XBRL Technical Primer

By

Charles Hoffman, CPA (Charles.Hoffman@me.com)
Raynier van Egmond (raynier@xbrlcp.com)

Last Revised – October 31, 2018 (DRAFT)

Executive summary:

- This document provides a high-level introduction to the XBRL technical syntax with just enough details to help you get your head around what XBRL actual is and how it works.
- The average professional accountant will never need to understand or work with XBRL at this technical, detailed level. However, if they want to understand some of the details this is a good document to read.
- This document is useful to both technical professionals and business professionals that have a little technical background that do want to understand the details and see how XBRL actually works.
- Throughout the document we point you to additional resources that help you build on the solid base that you learn from this high-level introduction.
- At the end of the document is a set of next steps for business professionals and for information technology professionals.
Copyright (full and complete release of copyright)

All content of this document is placed in the public domain. I hereby waive all claim of copyright in this work. This work may be used, altered or unaltered, in any manner by anyone without attribution or notice to me. To be clear, I am granting full permission to use any content in this work in any way you like. I fully and completely release all my rights to any copyright on this content. If you feel like distributing a copy of this work, you may do so without attribution or payment of any kind. All that said, attribution is appreciated should one feel so compelled. The copyrights of other works referenced by this document are established by the referenced work.
In this document we take a look at the physical and logical details of XBRL and provided an over-arching framework that is helpful in pulling the details together. The goal is to provide you with a high level understanding of XBRL and a path to greater understanding. We expose XBRL to you in layers with each level building on the previous. We are not going to teach you specific software applications in this document, you are on your own for finding the software you need.

We will not be covering the subtleties and nuances of XBRL here. We do provide a foundation upon which to build. You do not need to understand accounting or financial reporting to understand this document; but because many people understand the basics of a financial report and because this is targeted mainly at professional accountants and because I am a professional accountant; I will be providing a lot of financial reporting related examples.

**Hello World!**

Everyone these days knows what a hello world example is. We start our primer with a hello world example of XBRL. Consider this graphic below:

<table>
<thead>
<tr>
<th>Property, Plant and Equipment, by Component [Line Items]</th>
<th>Period [Axis]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2010-12-31</td>
</tr>
<tr>
<td>Property, Plant and Equipment, Net [Roll Up]</td>
<td></td>
</tr>
<tr>
<td>Land</td>
<td>5,347,000</td>
</tr>
<tr>
<td>Buildings, Net</td>
<td>244,508,000</td>
</tr>
<tr>
<td>Furniture and Fixtures, Net</td>
<td>34,457,000</td>
</tr>
<tr>
<td>Computer Equipment, Net</td>
<td>4,169,000</td>
</tr>
<tr>
<td>Other Property, Plant and Equipment, Net</td>
<td>6,702,000</td>
</tr>
<tr>
<td>Property, Plant and Equipment, Net, Total</td>
<td>295,183,000</td>
</tr>
</tbody>
</table>

What do you see? You see five items that add up to a total. Information is shown for two years. You can tell that this is a total because of the single and double underscores above and below the total value.

The screen shot above was generated using XBRL. You can fiddle with that XBRL-based report using this rendering which was generated from the XBRL¹. What we are going to do is explain to you how the information above was represented in XBRL. In order to enable a computer work with the information, we need to put the information into machine-readable form. We

---

¹ Hello World example rendering, [http://xbrlsite.azurewebsites.net/2018/HelloWorld/evidence-package/#Rendering-PropertyPlantAndEquipmentByComponent-pattern_PropertyPlantEquipmentByComponentTable.html](http://xbrlsite.azurewebsites.net/2018/HelloWorld/evidence-package/#Rendering-PropertyPlantAndEquipmentByComponent-pattern_PropertyPlantEquipmentByComponentTable.html)
will use the XBRL technical syntax. XBRL is a type of XML\textsuperscript{2}. Don’t worry about that now, just know that XBRL is XML.

The following is a taxonomy schema that is used as we begin to represent the information from that graphic above in machine readable form.

```xml
<?xml version="1.0" encoding="utf-8"?>
<schema
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:xbrli="http://www.xbrl.org/2003/instance"
  <element
      id="pattern_Land"
      name="Land"
      type="xbrli:monetaryItemType"
      substitutionGroup="xbrli:item"
      xbrli:periodType="Instant" />
  <element
      id="pattern_BuildingsNet"
      name="BuildingsNet"
      type="xbrli:monetaryItemType"
      substitutionGroup="xbrli:item"
      xbrli:periodType="Instant" />
  <element
      id="pattern_FurnitureAndFixturesNet"
      name="FurnitureAndFixturesNet"
      type="xbrli:monetaryItemType"
      substitutionGroup="xbrli:item"
      xbrli:periodType="Instant" />
  <element
      id="pattern_ComputerEquipmentNet"
      name="ComputerEquipmentNet"
      type="xbrli:monetaryItemType"
      substitutionGroup="xbrli:item"
      xbrli:periodType="Instant" />
  <element
      id="pattern_OtherPropertyPlantAndEquipmentNet"
      name="OtherPropertyPlantAndEquipmentNet"
      type="xbrli:monetaryItemType"
      substitutionGroup="xbrli:item"
      xbrli:periodType="Instant" />
  <element
      id="pattern_PropertyPlantAndEquipmentNet"
      name="PropertyPlantAndEquipmentNet"
      type="xbrli:monetaryItemType"
      substitutionGroup="xbrli:item"
      xbrli:periodType="Instant" />
</schema>
```

Note the “name” in the XBRL in what is called the taxonomy schema above. Note that there is an element which has a “name” for every line in the graphic on the prior page. You see elements with the name “Land”, “BuildingsNet”, “FurnitureAndFixturesNet”, “ComputerEquipmentNet”, “OtherPropertyPlantAndEquipmentNet”, and “PropertyPlantAndEquipmentNet”.

An XBRL taxonomy schema is an XML Schema\textsuperscript{3}.

\textsuperscript{2}W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), \url{https://www.w3.org/TR/xml/}

\textsuperscript{3}
Those names are sort of ugly to work with, so let’s add labels to make the human readable rendering of the information easier to read. Those labels are created using what is called a linkbase. Below you see a linkbase of labels, called an XBRL label linkbase:

```xml
<?xml version="1.0" encoding="utf-8"?>
<linkbase
    xmlns="http://www.xbrl.org/2003/linkbase"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xlink="http://www.w3.org/1999/xlink">
    <labelLink xlink:type="extended" xlink:role="http://www.xbrl.org/2003/role/link">
        <loc xlink:type="locator" xlink:label="pattern_BuildingsNet" xlink:href="RollUp.xsd#pattern_BuildingsNet"/>
    </labelLink>
    <loc xlink:type="locator" xlink:label="pattern_ComputerEquipmentNet" xlink:href="RollUp.xsd#pattern_ComputerEquipmentNet"/>
    <loc xlink:type="locator" xlink:label="pattern_FurnitureAndFixturesNet" xlink:href="RollUp.xsd#pattern_FurnitureAndFixturesNet"/>
    <labelArc xlink:type="arc" xlink:arctype="http://www.xbrl.org/2003/arctype/concept-label" xlink:from="pattern_FurnitureAndFixturesNet" xlink:to="pattern_FurnitureAndFixturesNet_lbl"/>
    <loc xlink:type="locator" xlink:label="pattern_Land" xlink:href="RollUp.xsd#pattern_Land"/>
    <loc xlink:type="locator" xlink:label="pattern_OtherPropertyPlantAndEquipmentNet" xlink:href="RollUp.xsd#pattern_OtherPropertyPlantAndEquipmentNet"/>
    <labelArc xlink:type="arc" xlink:arctype="http://www.xbrl.org/2003/arctype/concept-label" xlink:from="pattern_OtherPropertyPlantAndEquipmentNet" xlink:to="pattern_OtherPropertyPlantAndEquipmentNet_lbl"/>
    <loc xlink:type="locator" xlink:label="pattern_PropertyPlantAndEquipmentNet" xlink:href="RollUp.xsd#pattern_PropertyPlantAndEquipmentNet"/>
</linkbase>
```

If you look closely at the information in the linkbase above, you can see the names of the concepts and you can see labels. The human readable labels are associated with the names of the taxonomy schema. Each name in the taxonomy schema has a human readable label associated with it in the XLink linkbase that is shown above.

