

A large, artistic photograph of fiber optic cables dominates the background. The cables are bundled together and fan out from the bottom left towards the top right, creating a sense of depth and movement. The tips of the cables are illuminated, appearing as bright, glowing points of light against the dark background. The overall aesthetic is clean, modern, and high-tech.

UNDERSTANDING XBRL:
THE USE OF XLink

AUDIT AND RISK ADVISORY SERVICES



CONTENTS

- 1 **Overview**
- 2 **The XLink Standard**
- 2 **Linkages in XBRL**
 - Links Between Schemas and Linkbases
 - Links Between Facts in Instance Documents
 - Linking Meaning to Items and Tuples
- 7 **Tips and Tricks**
 - Keeping Semantics Consistent
 - Avoiding Anonymous Schemas
 - Watching the XPointers for Local Taxonomy Caches
- 9 **Related Materials**

OVERVIEW

The XBRL specification has consistently worked hard to leverage emerging XML standards. For example, XBRL made extensive use of the syntax of XML Schema <<http://www.w3.org/XML/Schema>> while that specification had only candidate recommendation status within the W3C (World Wide Web Consortium) <<http://www.w3.org/>>. Looking forward, XBRL is already experimenting with leveraging the XPath 2.0 specification <<http://www.w3.org/TR/xpath20>>, in particular its ability to enable documentation of computational relationships.

In a similar vein, when the XBRL specification needed a syntax for expressing relationships between concepts, this determination to conform with other XML standards drove a decision to adopt the XLink syntax even before the XLink standard <<http://www.w3.org/TR/xlink>> moved beyond candidate recommendation status at the W3C.

This early and comprehensive adoption of the XLink standard distinguished XBRL from a number of other XML grammar specifications, such as XHTML <<http://www.w3.org/xhtml1>> and SVG <<http://www.w3.org/svg11>>, which were more explicitly within scope for XLink during the XLink design process.

Even while XBRL has moved forward to embrace the generality of the XLink syntax for relating information, other XML standards, such as XHTML, XML Schema, and Docbook <<http://www.docbook.org>>, have expressed reservations <<http://www.w3.org/2003/01/16-tag-xlink>> about the suitability of XLink for their own purposes.

This document explores the motivation behind the use of XLink by XBRL and its consequences:

- Section 1, “The XLink Standard,” covers the relevant features of the XLink standard that make it suitable for expressing relationships in XBRL.
- Section 2, “Linkage in XBRL,” explores all the different ways in which XBRL uses XLink. It interprets XBRL as a framework for developing XML standards where the syntax of the standard is constrained using XML Schema, and the semantics for the standard are described using XLink.
- Section 3, “Tips and Tricks,” covers a small number of tips for getting along well with XLink.
- Section 4, “Related Materials,” supplies references to related topics.

If you are looking for explicit guidance on the syntax of XBRL linkbases or the most appropriate ways to use that syntax to express your own XBRL taxonomies, then we recommend that you review *Understanding XBRL: A Technical Guide to Working with XBRL 2.0 and XLink* <www.kpmg.com/xbrl/xlink2.pdf>, which covers these issues exhaustively.

THE XLink STANDARD

The XLink standard was developed by the W3C as a syntax for linking XML documents and other internet resources. It also defines some of the behavior for XLink-compliant applications. Most simply, these kinds of links are already being expressed using anchor tags, <a>, and source tags, <src>, in the billions of HTML documents that make up the World Wide Web.

As the emergence of XML spawned a plethora of XML grammars, there was a recognition that if each grammar used a different syntax to document links between XML fragments*, then links will be harder to recognize and process.

Tim Berners-Lee captures this motivation for the XLink standard in his perspective on when it is appropriate to use XLink <<http://www.w3.org/DesignIssues/XLink.html>>:

"...you should use XLink whenever your application [of XML] is one of hypertext linking, as XLink functionality such as power to control user interface behavior on link traversal is useful and should be implemented in a standard way to allow interoperability...When an application uses functionality that is within the scope of XLink, it should use XLink. To do otherwise breaks the principle that we are trying to make an interoperable web."

