# Ecological Metadata Language:
## Practical Application for Scientists

Written by
David Blankman
Jeanine McGann
LTER Network Office
Albuquerque, NM

# Table of Contents

# EML Handbook

### Who should read this?

This document is designed for ecologists, ecological information managers, and ecology students.

### Assumptions about the reader:

Is computer literate.

Has an understanding of ecological terminology.

May or may not know anything about XML.

## Introduction

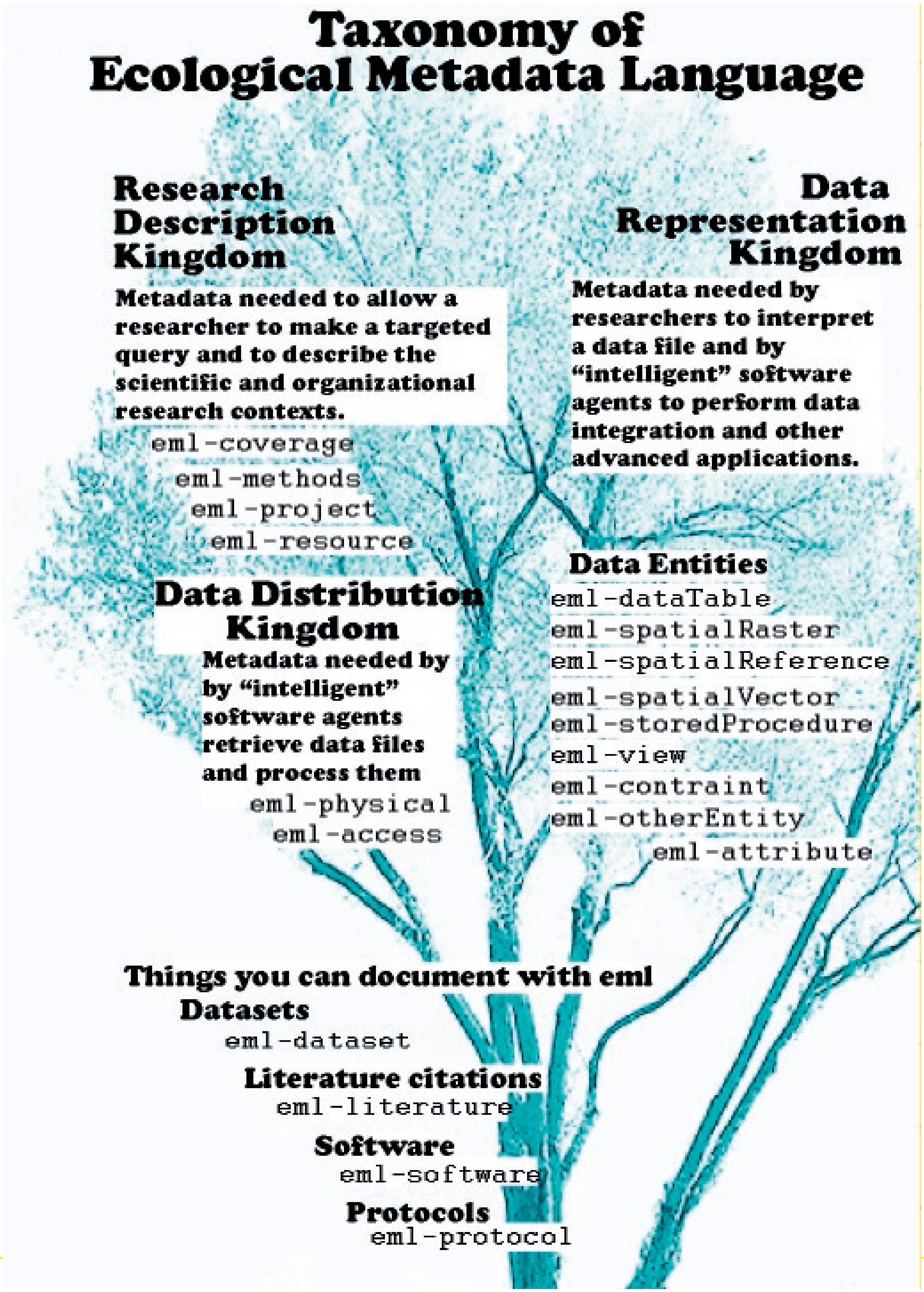
### What is EML?

Ecological Metadata Language (EML) has been designed using XML (Extensible Markup Language) schemas for use specifically with ecological data in order to fulfill two main purposes. The first and most important purpose is to define a common structure that all ecologists can use to document ecological data so that other ecologists can correctly interpret the data. The second purpose for EML is to provide a structure so that software applications can be developed. The kinds of applications anticipated range from a basic tool that allows a user to perform very specific targeted searches for datasets to advanced applications that could perform automatic integration of datasets. This will be a tremendous help to ecologists in organizing and implementing long-term research studies where the ability to search for and synthesize large amounts of data is crucial.

EML is a content standard for documenting ecological data that is implemented using XML schemas to define the structure. Many people are familiar with Hypertext Markup Language (HTML), and XML is similar in that it uses a system of coded tags <xxx>…..</xxx> that contain information. However, while HTML is a formatting language that allows one to use a series of standardized tags to display text in an electronic (web-based) environment, XML tags are created according to the data they define and they are used to define the content rather than the format of data.

EML, then, is XML written with tags that correspond to ecological data or metadata. To say that a document is an EML document means that it is a valid XML document and that the document follows the structure defined in a series of XML schemas.

*Figure 1*



## Taxonomy of Ecological Metadata Language

**Research Description Kingdom**

**Metadata needed to allow a researcher to make a targeted query and to describe the scientific and organizational research contexts.**

eml-coverage
eml-methods
eml-project
eml-resource

**Data Representation Kingdom**

**Metadata needed by researchers to interpret a data file and by "intelligent" software agents to perform data integration and other advanced applications.**

**Data Distribution Kingdom**

**Metadata needed by by "intelligent" software agents retrieve data files and process them**

eml-physical
eml-access

**Data Entities**

eml-dataTable
eml-spatialRaster
eml-spatialReference
eml-spatialVector
eml-storedProcedure
eml-view
eml-contraint
eml-otherEntity
eml-attribute

**Things you can document with eml**

**Datasets**
eml-dataset

**Literature citations**
eml-literature

**Software**
eml-software

**Protocols**
eml-protocol

## Taxonomy

The two purposes behind the creation of EML can be looked at as forming two of the basic branches in the taxonomy. As can be seen in Figure 1, EML has a scientific description "kingdom," a data representation "kingdom," and a data distribution "kingdom."

The scientific description branch contains those modules that answer questions like:

Who did the research? <creator>

In general terms, what is the research about? <abstract>

What are some of the key concepts that refer to the data? <keywords>

Where was the research done? <coverage>/<geographicCoverage>

What time periods are covered in the data? <coverage>/<temporalCoverage>

What species are represented in this research/data? <coverage>/<taxonomicCoverage>

What methods were used? <methods>/<methodStep>,<sampling>,<qualityControl>

Is the data being documented part of a larger project <project>

The data representation branch contains modules that are used to describe:

What kind of data entity is it? <entity>/<dataTable>,<spatialVector>,<spatialRaster>

How is the actual file structured? <physical>/<dataFormat>

The data distribution branch contains modules that show how the data is dissimilated:

How would I retrieve this file? <physical>/<distribution>

Who will I allow access to the data? <access>

## A Brief History of EML

Michener (2000) stressed the importance of well-documented metadata to the continued usefulness of datasets in all scientific fields, noting how the rate of attrition of knowledge increases over time as the documentation used to fully comprehend and interpret various data sets is lost due to accidents, changes in storage, or simple human memory failure. To this end, the standardization of metadata documentation provides a structure that all researchers can understand, keeping important information from becoming lost or incomprehensible.

The origins of Ecological Metadata Language can be traced, in part, to the work of Michener et al. (1997), in which a set of metadata descriptors was developed specific to the ecological sciences. Many of the current EML tags are based on these descriptors, which are organized into five classes as follows: Class I: Data set descriptors, Class II: Research origin descriptors, Class III: Data set status and accessibility, Class IV: Data structural descriptors, and Class V: Supplemental descriptors. Some metadata sets developed prior to EML standardization also may have used tags based on these descriptors.

For our purposes, it is possible to see the basic structure of EML emerge from these general classifications. For example, Class I & II both contain some of the basic elements that compose the current EML 2.0.0 ResourceGroup, and can be mapped as follows:

| **Ecological Metadata Descriptors** (Michener et al, 1997) | **Current EML tags** |
|---|---|
| Data set identity | <title> |
| Data set identification code | <alternateIdentifier> |
| Originator(s) | <creator> |
| Abstract | <abstract> |
| Keywords | <keywordSet>, <keyword> |

Other Class II elements also form the basis of the current <coverage> and <methods> modules. The original metadata descriptors contain a series of site descriptors such as location, physiographic region, landform component, and topographic attributes, which all are elements of the current EML <geographicCoverage>. Michener et al. also included taxonomic and temporal elements, which correspond to the other aspects of EML coverage. Research methods, design, instrumentation, and protocols all were included in Michener et al.'s original metadata descriptors, and all have their counterparts in the EML <methods> module.

Many of the elements found among the Class III descriptors--"metadata status," "storage location and medium," "proprietary restrictions," and "citation"--relate, respectively, to current EML modules such as <maintenance>, <distribution>, <access>, and <citation>. "Contact person," also included in Class III, became the basis for the EML version of <contact>.

Class IV contains several descriptors that have become the basis for many EML entity and attribute elements. Most of the Class IV "Data set file" descriptors can be found in the current EML EntityGroup, especially in the <physical> module. The "Variable information" descriptors include attribute-type values such as "units of measurement" (<measurementScale>), "variable codes" (<codeDefinition>), "precision" (<precision>), and "range of numeric values" (<bounds>). Descriptors also specify data format and codes for missing values, both of which are included in the current version of EML.

Other elements included in the supplemental descriptors in Class V are also mapped to various modules in EML, including "Quality assurance/quality control procedures," which ended up in the <methods> module as <qualityControl>, and "data set update history," which became part of the <maintenance> module's <changeHistory>.

All of these elements, based on Michener et al. (1997), have much to contribute to the lineage of EML and help to broaden our understanding of the structure and purpose of this metadata language. Though the EML tags may appear arbitrary at first, all of the elements are rooted in a thoughtful and logical system, one that was created with the specific needs of the ecological science community in mind.

## A Quick Look at the Basics

While there are some people who take pleasure in documentation, most ecologists are more interested in doing research than in documenting it. To illustrate a typical documentation scenario with a first-time EML user, let's take the case of Pat:

Pat Ecosis is an ecologist. S/he knows that providing metadata is important, but has never done so in any organized way. Pat is computer literate, but has never worked with XML, "It's like HTML, right?"

The rest of the cast of characters include:

Eco Informatics: Eco is an information manager with a strong interest in designing advanced software tools to help Pat with the analysis and modeling of ecologically interesting data. Eco is also a representative of KNB (Knowledge Network for Biocomplexity), a mythical association of ecological organizations that is committed to making ecological data available to researchers around the world.

emlMaven: emlMaven is an expert in EML

NSF: A mythical federal agency.

## Metalog 1

**Pat:** I just completed a research project on rodent populations in northwestern New Mexico. NSF tells me that I have to provide EML compliant metadata to document my research, but I want to get back to the field as soon as possible. What is the minimum I have to do to produce a valid EML document?