Pretty ugly stuff, right? Well, it gets even uglier...because there are more linkbases. Again, looking at the human readable rendering of the information you see that there are mathematical computations in the information that is being represented. In this case, there is a common type of mathematical computation called a roll up. XBRL uses XLink, just like for the labels, to articulate the mathematical computation, the roll up in this case, that the information we are trying to represent has. You can see the XBRL calculation relations below:

---

3 W3C, W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures, [https://www.w3.org/TR/xmlschema11-1/](https://www.w3.org/TR/xmlschema11-1/)

4 W3C, XML Linking Language (XLink) Version 1.1, [https://www.w3.org/TR/xlink11/](https://www.w3.org/TR/xlink11/)
Unfortunately, there is one other ugly set of XLink linkbase relations we need to show you. Notice that the lines in that graphic of the property, plant, and equipment roll up are in a specific order. **XBRL presentation relations** are used to describe the ordering of the information you want to represent.
There is another linkbase, the **XBRL definition relations** linkbase which is more advanced in nature, we will explain that and a few other linkbases later.

Finally, we look at the XBRL instance.

```xml
<?xml version="1.0" encoding="utf-8"?><xbrl xmlns="http://www.xbrl.org/2003/instance"
    xmlns:xbrli="http://www.xbrl.org/2003/instance"
    xmlns:link="http://www.xbrl.org/2003/linkbase"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xbrldi="http://xbrl.org/2006/xbrldi"
    xmlns:brm="http://www.xbrlsite.com/Schemas/brm"
    xmlns:frm="http://www.xbrlsite.com/Schemas/frm"
    xmlns:iso4217="http://www.xbrl.org/2003/iso4217"
    <link:schemaRef xlink:type="simple" xlink:href="RollUp.xsd"/>
    <context id="I-2010">
        <entity>
            <identifier scheme="http://www.SampleCompany.com">SAMP</identifier>
            <segment>
                <xbrldi:explicitMember dimension="frm:LegalEntityAxis">frm:ConsolidatedEntityMember</xbrldi:explicitMember>
            </segment>
        </entity>
        <period>
            <instant>2010-12-31</instant>
        </period>
    </context>
    <context id="I-2009">
        <entity>
            <identifier scheme="http://www.SampleCompany.com">SAMP</identifier>
            <segment>
                <xbrldi:explicitMember dimension="frm:LegalEntityAxis">frm:ConsolidatedEntityMember</xbrldi:explicitMember>
            </segment>
        </entity>
        <period>
            <instant>2009-12-31</instant>
        </period>
    </context>
    <unit id="U-Monetary">
        <measure iso4217:USD/>
    </unit>
    <pattern:Land contextRef="I-2010" unitRef="U-Monetary" decimals="INF">5347000</pattern:Land>
    <pattern:Land contextRef="I-2009" unitRef="U-Monetary" decimals="INF">1147000</pattern:Land>
    <pattern:BuildingsNet contextRef="I-2010" unitRef="U-Monetary" decimals="INF">244508000</pattern:BuildingsNet>
    <pattern:BuildingsNet contextRef="I-2009" unitRef="U-Monetary" decimals="INF">366375000</pattern:BuildingsNet>
    <pattern:FurnitureAndFixturesNet contextRef="I-2010" unitRef="U-Monetary" decimals="INF">34457000</pattern:FurnitureAndFixturesNet>
    <pattern:FurnitureAndFixturesNet contextRef="I-2009" unitRef="U-Monetary" decimals="INF">34457000</pattern:FurnitureAndFixturesNet>
    <pattern:ComputerEquipmentNet contextRef="I-2010" unitRef="U-Monetary" decimals="INF">4169000</pattern:ComputerEquipmentNet>
    <pattern:ComputerEquipmentNet contextRef="I-2009" unitRef="U-Monetary" decimals="INF">5313000</pattern:ComputerEquipmentNet>
    <pattern:OtherPropertyPlantAndEquipmentNet contextRef="I-2010" unitRef="U-Monetary" decimals="INF">6702000</pattern:OtherPropertyPlantAndEquipmentNet>
    <pattern:OtherPropertyPlantAndEquipmentNet contextRef="I-2009" unitRef="U-Monetary" decimals="INF">295183000</pattern:OtherPropertyPlantAndEquipmentNet>
</xbrl>
```

The XBRL instance references the XBRL taxonomy. You can see the same names in the XBRL instance as was in the XBRL taxonomy. We will explain the technical details later.
So, those are the ugly technical details. There is great news! Professional accountants will never have to deal with those ugly technical details. Why? Because there is this other technical thing called a conceptual model.

You have used a conceptual model if you are an accountant and probably don’t even realize it. Electronic spreadsheets are broken down into the rows, columns, cells, sheets, and workbooks that make up the pieces of an electronic version of a spreadsheet. Those ideas came from the paper-based spreadsheet that likewise had rows, columns, cells, and sheets.

Now, in the explanation of the technical details of taxonomy schemas and linkbases I left a few details out to make the bigger picture easier to grasp. As I said, we will drill down into many of those details in this document. But to master all of the technical details will require more work that just reading this very basic primer.

But, let’s go through the technical details again but this time applying a conceptual model and see how much easier all this stuff is to read.

Remember the taxonomy schema and the label linkbase we talked about. Well, here that information is again in a nicer computer-generated rendering:
And, remember the mathematical relations represented by the XBRL calculation relations.  
Well, here are those in an easier to read form:

Finally, here are the XBRL presentation relations represented in an easier to read form:

A little more computer magic⁵; you take all those machine-readable details, run them through what is called an XBRL processor, and presto-chango; you get that nice looking, easy to read computer-generated rendering.  And remember, the information is also readable by machine-based processes.

⁵ YouTube.com, How XBRL Works, https://www.youtube.com/watch?v=nATJBPOiTxM (it really is not using magic)
Again, remember that I left some of the details out to help provide an easy to understand introduction to XBRL. One commercial software vendor provides a static rendering and other reports that explains the information in an XBRL document that I can provide here for you to fiddle with\(^6\). If you want to dig into the XBRL technical syntax, here is a ZIP file that provides the actual XBRL that is behind the human readable reports\(^7\).

And so what does all this mean? Well, if it is not obvious to you what this means, then I would encourage you to read through the document *Computer Empathy*\(^8\) which helps you understand how computers work and what it takes to get a computer to perform useful work. That will help you understand what XBRL means in detail.

But in a nutshell, what XBRL means is that information can be effectively structured in machine-readable form which will enable computers to perform useful work. Couple that with artificial intelligence technology, digital distributed ledgers, and triple-entry accounting; and you get an ability to automate accounting, reporting, auditing, and analysis tasks which were here-to-for impossible to effectively automate.

**Grasping the XBRL Framework**

The core of XBRL is the XBRL 2.1 Specification\(^9\). Additional modules build upon the base XBRL specification, providing additional functionality\(^10\). You choose whether to use other XBRL...


\(^7\) Hello World example machine readable XBRL, http://xbrlsite.azurewebsites.net/2018/HelloWorld/RollUp.zip


modules based on your needs. In this primer I will stick with the base XBRL Specification but will point out important modules of XBRL as deemed necessary.

The XBRL Specification provides a framework that divides XBRL into two main parts:

- **XBRL taxonomies**, which are XML schemas that define the concepts, articulate a controlled vocabulary, define relations between concepts, and define business logic used by XBRL instances.
- **XBRL instances**, which contain the facts being reported, along with contextual information for those facts which helps you distinguish one fact from another.

And so we break our initial explanation of XBRL into two parts: XBRL taxonomies and XBRL instances. An XBRL taxonomy has a lot more moving parts to it than an XBRL instance. But XBRL taxonomies and XBRL instances interact with each other. We put these two pieces of the XBRL framework together and explain how the two interact.

After we cover these key parts, we will drill into the pieces that make up an XBRL taxonomy and an XBRL instance. We give you the level of understanding of the important details that you need without overwhelming you with areas of XBRL you’d likely never run across.

A big part of the **XBRL framework** is the flexibility offered by XBRL’s extensibility. We show you how XBRL’s extension capabilities are provided by XBRL taxonomies and used by XBRL instances. After introducing the important key ideas, we drill into the key notions to expand your level of understanding in a few specific areas.

The two most important modules of XBRL to understand is the XBRL Dimensions\(^{11}\) specification and XBRL Formula\(^{12}\) specification. XBRL Dimensions defines a multidimensional model that can be leveraged by XBRL taxonomies. XBRL Formula allows those creating XBRL taxonomies and XBRL instances to create business rules for describing and verifying information against a description.