The XLink standard finesses this design, providing the ability to navigate a web of associations between fragments of XML documents. It does so by defining a set of global XLink attributes that can be used on XML elements defined within any XML namespace <<http://www.w3.org/TR/REC-xml-names/>>. These XLink attributes identify the XML elements as participating in either a simple link between two Internet resources or an extended link associating two or more Internet resources.

Understanding the motivation for XLink helps us understand how it meets several XBRL requirements. In all situations where XBRL has identified a need for links to be established between different fragments of XBRL documents, XLink has been the first stop in the search for an appropriate mark-up syntax.

* If you are having trouble with the XML-related terminology in this article, you may find it useful to refer to the XML FAQ <<http://www.ucc.ie:8080/cocoon/xmlfaq>> maintained by Peter Flynn, which provides an easy introduction to the terminology that comes with the XML territory.

LINKAGES IN XBRL

Links turn up in all sorts of guises in XBRL: some fit closely with the original scope of XLink and some push the boundaries of XLink usage.

LINKS BETWEEN SCHEMAS AND LINKBASES

Simply put, XBRL requires links to navigate from XML Schema documents that describe the syntax for XBRL items and tuples to the XBRL linkbase files that document the semantics that must go hand in hand with the syntax. These links from schema documents to linkbase documents align closely with the scope of XLink.

In this situation, the "simple link" syntax of XLink is used to navigate from a schema document to a linkbase document. This is exactly the kind of scenario envisaged by Tim Berners-Lee, the inventor of the Internet, in his essay about appropriate use of XLink and corresponds very closely to the kind of navigation that users experience with HTML hyperlinks.

LINKS BETWEEN FACTS IN INSTANCE DOCUMENTS

XBRL also has straightforward requirements for connecting facts in one or more instance documents with an XML fragment that describes the commonalities between the connected facts.

For linking facts in instance documents, the extended link syntax of XLink is used in the form of a "footnote" linkbase. The extended link syntax is used because the relationships in instance documents can sometimes be among several facts, rather than from one fact to some documentation about that fact.

Footnote links enable users to navigate between related facts and to view the documentation that conveys the "sense" in which the facts are related. As with links from schema documents to linkbase documents, this use of XLink in XBRL fits closely with its design principles.

LINKING MEANING TO ITEMS AND TUPLES

XBRL has yet another linking requirement, one that causes much more trouble to those trying to understand how to create and work with both XBRL taxonomies and XBRL instance documents. This requirement is to link the semantics for a given XBRL taxonomy with the syntax for that taxonomy.

While the simple links from XML Schema documents to linkbase documents enable XBRL taxonomy users to find linkbase documents, they do not provide sufficient granularity to determine which parts of the linkbases provide the semantics for which XBRL item and tuple elements.

To fully understand the need for linking at this more granular level, it is worth spending a short while coming to grips with the idea that XBRL really is a framework for developing new XML vocabularies in a standardized way.

XBRL Standardizes XML Standards

Each XBRL taxonomy defines an XML standard. It does so in the increasingly traditional way by:

- Producing a set of XML Schema documents setting out the constraints on the XML format that must be adhered to by documents conforming to the defined standard
- Providing a comprehensive set of documentation explaining how the syntax defined in the schemas is to be interpreted.

That said, XBRL takes a number of steps to standardize:

- The kinds of syntax that can be defined in the schema documents (via a set of XML Schemas that form a normative part of the XBRL specification)
- The way in which humans and computers can expect to locate and understand the semantics behind the syntax.

Capturing the Syntax in XML Schemas

While XBRL taxonomy developers can exploit the full power of XML Schema, they can use the XBRL approach to semantics only for elements that are defined to be in the substitution group for XBRL item elements or XBRL tuple elements. If the declaration that one element (A) is in the substitution group of another element (B), then element A can be used as a substitute for element B in instance documents.

