



# Table of Contents

<a href="#"><u>A Stroll Through Some of the Ideas and Terminology Surrounding Xml and Alexis.....</u></a>	<a href="#"><u>1</u></a>
---	--------------------------

# A Stroll Through Some of the Ideas and Terminology Surrounding Xml and Alexis

please send comments and questions to: [matthew@ella-associates.org](mailto:matthew@ella-associates.org)

## INTRODUCTION

This document attempts to explain some of the concepts and technical ideas behind the Alexis System. Alexis is java-based xml-oriented system designed to facilitate the collaborative writing of dictionaries and other xml-oriented projects. This document tries to follow a middle ground between technical detail and clear explanation. It is designed for those people who need to approximately understand how the Alexis system works and what advantages it can bring to teams of lexicographers and editors. There is also an emphasis in this document on explaining terminology which is often confusing. In reality, there is often no exact or precise definition for a piece of terminology and it is only necessary to understand its general or approximate meaning.

## AN EXPLANATION OF XML AND WHERE IT CAME FROM.

I will begin this explanation of XML by referring heavily to HTML. This is because a great deal of the terminology and concepts behind HTML are applicable to XML. (As will be made clear later, in some cases HTML actually is a form of XML)

If you click on the menu item 'view source' or 'view page source' in your Browser (such as Microsoft Internet Explorer or Netscape Navigator, or Opera, etc) when you are viewing a web-page on the Internet, you will see something like this:

```
<html><body>
<table>
  <tr valign=top>
    <td nowrap><font face=arial size=-1><b>Fun</b>
    </font>
    </td>
    <td colspan=2><font face=arial size=-1>
      <a href="http://www.yahoo.com">Games</a> ,
```

```

    <a href="http://www.ella-associates.org">Horoscopes</a>,
    <a href="http://www.google.com">Kids</a></font><hr>
  </td>
</tr>
</table>

</body>
</html>

```

This is a small example of something called HTML, which stands for Hyper-Text Mark-up Language. All the text between the 'pointy brackets' (<>), which are also known as 'angle brackets' or 'greater and less than signs', is called 'mark-up'. When I say 'between' I mean all the text which begins with a < and ends with a >. A piece of mark-up, such as '<tr valign=top>', is referred to as a 'tag', or 'mark-up tag'. You will notice above that some of the text is not enclosed by greater and less than signs; For example, the word 'Fun' and the words 'Games' and 'Horoscopes'. This text is called 'the content', or 'the content of the HTML page' or something similar. When you look at the HTML page in your Browser you will not actually see any of the mark-up. You will only see the 'content'. But the mark-up affects the way that the content appears or behaves. For example, in the HTML above, the word 'Fun' will appear to be 'bold', that is the text will be darker and thicker than the normal text on the page. This is because of the mark-up tags <b> and </b>. These tags tell your Browser that the text between them should be displayed as bold. This process, which your Browser carries out, of turning mark-up tags into a web-page is called 'rendering' or 'rendering the html'.

The first tag '<b>' is called the 'opening tag' and the second tag '</b>' is called the 'closing tag'. All 'closing tags' have the '/' character immediately after the first 'angle bracket' (<). Almost all HTML tags are matching pairs of opening and closing tags, but not all. In the example above there is an opening tag '<a href="http://www.yahoo.com">' and its matching closing tag is '</a>'. The word 'Fun' is said to be 'enclosed' by the two tags.

In the HTML above, we can see the line

```
<a href="http://www.yahoo.com">Games</a>,
```

In this line, the text 'href="http://www.yahoo.com"' is known as an 'attribute' of the <a> tag. The text 'href' is called the 'name' of the 'attribute' and the text 'http://www.yahoo.com' is called the 'value' of the attribute. Together this is referred to as a 'name/value pair'. You may think that I am un-necessarily introducing confusing terminology for no reason. But all this terminology is important when talking about XML, which, in the long run, is more important than Html.

A set of mark-up tags is referred to as a 'mark-up language'. HTML is not the only mark-up language. For example, there is a mark-up language called 'SGML' which stands for Standard Generalised Mark-up Language, which has been used by publishing companies for many years. What is the point of these mark-up languages? Why do we need them? I'm glad you asked. In the past mark-up languages have been mainly used to 'format' text documents. By 'formatting' I mean the process of taking a text document that looks as if it has been written on a typewriter, and applying

effects like making some text 'italic', making the title bigger, putting different parts of the document into different 'fonts' and so on. The majority of HTML tags are designed for this purpose; they change the appearance or the 'presentation' of the text which they enclose. You may well ask at this point, why don't we just use 'Microsoft Word' if we want to format text documents? We don't have to learn some obscure mark-up language when we use Microsoft Word. Microsoft Word uses its own mark-up language, which the user never sees. One advantage of a mark-up language like HTML is that it is a system which no one company or person controls.