The XBRL Open Information Model\(^{13}\) and the XBRL Abstract Model 2.0\(^{14}\) describe the logical model of a business report.


\(^{12}\) XBRL International, *XBRL Formula 1.0*, [https://specifications.xbrl.org/work-product-index-formula-formula-1.0.html](https://specifications.xbrl.org/work-product-index-formula-formula-1.0.html)

\(^{13}\) XBRL International, XBRL Open Information Model 1.0, [https://specifications.xbrl.org/work-product-index-open-information-model-open-information-model.html](https://specifications.xbrl.org/work-product-index-open-information-model-open-information-model.html)

Fundamentals of XML for XBRL Users

As XBRL is an XML language, we need to tell you a little about eXtensible Markup Language (XML)\(^\text{15}\). XML is a language for creating markup languages, which XBRL is. XBRL is an XML language. XBRL uses other pieces of the XML family, including namespaces, XML Schema, XML Base, XLink, XPath, and XPointer.

If you happen to be knowledgeable about XML, you may find these tips useful:

- XBRL has an XML schema\(^\text{16}\), which defines how you create XBRL taxonomies and XBRL instances.
- Another schema describes how to create XBRL Linkbases\(^\text{17}\).
- If you’re familiar with traditional approaches to creating XML languages, when you first look at XBRL, you may get confused. XBRL doesn’t just build a traditional XML schema that explains how instances are to be created. What XBRL does is build a layer on top of what XML provides that is used to create a metalanguage, which is used to create in part something you may be familiar with, an XML Schema.
- In XBRL, a taxonomy schema (which is part of an XBRL taxonomy) serves the same function as an XML Schema. A taxonomy schema defines XML elements and attributes, just like XML languages normally do. But XBRL adds metadata needed for business reporting, such as period information and whether a concept is a point in time, for a period of time, and so on. This additional metadata is common in business reporting, so XBRL adds it to the schema. The taxonomy schema is the XML Schema, which articulates how to create XML instances in XBRL.
- Another part of an XBRL taxonomy is the linkbase. You can add metadata to the XBRL taxonomy via the use of linkbases. While not required, linkbases provide useful information for understanding an XBRL taxonomy and using an XBRL instance. Linkbases enable the expression of many kinds of semantics.
- XML Schema elements with a specific substitutionGroup attribute value are XBRL concepts. All XBRL concepts are global because of the way XML Schema substitutionGroup attributes work. This makes the elements of a taxonomy schema a flat list. You can use linkbases to add any number of hierarchies to that flat schema.
- An XBRL user should work with the help of an XBRL processor, not just an XML parser. In most cases, you’ll drive yourself mad if you try to use XBRL with only an XML parser.

---

\(^{15}\) W3C, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, https://www.w3.org/TR/xml/


XBRL Builds on Top of XML

XBRL is an approach to using XML\textsuperscript{18}. XBRL is actually more similar to RDF than it is to traditional approaches to using XML. XBRL is XML; XBRL uses the XML syntax. Therefore, XBRL can leverage the entire family of XML specifications\textsuperscript{19}. XML allows you to construct "trees". XBRL and RDF allow you to construct "graphs"\textsuperscript{20}. This primer assumes a familiarity with XML. If you are not familiar with XML, consider the book *Communicating with XML*\textsuperscript{21}.

XBRL expresses semantics (meaning, business logic) in a global standard format. XML only articulates syntax. You can create mechanisms using XML to represent meaning. For others to do what XBRL does with XML, you would basically have to reinvent what XBRL has already created. Because this meaning, the business logic can be expressed in a global standard format that meaning and business logic can be effectively exchanged.

XBRL allows validation against the expressed meaning and business logic also. Because the meaning (semantics) are expressed in machine readable form; it is possible to validate XBRL.

\textsuperscript{18} XBRL Builds on Top of XML, \url{http://xbrl.squarespace.com/journal/2009/4/25/xbrl-builds-on-top-of-xml.html}
\textsuperscript{19} Airi Salminen, *XML Family of Languages*, \url{http://users.jyu.fi/~airi/xmlfamily.html}
\textsuperscript{20} If you don’t understand the difference between a tree and a graph, please read the *Computer Empathy* document.
\textsuperscript{21} Airi Saminen and Frank Tompa, *Communicating with XML*, \url{https://www.amazon.com/Communicating-XML-Airi-Salminen/dp/1461409918}
instances against that expressed meaning or business logic. And XBRL has created global standard mechanisms for performing this validation, such as XBRL calculations, XBRL definition relations, and XBRL Formulas.

XBRL separates concept definitions from the content model. Typically with XML languages, the concept definitions and the content model are mixed together within an XML schema. This makes it difficult to reuse either the concepts or the concept definition. Further, XML provides you with only one implicit set of relations (because it has only one content model) and the definition of those relations is mixed with the definition of elements and attributes. XBRL, on the other hand, uses an atomic approach (flat XML content model) similar to RDF in defining concepts and moves the expression of relations away from the XML schema. This separation of concept and relation definition leads to the next benefit of XBRL, you can express more than one set of relations and each of those sets of relations can be explicitly identified as being for a specific purpose.

XBRL can express multiple hierarchies of explicit relations. Because XBRL separates concept and relation definitions, you can define more than one hierarchy of such relations. Further, the hierarchies of relations defined are explicit rather than XML's implicit content model. But, you must keep those XBRL hierarchies consistent in terms of meaning. One representation of meaning or business logic should not be inconsistent with another representation.

XBRL provides prescriptive extensibility. XML's greatest strength is also its greatest weakness. XML is extensible everywhere, in every direction. XBRL is extensible in a specific, prescriptive, and therefore predictable manner. As such, the extensibility is usable without modifying software for the extension. You can think of this as XBRL always having the same "shape".

XBRL provides a multidimensional models. The multidimensional provides a consistent model with flexibility in the right areas which provides for flexible presentation of information and the ability to "slice and dice" information. Although XML can be made to fit into a multidimensional model in many cases, doing so can be a struggle.

XBRL enables "intelligent", metadata driven connections to information. With XBRL, connections to information can be created by business users adjusting metadata rather than by requiring technical people writing code. As such, rather than building multiple point solutions, XBRL enables the creation of effective and efficient solutions that allow extendibility and don't require programming modifications to connect to new information or new information models. Again, this is because of the prescriptive manner of XBRL's extensibility, the predictable "shape" of XBRL is always the same. With XML, every new connection pretty much has to be enabled by a programmer writing code because XML only communicates technical syntax and does so at
the data level, not the meaning or business logic level, of information and because the shape of different implementations of each XML implementation can be so varied.

JSON is similar to XML in that it is a syntax. Some people prefer the JSON syntax over the XML syntax; but to use all the features offered by XBRL in JSON; just like if you wanted to use XML to do what XBRL does; to use JSON like XBRL can be used, you would basically need to reinvent XBRL in the JSON syntax.

**Exploring XBRL Taxonomy Parts**

XBRL taxonomies have various physical pieces and express concepts, resources, and relations (see Chapter 3). These pieces, described in the following sections, work together to provide the functionality you need to express the meaning of business information that is to be exchanged.

**Taxonomy schemas and linkbases**

XBRL taxonomies are comprised of two parts:

- **Taxonomy schemas** are the XML Schema part of the XBRL taxonomy. Taxonomy schemas contain concept definitions that take the form of XML Schema elements. For example, the business concept Cash may be an XML Schema element that has a name attribute value of “Cash” and other attributes.

- **Linkbases** are the XLink part of the XBRL taxonomy and are also XML documents. The term *linkbase* is an abbreviation for link database. Linkbases are physical things used to express a logical thing, networks. Networks come in two types: resource and relation. (We explain networks, resources, and relations in the upcoming section “Networks and extended links.”) Resource and relation networks are expressed in the XLink syntax in the form of what is called an extended link.

Extended links are basically containers that hold the data contained within linkbases. We don’t want to get into a big technical discussion of extended links here; if you want that, read the XLink specification²².

**Discoverable taxonomy sets**

A single XBRL taxonomy may be comprised of a set of multiple taxonomy schemas and linkbases. This set has a specific and important name in XBRL, the discoverable taxonomy set (DTS).