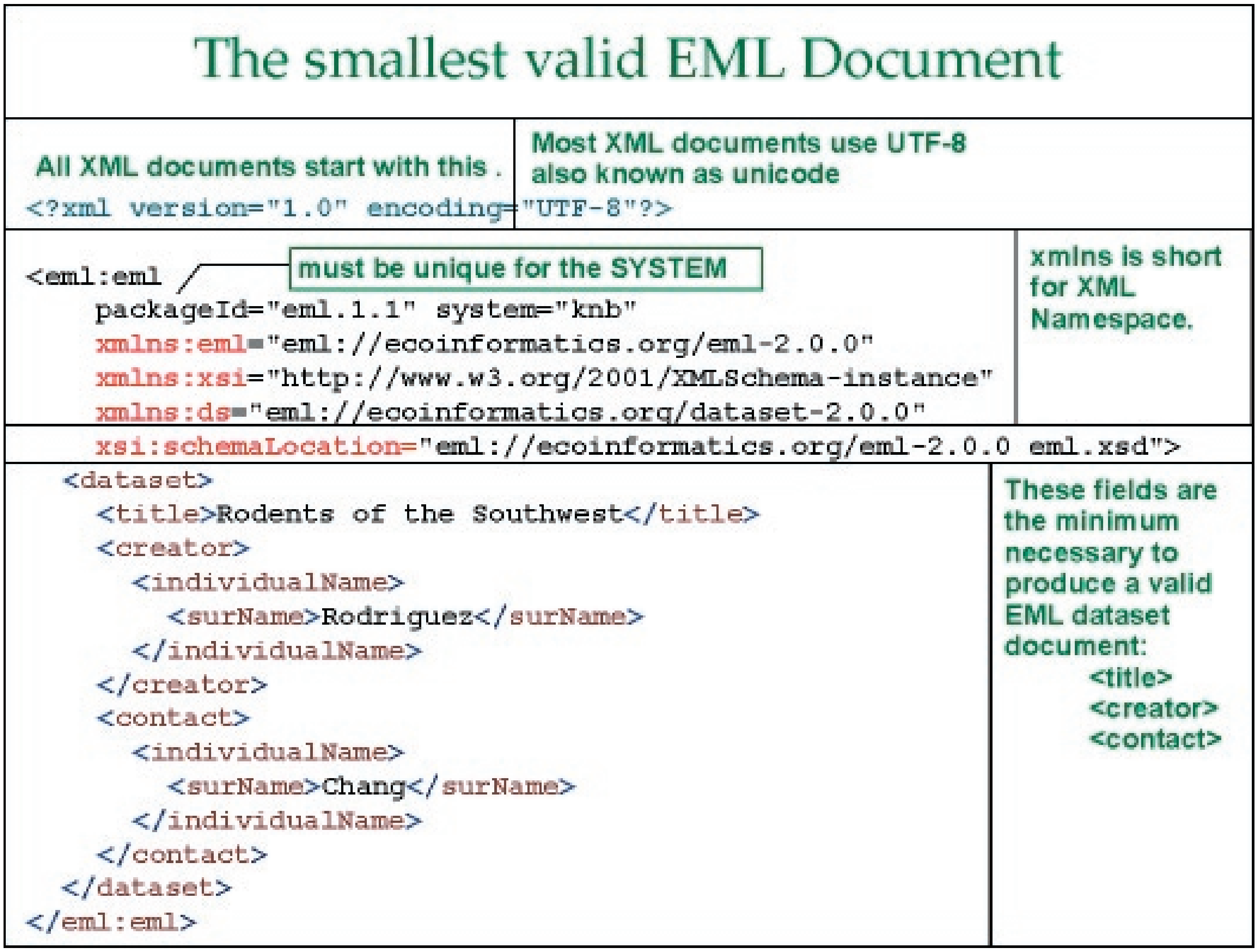
**emlMaven:** The document in Figure 2 shows the smallest valid EML document.

**Eco Informatics:** That doesn't tell me very much. How can I design any useful tools with just a title and a last name?

**Pat:** I know I said minimal, but I can tell that NSF probably won't be satisfied with that. So, let me rephrase my question: What would an EML document look like that contains enough information about my research to make NSF happy and gives another ecologist something to work with?

**emlMaven:** That's a longer story…and it's going to require some definitions and more detailed explanations. It will help us to look more closely at the structure of the minimal EML document.

*Figure 2*


The smallest valid EML Document

All XML documents start with this .

Most XML documents use UTF-8 also known as unicode

must be unique for the SYSTEM

xmlns is short for XML Namespace.

These fields are the minimum necessary to produce a valid EML dataset document:
<title>
<creator>
<contact>

```
<?xml version="1.0" encoding="UTF-8"?>
<eml:eml
    packageId="eml.1.1" system="knb"
    xmlns:eml="eml://ecoinformatics.org/eml-2.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ds="eml://ecoinformatics.org/dataset-2.0.0"
    xsi:schemaLocation="eml://ecoinformatics.org/eml-2.0.0 eml.xsd">
  <dataset>
    <title>Rodents of the Southwest</title>
    <creator>
      <individualName>
        <surName>Rodriguez</surName>
      </individualName>
    </creator>
    <contact>
      <individualName>
        <surName>Chang</surName>
      </individualName>
    </contact>
  </dataset>
</eml:eml>
```

## Anatomy of an EML Document

While most EML users will probably be using a tool such as Morpho or Xylographa, which take care of the formatting chores and assure that all of the metadata documents produced are valid EML documents, it helps to know what an EML document looks like "under the skin."

### A word about formatting conventions

Since EML documents are XML documents, they use XML tagging notation. Throughout this document, you will find references to EML elements. Unless otherwise stated, all of the examples will refer to documenting data objects. EML can also be used to document literature citations, research protocols, or software independent of any dataset.

When <eml> (lower case, inside <>) is used, it refers to the top-level element of an EML document. All EML documents begin with the <eml:eml> tag and end with the </eml:eml> tag.

A reference in the form of ...<xxxx> means that the element in question is an arbitrary number of levels deep in the EML structure. For example, a reference to ..<coverage>/<geographicCoverage> would be interpreted as: <eml> <dataset> <coverage> <geographicCoverage></geographicCoverage> </coverage> </dataset> </eml>.

XML and EML tags are case sensitive. For example, the tags <species> and <Species> are not the same.

### Technical specifications

XML documents open with the XML version and encoding statements shown in Figure 2, followed by the <eml:eml> tag that identifies the document as EML. The packageId is a unique identifier for a given system, with the term "system" referring to the catalog or storage system to which the metadata document is being contributed. The system in this example is "knb," the Knowledge Network for Biocomplexity. Any metadata document from an LTER, OBFS, or California NRS site should be considered as being contributed to KNB.

The next three terms each begin with xmlns and are references to XML namespaces. A namespace can be thought of as a way to specify where the element definitions come from. Since anyone can define their own tags, the use of namespaces allows XML processors to identify the specific vocabulary being used. A namespace is uniquely identified using a Uniform Resource Identifier (URI). A URI should be globally unique, but, so far, there is no method of enforcing this uniqueness; that is, there is no central registry of URIs. Often a URI is expressed as a URL (Uniform Resource Locator), such as `http://www.w3.org/2001/XMLSchema-instance` in Figure 2.

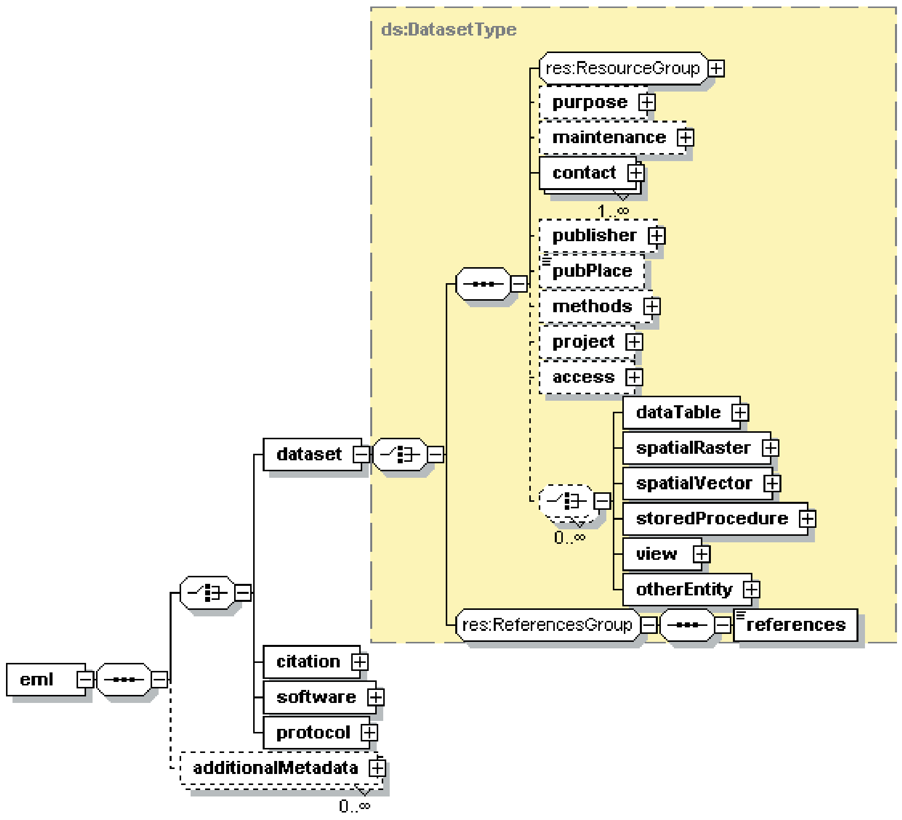
The schemaLocation refers to the location of the specific EML schema being used. For the curious or obsessive, the technical specifications for URI, URL and the even more abstract URN (Uniform Resource Name) can be found at: http://www.w3.org/Addressing.

Once the technical specifications have been defined, we can begin to look at the body of the EML document, that is, the dataset to be documented.

## Documenting a dataset

The term <dataset> means different things to different people. In EML, the term dataset may contain general information such as the title, creator, and contacts, as well as one or more data entities, such as data tables, that provide more specific research details. Take a look at the expanded EML dataset schema displayed in Figure 3. This schema provides a graphical representation or "map" of the actual XML schema that shows where the different elements fit in the hierarchy of the language.

*Figure 3*



For the moment, let's focus on the ResourceGroup schema, which is displayed in Figure 4. The <title> field provides a description of the resource being documented that is long enough to differentiate it from other similar resources. Multiple titles may be provided, particularly when trying to express the title in more than one language. If a title is in a language other than English, use the "xml:lang" attribute. For example,

```
<title>Rodents of the Southwest</title>
<title>xml:lang="de"Rodents des Südwestens</title>
```

In the second <title>, the text 'xml:lang="de"' is an XML attribute that says: language =german (de=deutsche). There are two versions of ISO 639 language codes: two-letter or three-letter codes. The three-letter system was designed to expand the list of languages that can be represented. Currently, the two-letter system is used more often, but over time it is likely that the three-letter system will be preferred.

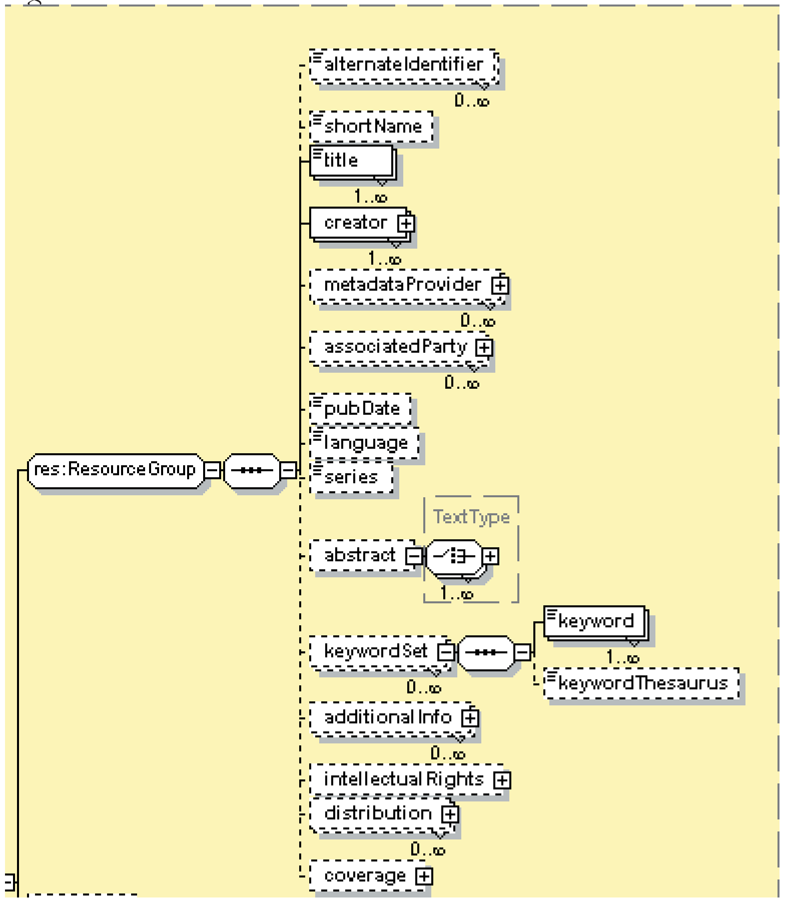
A clear discussion of the use of the language attribute can be found at:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/prompting_beta2/html/SSML_Elements_lang.asp

The actual language codes can be found at:

http://www.w3.org/WAI/ER/IG/ert/iso639.htm.

*Figure 4*



A <creator> is a structure for representing the owner of the data. A <creator> can be a person, an organization, or an organizational role. Examples of organizational roles are: Executive Director, Department Administrator, or Information Manager.