More precisely, XBRL only gives guidance on how semantics should be provided for elements that are in the item substitution group or the tuple substitution group.

The XBRL instance schema <<http://www.xbrl.org/2001/xbrl-instance.xsd>> defines the item element as:

```
<element
  xmlns="http://www.w3.org/2001/XMLSchema"
  name="item"
  type="anySimpleType"
  abstract="true"/>
```

The "anySimpleType" is a primitive data type, defined in the XML Schema specification, allowing any simple content within item elements and any configuration of attributes on item elements. It explicitly rules out elements that contain child elements. If this is too restrictive, it is possible to define the element as being in the tuple element substitution group instead.

The core XBRL instance schema <<http://www.xbrl.org/2001/xbrl-instance.xsd>> defines the tuple element as:

```
<element
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xbrli="http://www.xbrl.org/2001/instance"
  name="tuple"
  type="xbrli:tupleType"
  abstract="true"/>
```

where the tupleType is the following complex type definition also provided by the XBRL instance schema:

```
<complexType
  xmlns="http://www.w3.org/2001/XMLSchema"
  name="tupleType">
  <choice minOccurs="0" maxOccurs="unbounded">
    <element
      ref="xbrli:item"
      minOccurs="0"
      maxOccurs="unbounded"/>
    <element
      ref="xbrli:tuple"
      minOccurs="0"
      maxOccurs="unbounded"/>
  </choice>
  <anyAttribute
    namespace="##any"
    processContents="lax"/>
</complexType>
```

The tuple element definition is more intricate than that for the item element because its data type is "complex" in the sense that it allows children elements. Specifically, a tuple element can have any attributes and it can contain zero or more tuple or item elements (or elements that are in their substitution groups).

If you use XBRL to define the syntax of an item or tuple and to associate meaning with that syntax, then you have to use a syntax made up of items that contain only unstructured content or tuples that contain only other XBRL items or tuples. You cannot use XBRL to develop a syntax that includes elements with mixed content (allowing both child elements and child text nodes).

This is not to say that you cannot have such syntax definitions in XML Schemas that also include XBRL element definitions. It just says that you cannot use the XBRL approach to define semantics for elements with mixed content. Instead, you have to make up your own way of defining such meaning and communicating it to software developers.

This is not quite the full story. XBRL recognizes that to fully understand the meaning conveyed by the value of each element defined in your XML standard, the value generally needs to be accompanied by a set of contextual material. XBRL provides a very rigid, complex content model for this contextual material, enabling each element to be used in conjunction with a set of context describing the:

- Entity (and the segment if the entity is not sufficiently granular) being described by the value
- Period of time over which the value has been measured
- Scenario of the value.

When deciding whether it is sensible to develop an XML language within the XBRL framework, one of the best guides is an assessment of whether each of the unstructured pieces of information being conveyed by your language needs this kind of context to be understood.

Capturing the Semantics in XLink Linkbases

Just as XBRL standardizes the syntax structures in XBRL-based XML vocabularies, it also standardizes how the semantics of the syntax are to be found and processed.

Standardization of the way in which the language semantics can be found and understood makes it much easier to write applications that can work across and respond appropriately to the semantics for all XML languages that use the XBRL framework.

As a simple example, consider an application that needs to determine totals from a number of values contained in XML documents. The application does not know what values are going to be processed or what XML syntax is going to be used. It just has to find all values in an XML document that are supposed to be summed, add them, and report the totals. If the values are reported using XML syntaxes that are not based on the XBRL framework, the application needs to incorporate an understanding of:

- The syntax or syntaxes that have been used for each value that could contribute to a total
- What values are to be added together.

All of this understanding generally needs to be constructed by humans when putting together the functional requirements for the application.

In contrast, when the values are reported using XML syntaxes that conform to the XBRL specification, the application only needs to have:

- The XBRL item specification built into it (so that it can find values that may contribute to a total)
- The XBRL syntax for expressing the semantics of additive relationships between the values of elements.