---

²² W3C, XML Linking Language (XLink) Version 1.1, [http://www.w3.org/TR/xlink/](http://www.w3.org/TR/xlink/)
A DTS is governed by discovery rules, specified by the XBRL Specification, that XBRL processors understand. A DTS can contain any number of taxonomy schemas and/or linkbases and can start from either a taxonomy schema, a linkbase, or an XBRL instance.

**Networks and extended links**

Networks are a logical aspect of XBRL expressed physically as a set of linkbases. Linkbases exist within the physical model and are collections of extended links. Extended links work slightly differently in XLink than they do in XBRL. In XLink, each extended link is physically separated. In XBRL, a role attribute is added to an extended link. A network is a collection of all the extended links of a specific type with the same extended link role. The set of linkbases that are of the same type and have the same extended link role is called a base set. An extended link role is nothing more than a unique identifier expressed as a role attribute of an extended link.

You use networks to separate and organize resources and relations. Networks have different extended link roles and different unique identifiers, which creates the separation. Network roles and extended link roles are exactly the same thing. The creators of XBRL taxonomies define these roles within taxonomy schemas.

**Resources and relations within networks**

Physically, networks come in two types categorized by the type of extended links they use:

- **Resource** networks provide additional information about a concept. The additional information is in the form of an XLink resource. Of the five standard types of linkbases, label and reference are resource linkbases, and they express resource networks. Resources can have different resource roles to help further categorize resources. An example of additional information is an English label, Property, Plant and Equipment, with a standard role that is associated with the concept of the name PropertyPlantAndEquipment within a taxonomy schema. An example of a reference is a URL to a company’s accounting manual or, say, some descriptive information about the chapter, section, subsection, or page of that manual.

- **Relation** networks express relations between concepts using XLink arcs. Of the five standard types of linkbases, presentation, calculation, and definition are relation linkbases, which they express as relation networks. Relations (expressed as an XLink arc) can have different arc roles to help further categorize relations. An example of a relation network is the presentation network with the network role Balance Sheet that indicates that the concept PropertyPlantAndEquipment is part of that Balance Sheet and related to Assets.
Identifying XBRL Instance Parts
Compared to XBRL taxonomies, XBRL instances are simple and straightforward. XBRL instances are single physical files that contain the following pieces: references to DTS information, facts, contexts, units, and footnotes. All these components help you use that business information on your terms.

Achieving Flexible Business Information Exchange
XBRL is a knowledge media. XBRL taxonomies and XBRL instances provide for business information exchanges, the fundamental objective of XBRL. Key relationships between the parts of XBRL explained in this document is critical to understanding XBRL. This understanding can make the difference between making XBRL meet your needs and fighting XBRL at every step along the way. This section explains these key ideas both logically and how they’re physically implemented.

Defining concepts and organizing with taxonomy schemas
Concepts are defined in taxonomy schemas, which are really XML schemas. The XBRL concepts are XML Schema elements. Basically, XBRL leverages XML Schema as the means of defining XBRL concepts. XBRL adds a few attributes to XML Schema elements.

Taxonomy schemas can point to other taxonomy schemas already defined. XBRL uses the XML Schema mechanisms import and include to combine sets of concepts defined in separate physical files. XBRL also provides mechanisms schemaRef and linkbaseRef for combining DTS components. All these taxonomy schemas become part of the DTS. These mechanisms allow you to build modular taxonomy schemas. Another discovery mechanism for taxonomy schemas is any reference from a linkbase to a taxonomy schema.

Taxonomy schemas can also point to resource or relation networks that likewise become part of the DTS.

Using networks to separate and organize sets of resources
Resource networks are physically contained in XLink linkbases in the form of extended links with a specific extended link role. A resource defined within these networks allows you to add information to the XBRL concepts defined in the taxonomy schema. XBRL provides two types of resource networks in the XBRL specification: labels and references. The XBRL modules add other types of resources, such as formulas and formatting information. You can add your own

---

23 Charles Hoffman, Understanding that XBRL is a Knowledge Media, http://xbrl.squarespace.com/journal/2017/1/16/understanding-that-xbrl-is-a-knowledge-media.html
types of resources using the Generic Linkbase specification. Taxonomy schemas or XBRL instances can connect a resource network to the DTS.

Separating and organizing sets of relations by using networks
XLink linkbases physically contain relation networks in the form of extended links with a specific extended link role. Relation networks allow you to associate one concept with another concept in various precise ways for many purposes. XBRL provides three types of relation networks in the XBRL specification: presentation, calculation, and definition. You can add your own types of relations using the Generic Linkbase specification. Taxonomy schemas or XBRL instances can connect a relations network to the DTS.

Exchanging facts with XBRL instances
XBRL instances contain the information that is being exchanged. That information is expressed in the form of facts. Each fact is associated with a concept from an XBRL taxonomy, which expresses the concept and either defines it or points to a definition of the concept external to the XBRL taxonomy by using one or more XBRL references. Concepts are associated with an XBRL instance by being part of the DTS. Other information in networks connected to the XBRL instance provides additional information that helps you understand and use the facts within an XBRL instance. Contextual information is also associated with facts to help further explain things like which entity and which period a fact relates to. Numeric information has additional contextual information called units. You can use XBRL footnotes (not to be confused with what an accountant calls a footnote) to communicate additional information, such as comments.

Gaining flexibility through extension
The ability to express concepts, resources, and relations is well and good, but if the XBRL taxonomy isn’t exactly what you want, guess what? You can change it. This characteristic is why XBRL is called the Extensible Business Reporting Language. One of the most powerful features of XBRL is that it’s dynamic when it needs to be. Although this extensibility is there if you need it, you don’t have to use it.

XBRL’s need for extensibility makes the physical syntax level seem somewhat confusing. All these separate pieces provide the required flexibility needed to make XBRL’s extensibility work as needed. The good news is that XBRL processors hide much of this physical syntax from you. The logical view of XBRL is much easier to work with.

You never change someone else’s taxonomy schema or linkbases, but you can create your own taxonomy schemas or linkbases, which add to or change things others have defined by

---

becoming part of the DTS. You can even nullify what exists in someone else’s taxonomy, rendering it unusable. As such, you can change the overall DTS, effectively virtually changing what you should never change physically. Further, the users of the XBRL instance information can clearly see the changes you’ve made because you communicated those changes in the form of an XBRL taxonomy that is part of the DTS.

**Achieving interoperability through validation**
Does all this stuff seem complex? Business reporting is complex; sometimes incredibly complex. The good news is that computers, not you, deal with this complexity. Software guides you through the process, makes sure that you follow the rules of XBRL, and otherwise helps you achieve your goals. Software verifies that everything is okay by checking to be sure that your XBRL instances and XBRL taxonomies don’t have any logical, structural, mechanical, or mathematical errors, a process known as validation. You can make a mistake in lots of different ways and places, so you can check for errors in many different ways.

**Demystifying the DTS**
XBRL requires everything in an XBRL instance to be explicitly defined. An XBRL instance must directly or indirectly physically reference all those XBRL taxonomies whose concepts, resources, and relations have bearing on the XBRL instance’s contents. The first thing an XBRL processor does when it encounters an XBRL instance is attempt to discover all its related XBRL taxonomies; if it can’t, it goes no further!

Schema files in XML and XBRL taxonomies work differently. The XBRL taxonomies must exist and be discoverable. XML can provide hints about the schema or point to a schema, or it may not provide a schema at all. Not with XBRL. Those schemas that are XBRL taxonomies must exist because the XBRL taxonomies enable you to understand the information in the XBRL instance. The graphic below shows how multiple taxonomy schemas and linkbases are brought together into a DTS. The XBRL instance is connected to a set of two XBRL taxonomies. Each XBRL taxonomy is comprised of one taxonomy schema and two linkbases. The DTS is the combined set of all taxonomy schemas and linkbases from both XBRL taxonomies. Note that an XBRL instance is never considered part of the DTS, but it can contain references to taxonomy schemas, XML Schemas, and linkbases that are part of the DTS.
Grasping the functioning of networks

When we talk about networks, we’re not talking about computer networks, broadcasting networks, or any other form of communication. Although network is clearly not an accounting or business term, it’s a way to describe something that does occur in business information. Networks have two important states:

- **Unresolved or raw form**: In XBRL jargon, this state is called the base set. A base set is the actual physical relations in what can best be described as an unresolved or raw state. For example, if a relation is expressed between two concepts, and then another relation is created to prohibit that relation, in the base set, two relations exist. These two relations are more the physical form of the actual relations. To better understand what we mean, contrast it to the resolved form.