Another advantage of a mark-up language such as html, as opposed to say the microsoft word markup language is that it is 'human-readable'. For example, if you get a Microsoft Word document and you open it in a text editor like Microsoft Notepad, you will notice that there are all sorts of strange codes and incomprehensible symbols. This is because Notepad is actually displaying the 'mark-up codes'. You will also be immediately aware that it is not possible to make any sense of the codes and symbols. On the other hand, if you look at the source for an HTML page, although the codes may look at first quite complicated, with a small amount of knowledge it is quite possible to understand what each of the codes is doing. This is what is meant by 'human-readability'. This is an important characteristic of HTML and XML.

## AN IMPORTANT CONCEPT

In some situations, there will be a set of rules that govern a particular area. But there may also be a set of rules which govern how those first rules should be made. An example of this is the constitution of a country and the laws which have effect in the country. The constitution is, in a sense, a set of rules about how other rules may be created. The Constitution of a country may state, for example, that you cannot make a law which takes away the right of a person to freedom of speech, and that any law which attempts to do that will not be a valid law. This is a very important idea when understanding XML.

XML stands for Extended Mark-up Language. But XML is not really a mark-up language like HTML. Actually it is a set of rules stating how mark-up languages like HTML should be created. It defines a set of rules that people should follow when creating mark-up languages. This may seem very abstract and difficult to understand at first, but its not. The rules that XML sets out are really pretty simple: it says,

Every opening tag must have a closing tag or, if not, the tag must look like this

`<sometag/>`

in other words it has to end in `'/>'`;

All closing tags must begin with `'</'`;

All tags must use the `<` and `>` signs as their starting and ending characters (the same as HTML);

Any tag may have 'attributes' but that the 'values' of all attributes must be enclosed in

'double quotes' (which is the `"` character);

And a few other rules as well.

If a text document follows these rules then it is called 'well-formed'. This is an important word. Remember it. The rules of XML are called 'strict' because if they are broken in even the smallest and most in-offensive possible way, then the document ceases to be

'XML' and becomes what is affectionately called 'Tag Soup'. Most HTML documents are actually 'Tag Soup' because they don't follow all the rules of XML. For example, the HTML displayed above breaks the rules of XML in a number of ways: The tag `<hr>` (which creates a horizontal line on the page) has no closing tag. Also, many of the values of attributes are not enclosed in the " character, such as in the case of the tag `<tr valign=top>`. In order to obey the rules of XML it should be `<tr valign="top">`. If we fixed all these problems, then we would have an HTML document which was also an XML document.

Lets look at an example of a 'well-formed' (but very short) XML document

```
<shopping-list>
  <item quantity="5">Dr. Boggles Magical Washing Powder</item>

  <item quantity="1">Bananas</item>
  <item quantity="2">Apples</item>
</shopping-list>
```

This is a mark-up language which obeys all the rules of XML. Strictly speaking it is called an 'instance of XML' or 'an implementation of XML' or 'an application of XML'. For example it might be given the name 'ShoppingML' which would stand for Shopping Mark-up Language. But in practice it is extremely common to hear the text above referred to simply as 'XML' or as an 'XML document'. When you hear the phrase 'XML' it is actually more likely that the speaker is referring to something like what you can see above, rather than referring to the set of rules which is what XML really is.

The purpose of XML, unlike HTML, is not to 'format' the appearance of text documents. The purpose of XML is to store and transmit data in a way that is reliable and yet flexible. One of the great problems that has occurred in computer systems, has been the inability of different types of computers and different types of programs to communicate with one another. For example if you create a Microsoft Word document on a Microsoft Windows computer and then open that same document on a Apple Macintosh computer you will often find that a lot of the formatting does not look the same. Because of the rise of the Internet, it has become more and more important that all different types of computers with different types of operating systems should be able to communicate to each other and transfer data without that data becoming corrupted or distorted in any way. This is one of the problems which XML solves.

XML is a powerful system because it allows data to be easily 'transformed', 'manipulated' and exchanged between different types of computers and programs.

XML is very important in the Alexis system for various reasons. Firstly, Alexis uses XML to transfer all data from the 'server' to the 'client'. (I will try to explain these terms later). Also, it is customary in the field of lexicography to store and manipulate the data for lists of words as XML or SGML. Alexis conforms to this standard by providing comprehensive XML editing facilities in the 'user-interface' and by storing the word data as XML in the back-end database.