Entirely new XBRL-based XML languages can be developed after the application has been written, and the application can still be expected to understand the syntax enough to find appropriate values and to understand the additive semantics of that language. In this way, applications can be future-proofed against innovations in certain classes of XML languages!

XBRL achieves this future-proofing by providing a formal XML language that can be understood by computers and in which the semantics of future XML languages can be expressed. If you conform with XBRL by using that syntax for your semantics, you have already taken a substantial step toward widespread application support of your new XML language. That is a major benefit of conforming to the XBRL framework.

Capturing Semantics with XLink

We have established that XBRL is a framework for developing XML languages, defining the syntax using XML Schema, and associating the syntax with appropriate semantics for each XBRL item and tuple definition. The need for this association sounds like another case for linking—this time linking the element syntax definition in the schema documents with XML mark-up that describes the semantics.

XBRL provides a broad range of XLink extended links to meet this requirement.

Semantics in Extended Links Containing Resources

Some of these extended links, such as the label and reference linkbases, contain XLink “resources,” which are the semantics, or provide references to other documentation and resources, which define the information that can be validly tagged with the XBRL elements in the taxonomy.

Looking more closely at the label linkbase, we can see that it provides labels for each of the item or tuple elements in XBRL taxonomies. These label resources are text that helps to document the content of XBRL elements for users.

For example, an element defined as:

```
<element
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xbri="http://www.xbrl.org/2001/instance"
  name="totalAssets"
  id="id_134"
  substitutionGroup="xbri:item"
  type="xbri:monetaryItemType"
/>
```

may be given a label “Total assets,” which is an English label that can be used to present information about this element when it is being used as a total of other element values.

Label resources are differentiated by their language (captured by an XML language attribute) and by the role of the label (captured by the XLink role attribute on the label resource). In the example above, the label would have an English “en” language attribute and could have an XLink role value of “http://www.xbrl.org/linkprops/label/total,” which distinguishes labels for totals from other kinds of labels.

The latest draft of the XBRL specification sets out the way in which language codes are defined (based on the standard XML language attribute) and meanings for a broad range of label roles ranging from presentation-purpose text strings, such as the one above, to concept documentation and concept definition purpose text strings.

The broad range of labels that can be associated with a given XBRL element, some of which may not even be known at the time the element's schema definition is created (e.g., Turkish language labels) make it necessary to use extended links to relate labels and the corresponding schema element definitions. As a result, label linkbases include not only the label resources themselves but also a collection of locators pointing to XML Schema element definitions for items and tuples and a collection of arcs that connect the label resources with the element syntax-definition locators.

Reference linkbases provide semantics using a mechanism much like label linkbases, except that the reference resources provide "references" to authoritative literature that defines valid content for the elements that are linked to them.

Note that both reference and label linkbases are aimed at providing structured semantics that can be found and presented to humans by computers, but cannot be understood by computers. This is a very important distinction. All of the other linkbases actually provide meaning that a computer can process and respond to appropriately.

Again, we see XBRL helping software developers to produce applications that are future-proofed against the creation of new XML vocabularies, as long as those vocabularies conform to the XBRL framework.

Using XLink to Create Meaningful Concept Relationships

Some of these extended links, such as the definition, calculation, and presentation linkbases, establish particular kinds of relationships between elements.

These extended links, without resources, also document semantics in a way that XBRL-enabled software can interpret without customization for specific XBRL taxonomies.

The presentation linkbase documents both parent-child relationships between elements and sibling-element ordering information. This information facilitates consistency in the presentation of the same materials across different XBRL applications.

The calculation linkbase documents summation relationships between elements in a way that enables XBRL software to understand and implement element value derivations from the values of other elements.

The definition linkbase provides semantic information about a number of inter-element relationships at the level of the definition of their valid content (as also described in the label and reference linkbases). These include relationship types that describe:

- Equivalencies between linked elements at the level of the definition of their valid content (e.g., element A content is always valid content for element B, and vice versa)
- Generalization-specialization relationships between the definitions of valid content for the elements (e.g., element A content is always valid content for element B, but the reverse does not hold).