- **Resolved form**: In XBRL jargon, this state is called a network or a network of relations. A network is the resolved form of the set of relations. If you have two relations; one that defines a relation and a second that prohibits that relation; the resolved form of this set of relations is no relation. The resolved form of the set of relations is more the logical result of all the physical relations.
Here’s an example of a network: Property, Plant and Equipment, Net is an accounting concept that would likely be designed in a calculation network in XBRL to denote their mathematical accounting relationship of the information represented within an XBRL instance:

\[
(=) \text{Property, Plant and Equipment, Net} \\
(+) \text{Land} \\
(+) \text{Buildings} \\
(+) \text{Furniture and Fixtures} \\
(+) \text{Other}
\]

The graphic shows that XBRL is saying that Property, Plant and Equipment, Net = Land + Buildings + Furniture and Fixtures + Other. To express relations in an XBRL taxonomy, you must have at least one network because all resources and relations exist within a network.

**When to use networks**

When you build an XBRL taxonomy, you express resources and relations within networks. Understanding why you’d use a network helps you understand what networks are. You separate relations into different networks for two general reasons:

- **Convenience:** The taxonomy creator may want to create several smaller networks rather than one big network. For example, in the US GAAP taxonomy, the information contained in the taxonomy is broken up into multiple networks. Another way that the taxonomy may have been created is by combining all those networks into one single network. Sometimes working with several smaller pieces rather than one big piece is easier. Networks allow you to create pieces, if you want.

- **Out of necessity:** Sometimes you *have* to create networks. Eliminating conflicts when one parent concept has two or more possible sets of child concepts is another reason to create a network. For example, it would be impossible to express these three different ways to arrive at the one value for Receivables, Net in one network:
  - Receivables, Net = Receivables, Gross minus Allowance for Bad Debts
  - Receivables, Net = Receivables, Net, Current plus Receivables, Net, Noncurrent
  - Receivables, Net = Trade Receivables, Net plus Financing Receivables, Net plus Other Receivables, Net, and so on

On the left side of the graphic below, nothing is physically separating the three ways to compute Receivables, Net. Humans can see this separation; we understand that they’re distinct. But how would a computer know that they’re separate? On the right side, see how networks physically separate the three different computations? Computers need to be told that the three computations are, in fact, separate. Otherwise, they’d collide. Networks provide that physical separation, communicating that separation to software applications.
Trying to express all three of these relations within one network would result in collisions in the relations. To get around this issue, you must use three different networks, one expressing each of the different calculation relations.

**Other important aspects of networks**

Here are some other important points about networks to keep in mind:

- You shouldn’t physically change networks others created, but you can create your own networks, hide original relationships to effectively nullify them, and redefine new relations. Networks exist within a set of physical files. You may not change someone else’s physical file. Instead, relations you create take precedence over the existing relations that existed, in effect creating new relations. You create new relations by prohibiting someone else’s relations, creating new relations, or both. This process, in effect, creates a new set of relations and also documents exactly how you’ve changed what previously existed.

- Although you may not change someone else’s taxonomy’s set of physical files, you don’t have to use the linkbase files that contain the relations. If you don’t refer to the linkbase files, for all practical purposes, the relations don’t exist in the context of the DTS because the file will never get discovered.

- You can’t order networks within XBRL. For example, say that you have three networks: Balance Sheet, Income Statement, and Cash Flow Statement. Users of an XBRL instance may use them in the order of Income Statement, Balance Sheet, and Cash Flow Statement. Although XBRL has no means for prescribing a specific order, you can use proprietary approaches to prescribe the specific order of networks. However, keep in
mind that they’re proprietary and shouldn’t be relied on outside of any specific piece of software that supports that proprietary approach.

- Users can fabricate a network to their liking without the creator of the XBRL instance being involved or otherwise consenting by using XBRL’s extensibility. As a result, XBRL is sometimes called interactive data. Users can make use of the XBRL instance information as they see fit, without consideration for how the creator of the information may want it used. However, when information users do reorganize information, they take on the responsibility of getting the relations correct.

### Drilling into Taxonomy Schemas

Taxonomy schemas are one of two parts of an XBRL taxonomy (linkbases being the other) for expressing the information model of XBRL instances. Taxonomy schemas have several pieces, each of which helps you express meaning relating to that information model. You don’t have to use all the pieces, but you can, if you want.

The important pieces of a taxonomy schema are concepts and pointers to other concepts, resources, and relations.

XML namespace prefixes add readability to XML markup. Although these prefixes aren’t required, they can be anything you want (clearly within the bounds of proper XML syntax). We use these namespace prefixes when we show you XML markup:

- `xs`: XML Schema namespace
- `xbrl`: XBRL Instance namespace
- `link`: XBRL Linkbase namespace
- `basic`: Taxonomy schema of our example

### Concepts

Essentially, taxonomy schemas are all about concepts, modeling the pertinent aspects of a collection of business concepts (controlled vocabulary) related to some exchange of business information. XBRL instances use concepts from a taxonomy schema to express fact values for the purpose of exchanging this information.

Concepts have the following characteristics expressed as attributes of XML Schema `xs:element` elements:

- **Name**: Every concept must have a name attribute. The name identifies the concept within a taxonomy schema.
• **ID:** Every concept may have an id attribute. If you’re using linkbases, you need that id when you add resources or express relations because this id is how the resource or relation is connected to the concept. The id does two things. First, it uniquely identifies a concept across all taxonomy schemas. Second, relations and resources use id’s (not names) as unique keys to create the connection between the relation or resource and the concept to which the relation or resource relates.

• **Type:** Every concept must have a type attribute. You may know the term as *data type* from other technologies. Data types constrain content, such as string, integer, or date. XBRL has similar primitive types, but it also has a number of more specialized types common to the business world, such as monetary or shares. You can also define your own types.

• **Substitution Group:** Every concept must have a substitutionGroup attribute, and the value of that attribute will most likely be xbrli:item, which means the element is an XBRL concept. Substitution groups can get pretty involved. Just realize that there is a lot to them and be careful how you use them. Ninety-nine percent of all the substitution groups you run across will be xbrli:item, which is straightforward.

• **Period Type:** Every concept must have an xbrli:periodType attribute. The xbrli:periodType provides information about the kind of period the concept relates to. Two values are most commonly used: instant and duration. An instant xbrli:periodType denotes a point in time. Conversely, a duration xbrli:periodType denotes a span of time between two instants. For example, a balance sheet reflects information as of a specific point in time, say December 31, 2008, and thus has a xbrli:periodType value of instant. An income statement reflects information for a span of time, such as “For the Year Ended December 31, 2008,” or a duration from January 1, 2008 to December 31, 2008, and thus has a periodType value of duration.

• **Balance:** Concepts may have a xbrli:balance attribute, but aren’t required to. If it does have a value, the value must be either debit or credit. If you’re an accountant, you get debits and credits. If not, you probably don’t have to deal with debits and credits, or you have an accountant helping you, so go ask them about debits and credits.

Here’s what a concept looks like within a taxonomy schema expressed as an XML Schema element:

```xml
<xsl:element
    name="NetIncomeOrLoss"
    id="basic_NetIncomeOrLoss"
    type="xbrli:monetaryItemType"
    substitutionGroup="xbrli:item"
    xbrli:periodType="duration"
    xbrli:balance="credit"
```
The preceding fragment expresses the concept NetIncomeOrLoss in the form of an XML Schema xs:element element within a taxonomy schema. The element has a number of attributes. Many of these attributes are provided by XML Schema, including the name, id, type, and substitutionGroup. The xbrli:periodtype and xbrli:balance are attributes added to XML Schema by XBRL. The values for the attributes are rather straightforward; refer to the list before the fragment.

Sometimes concepts are referred to as elements because they exist within the physical part of an XBRL taxonomy known as a taxonomy schema, which is an XML Schema. But not all XML Schema elements within a taxonomy are concepts. Single value XBRL facts always have a substitutionGroup value of xbrli:item, which is how you know they’re XBRL concepts.

**Pointers to other concepts, resources, or relations**

Another piece of the taxonomy schema that you may find are references to other taxonomy schemas and linkbases. You can create references using the following elements:

- **Import**: You can use an XML Schema xs:import element within a taxonomy schema file to refer to another taxonomy schema or XML schema file. Each imported XML Schema has a different namespace identifier.
- **Include**: An XML Schema xs:include element does something similar to the xs:import element. The difference is that included files pull elements into the namespace of the file containing the include element, and included files don’t contain a namespace identifier.
- **LinkbaseRef**: A link:linkbaseRef element or linkbase reference points to a linkbase, which contains networks of either resources or relations.