## OF SERVERS AND CLIENTS

Two words which are very frequently 'bandied about' with respect to computers: 'client' and 'server' and the compound form 'client/server'.

These two words represent quite an important concept when dealing with 'networked' software and this concept is important in understanding the Alexis system.

On the internet, some computers provide 'services' to other computers and to other programs. A simple example of a 'service' is a program (and computer) which outputs an accurate representation of the current time. In other words, some computer on the internet will ask another computer on the internet 'what is the time, precisely' and the computer that is asked will reply '4:20:22 am Greenwich Mean Time'. This is a simple but important example because it demonstrates a very common operation on the Internet. The computer which does the asking is called the 'Client' or the 'Client Computer' and the computer which replies is called the 'Server' or the 'Server Computer'. The process of asking a question of another computer is called 'making a request' and the reply is often called 'a response'. In reality, when a computer makes 'requests' they normally don't do so in plain english (or in any other human language). Instead they might send a message such as 'GET TIME' and the server will respond 'CURRENT TIME: 4:20:22 am'. Although these messages look like plain english, they are not really. For example, if the client send a message 'whats the time please', the server computer would not understand it and would probably return a message like 'INVALID REQUEST'. In other words, the server only understands a very limited number request messages and the requests have to be in exactly the right format. For example if the client computer asks 'GET Time' (using lower case letters), the server may well not understand this request at all.

This system of sending special request and response messages is called a 'protocol'. Protocols are extremely important on the Internet because they allow computers who otherwise know nothing about each other to communicate. In summary, a client computer makes requests for services of a server computer using a particular protocol, and the server computer responds to the request using the same protocol.

I have been referring so far to client and server computers. But computers dont actually do anything by themselves. It is actually the programs that carry out actions. In other words, it is actually a program which is running on the client computer which sends a request to the server computer, and it is actually a program running on the server computer which replies. For this reason, these programs are known respectively as 'client programs' (or 'client application', or 'client software') and 'server programs/applications/software'. To ensure that everybody becomes completely confused, it is very common not to include the second part of these phrases. That is, it is common to simply refer to a 'server' or a 'client'

without specifying if the speaker is referring to a computer or to a program. This does cause confusion and it is necessary to work out which is being referred to from the context of the sentence.

An example of a client program (or application) is a Web Browser like Netscape Navigator. The Web Browser makes requests for web pages from some Web Server located somewhere on the Internet. It does this using a special protocol called HTTP (Hyper Text Transfer Protocol). And the Web Server responds using the same protocol. The user of the Web Browser does not actually get to see the requests being made. They are made 'behind the scenes' as soon as you type a Web Address in the 'location bar' of your Browser. The actual requests are quite simple and look something like

```
GETwww.google.com/index.html
```

The Web Server (for example: Apache) which replies to this request is an example of a 'Server Application (or Program)'.

The Alexis system, like the majority of network based programs, has a client program and a server program. These are, in reality, completely separate programs which can (and probably should) run on completely different computers. While the Client and Server programs don't actually need each other to run, they are not able to do anything useful unless they are able to talk to each other using the Alexis protocol.

The Alexis protocol is actually an implementation of XML (or colloquially, is XML). In other words, the Alexis Client program will make a request to the Alexis Server program (the name of which is 'Chakriya') in approximately the following way

```
<alexisRequest>
  <filesystem-request>
    <word-pattern>a*</word-pattern>
  </filesystem-request>
</alexisRequest>
```

What this request means, is that the Client program is asking for all words which match a certain pattern. In this case the pattern is that the words should start with the letter 'a'.

The client sends this request to the Server which is located on some computer on the Internet and the Server responds by sending to the client all of the words in a particular dictionary which start with the letter 'a'. The user of the Alexis system (who is using the client application), can then edit those words and when he or she has finished editing them he or she can send a request to the Server that his/ her changes should be saved on the Server.

The advantage of this system of having separate server and client programs is that it allows for easy collaboration between different



users who can be located anywhere in the world, as long as they have a connection to the Internet. There is no reason why more than one Client program cannot be run at the same time and therefore many different users can use the system at the same time without being geographically located in the same place. The Server program (Chakriya) acts like a manager or orchestra conductor for each and all of the client programs. The Server program stores and manages the work carried out by each user. In other words, while a user (in this case, some lexicographer located anywhere in the world) carries out all her work on her own computer, all the changes that she makes to different entries (words) in the dictionary which she is working on, actually get saved to the 'central' Server Computer. In this way, all the other lexicographers and editors who are working on the same dictionary (or other xml related project) can see and use the work that has been done by that person.