At this point, we can expand the **minimal EML** document to look at the <creator> information in more detail.

```
<creator>
  <individualName>
    <salutation>Dr.</salutation>
    <givenName>Pat</givenName>
    <givenName>Morgan</givenName>
    <surName>Ecosis</surName>
  </individualName>
  <address>
    <deliveryPoint>Department of Southwestern Ecology</deliveryPoint>
    <deliveryPoint>Semiarid State University</deliveryPoint>
    <city>Noaguaville</city>
    <administrativeArea>NM</administrativeArea>
    <postalCode>78890</postalCode>
    <country>USA</country>
  </address>
  <phone>709-345-8970 x 254</phone>
  <electronicMailAddress>pat.ecosis@semiarid.edu</electronicMailAddress>
</creator>
<creator>
  <organizationName>USDA</organizationName>
  <address>
     <deliveryPoint>US Department of Agriculture</deliveryPoint>
    <deliveryPoint>Mailstop 3654</deliveryPoint>
    <deliveryPoint>25 Federal Plaza</deliveryPoint>
    <city>Washington</city>
    <administrativeArea>DC</administrativeArea>
    <postalCode>20025</postalCode>
    <country>USA</country>
  </address>
  <phone>phonetype=”voice ”709-345-8970 x 254</phone>
  <phone>phonetype=”fax ”709-345-8962</phone>
  <electronicMailAddress>dataservices@usda.gov</electronicMailAddress>
  <onlineUrl>http://www.usda.gov/ecoinformatics</onlineUrl>
</creator>
```

The element names may seem unfamiliar and you might be wondering why the developers of EML used such strange names. These elements have been taken from the International Standards Organization (ISO), schema for representing people: iso-party.

For example, a person can have more than one <givenName>, but only one <surname>. EML does not have a specific tag for middle names. Since <givenName> is optional, <surName> would be used for “Cher” or “Sting.”

An address can have more than one <deliveryPoint>. A <deliveryPoint> is that part of an address that precedes the <city>. The most common <deliveryPoint> is a street address. Other examples are company names, department names, or post office boxes. Because XML does not recognize carriage returns, the way to enter that part of an address that precedes the city is to use a separate <deliveryPoint> element for each item that you would want to appear on a separate line when displayed on a web page or printed report. An <administrativeArea> is the tag that would be used in the United States to represent a state or in Canada to represent a province.

A <creator> can have more than one address. Each address must be contained in its own set of <address></address> tags:

```
<creator>
  <individualName>
    <salutation>Dr.</salutation>
    <givenName>Joe</givenName>
    <surName>Ecologist</surName>
  </individualName>
  <address>
    <deliveryPoint>25 Oceans Avenue</deliveryPoint>
    <city>Oceans</city>
    <administrativeArea>CA</administrativeArea>
    <postalCode>98025</postalCode>
  </address>
  <address>
    <deliveryPoint>Department of Ecological Sciences</deliveryPoint>
    <deliveryPoint>University of the Oceans</deliveryPoint>
    <city>Oceans</city>
    <administrativeArea>CA</administrativeArea>
    <postalCode>98025</postalCode>
  </address>
</creator>
```

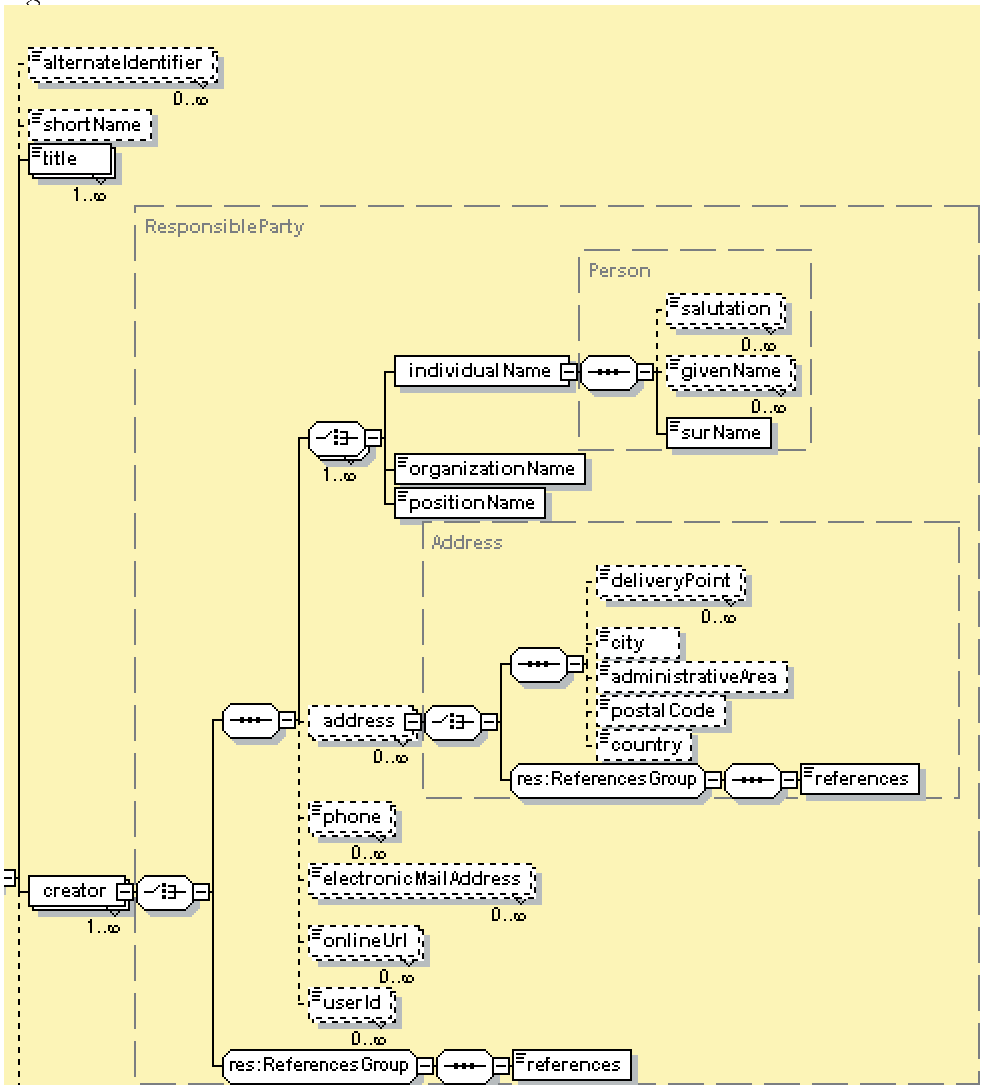
In terms of EML schema, the above code fragment would look something like the one shown in Figure 5:

There are two additional optional creator types that can be described by EML, <metadataProvider> and <associatedParty>. A <metadataProvider> is used if the person, organization, or organizational position that provided the metadata is not the one who created the data being documented. A <metadataProvider> could be an LTER information manager, a student, or a researcher who may or may not have been part of the data creation process. While this field is optional, it is recommended to use if the metadata was provided by someone other than the <creator>.

It is also a good idea to use this field if there are multiple creators, but one of them is the primary source of the metadata. It is especially important to use this structure if the metadata is being constructed after the fact. For example, if someone decides to create EML documentation for data that is 20 years old, some metadata may exist in a lab notebook, but the documenter also is developing new metadata.

An <associatedParty> is a person, organization, or organizational position that is involved in the creation of the data, but is neither a <creator> nor a <metadataProvider>. An associated party could be a researcher, statistician, technician or an advisor or consultant. If the research data was developed by a team of researchers, the principal investigator might be the <creator>, while her associates could be described by <associatedParty>. The schemas

*Figure 5*



for both <metadataProvider> and <associatedParty> have essentially the same structure as that of <creator>, however, it should be noted that <associatedParty> also contains a mandatory element called <role>, which is used to define the role that the party played in relation to the research being documented, such as "research assistant" or "technician."

As mentioned earlier, the structure of an EML document is defined by a series of XML schemas. The schema determines not only the names of valid tags, but also the order of the various structures. As can be seen in Figure 4, a <creator> must be described before a <metadataProvider>, and a <metadataProvider> before an <associatedParty>.

Aside from names and addresses, the ResourceGroup schema is also used to provide an overview of the information in the dataset. Some of the more useful ResourceGroup elements include <abstract>, <keywordSet>, <distribution>, and <coverage>. While they are not required elements, both <abstract> and <keywordSet> are useful for describing the general nature of the research being documented, especially when the data is being designed for use with a search engine. The <distribution> and <coverage> elements reappear in later modules, and as they are also optional, they will be discussed later. All of these elements, however, should be listed after the <creator>, <metadataProvider>, and <associatedParty>.

In the <abstract> section, each section is enclosed in the <section></section> tagset, while each paragraph is separated by the <para></para> tagset. For most abstracts, there will be only one section.

As for keywords, the entire set is defined by the <keywordSet> tags, while each individual keyword is separated by a <keyword></keyword> tagset.

```
<abstract>
  <section>
    <para>Small mammal species may be affected by changes in vegetation
    composition and structure. In the southwestern United States, many large
    recent burns have altered the structure and function of pinon ecosystems.
    These burns have affected the structure of dead wood habitats of many mouse
    species. In this study, small mammals in a northern New Mexico pinon and
    juniper forest were live-trapped in both burn and non-burn areas and it was found that
    mouse species (especially Peromyscus species) had been displaced in large burn areas.
    </para>
  </section>
</abstract>

<keywordSet>
  <keyword>pinon forest</keyword>
  <keyword>Peromyscus maniculatus</keyword>
  <keyword>deer mouse</keyword>
  <keyword>New Mexico</keyword>
</keywordSet>
```

The next element to be added after the abstract and keywords is the contact information. A <contact> is a structure for representing the person, organization, or organizational role to contact regarding the use of the data. Although <contact> followed <creator> in the minimal EML document in Figure 2, a closer look at Figures 3 & 4 reveals that <creator>, <metadataProvider>, and <associatedParty> are part of the larger structure of the ResourceGroup, while <contact> is located just below the ResourceGroup in the schema. Any elements that are used in the ResourceGroup must be entered before the next set of elements: <purpose> (optional), <maintenance> (optional), and <contact> (required).

Without getting too deep into the language of XML schema, we can say that both <contact> and <creator> elements have the same structure, that is, both are "of type responsibleParty." The <contact> information follows the same structure as the <creator> information with regard to address, phone number, and online resources.

```
<contact>
  <positionName>Information Manager</positionName>
  <address>
    <deliveryPoint>Oceans LTER</deliveryPoint>
    <deliveryPoint>Department of Marine Ecology</deliveryPoint>
    <deliveryPoint>University of the Oceans</deliveryPoint>
    <deliveryPoint>1514 San Ysidro</deliveryPoint>
    <city>Seaside</city>
    <administrativeArea>LA</administrativeArea>
    <postalCode>78890</postalCode>
    <country>USA</country>
  </address>
  <phone>709-345-8970 x 254</phone>
  <electronicMailAddress>imanager@oceans.edu</electronicMailAddress>
  <onlineUrl>http://oceanslter.lternet.edu</onlineUrl>
</contact>
```

Following the description of the <contact> information for a particular dataset, the optional <project> module is used to provide an overall description of the larger-scale project with which that dataset is associated. For our purposes, the <project> will most often be the LTER site that directed the research.

Accordingly, the <title> here will consist of the name of the LTER site, i.e. "Andrews Experimental Forest." The <personnel> group contains the same elements as <creator> and <contact>, with the addition of a mandatory <role> element, and should be used to identify the lead PI and/or information manager on the site. Other optional elements in the <project> module include <abstract>, <funding>, <studyAreaDescription>, and <designDescription>, each of which can be used to provide a richer textual description of the LTER site that is responsible for the research project being documented.

If used, the <abstract> will include basic information about the LTER site, such as its general history and administration, while <studyAreaDescription> is more of a physical description of the area where the site is located. This description may also include the <coverage> module, which is fully discussed on page XX of this handbook, or the <citation> module, covered on page XX. The <funding> tag is textual and self-explanatory, but <designDescription> is best used for a description of the site's database information and availability.