TIPS AND TRICKS

Having explored the topic of XLink usage by XBRL in detail, it should be clear that there is some complexity involved, even if the payoffs are adequate compensation.

This complexity brings with it a small number of pitfalls that XBRL users should take care to avoid. Most of these issues affect those in the business of creating XBRL taxonomies (and those writing software to help them).

This section discusses some of the common issues and looks at how best to deal with them.

KEEPING SEMANTICS CONSISTENT

We know that linkbases provide semantics for XML languages based on the XBRL framework. We also know that the linkbases are conveying information to both humans and computers. Some linkbases are better at conveying semantic information to humans and some are better at conveying semantic information to computers.

This makes it extremely common to find redundant information in linkbases. For example, a careful reading of labels and reference materials for two elements could establish to a human that they have definition-equivalent content. However, no computer can ascertain this information from labels and references. To inform XBRL software of this equivalency, it is important to document the equivalency in the definition linkbase.

With information being documented in several ways and in several places, there is also the potential for errors and semantic conflicts.

Extreme care should be taken to synchronize the semantics embodied in label and reference linkbases with the semantics embodied in the other linkbases that will dictate the actions of computer software.

AVOIDING ANONYMOUS SCHEMAS

XBRL does not restrict the way that XML Schema can be used. Some features of XML Schema, however, should be used with caution. This general statement is particularly important in regard to “anonymous schemas,” which are schemas that do not declare a target namespace.

Anonymous schemas are useful in situations where the same XML grammar constructs are appropriate for a range of XML languages. By including the anonymous schemas in the schemas for each of those XML languages, the duplicate grammar constructs can be defined in one place and reused.

In the above discussion about the value of anonymous schemas, note that nothing was said about duplicate grammar constructs carrying the same meaning. Elements with the same syntactical structure but with different namespaces can be defined using anonymous schemas and given completely different meanings.

The XML Namespace specification `<http://www.w3.org/TR/REC-xml-names/>` directs XML parsers and XML validation software to treat such elements as unrelated in meaning and purpose, despite their identical grammatical structure. To further the point, XML software will not recognize any connection between elements when they have different namespaces. The meaning of the element is always associated with the element name and namespace, not the element name and the XML document in which its syntax is defined.

Now think back to what XBRL is doing with XLink when it associates semantics with the XBRL items and tuples defined in XML Schemas: It is creating links between XML documents. Specifically it links the XML fragment that is the XML Schema element definition with the XML fragments that describe the meaning of that element.

More generally, XLink is an effective standard for linking specific XML fragments that are identified by XPointer expressions. XPointer expressions operate with URLs rather than with URIs.

This can be a problem because all other XML specifications define meaning for elements within a namespace rather than within a specific XML Schema document at a specific URL. Most of the time there is a one-to-one relationship between an element namespace and the XML document in which its syntax is defined. This rule breaks down only when schemas are anonymous.

By associating meaning at the document level, XBRL forces the XBRL item and tuple elements declared in an anonymous schema to have the same semantics across all namespaces that include them. Worse still, while XBRL imposes this restriction, the design of XML and XML Namespaces is such that this equivalency in meaning will not carry through to generic XML software that users might expect would treat the elements identically.

This means that processing XBRL calculation links for an instance document, where the same element with the same semantics can have a variety of namespaces, becomes much harder. All the common off-the-shelf (COTS) XML software is geared toward searching for elements based on their element name and namespace: One meaning—one name/namespace pair. Using XBRL with anonymous schemas would cause this approach to fail. Finding the right elements would become a complex analysis of schema hierarchies and linkbase structures.

How do we dodge this pitfall? The only answer right now is to avoid using anonymous schemas to define elements that are in the XBRL item or tuple substitution groups.