This use of a client/ server 'architecture' (as it is known) also eliminates the possibility for conflicts between the work carried out by different users. That is, the Server program makes sure that one user does not over-write or destroy the work carried out by another user.

## ALL ABOUT DATABASES

Data is any set of patterns that have meaning to a human being. A 'database' is a collection of data usually about a particular subject or field. In information technology, a computer program which stores and manages data is referred to as a 'database system' or 'database management system' or 'database server'. As the last term suggests, most modern database programs use the Client/ Server system which I have explained above. That is, there is one part of the program which stores and manages the data which is called the database server and which responds to requests for that data from 'Client' programs. The Server program and the Client program do not have to be on the same computer.

In the same many as other client and server programs the Database Server and Client use a 'protocol' to communicate to each other.

There are two main types of modern Database systems: The 'hierarchical' database, and the 'relational' database. A hierarchical database 'looks' like a (up-side down) tree, something like this:

```
village
  -- name = almetlla de mar
  -- residents
    person1
      -- first name = paz
      -- second name = martinez
```

```

person2
-- first name = nuria
-- second name = fill
person2
-- first name = james
-- second name = sweetapple
-- bars
bar1
-- name = pica pica
-- proprietor = montse
bar2
... etc

```

Another example of a hierarchical database is the files on a computer. If you expand each of the folders (directories) on your computer using a program like Microsoft Explorer you will see the same 'tree' like structure. Each of the folders is like a branch of the tree and the files are like the leaves on a tree (but not as attractive).

With a small or large amount of brain-racking you will also notice that the XML documents which are displayed above are also examples of a hierarchical database structure. Historically (in the 1970's and early 1980's) hierarchical databases used to be the most important and most used. However, more recently the other type of database has become the most common, that is the 'relational' database. Finally, with the rise of XML, the pendulum is swinging back towards hierarchical databases (please excuse the cliché).

In order to understand Alexis you really don't have to know anything about relational databases since only the programmer actually has to deal with them. But in any case here is a brief explanation.

A 'relational' database looks like a collection of 'tables', in other words, bits of data divided into rows and columns, like this

first name -----	last name -----	place of residence -----	favorite bar -----
paz	martinez	almetlla de mar	pica pica
nuria	fill	almetlla de mar	The English Bar
james	sweetapple	almetlla de mar	N/A

As you can see, the actual data is pretty much the same as for the hierarchical database, but the way that it is arranged is different.

To get data in and out of a relational database program you use a thing called 'Structured Query Language', which is usually referred to as 'SQL'. It looks like this:

```

SELECT 'first name', 'last name'
FROM 'Residents'
WHERE 'place of residence' = 'almetlla de mar'

```

This is called a 'query'. As you can see it is a very simple language to use since it is almost the same as English. The 'query' above would get you the following data:

first name	last name
-----	-----
paz	martinez
nuria	fill
james	sweetapple

Alexis uses a combination of a hierarchical data structure (XML) and a 'relational' data structure in order to store the words of a dictionary or other data. But the user of the Alexis system will only ever have anything to do with the hierarchical system, since the relational database program (called 'MySQL') is hidden or 'behind-the-scenes'.

... possibly to be continued

## A QUICK GLOSSARY

### mark-up

Mark-up is a method of adding more information to a text document.

This information can be added for different purposes. One main purpose is to change the appearance of the text document. This is the case for HTML. Another purpose is to insert data which will allow the document to be exchanged with other people, computers, or programs.

This is the case for XML.

### tags

Tags are specific examples of mark-up, such as `<i>` or `<a href="something">`

### xml

This is a way of adding extra information to a document in order to allow it to be exchanged with other computers, programs or people, or to allow it to be transformed into other formats, such as HTML, plain text, Adobe PDF or WAP (which is the format used by mobile phones).

### Server

A program (or computer) which manages information and responds to requests from other programs which may or may not be on the same computer.

### Client

A program which makes requests to another computer for some kind of information.

### Parsing

The process of turning an XML document into something that a computer can actually understand

### Relation Database

A system of tables (things which contain rows and columns) which contain information about a particular subject area.

### Rendering

The process of turning mark-up tags into some kind of visual display.

Hierarchical Database

A kind of upside down tree of data.

## WHATS MISSING

This section contains concepts that possibly should be covered but aren't

validation of xml, xml schemas and dtDs, parsing of xml.

Methods of transforming xml.

... possibly to be continued

please send comments or corrections to: [matthew@ella-associates.org](mailto:matthew@ella-associates.org)