An EML code fragment that incorporates these various elements might look something like this:

```
<project>
   <title>HJ Andrews Experimental Forest</title>
   <personnel>
      <individualName>
         <givenName>Don</givenName>
         <surName>Henshaw</surName>
      </individualName>
      <electronicMailAddress>henshaw@fsl.orst.edu</electronicMailAddress>
      <role>Data Manager</role>
   </personnel>
   <abstract>
      <para>Several distinctive aspects of the Andrews Forest environment and research
program have placed it center stage in the science and politics of natural resource
management in the region. Basic watershed research in the Pacific Northwest has its roots
in small watershed experiments involving forestry treatments initiated in the 1950s.
The Forest contains extensive examples of old-growth (500 year old) forests, which
were subject of intensive basic research beginning in the 1970s. The original work on
northern spotted owl and its relations with forest habitat was conducted at the Andrews
in the 1970s, setting the stage for extensive monitoring studies that continue. Studies
of carbon cycling over the past two decades have revealed the exceptional properties of
Pacific Northwest forests at the scale of a single tree to the regional scale in terms
of carbon sequestration. The Andrews Experimental Forest serves as a science benchmark
for each of these themes and thus has been examined in terms of its regional context and
representativeness. The Andrews Forest is administered cooperatively by the USDA Forest
Service's Pacific Northwest Research Station (USFS Research), Oregon State University (OSU)
and the Willamette National Forest.</para>
   </abstract>
   <funding>
      <para>Funding for the research program comes from the National Science Foundation
(NSF), Pacific Northwest Research Station, Oregon State University, and other sources.
The Andrews Forest is one of the 24 major ecosystem research sites in the United
States funded through NSF's Long-Term Ecological Research (LTER) Program. The National
Science Foundation-sponsored LTER Program provides relatively stable (6 year funding
cycle) support to the basic science program at the Andrews Forest, and a mechanism for
integration across the larger Andrews research program.</para>
   </funding>
   <studyAreaDescription>
      <descriptor>
         <descriptorValue>Pacific Northwest old-growth forest</descriptorValue>
```

```
    </descriptor>
  </studyAreaDescription>
  <designDescription>
    <description>
      <para>The Forest Science Data Bank (FSDB) is accessible online and includes all
core data sets of the Andrews LTER. Data sets consist of long-term research study data,
spatial data, and analytical tools including software and models.The Forest Science Data
Bank is currently transitioning to a relational database management system. During the
transition, data will be available in a variety of ways. Data sets are listed by research
categories and in the master catalog. Data sets accessible through the new system will be
produced dynamically and will require a one-time user registration. Data sets not yet in
the new system will continue to link to existing static pages.</para>
    </description>
  </designDescription>
</project>
```

One other element that should be discussed relevant to the entire dataset is <access>. This element specifies permissions given to certain groups or individuals regarding access to the data or metadata contained in a particular file. Within <access>, there is a choice of <allow> or <deny>, both of which then require both <principal> and <permission>. The <principal> refers to the name of the individual or organization to which the rule applies, while <permission> specifies the level of access that is being granted, either "read," "write," "changePermission," or "all."

```
<access>
  <allow>
    <principal>Sevilleta LTER</principal>
    <permission>read</permission>
    <permission>write</permission>
  </allow>
</access>
```

## Introducing the data entity

Now that we have defined the basic informational elements of EML, we can add to this with an explanation of the concept of the data entity. As stated earlier, a dataset consists of one or more data entities, and the most common data entity is a <dataTable>. A data table is something that looks like this

Table 1

| OBSERVATION_DATE | OBSERVATION_ LOCATION | SPECIES | SPECIES_COUNT |
|---|---|---|---|
| 2002-10-29 | FCE_001 | ALLIGATOR | 1 |
| 2002-10-29 | FCE_002 | ALLIGATOR | 2 |
| 2002-10-29 | FCE_003 | ALLIGATOR | 0 |

Data tables are produced by people using software applications such as text or word processors, and they often are saved as text files (ascii) or spreadsheets (Excel), by statistical packages (SAS, SYSTAT, SPSS), by database systems (MS Access, My SQL, Oracle, MS SQL Server), or by certain kinds of sensors (data loggers).

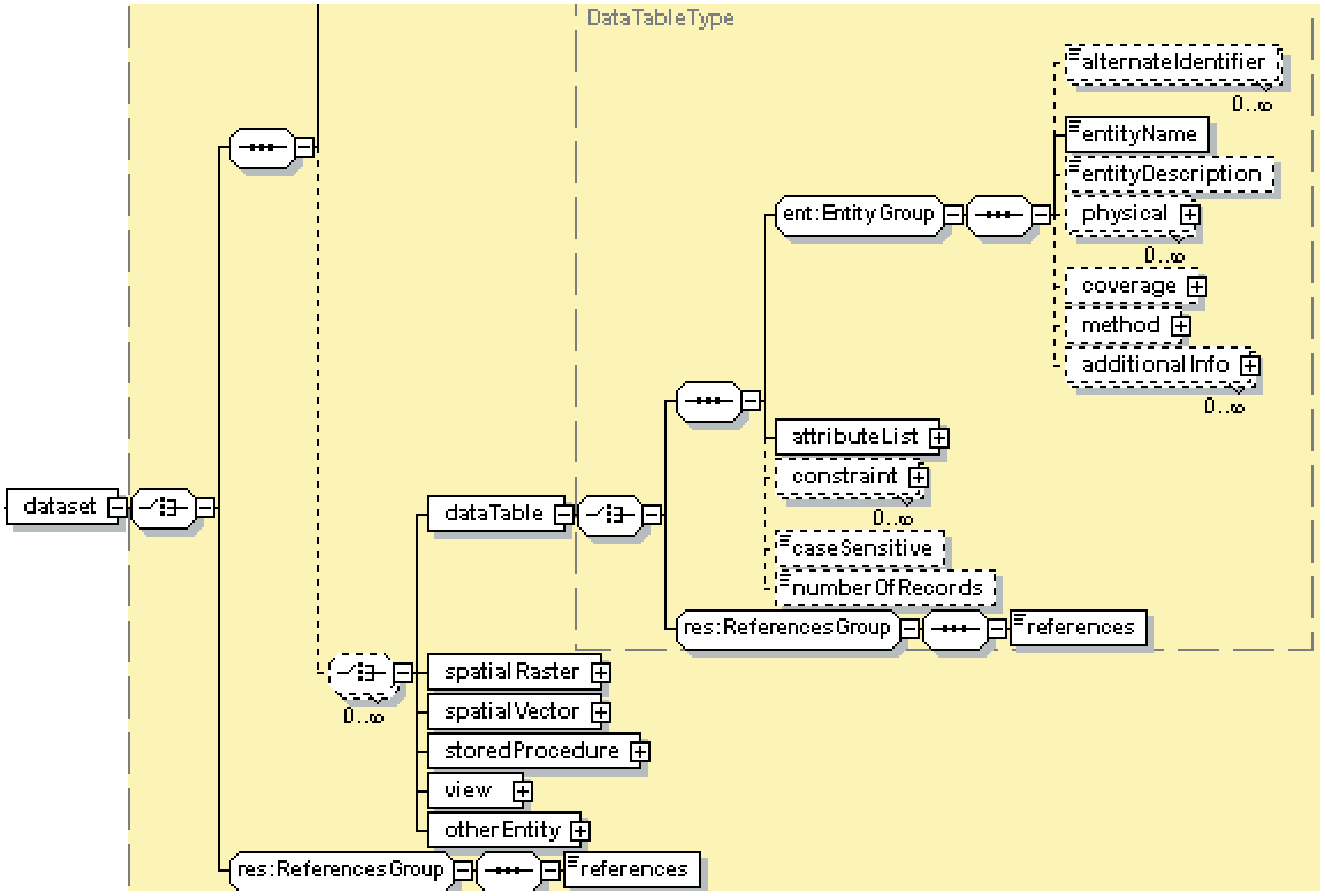
In addition to data tables, people using database applications may also produce a <view> from a database management system or a <storedProcedure> that results in data output.

People using GIS (geographical information system) applications generate both <spatialVector>, also referred to as boundary or shape files, and <spatialRaster>. A <spatialRaster> is a geo-referenced image usually produced by a camera on a satellite or other remote sensing device.

The final kind of data entity is <otherEntity>. An <otherEntity> is a data entity that cannot be represented by any of the previously defined data entity structures. A non-geo-referenced photograph is an <otherEntity>, e.g., a photograph of two different types of butterflies.

The elements must be entered in the sequence shown in Figure 6, and any element that was specified in the ResourceGroup applies to the entire dataset that is being documented. This is especially significant if the dataset has more than one data entity. For example, assume the dataset consists of three files: Climate.xls, Biodiversity.xls, and Productivity.xls. In this case, ResourceGroup elements such as <distribution> or <coverage> must be general enough to apply to all of the files. This would be true also if Climate.xls, Biodiversity.xls, and Productivity.xls were the names of tables in a database.

*Figure 6*



## Metalog 2

**Pat:** Hold on a minute…I'm getting confused. If I am documenting a dataset aren't I documenting just one thing?

**Eml Maven:** Not necessarily. Remember that, "In EML, the term dataset refers to one or more data entities." There is no generally accepted practice on what to include in an EML-dataset document. Some people will choose to have a one-to-one correspondence between an EML document and a data entity and a single physical file. Others will document several data entities in one dataset document.

**Pat:** Let me get very concrete. I go out in the field once a month for a year with a notebook. Let 's say I am doing research on biodiversity and productivity. Each time I go out in the field, I record the temperature and other climate attributes. I also make measurements of primary productivity and do species counts. When I get back to my office I open up my spreadsheet program and enter the climate information in climate.xls, then I record the species counts in biodiversity.xls, and finally enter the productivity information in productivity.xls.

**Eco Informatics:** Let me interrupt for a moment. It is important to recognize that there is a distinction between physical files and data entities. A data entity is a logical category. Let me show you what I mean. Suppose that there are five data attributes: LocationCode, Temperature, Rainfall, Productivity_Index, and Biodiversity_Index. In Example One, all five attributes are in a single table. In this example there is a one-to-one relationship between the data entity and the physical file. In Example Two, the attributes have been broken up into different tables: a table for the climate data, a table for the productivity data, and a third table for the biodiversity data. However, each of these tables has been included in a single Excel file. There is no magic to this being an Excel file; the tables could just as easily be stored sequentially in a simple text file. In this case there are three data entities (logical) in one file (physical). In Example Three, each table has been saved in a different physical file. Once again there is a one-to-one relationship between data entities and physical files.

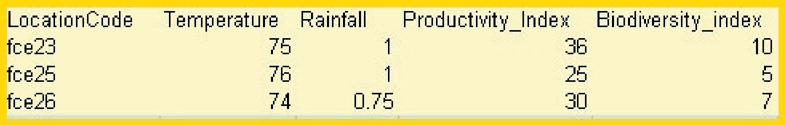
**Eml Maven:** So, if you choose to organize your data as in examples two or three, then you have a data set that consists of three table entities, each containing several data attributes.

**Pat:** Okay, I understand. But what I'm really interested in is providing data that will be useful to other ecologists working on similar projects.

**Eml Maven:** Then you're in luck. The EML schema of data entities and attributes includes all kinds of opportunities for defining and describing your research projects, from the actual physical file to the distribution of the research to the geographic and taxonomic areas that the research covers. But for that, we need to see the modules in action.