WATCHING THE XPointers FOR LOCAL TAXONOMY CACHES

A final pitfall in the context of XBRL and XLink is the potential for broken XPointers in the XLink simple and extended links when taxonomy documents need to be cached locally. This is a problem that arises generally with inter-document links and XML, but it is exacerbated in XBRL where:

- XLink leads to a proliferation of XPointer expressions
- Taxonomy documents are often large enough to necessitate local caching to ensure acceptable software performance.

During the processing of taxonomies, numerous URL references need to be handled. Many will have fragment identifiers. In addition, these URL references can come in the form of absolute or relative URLs, from within documents that may or may not contain an XML base element. These documents will be served from originating servers across the Internet with varying cache control directives in place. A typical XBRL processing scenario may involve several such servers.

Faced with this situation, the software developer needs to protect all the applications that are part of the XBRL processing pipeline from the vagaries outlined above in a consistent way. Local caching of the taxonomy documents is almost mandatory. However, there are several ways to implement this requirement.

One solution is for the pipeline to rely upon the existence of a caching service, such as a proxy server. In this scenario, the XBRL applications and the underlying libraries are not modified significantly. They assume that the issues are all resolved by another service in a robust and consistent way that is always available. In a production server environment, this may be the case.

Another solution is to use caching services that may be available as part of the functionality of the development libraries for handling Internet-related services. Each individual application would have to be carefully written or configured to do this in the identical fashion. Even assuming

RELATED MATERIALS

that these functions are available, this is a relatively brittle solution as the number of applications expands. A cache that persists across all the applications in the pipeline is much more useful.

Between these two choices lies the possibility of customizing the middleware to handle XBRL documents specifically with a lightweight caching mechanism. Customizing the URL resolution strategy is such an approach. As a modification to a library, it can be implemented with minimal impact on the source of each application. It can address issues such as ignoring fragment identifiers that standard functionality (in proxy caches or development libraries) does not.

Most XML parsers and some XSL processors provide hooks for users to specify the URL resolution method they wish to use.

Unless you take this extra step to establish a robust XBRL taxonomy-processing environment, you will need to be able to ensure:

- Permanent access to taxonomy document primary locations
- Unchanged relative file locations for all taxonomies
- Plenty of bandwidth.

The following materials may help enhance your understanding of XBRL and XLink.

KPMG's Knowledge Base

- *Understanding XBRL: A Technical Guide to Working with XBRL 2.0 and XLink* <www.kpmg.com/xbrl/xlink2.pdf>
- KPMG's XBRL taxonomy and instance document tutorial <www.kpmg.com/xbrl/kkb.asp>

Other Resources

- *When to Use XLink* by Tim Berners-Lee <<http://www.w3.org/DesignIssues/XLink.html>>
- Docbook <<http://www.docbook.org>>
- Documentation of the scope of the XLink specification <<http://www.w3.org/2001/tag/ilist#xlinkScope-23>>
- Minutes of the January 16 meeting on linking in XML organized by the W3C Technical Architecture Group to discuss the scope of the XLink specification <<http://www.w3.org/2003/01/16-tag-xlink>>
- URI definition <<http://www.ietf.org/rfc/rfc1738.txt>>
- URL definition <<http://www.ietf.org/rfc/rfc1738.txt>>
- XBRL specification <<http://www.xbrl.org/resourcecenter/specifications.asp?sid=22>>
- XLink design considerations <<http://www.w3.org/DesignIssues/XLink.html>>
- XLink requirements documentation <<http://www.w3.org/TR/NOTE-xlink-req>>
- XLink specification <<http://www.w3.org/TR/xlink>>
- XML specification <<http://www.w3.org/TR/REC-xml>>
- SVG specification <<http://www.w3.org/TR/svg11>>
- Xpath 2.0 specification <<http://www.w3.org/TR/xpath20>>
- XHTML specification <<http://www.w3.org/TR/xhtml1>>
- XML Namespaces specification <<http://www.w3.org/TR/REC-xml-names/>>
- XML Schema specification <<http://www.w3.org/XML/Schema>>