These pointers allow you to use taxonomy schemas, XML schemas, or linkbases others have created; let you modularize your taxonomy so that you or others can use some parts of your XBRL taxonomy without being forced to use other parts; and otherwise physically separate the files that make up your XBRL taxonomy.

**Drilling into Resource Networks**

Resource networks let you express additional information for concepts. Labels and references are two types of resource networks that the XBRL standard provides. The XBRL Formulas specification provides a third type of resource. This list explains these different types of resource networks:
• **Label**: Label resources, as the name suggests, allow the creator of an XBRL taxonomy to create labels for each concept in the taxonomy. Basically, they provide a more user-friendly label in place of the ugly XML element names. Another use for labels is to provide multilingual support and multidialect support. Labels also provide documentation for a concept, such as a human-readable definition of the concept.

• **Reference**: Reference resources allow the creator of an XBRL taxonomy to express references to external sources (such as a paragraph in a manual) that explain or further define a concept in human terms. References are pointers to a reference, not the references themselves.

• **Formula**: Not in the base XBRL Specification but added by the XBRL Formula Specification, formula resources allow the creator of an XBRL taxonomy to express various types of business rules. XBRL instances that make use of an XBRL taxonomy that contains these rules must comply with these rules.

You can express an infinite number of different types of resource networks. You can specify your own type of resource networks using the XBRL Generic Linkbase Specification.

Resources are connected to concepts in a taxonomy schema. Resources can have different resource roles, many of which XBRL predefines, but taxonomy creators can also define their own resource roles. These roles let taxonomy creators further categorize resources, if needed.

You can categorize resources by the preceding resource roles. This categorization helps when partitioning one type of resource from another. This resource role is different than the role of the network (extended link role), which contains the resources. Finally, remember that the resources are connected to the id element of a concept expressed within a taxonomy schema.

The documentation for each concept expressed in an XBRL taxonomy is commonly implemented as labels with a specific label role of “http://www.xbrl.org/2003/role/documentation”. XML Schema offers documentation in the form of an xs:appinfo element, but it’s not standardized, not extensible, is generally ignored by most XBRL processors, can’t be prohibited, and therefore is a bad idea to use.

**Drilling into Relation Networks**

You can use the following types of standard relation networks to express relations between concepts:

• **Presentation**: Presentation relations allow you to express a simple parent-child type of relationship, basically a hierarchy. Presentation relations are primarily intended to help
organize the XBRL taxonomy. You can also use presentation-type relations to help generate human-readable renderings of an XBRL instance.

- **Calculation**: Calculation relations allow you to express certain types of computations between concepts within an XBRL taxonomy. XBRL calculation relations handle only addition and subtraction, and concepts must be in the same XBRL context. This is many times helpful, but can also be somewhat limiting. Need more power in your computations? Then use XBRL Formulas.

- **Definition**: You can use definition relations for many purposes. Definition relations basically let you express any type of relation. You can define any arcrole, which explains what type of relation you’ve created. For example, the XBRL Dimensions Specification uses definition relations to express multidimensional type relations.

You can express an infinite number of different types of relation networks. You can specify your own type of relation networks using the XBRL Generic Linkbase Specification.

Relations and the concept in a taxonomy schema to which the relation relates are connected. XBRL processors understand this connection. Relations can have different arcroles, many of which XBRL predefines, but taxonomy creators can also define their own arcroles. Arcroles differentiate or categorize relations, should you have that need.

**Drilling into XBRL Instances**

XBRL instances are the physical documents that contain business information that is published, transferred, or otherwise exchanged. Here are the parts of an XBRL instance:

- Reference(s) to XBRL taxonomy(s)
- Contexts
- Units
- Facts
- XBRL Footnotes (that is, comments)
- Other technical odds and ends

The graphic below shows the parts of an XBRL instance and the relation to XBRL taxonomies that support the instance. The figure shows the parts of an XBRL instance and the relations between the parts on the left, the connection between the XBRL instance and the XBRL taxonomy, the relations between the XBRL instance and the XBRL taxonomy, and the relations within the XBRL taxonomy on the right.
References to XBRL taxonomies

XBRL instances always refer to an XBRL taxonomy or a set of XBRL taxonomies, collectively known as the DTS. Because the XBRL taxonomies make up the DTS that determines the controlled vocabulary and the other semantic meaning expressed by the XBRL taxonomies, you’ll want to know where to get those all those XBRL taxonomy pieces. The references to these physical files tell you where to get the XBRL taxonomy pieces that explain the controlled vocabulary the XBRL instance uses and all the other meaning expressed for the information model by the DTS.

An XBRL instance can refer to pieces of the DTS in two ways:

- **SchemaRef**: A link:schemaRef element, or schema reference, points to a taxonomy schema file or an XML schema file. Those files can contain references to even more files. These physical files may be components others created or components created by you that express new concepts, relations, or resources used by your information model or that modify the information of other XBRL taxonomies you refer to. Your modifications are commonly referred to as an *extension*.

- **LinkbaseRef**: A link:linkbaseRef element, or linkbase reference, points to a linkbase that contains relations or resources. The linkbases themselves generally reference the taxonomy schemas that contain the concepts associated with that linkbase.
You can connect linkbases directly to XBRL instances with a linkbaseRef. Alternatively, you can use a schemaRef to connect linkbases to a taxonomy schema, which then references one or more linkbases.

An XBRL processor is responsible for grabbing all the referenced XBRL taxonomy pieces that make up the DTS, putting all the pieces together correctly, and communicating any issues relating to that process to the user of the XBRL instance.

Here is an example schemaRef:

```
<link:schemaRef xlink:type="simple" xlink:href="Example.xsd"/>
```

The preceding code fragment shows a schema reference to the taxonomy schema with the name Example.xsd. A linkbase reference is similar, with the only difference being a different element name.

Software can add linkbases to a DTS dynamically by connecting the linkbase to the XBRL instance. For example, consider an XBRL instance that refers to an XBRL taxonomy that has only English labels in it. Suppose that someone else creates a linkbase that provides Spanish labels for that same XBRL taxonomy. Further, suppose that your native language is Spanish, and you’d prefer to use those Spanish labels, but the linkbase isn’t physically connected to the XBRL instance. No problem! All you have to do is load the XBRL instance, which will automatically load the DTS that the XBRL instance points to. Then you use your software to point to the linkbase with the Spanish labels. Voilà! You can use the Spanish labels! Pretty slick, eh? That is why XBRL is sometimes referred to as interactive data.

You can use the same approach to reorganize the way you view XBRL instance information. If you don’t like the way the creator of the XBRL instance or XBRL taxonomy organized the information, you can simply reorganize the instance information by creating your own XBRL taxonomy extension. You can then use your view, provided by your XBRL taxonomy, instead of the XBRL instance creator’s view of the information it contains.

**Giving facts context**

Contextual information helps users of the information in the XBRL instance understand the context in which the information is being used. Context is provided using the xbrli:context element, which contains

- **ID**: Every context within an XBRL instance has an id that uniquely identifies the context so that one or more fact values can refer to it. A context id attribute can look something like X10B72009. These IDs have no meaning, other than to uniquely identify the
context. They’re like an identification number. The meaning is expressed within the elements and attributes of the context.

- **Entity**: The `xbrli:entity` element is part of the context and has two pieces; the first is required, and the second is optional. The required piece is the `xbrli:identifier` element. The scheme attribute of the `xbrli:identifier` element communicates which identification scheme is being used. The value of the `xbrli:identifier` is the actual identifier from that scheme. For example, the U.S. SEC has a numbering system called a CIK number. XBRL instances going to the SEC use a pointer to that scheme, which may be something like http://www.sec.gov/CIK. The identifier value is the actual CIK number, such as 0000066740. The second and optional piece of information is the entity `xbrli:segment` element. You can put any information you desire into the `xbrli:segment` element in order to provide information that helps users gain needed contextual information to understand the fact value. Note that the XBRL Dimensions specification is a way to express this type of contextual information, and the use of XBRL Dimensions information in the `xbrli:segment` element is emerging as a best practice. However, any valid XML is allowable, other than XBRL. You can express an infinite number of elements within a segment.