**EXAMPLE ONE**

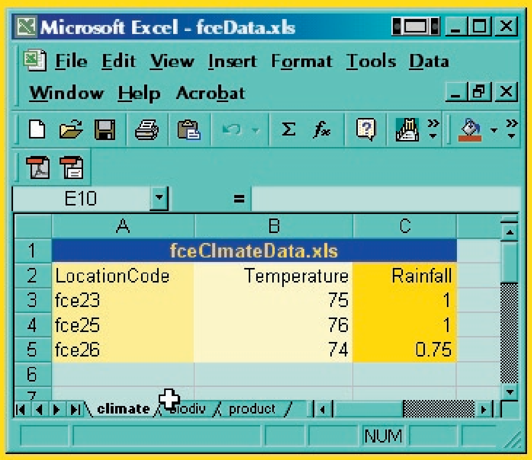
**All Data in a Single Data Entity and a Single Physical File**



| LocationCode | Temperature | Rainfall | Productivity_Index | Biodiversity_index |
|---|---|---|---|---|
| fce23 | 75 | 1 | 36 | 10 |
| fce25 | 76 | 1 | 25 | 5 |
| fce26 | 74 | 0.75 | 30 | 7 |

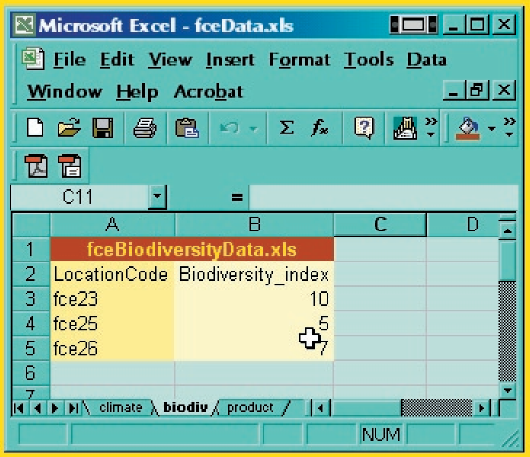
## Example Two

### Three Entities in a Single Physical File


Microsoft Excel - fceData.xls
File Edit View Insert Format Tools Data
Window Help Acrobat
climate biodiv product
NUM

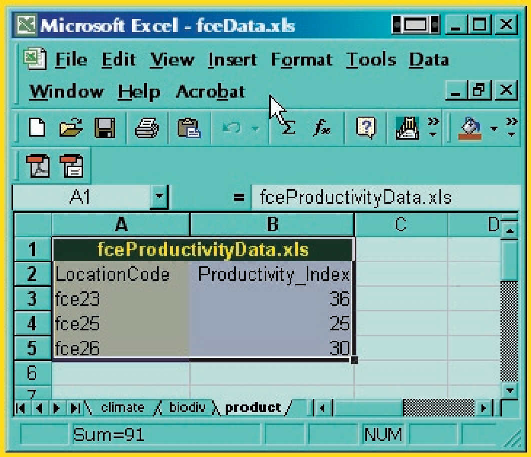
E10 =

| | A | B | C |
|---|---|---|---|
| 1 | fceClmateData.xls | | |
| 2 | LocationCode | Temperature | Rainfall |
| 3 | fce23 | 75 | 1 |
| 4 | fce25 | 76 | 1 |
| 5 | fce26 | 74 | 0.75 |
| 6 | | | |


Microsoft Excel - fceData.xls
File Edit View Insert Format Tools Data
Window Help Acrobat
climate biodiv product
NUM

C11 =

| | A | B | C | D |
|---|---|---|---|---|
| 1 | fceBiodiversityData.xls | | | |
| 2 | LocationCode | Biodiversity_index | | |
| 3 | fce23 | 10 | | |
| 4 | fce25 | 5 | | |
| 5 | fce26 | 7 | | |
| 6 | | | | |

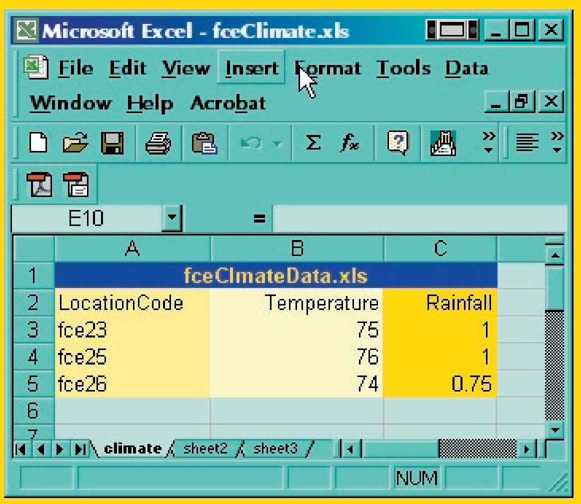

Microsoft Excel - fceData.xls
File Edit View Insert Format Tools Data
Window Help Acrobat
climate biodiv product
Sum=91
NUM

A1 = fceProductivityData.xls

| | A | B | C | D |
|---|---|---|---|---|
| 1 | fceProductivityData.xls | | | |
| 2 | LocationCode | Productivity_Index | | |
| 3 | fce23 | 36 | | |
| 4 | fce25 | 25 | | |
| 5 | fce26 | 30 | | |
| 6 | | | | |

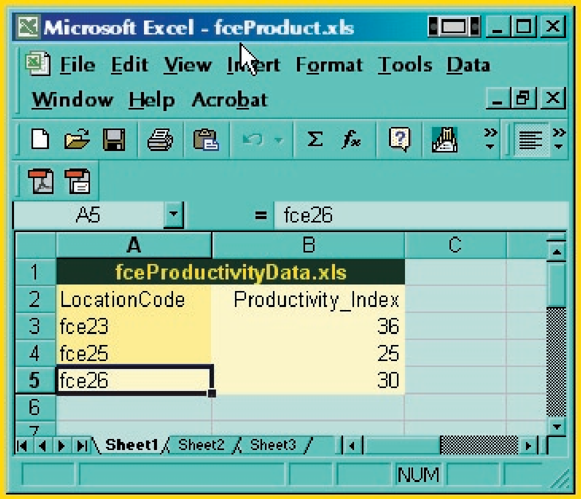
## EXAMPLE THREE

### Three Data Entities (logical) each in a separate Excel File (physical).


Microsoft Excel - fceClimate.xls
File Edit View Insert Format Tools Data
Window Help Acrobat
climate sheet2 sheet3
NUM

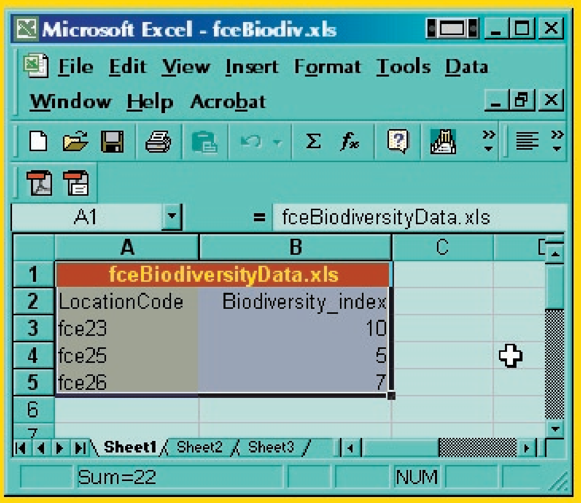
E10 =

| | A | B | C |
|---|---|---|---|
| 1 | fceClmateData.xls | | |
| 2 | LocationCode | Temperature | Rainfall |
| 3 | fce23 | 75 | 1 |
| 4 | fce25 | 76 | 1 |
| 5 | fce26 | 74 | 0.75 |
| 6 | | | |


Microsoft Excel - fceProduct.xls
File Edit View Insert Format Tools Data
Window Help Acrobat
Sheet1 Sheet2 Sheet3
NUM

A5 = fce26

| | A | B | C |
|---|---|---|---|
| 1 | fceProductivityData.xls | | |
| 2 | LocationCode | Productivity_Index | |
| 3 | fce23 | 36 | |
| 4 | fce25 | 25 | |
| 5 | fce26 | 30 | |
| 6 | | | |


Microsoft Excel - fceBiodiv.xls
File Edit View Insert Format Tools Data
Window Help Acrobat
Sheet1 Sheet2 Sheet3
Sum=22
NUM

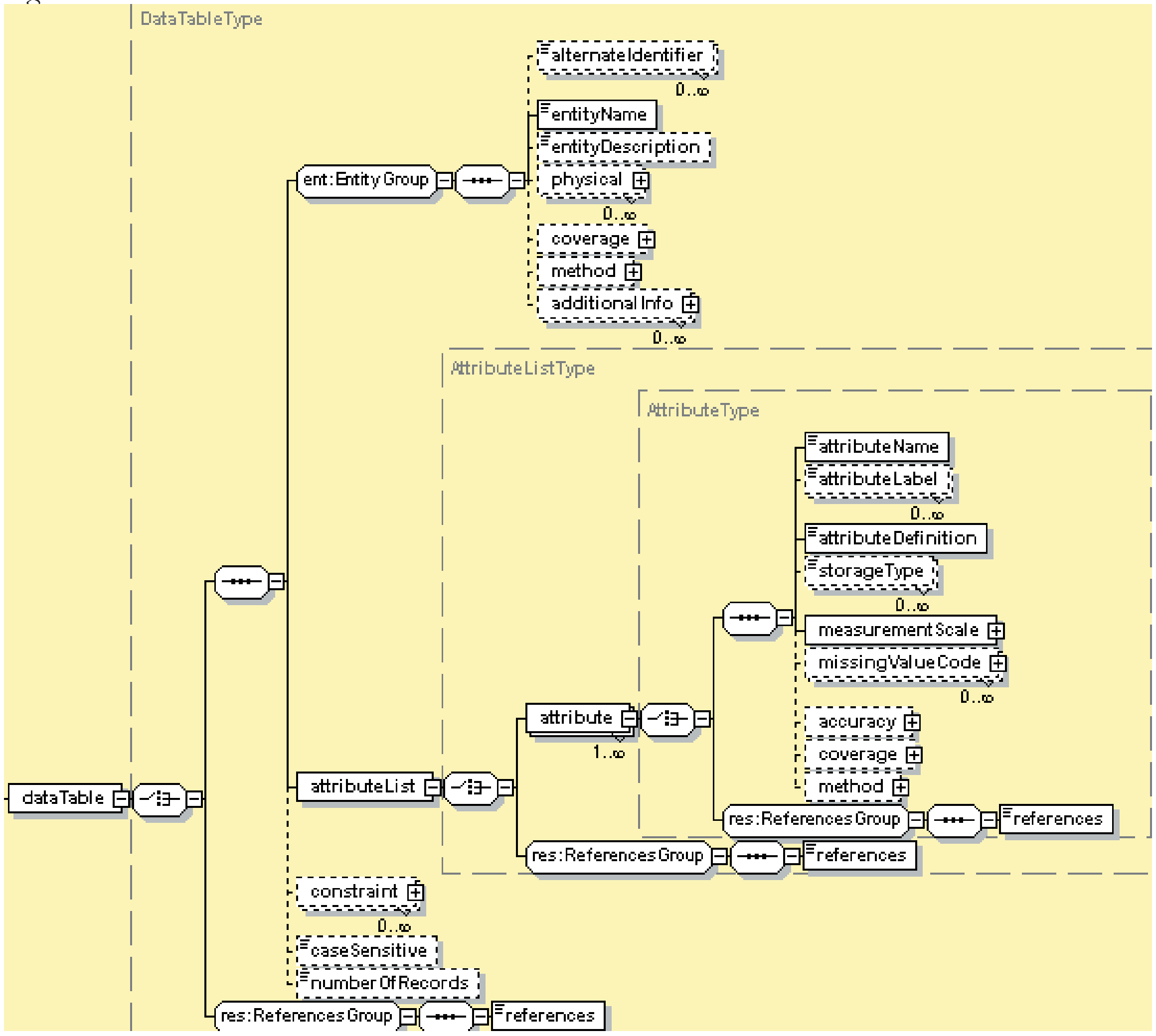
A1 = fceBiodiversityData.xls

| | A | B | C |
|---|---|---|---|
| 1 | fceBiodiversityData.xls | | |
| 2 | LocationCode | Biodiversity_index | |
| 3 | fce23 | 10 | |
| 4 | fce25 | 5 | |
| 5 | fce26 | 7 | |
| 6 | | | |

## Entity and attribute

A look at the expanded schema in Figure 7 reveals the hierarchy of entity to attribute and illustrates that each attribute must correspond to a previously defined entity.

*Figure 7*



In terms of actual code, the entity-attribute hierarchy translates into something like this:

```
<dataset>
     <title/>
     <creator>
        <individualName>
           <surName/>
        </individualName>
     </creator>
     <contact>
        <individualName>
           <surName/>
        </individualName>
     </contact>
     <dataTable>
        <entityName>Groundwater Levels 2001</entityName>
        <entityDescription>Groundwater levels recorded in sample wells throughout the
Trout Lake hydrological basin, North Temperate Lakes LTER study sites</entityDescription>
        <attributeList>
           <attribute>
              <attributeName>SMPLDATE</attributeName>
```

```
         <attributeDefinition>date sample was taken</attributeDefinition>
         <measurementScale>
            <datetime>
               <formatString>YYYY-MM-DD</formatString>
               <dateTimePrecision>1</dateTimePrecision>
               <dateTimeDomain>
                  <bounds>
                     <minimum exclusive="false">1989-01-01</minimum>
                     <maximum exclusive="false">2050-12-31</maximum>
                  </bounds>
               </dateTimeDomain>
            </datetime>
         </measurementScale>
      </attribute>
      <attribute>
         <attributeName>WELLID</attributeName>
         <attributeDefinition>well location identification code</attributeDefinition>
         <measurementScale>
            <nominal>
               <nonNumericDomain>
                  <enumeratedDomain>
                     <codeDefinition>
                        <code>K1</code>
                        <definition>Well at Tyler Creek Station</definition>
                     </codeDefinition>
                  </enumeratedDomain>
               </nonNumericDomain>
            </nominal>
         </measurementScale>
      </attribute>
      <attribute>
         <attributeName>WELL_LEVEL</attributeName>
         <attributeDefinition>water level above the datum</attributeDefinition>
         <measurementScale>
            <ratio>
               <unit>
                  <standardUnit>meter</standardUnit>
               </unit>
                  <precision>0.01</precision>
               <numericDomain>
                  <numberType>real</numberType>
                  <bounds>
                     <minimum exclusive="false">0.0</minimum>
                     <maximum exclusive="false">700.0</maximum>
                  </bounds>
               </numericDomain>
            </ratio>
         </measurementScale>
      </attribute>
   </attributeList>
   <numberOfRecords>2602</numberOfRecords>
  </dataTable>
</dataset>
```

For a complete listing of the ISO 8601 preferred date time formats, go to: **http://www.iso.org/iso/en/prods-services/popstds/datesandtime.html#three**

In this example, there is one entity named "Groundwater Level" and three attributes, "SMPLDATE," "WELLID," and "WELL_LEVEL." The entire list of attributes is enclosed in the <attributeList></attributeList> tagset, but each attribute is part of the overall entity, which, in this case, is stored in a single physical file. This file was written to return data in a format such as the one shown in Table 2 when used in conjunction with a search program.

