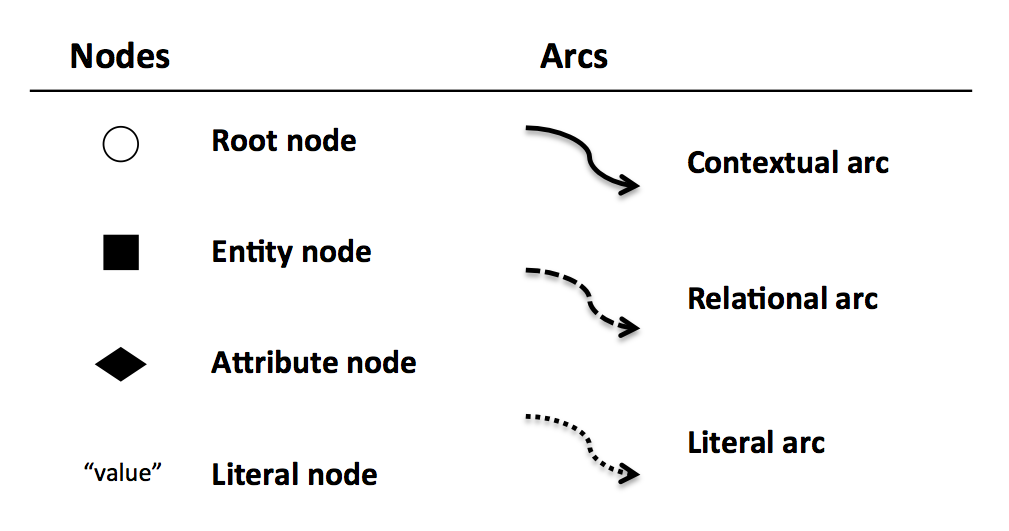
]

}

}

## Visual Graph Diagramming Notation

For consistency across implementations, the XDI Technical Committee RECOMMENDS the notation shown in Figure \_\_\_ for visual diagramming of XDI graphs.



The root node symbol (a circle) is suggestive of the parentheses ( ) used in XDI syntax, and the attribute node symbol (a diamond) is suggestive of the chevron brackets < > used in XDI syntax. The entity and attribute node symbols are solid to represent concrete identities and properties. The root node symbol is open to represent that an XDI graph is only a container of XDI statements.

For diagrams that support color, it is RECOMMENDED to use:

1. A red outlined circle for the common root node.
2. A blue outlined circle for a peer root node.
3. A green outlined circle for an inner root node.

Literal nodes are a direct representation of the JSON value. If the value is truncated to save space, it is RECOMMENDED that the portion shown end in ellipses.

All contextual and relational arcs MUST be labeled. A literal arc MAY be labeled with the ampersand symbol, but it is not recommended. For a contextual arc, the label MUST be the unique XDI identifier of the object context node. For a relational arc, the label MUST be the predicate of the relational statement.

Since there are many ways to orient an XDI graph diagram that uses this notation, the following two options are RECOMMENDED:

1. **Radial format.** In this orientation, the common root appears roughly in the center of the diagram, and arcs radiate outward.
2. **Tree format.** This orientation mimics a typical file or directory tree layout. The common root node appears in the upper-left-hand corner, contextual and literal arcs follow a grid, and only relational arcs are curved.

The choice of format depends on the particular XDI graph being shown. It is RECOMMENDED that viewing/editing tools support both formats and enable viewers to switch between them dynamically.

Figure \_\_\_ shows a simple XDI graph in radial format.

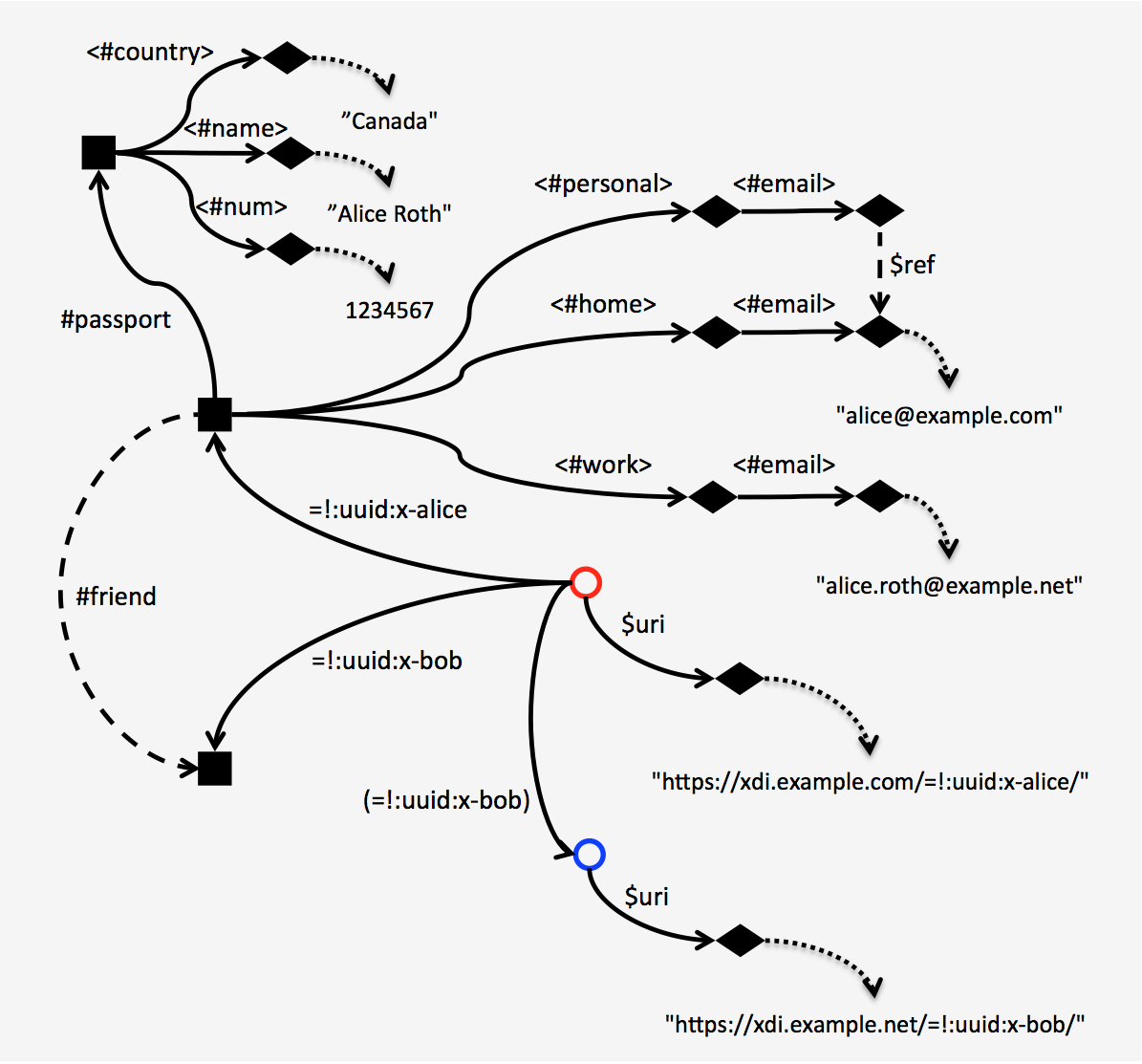


Figure \_\_\_ shows the same graph diagrammed in tree format.