- **Period**: The `xbrli:period` element communicates contextual information about the period to which a fact value relates. The `xbrli:periodType` attribute is defined within a taxonomy schema. The `xbrli:periodType` on a concept within the taxonomy schema must match the period used within a context used on that concept. The two common `xbrli:periodType` values are
  - **Instant**: If the XBRL taxonomy concept defines the concept as an instant, the element used within the period will be `xbrli:instant` and will contain a single date — for example, `<xbrli:instant>2009-12-31</xbrli:instant>`.
  - **Duration**: If the taxonomy concept says duration, two elements will be within the `xbrli:period`, `xbrli:startDate`, and `xbrli:endDate`. Each one has a value to communicate the starting and ending dates of the duration (date range). The start date may look like `<xbrli:startDate>2009-01-01</xbrli:startDate>`, and the end date may look like `<xbrli:endDate>2009-12-31</xbrli:endDate>`. There is one other possibility for period, but its usage is rare. The value `xbrli:forever` means no date is provided, and that context is always applicable.

- **Scenario**: The `xbrli:scenario` element works in a manner similar to the `xbrli:segment` element of the `xbrli:entity` context described earlier in this list. The `xbrli:scenario` element is optional. The `xbrli:scenario` element can contain any valid XML other than XBRL. One way of expressing this information is to use XBRL Dimensions, and it’s becoming a best practice to use this approach. Literally, you can put any type of contextual information into the `xbrli:scenario` element, such as whether information is
actual, budgeted, and so on. You can express an infinite number of elements within a
scenario.

Putting these pieces together, here’s an example of a xbrli:context element:

```xml
<xbrli:context id="D-2009">
  <xbbrli:entity>
    <xbbrli:segment>
      >>>>>>segment information would go here<<<<<<
    </xbbrli:segment>
  </xbbrli:entity>
  <xbbrli:period>
    <xbbrli:startDate>2009-01-01</xbbrli:startDate>
    <xbbrli:endDate>2009-12-31</xbbrli:endDate>
  </xbbrli:period>
  <xbbrli:scenario>
    >>>>>>scenario information would go here<<<<<<
  </xbbrli:scenario>
</xbbrli:context>
```

The preceding code sample shows the components a xbrli:context element from an XBRL
instance. The context has an id value of D-2009, which is the same value of the contextRef on
one or more facts, which we discuss in the upcoming section “Expressing the values for facts.”
The entity xbrli:identifier element has a value of SAMP, which comes from the scheme
http://www.Sample.com. The segment element in our example is empty (we just wanted to
show you were it would be located). The xbrli:period element has a duration (a period of time)
value with a start date of 2009-01-01 and an end date of 2009-12-31. As with the xbrli:segment,
the xbrli:scenario element values aren’t shown, as we don’t want to get into this topic at this
time.

**Units provide additional context for numeric facts**

In order to provide the proper context for a numeric value, you need to know the units for the
numeric values. For example, say that you have a value of 1,000. Is that some particular
currency, and if so, is it U.S. dollars, euros, yen, or some other currency? Or, is it square feet,
the number of employees, or something else?

An XBRL instance contains units whenever you have any numeric fact values. A xbrli:unit
element contains the additional context:
• **ID:** Every `xbrli:unit` element has an id attribute that uniquely identifies the unit so that numeric fact values can point to it. The id on the units works exactly the same way that a context id attribute works.

• **Measure:** Units generally have one `xbrli:measure` element, but they can have any number of measures. The value of the `xbrli:measure` element is the actual units for the fact value — for example, `iso4217:EUR`, which is the ISO currency code for the euro. If the fact value has no units, such as a percentage, the measure would have a value of `xbrli:pure`, which indicates that it’s a pure number with no real units.

Have a look at how you can express units within an XBRL instance:

```xml
<xbrli:unit id="U-Monetary">
  <xbrli:measure>iso4217:EUR</xbrli:measure>
</xbrli:unit>
```

The code sample of unit shows the unit element with an id attribute value of U-Monetary that would correspond to one or more fact unitRef attribute value, which ties the fact to this unit. The `xbrli:unit` element has a `xbrli:measure` element with the value of `iso4217:EUR`, which means that the ISO 4217 standard currency code value of EUR for euros is the unit of measure.

**Expressing the values for facts**

An XBRL instance’s meat is the fact that the XBRL instance is communicating. What a fact looks like can be slightly different depending on whether it’s numeric. Numeric facts need to indicate the units in which the numeric fact value is expressed and the decimal places of the numeric value.

These parts make up a fact in an XBRL instance:

• **Concept name:** Every fact is data for a concept defined by one of the taxonomy schemas within the DTS of the XBRL instance. For example, a concept name may be something like `gaap:CashAndCashEquivalents`. The concept name is actually not exactly the name from the taxonomy, but rather the qualified name (sometimes called the QName), which is the name plus what is called the namespace prefix of the taxonomy schema from which the concept came. A **namespace prefix** is basically shorthand for referring to a namespace identifier. The **namespace identifier** basically identifies the taxonomy schema. We know, boring technical stuff. But the technical people dig this sort of thing and make all these connections behind the scenes for you.

• **ID:** Facts can have an id attribute that uniquely identifies the fact, but it’s optional. Generally, the id isn’t really necessary on a fact.
• **Context reference**: Every fact has a contextRef attribute. The contextRef attribute refers to one of the id attributes of a context element defined in your XBRL instance. The context element contains information that’s necessary to understanding the context in which the fact value is being used.

• **Unit reference**: Every fact with a numeric concept has a unitRef attribute. The unitRef refers to one of the id attribute of one of the unit elements defined in your XBRL instance. Concepts like strings, dates, and a few other odds and ends don’t need unit information, so you don’t need to provide one for facts with those types. Your software can help you sort out whether units are needed on a fact.

• **Decimals**: Every fact with a numeric concept must have either a decimals or possibly a precision attribute. Both the decimals and precision attributes provide virtually the same functionality, explaining the number of decimal places to which the fact value is accurate. Decimals are preferred because most people to understand them more easily. You can specify, say, 2 as the decimals value to indicate that a fact value is accurate to 2 decimal places, or -3 to indicate that a number is accurate to thousands. Software applications generally guide you to the appropriate choice.

• **Fact value**: Facts have a value, which is referred to as the fact value. A fact value can be a number, such as 1000, a text, such as LIFO, or any other data type, including an entire narrative with several paragraphs of text. The taxonomy schema determines and enforces the data type.

Here is an example of a numeric fact:

```
<basic:NetIncomeOrLoss
  contextRef="D-2009"
  unitRef="U-Monetary"
  decimals="-3">5347000</basic:NetIncomeOrLoss>
```

The preceding XBRL instance fragment shows a fact with the value of 5347000. That fact expresses information for the concept basic:NetIncomeOrLoss. It points to a context with the contextRef of D-2009 and units information with the unitRef of U-Monetary. The numeric fact is accurate to –3 decimal places (in thousands), as you can see in the decimals attribute value of -3.

**Commenting with XBRL footnotes**

XBRL instances can also contain what are basically comments but are referred to in XBRL as *footnotes*. XBRL footnotes are used in the XBRL instance as comments or notations that refer to one or more fact values.
XBRL footnotes are effectively a resource network physically contained within an XBRL instance. The footnote network is expressed as a linkbase and operates just like any other resource type linkbase.

Footnotes are good to know about and useful if you have a special case where you need them, but in the grand scheme of things, they really aren’t that important.

Many people confuse XBRL footnotes with the term footnote as it applies to a financial statement. They’re not the same thing. What XBRL footnotes actually provide in terms of functionality, how people use them, and how software vendors implement them are rather inconsistent, and you should avoid using footnotes, if possible. When a specific regulator or someone does specify what they should be used for and how they operate, however, you should follow the guidance of that regulator.

**Gaining Flexibility through Extension**

Many businesses need XBRL-required information models to be dynamic (flexible), and XBRL was created to fulfill that need. XBRL lets you modify XBRL taxonomies, but you don’t have to if you don’t need to.

We use the term modified in a special way here. These modifications don’t change other people’s physical files, but rather those creating extensions add new XBRL taxonomy components that can virtually change what others have created. It also articulates how you changed it.

Here are two terms you need to know to understand how extension works:

- **Base XBRL taxonomy:** You can think of a base XBRL taxonomy as an XBRL taxonomy that another XBRL taxonomy is extending. Other terms sometimes used to refer to these taxonomies are standard taxonomies or anchor taxonomies. For example, a taxonomy mandated by a regulator may be a base taxonomy.