*Table 2*

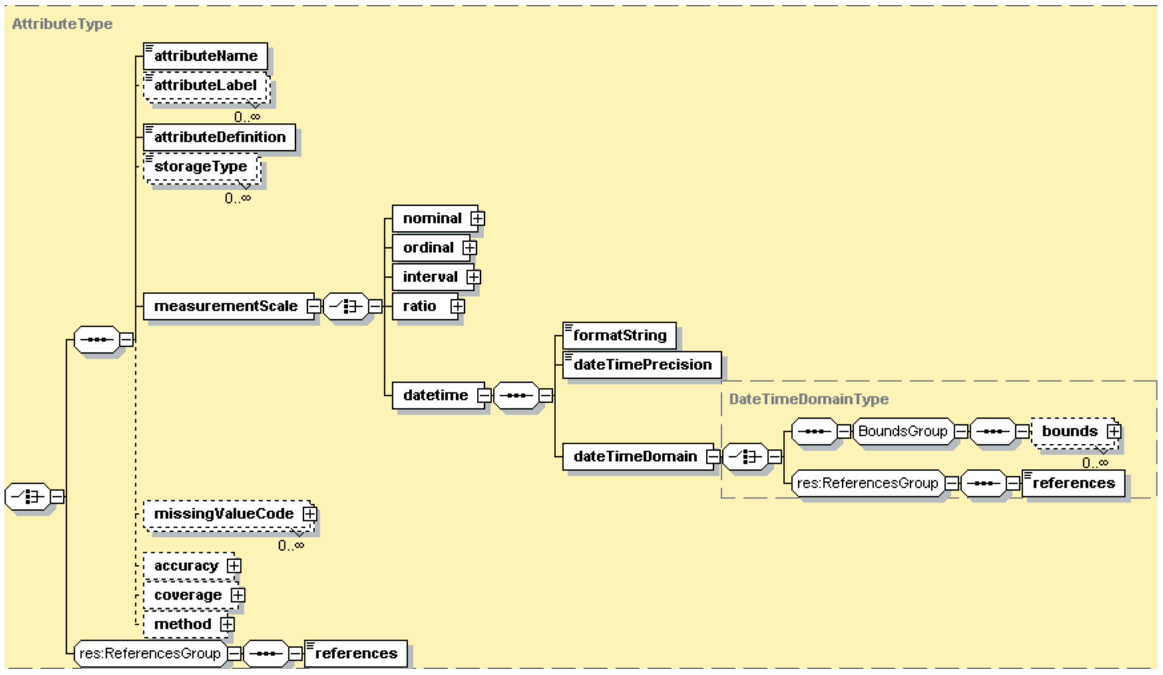
| SMPLDATE | WELLID | WELL_LEVEL |
|---|---|---|
| 03/04/1997 | K1 | 501.93 |
| 04/12/1997 | K1 | 501.99 |
| 03/04/1997 | K2 | 501.51 |
| 04/12/1997 | K2 | 501.69 |
| 03/04/1997 | K5 | 500.92 |
| 04/12/1997 | K5 | 500.99 |
| 03/04/1997 | K6 | 501.24 |
| 04/12/1997 | K6 | 501.38 |
| 03/04/1997 | K9 | 500.48 |

## Inside the data attribute

The actual data contained in these tables generally is not defined in the metadata document, because the purpose of the metadata is mainly to describe data. The information contained within each attribute element offers a description of the type of data it contains. For example, the following section of the previous code describes the format of the SMPLDATE column, using the required attribute fields: <attributeName>, <attributeDefinition>, and <measurementScale>.

```
<attribute>
  <attributeName>SMPLDATE</attributeName>
  <attributeDefinition>date sample was taken</attributeDefinition>
  <measurementScale>
    <datetime>
      <formatString>YYYY-MM-DD</formatString>
      <dateTimePrecision>1</dateTimePrecision>
      <dateTimeDomain>
        <bounds>
          <minimum exclusive=”false”>1989-01-01</minimum>
          <maximum exclusive=”false”>2050-12-31</maximum>
        </bounds>
      </dateTimeDomain>
    </datetime>
  </measurementScale>
</attribute>
```

*Figure 8*



The fields containing the attribute's name and definition are fairly self-explanatory, however, the <measurementScale> field warrants further explanation.

The <measurementScale> tagset is required, and it is a bit more complicated as it contains several other required fields within it. First, the type of <measurementScale> must be selected, be it <nominal>, <ordinal>, <interval>, <ratio>, or <datetime> (see Figure 8). In the short EML fragment above, the selected scale is <datetime>.

One of the required elements in <datetime> consists of the <formatString>, which specifies the format of the date to be returned; here it is in the "YYYY-MM-DD" format. The <dateTimePrecision> tagset defines the smallest unit represented by the <formatString>. In this example, the "1" value means that the dates are precise to the nearest day. If it is necessary to have a format string that is more precise, let's say to the nearest tenth of a second, the <formatString> "hh:mm:ss.ss" would then have a <dateTimePrecision> of 0.1, that is, to the nearest tenth. For most values expressed in days, however, the precision value will be "1."

The exception to this rule occurs when the <datetime> originates from a file that automatically adds values for hours, minutes and seconds to the simple YYYY-MM-DD format. Some types of software, such as Microsoft Access, will automatically do this, adding 00:00:00 to the date when no values for hours, minutes or seconds have actually been specified by the person entering the data. In this instance, the <dateTimePrecision> needs to be entered as "86400" (60 sec. x 60 min. x 24 hours), which is the value that seconds must be multiplied by in order to convert them to days and return a value with the correct precision.

The <dateTimeDomain> then specifies the <bounds> of the valid dates to be returned for a particular attribute. In this case, no dates will be returned that are earlier than January 1, 1989 or later than December 31, 2050.

A closer look at the other two attributes, WELLID and WELL_LEVEL, provides an example of the <nominal> and <ratio> measurement scales.

```
<attributeList>
   <attribute>
      <attributeName>WELLID</attributeName>
      <attributeDefinition>well location identification code</attributeDefinition>
      <measurementScale>
         <nominal>
            <nonNumericDomain>
               <enumeratedDomain>
                  <codeDefinition>
                     <code>K1</code>
                     <definition>Well at Tyler Creek station</definition>
                  </codeDefinition>
               </enumeratedDomain>
            </nonNumericDomain>
         </nominal>
      </measurementScale>
   </attribute>
   <attribute>
      <attributeName>WELL_LEVEL</attributeName>
      <attributeDefinition>water level above the datum</attributeDefinition>
      <measurementScale>
         <ratio>
            <unit>
               <standardUnit>meter</standardUnit>
            </unit>
            <precision>0.01</precision>
            <numericDomain>
               <numberType>real</numberType>
               <bounds>
                  <minimum exclusive="false">0.0</minimum>
                  <maximum exclusive="false">700.0</maximum>
```

```
              </bounds>
            </numericDomain>
          </ratio>
        </measurementScale>
      </attribute>
    </attributeList>
```

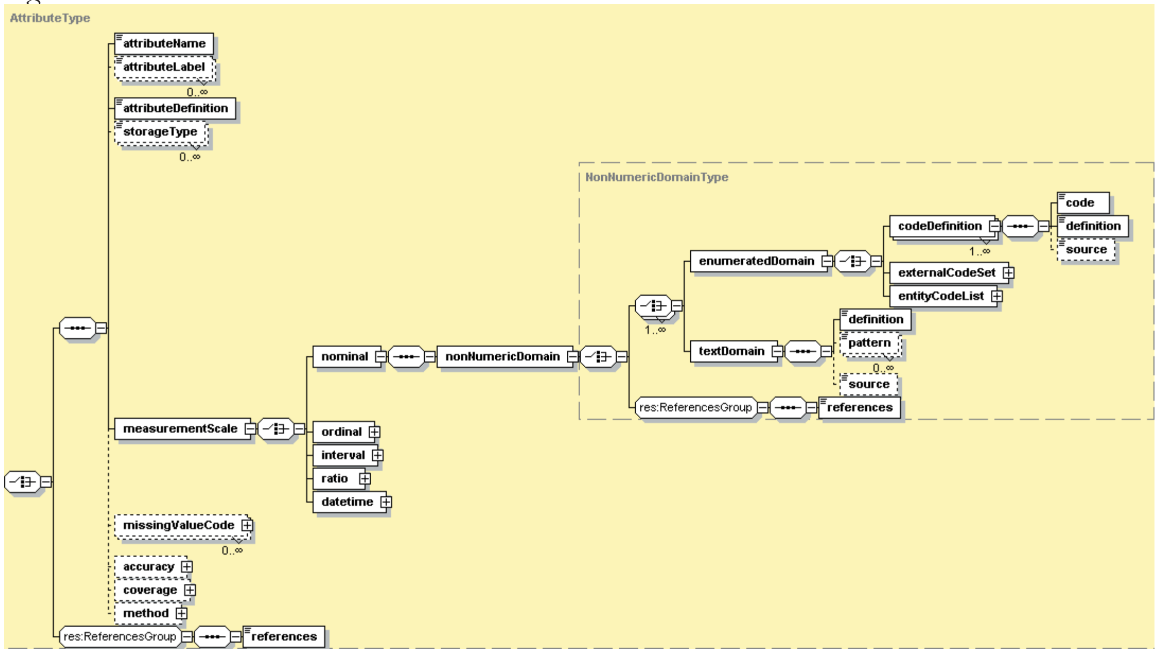
The <nominal> scale is used when data is labeled, either numerically or non-numerically, by a code without any quantitative value, such as in the WELLIDs "K1," "K2," etc. The <nominal> value is further specified as a <nonNumericDomain>, which can be either a <textDomain> or an <enumeratedDomain>, as seen in Figure 9. A <textDomain> only requires a textual <definition> to be complete, but in this case we have an <enumeratedDomain> with a <codeDefinition> consisting of the <code> "K1" and the <definition> "Well at Tyler Creek station."

Other choices of <enumeratedDomain> include <externalCodeSet> and <entityCodeList>. The <externalCodeSet> is selected when the code set being used exists in a separate document, and it is defined with a <codesetName>, a <citation> or a <codesetURL>. The <entityCodeList> consists of a choice of <entityReference>, <valueAttributeReference>, and <definitionAttributeReference>, and it is used to reference a code set that already exists in some other part of the dataset.

The <ordinal> scale has essentially the same schema as the <nominal> in terms of EML code, although it is used when the coded values are part of a ranked scale with some quantitative, but non-arithmetic value, such as "1=poor water quality," "2=fair water quality," etc.

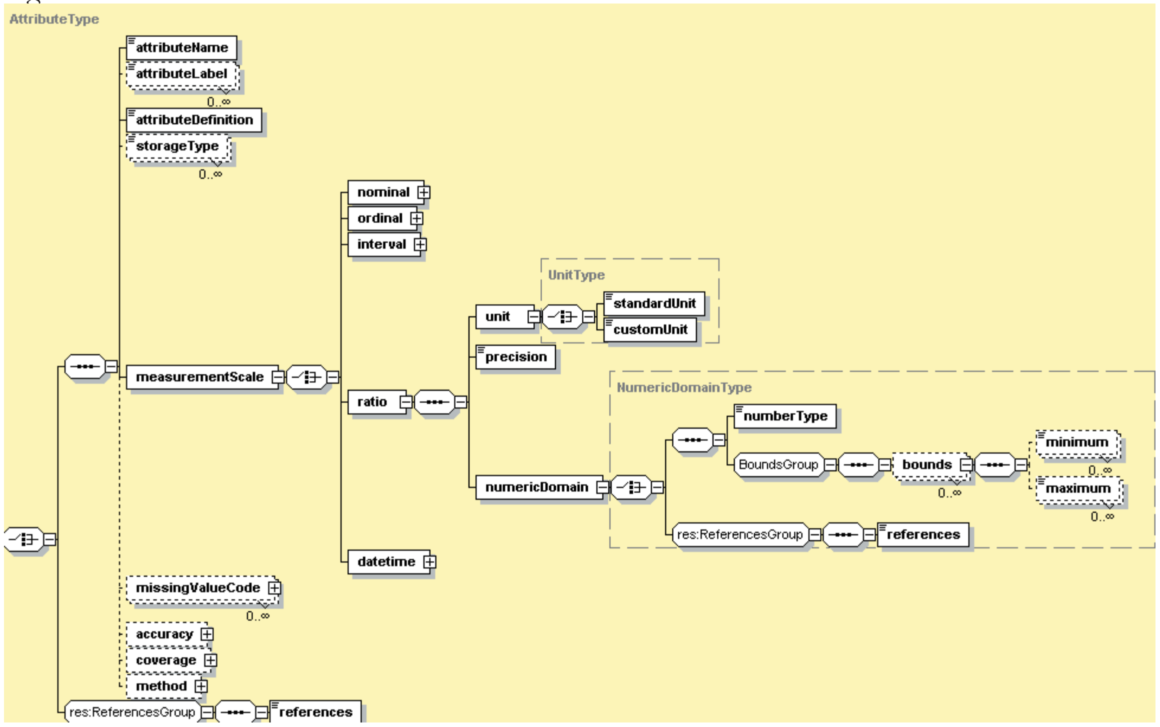
The WELL_LEVEL attribute is defined on the <ratio> measurement scale, however, which differs from the <nominal> and <ordinal> in that the measurements it describes correspond to a real numeric scale that has an actual zero point. Here, the water is being measured at some height relative to 0 meters, so the actual numbers returned are a ratio. This <ratio> scale also works for measurements such as degrees Kelvin or the weight of the biomass in a lake.

*Figure 9*



Required elements for the <ratio> scale include <unit>, <precision>, and <numericDomain> (see Figure 10). The <unit> tag specifies the units being used, in this example the <standardUnit> of a meter. The very useful <customUnit> is also available, which can be used to define units useful to ecologists such as "milligramsPerSquar eMeterPerDay." The <precision> is similar to the <dateTimePrecision> in that it is used to specify the precision of the units being used, such as one hundredth of a meter "0.01" in this example.

*Figure 10*



The <numericDomain> is a small module that requires a <numberType>, which can be "real," as in this example, or "natural," "whole," or "integer." All numbers that contain fractional or decimal elements are described as real, while all non-fractional positive or negative numbers are integers, and whole numbers consist only of positive integers. Natural numbers are all of the whole numbers except zero. When defining a <numberType>, the most restrictive description of the numbers used should be given. For a more in depth discussion of number types, go to (http://www.purplemath.com/modules/numtypes.htm) or see Table 3 below.

Table 3 : Number Types

| NATURAL | WHOLE | INTEGER | REAL |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 4 | 4 | 4 | 4 |
| 23 | 23 | 23 | 0 |
| 10 | 10 | 0 | -24 |
| 75 | 0 | -24 | -735.26 |

The <bounds>, while not required, are then used to describe the <minimum> and <maximum> possible values for a particular attribute. These are helpful to prevent obviously wrong values from being included in the data. In this example, unacceptable values are any values less than 0 or higher than 700. You may notice that the schema does not specifically require either minimum or maximum <bounds>. This is because the <bounds> represent the logically possible range of values and not the actual range of values found in the data. For a well of 700 meters, water levels could logically range from 0-700, but a variable such as water temperature on the Celsius scale could only have the logical range of 0-100, as any value less than 0 would be defined as ice and any value greater than 100 would be defined as water vapor. In certain instances, there may not be a logical minimum or maximum, and so either one or both of those tags could be left out of the <bounds> definition. This is especially useful if the <maximum> bound is unknown or infinite.

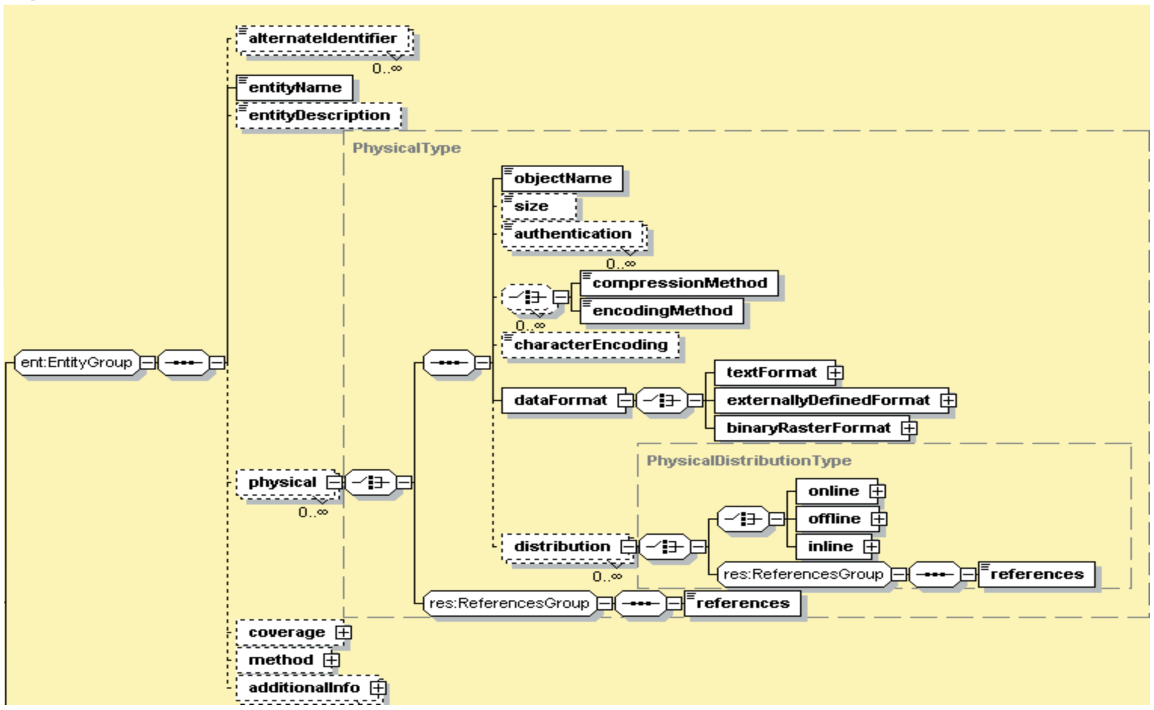
The <interval> scale contains all the same elements as the <ratio> scale, the only difference being that it is used for measurements that do not have an absolute zero value, such as the Celsius or Fahrenheit temperature scales.

## Data description on different levels

Most metadata-defining EML documents need not be extensive; it is up to the individual metadata provider to decide how much description he or she wants to include. As a general rule, however, the more description that is provided, the better, and the structure of EML contains many opportunities to describe the data being presented. Descriptions of data can be provided on the dataset level, the entity level, or the attribute level, depending on how broadly those descriptions apply to the documented research.

For example, between the <entity> and <attribute> levels in the EML schema, there is an optional module that begins and ends with the <physical></physical> tagset (see Figure 11 for this schema).

*Figure 11*



This tagset, while not required, is helpful in that it defines the physical format of the actual data file for which the entity and attribute metadata describe the content. The <physical> module requires both an <objectName>, which is the actual name of the physical data file, ("SO42000.txt" in this example), and a <dataFormat>, which describes the way in which the data is organized in that file. There are several choices of <dataFormat> available in the EML schema: <textFormat>, <externallyDefinedFormat>, or <binaryRasterFormat>.

Of these, the <textFormat> and <externallyDefined Format> are the most straightforward, and will be discussed briefly. The <binaryRasterFormat> is for use with GIS data and is more complex. Researchers who are interested in defining <binaryRasterFormat> for EML should refer to the technical specifications manual.

```
<physical>
   <objectNames>SO42000.txt</objectName>
   <dataFormat>
      <textFormat>
         <attributeOrientation>column</attributeOrientation>
         <simpleDelimited>
            <fieldDelimiter>,</fieldDelimiter>
         </simpleDelimited>
      </textFormat>
   </dataFormat>
   <distribution>
      <online>
         <url function="information">http://www.hubbardbrook.org/research/data/stream/so/
so4/SO42000.txt</url>
      </online>
   </distribution>
</physical>
```

The above EML fragment sample uses a <textFormat>, which merely requires an <attributeOrientation>, either a "column" or a "row," and a mention of how the fields between each row or column are delimited. The <textFormat> offers a choice of <simpleDelimited> or <complex>. The <simpleDelimited> choice requires only a <fieldDelimiter>,

## Metalog 3

| | |
|---|---|
| Pat: | Okay, so let's say my study covers several Peromyscus species and their relationship to pinon and juniper forest ecosystems. A description of this coverage would be really helpful to include in my metadata. How would I go about coding it into EML? |
| EMLMaven: | First, you need to decide what level you want to define the data for. |
| Pat: | Based on what we've discussed so far, I think it would be on the entity level. This study continued over three different summers, so there are three actual data files containing my findings during each time frame. Within each of these files are my species counts for each trapping location, as well as information about pinon and juniper biomass. |
| Eco Informatics: | Based on what you just described, I'd say you were correct in your estimation of the metadata levels. Defining your coverage at the dataset level would be too broad, since there are several different sets of <temporalCoverage>. Each one would need to be defined as a separate entity. What about your locations? Was the <geographicCoverage> the same each summer? |
| Pat: | Yes, I used the same trapping locations each year, and I covered the same species. |
| EMLMaven: | Then you have three different choices of how to integrate this coverage into your metadata: You could define the taxonomic and geographic coverage at the dataset level and then define temporal coverage at each of the entity levels; you could simply repeat the same geographic and taxonomic coverage information for each entity, changing the temporal coverage each time; or, instead of repeating the geographic and taxonomic coverage for each entity, you could define them only once and then use EML references. |
| Pat: | Which option do you recommend? |
| Eco Informatics: | Defining the taxonomic and geographic coverage at the dataset level may look easier and faster, but it is going to make processing the information more difficult for database software and require more complicated logic to interpret. Repeating the information for each entity is the most straightforward, and the one-to-one correspondence between entity and coverage makes it easier for database software to interpret the EML tags. The third option, using references, is probably the most elegant, in that it cuts down on the amount of repetition at the EML level but does not require significantly more code to display at the software level. |
| Pat: | Are references difficult to do? |
| EMLMaven: | Not at all. As we go through the EML coverage module, we'll see just how simple they are to use and apply to many areas of your metadata. |

which in our example is a comma "," the simplest field delimiter. The <complex> delimiters require further information that again can be found in the technical specifications manual for those who are interested in data file minutiae. The <externallyDefinedFormat> requires only a <formatName>, such as "Excel 97" or "Access 2000," which refers the user to an outside file as the template for the data format.