- **Extension XBRL taxonomy:** You can think of an extension XBRL taxonomy as an XBRL taxonomy that extends some base XBRL taxonomy. Other terms sometimes used to refer to these taxonomies are custom taxonomy or company taxonomy. Extension XBRL taxonomies are generally customizations created of some base XBRL taxonomy for someone’s specific use. For example, a company that files with a regulator who allows extension may create a company extension taxonomy that extends the base taxonomy created by the regulator.

An XBRL taxonomy that is a base has no real physical difference from a taxonomy that is an extension. In fact, an XBRL taxonomy can be both a base and an extension. For example, an
XBRL taxonomy may extend some other XBRL taxonomy, and then someone else can use those two XBRL taxonomies as the base for its extension XBRL taxonomy.

XBRL extensibility has two fundamental facets:

- **Extension**: Adding to or altering, through extension, an existing DTS. Extension entails creating a new extending XBRL taxonomy that references (but doesn’t directly impact or replace) the original, which is often referred to as the *base* XBRL taxonomy. The extending XBRL taxonomy adds concepts, resources, relations, or resource/relation networks.

- **Prohibition**: Removing by nullifying relations or resources from an existing taxonomy, through extension XBRL taxonomy. The key point is that you’re getting rid of something by adding something. Note that you can’t prohibit concepts themselves. Prohibiting a relation or resource entails creating one or more new resources or relations in the new extending XBRL taxonomy that voids and nullifies a specific resource or relation in the referenced pre-existing base XBRL taxonomy.

Neither type of extension has any impact on the pre-existing base XBRL taxonomy. Many organizations can concurrently extend the same base XBRL taxonomy in entirely different ways with zero impact on each other or the base XBRL taxonomy. But, extensions can impact the overall DTS.

**Understanding extension**

Suppose that someone created an XBRL taxonomy that included concepts, resources, and relations. An extension can

- Add one or more new concepts.
- Prohibit the use of existing resources, effectively nullifying them.
- Prohibit the use of existing relations, effectively nullifying them.
- Add one or more new resources or relations.
- Add one or more new resource or relation network.

The user who wants to extend some base XBRL taxonomy creates a new XBRL taxonomy, pulls that base XBRL taxonomy into her extension XBRL taxonomy by referencing it, and thus modifies the DTS with the extension and/or prohibition information from the extension XBRL taxonomy she created. So basically what you’re doing with either extension or prohibition is adding pieces (taxonomy schema or linkbase) to the set that comprises the DTS, thus modifying the overall DTS.
The exact opposite of an extension is a form. A form is static. The person completing the form can’t change a form to meet his needs. (The creator, though, can update the form from year to year to add new things.) If your information is static like a form, the extensibility features of XBRL may not be that useful to you. But other XBRL features, such as the ability to express the semantics (meaning) of the form, might still be useful, such as form computations or other relations. The underlying point here is that forms are static and don’t need extension, whereas many types of business information aren’t forms; they are dynamic.

Here are two examples of static versus dynamic data:

- Static: Most tax forms
- Dynamic: Most financial statements of U.S. public companies

**Reasons to extend**

The three primary reasons why you may want to create an extension XBRL taxonomy are to

- **Tweak existing information models.** If need be, you can totally redefine relations or create your own XBRL taxonomies, which is fine. However, if another XBRL taxonomy exists and it’s close to what you need, rather than creating a new XBRL taxonomy, simply extend the existing one to get what you desire. An example is a form that you might like to tweak a little. You like 90 percent of what exists in that form, but you don’t like the other 10 percent and want to add another 5 percent, rather than starting from scratch. So instead you simply create an XBRL extension. You’re extending the form to create a new, modified form based on another form.

- **Express information your way.** Often, organizations even in the same industry use different terminology — for example, different labels for the same concept or different ways to compute values. If need be, maybe you can tweak those models, if the system in which you operate allows it. Or, you may simply want to add labels in a different language than what someone else provided. How you tweaked the base XBRL taxonomy information model is articulated within your extension XBRL taxonomy information model, and those who use your XBRL instance information can see and understand the tweaks you made.

- **View information your way.** You can totally reorganize someone’s taxonomy and not change the meaning of the information expressed by taxonomy. For example, the US GAAP XBRL taxonomy is a consensus-based taxonomy created by accountants. Analysts have a different view of exactly the same information. Analysts could reorganize portions of the US GAAP XBRL taxonomy or totally reorganize the entire taxonomy if they see fit and not change the meaning of the information model at all.
What is allowed with respect to XBRL taxonomy extension is typically based on the agreed practices of business partners or regulators for the system with which you interact.

One scenario in which extensibility is valuable is in the financial reporting of publically traded companies. Particularly in the United States, even companies in the same industry don’t all report similar information in precisely the same way. In other words, financial statements aren’t forms that must be filled in; they’re more dynamic. Financial statements have a lot in common, but companies have flexibility in reporting their financial information, taking into consideration the unique aspects of their business. Flexibility, in this case, may mean including a concept that no other business uses, adding things up slightly differently, or putting things in one place rather than in another. XBRL was built to handle this flexibility.

**Next Steps for Business Professionals**

The following is a summary of some of the next steps which might be useful for a business professional who might want to learn more:

- **Download and fiddle with Pesseract**\(^{25}\): Pesseract is a working proof of concept which a software engineer and I are creating to construct financial reports. This software helps you understand digital financial reports.
- **Intelligent XBRL-based Digital Financial Reporting**\(^{26}\): This resource is about 700 pages of material that I have collected, summarized, organized, and synthesized over a period of about 20 years. If you are motivated, this is a gold mine.
- **Edgar Dashboard**\(^{27}\): The Edgar Dashboard has thousands of XBRL-based reports that you can check out using an easy to use viewer tool. Go to the line, look for the column “Interactive Reviewer” and click on any of the BLUE buttons to open up the viewer for the report.
- **Computer Empathy**\(^{28}\): Again, I cannot over emphasize the utility of this document. There is a lot of room for improvement, but all the moving pieces of the puzzle are there.
- **Blueprint for Creating Zero-defect XBRL-based Digital Financial Reports**\(^{29}\): There are a lot of snake oil salesmen out their today when it comes to XBRL-capabilities. This


\(^{26}\) Charles Hoffman and Rene van Egmond, [Intelligent XBRL-based Digital Financial Reporting](http://xbrl.squarespace.com/intelligent-xbrl/)

\(^{27}\) Edgar Dashboard, [https://edgardashboard.xbrlcloud.com/edgar-dashboard/](https://edgardashboard.xbrlcloud.com/edgar-dashboard/)

\(^{28}\) Charles Hoffman, [Computer Empathy](http://xbrlsite.azurewebsites.net/2018/Library/ComputerEmpathy.pdf)

document will help you learn good questions to ask software vendors that are selling XBRL related software for creating financial reports.

Next Steps for Information Technology Professionals

The following is a summary of some of the next steps which might be useful for an information technology professional who might want to learn more:

- **Hundreds of examples of XBRL instances and XBRL taxonomies**[^30]: This blog post provides literally hundreds of high-quality examples of XBRL.
- **Sophisticated example**[^31]: This ZIP archive contains a rather sophisticated example of XBRL that uses the power of XBRL definition relations and XBRL formula.
- **Putting the Expertise into an XBRL-based Knowledge Based System for Creating Financial Reports**[^32]: Helps you understand how to create and leverage higher-level objects.
- **Guide to Building an Expert System for Creating Financial Reports**[^33]: Walks you through how a software engineer and I created an expert system for constructing financial reports.
- **Understanding and Appreciating the Capability to Recognize and Identify Blocks**[^34]: This is a really good way to come to appreciate the use of a high-level model to make working with XBRL easier for business professionals.
- **The XBRL Book: Technical, Simple, and Precise**[^35]: This is one of the better books that explains XBRL to software engineers.

Acknowledgements

Most of the ideas in this document come from discussions and feedback that I received over the past 15 or so years from many, many colleagues who are too numerous to list here. That input was critical to shaping the thoughts expressed in this document. Thank you to the entire XBRL community!

[^31]: Sophisticated example, [http://xbrlsite.azurewebsites.net/2018/Pesseract/DynamicRules_AllFilesLocal.zip](http://xbrlsite.azurewebsites.net/2018/Pesseract/DynamicRules_AllFilesLocal.zip)