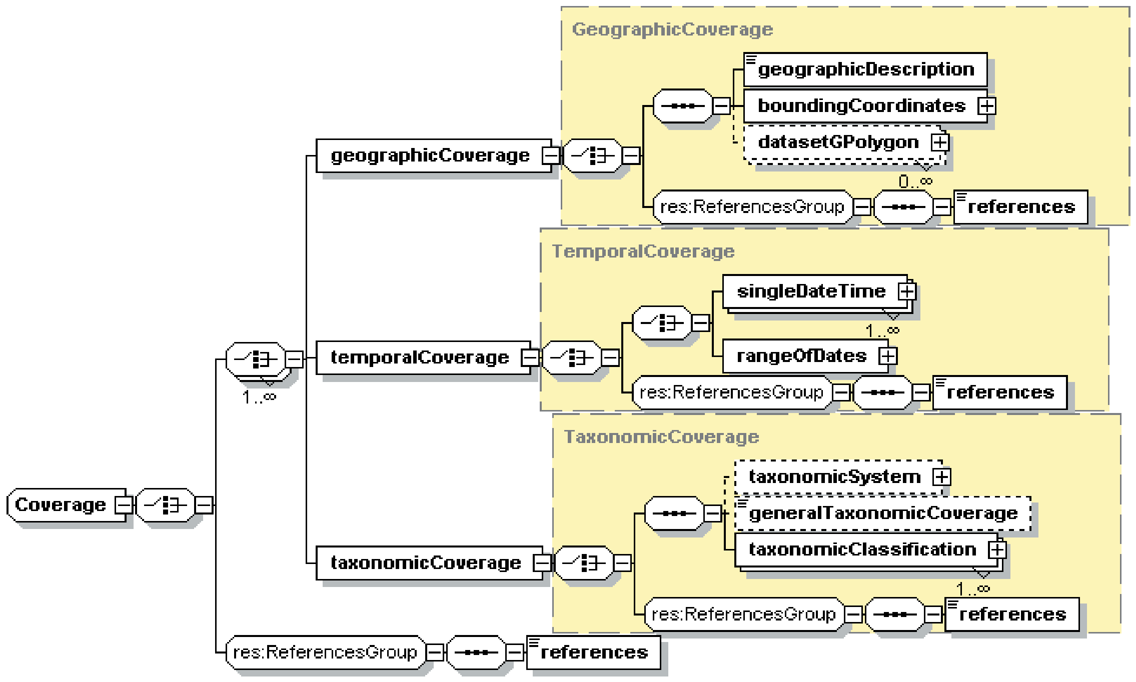
Following the <dataFormat> is a tagset called <distribution>, which warrants further explanation. The <distribution> describes where the data file can be found, either in <online> or <offline> source material. The <online> tag allows a metadata provider to specify a <url> where the data may be accessed, while the <offline> tag offers a choice of the name, volume, and format of the medium where the data can be found. In this example, the data is located in an <online> file at `http://www.hubbardbrook.org/research/data/stream/so/so4/SO42000.txt`. An additional <inline> tag may be used to specify data contained within the metadata file, and it is being further developed for EML usage.

The entire <physical> tagset is important in that it is an example of a description that is defined at the entity level. That is, the <objectName> and <distribution> now apply to all of the attributes within that entity. If one were then to refer to a different file in a different location, a new <entity> containing a new <physical> description and distribution would need to be defined.

The tricky part is that <distribution> may also be defined earlier, within the ResourceGroup module. If one were to define <distribution> in this earlier module, the <online> or <offline> location specified would then apply to the entire <dataset>, not just one <dataTable> entity.

This is even more important with the <coverage> module, which may be defined at the <dataset>, <entity>, or <attribute> level. Coverage is broken down into <geographicCoverage>, <temporalCoverage>, and <taxonomicCoverage> (see Figure 12). The <geographicCoverage> tagset requires a <geographicDescription> and a set of <boundingCoordinates>, both of which provide information about the area being studied. The timeframe of the study is described using <temporalCoverage>, which offers a choice of a <singleDateTime> or a <rangeofDates>. For <taxonomicCoverage>, all that is required is a <taxonomicClassification>. It is up to the metadata provider to decide what type of coverage, if any, is appropriate for each level of metadata; that is, whether the coverage applies to an entire dataset, or just to one specific entity or attribute.

*Figure 12*



## Using coverage to describe data

The coverage module utilizes the same format no matter what level it is located on within the dataset. There is a choice among <geographicCoverage>, <temporalCoverage>, and/or <taxonomicCoverage> within each module, but all three are not required at a single level. Theoretically, this means that <geographicCoverage> could be defined at the dataset level, while <temporalCoverage> could be defined at the entity level, and <taxonomicCoverage> could be specified at the <attribute> level. Most likely, though, at least two of the three coverage elements will be defined at the same level.

An EML fragment that describes all three types of coverage at the entity level is shown below:

```
<dataTable>
  <entityName>Rodents of the Southwest Summer 1999</entityName>
  <entityDescription>Rodent species in central New Mexico and their relationship to pinon
and juniper forest</entityDescription>
  <coverage>
    <geographicCoverage>
      <geographicDescription>northeast section of Sevilleta National Wildlife Refuge,
Socorro County, New Mexico, Deep Well to Cerro Montoso. Grass/shrubland/desert to pinon-
juniper woodland.</geographicDescription>
      <boundingCoordinates>
        <westBoundingCoordinate>106.689</westBoundingCoordinate>
        <eastBoundingCoordinate>106.522</eastBoundingCoordinate>
        <northBoundingCoordinate>34.408</northBoundingCoordinate>
        <southBoundingCoordinate>34.305</southBoundingCoordinate>
        <boundingAltitudes>
          <altitudeMinimum>1601</altitudeMinimum>
          <altitudeMaximum>1976</altitudeMaximum>
          <altitudeUnits>meters (above sea level)</altitudeUnits>
        </boundingAltitudes>
      </boundingCoordinates>
    </geographicCoverage>
    <temporalCoverage>
      <rangeOfDates>
        <beginDate>
          <calendarDate>1999-05-23</calendarDate>
        </beginDate>
        <endDate>
          <calendarDate>1999-08-15</calendarDate>
        </endDate>
      </rangeOfDates>
    </temporalCoverage>
    <taxonomicCoverage>
      <generalTaxonomicCoverage>desert/shrubland and pinon-juniper forest dwelling mouse
species of genus Peromyscus</generaltaxonomicCoverage>
      <taxonomicClassification>
        <taxonRankName>genus</taxonRankName>
        <taxonRankValue>Peromyscus</taxonRankValue>
        <commonName>deer mice, white-footed mice</commonName>
        <taxonomicClassification>
          <taxonRankName>species</taxonRankName>
          <taxonRankValue>Peromyscus maniculatus</taxonRankValue>
          <commonName>deer mouse</commonName>
        </taxonomicClassification>
        <taxonomicClassification>
          <taxonRankName>species</taxonRankName>
          <taxonRankValue>Peromyscus boylii</taxonRankValue>
          <commonName>brush mouse</commonName>
        </taxonomicClassification>
```

```
          <taxonomicClassification>
            <taxonRankName>species</taxonRankName>
            <taxonRankValue>Peromyscus difficilis</taxonRankValue>
            <commonName>rock mouse</commonName>
          </taxonomicClassification>
          <taxonomicClassification>
            <taxonRankName>species</taxonRankName>
            <taxonRankValue>Peromyscus eremicus</taxonRankValue>
            <commonName>cactus mouse</commonName>
          </taxonomicClassification>
          <taxonomicClassification>
            <taxonRankName>species</taxonRankName>
            <taxonRankValue>Peromyscus leucopus</taxonRankValue>
            <commonName>white-footed mouse</commonName>
          </taxonomicClassification>
          <taxonomicClassification>
            <taxonRankName>species</taxonRankName>
            <taxonRankValue>Peromyscus truei</taxonRankValue>
            <commonName>pinon mouse</commonName>
          </taxonomicClassification>
        </taxonomicClassification>
      </taxonomicCoverage>
    </coverage>
</dataTable>
```

Here, <coverage> is located after the <entityName> and <entityDescription> are defined, but before any attributes are listed. The <geographicCoverage> is listed first, and it requires a <geographicDescription>, with a sentence or two describing the general area of the study, and a set of <boundingCoordinates>, which use latitude and longitude values at the east, west, north and south boundaries to block off a rough rectangular location for the study area. Metadata providers should note that the original EML technical specifications require values in decimal degrees rather than the standard degrees, minutes, seconds for latitude and longitude. Decimal degree values can be easily obtained using an online converter such as the one located at http://www.beg.utexas.edu/GIS/tools/DMS_DD.htm, which also performs conversions back from decimal degrees to degees, minutes, and seconds.

The <boundingCoordinates> also include elements of altitude, which are optional. These <boundingAltitudes> consist of the three required fields <altitudeMinimum>, <altitudeMaximum>, and <altitudeUnits>. These altitude values can be describe in whatever units are preferred, but the <altitudeUnits> field must specify those units in relation to some datum.

The <temporalCoverage> offers the choice of <singleDateTime> or <rangeOfDates>, both of which require a <calendarDate> in a simple YYYY-MM-DD format. The <rangeOfDates> option requires both a <beginDate> and an <endDate>, as shown in the above example.

Both <singleDateTime> and <rangeofDates> have an optional <time> element which, if used, should be entered using the hh:mm:ss format followed by a time zone designator. If Coordinated Universal Time (UTC) or Greenwich Mean Time is used, the time zone is designated by a "Z", as in hh:mm:ssZ. All other time zones are indicated with a + or - followed by the number of hours they are ahead or behind the UTC, such as hh:mm:ss-8:00, which indicates Pacific Standard Time (8 hours behind UTC).

Following the <time> field, there is also an optional <alternativeTimeScale> element, which is useful for describing dates that do not meet the standard YYYY-MM-DD format, such as the Geologic Time Scale. To use the <alternativeTimeScale>, both a <timeScaleName> and <timeScaleAgeEstimate> must be specified. For example, if the <timeScaleName> was "International Geologic Time Scale," the <timeScaleAgeEstimate> might be given as "Jurassic."

The <taxonomicCoverage> requires a <taxonomicClassificiation>, which consists of a <taxonRankName> indicating the taxonomic level being defined, in this case "genus," and a <taxonRankValue>, which is the actual name of the genus, "Peromyscus." The <commonName> (or names, if there are several) can also be included. The structure of the <taxonomicCoverage> module also allows for a nesting of listings, such as in the above example, where the <taxonRankValue> and <commonName> of each separate species in the study is then listed individually in its own <taxonomicClassification> within the larger classification. This can be done at every level, beginning with Kingdom, and defining the rank and value for each.

## References vs. repetition

The previous EML fragment presents all three types of <coverage> defined at the level of a single entity. This type of structure is perfectly suited for the dataset that contains only one entity and several attributes, all of which cover the same material. When there are several entities that share some of the same coverage elements, however, as in the example described in Metalog 3, it may not be desirable to repeat the same coverage information several times as it can become unwieldy in the EML code.

While you could define some types of coverage at the dataset level and others at the entity level, the best way to handle this situation is to use EML references within the entities.

By this method, you can define all the coverage elements within the first entity to which they apply and then, instead of repeating the information, use references in the following entities to note coverage information that is the same.

Let's say that the EML fragment above represents the first of three entities for which you are defining coverage. The <geographicCoverage> and <taxonomicCoverage> are the same for all three entities; it is only the <temporalCoverage> that changes. The only change you would need to make to the above code would be to define an ID for both geographic and taxonomic coverage in the following format:

```
<geographicCoverage id="sevill.1">
  <geographicDescription>northeast section of Sevilleta National Wildlife Refuge, near
Socorro, NM</geographicDescription>
  <boundingCoordinates>
    <westBoundingCoordinate>106.689<westBoundingCoordinate>
    <eastBoundingCoordinate>106.522<eastBoundingCoordinate>
    <northBoundingCoordinate>34.408<northBoundingCoordinate>
    <southBoundingCoordinate>34.305<southBoundingCoordinate>
  </boundingCoordinates>
</geographicCoverage>
```

The only difference is that instead of using the tag <geographicCoverage> to open your definition of that element in the first entity, you use <geographicCoverage id="xxx">, with the ID name you give to your reference included within the quotation marks. If you then opened taxonomic coverage with <taxonomicCoverage id="peromys.1">, the next two entities would then have coverage defined like this:

Second Entity

```
<dataTable>
  <entityName>Rodents of the Southwest Summer 2000</entityName>
  <entityDescription>Rodent species in central New Mexico and their relationship to pinon
and juniper forest</entityDescription>
  <coverage>
    <geographicCoverage>
      <references>sevill.1</references>
    </geographicCoverage>
    <temporalCoverage>
      <rangeOfDates>
        <beginDate>
          <calendarDate>2000-05-18</calendarDate>
        </beginDate>
        <endDate>
          <calendarDate>2000-08-22</calendarDate>
        </endDate>
      </rangeOfDates>
    </temporalCoverage>
    <taxonomicCoverage>
      <references>peromys.1</references>
    </taxonomicCoverage>
  </coverage>…
```

**Third Entity**

```
<dataTable>
   <entityName>Rodents of the Southwest Summer 2001</entityName>
   <entityDescription>Rodent species in central New Mexico and their relationship to pinon
and juniper forest
   </entityDescription>
   <coverage>
      <geographicCoverage>
         <references>sevill.1</references>
      </geographicCoverage>
      <temporalCoverage>
         <rangeOfDates>
            <beginDate>
               <calendarDate>2001-05-31</calendarDate>
            </beginDate>
            <endDate>
               <calendarDate>2001-08-28</calendarDate>
            </endDate>
         </rangeOfDates>
      </temporalCoverage>
      <taxonomicCoverage>
         <references>peromys.1</references>
      </taxonomicCoverage>
   </coverage>
...</dataTable>
```

The reference module does not exist for the sole purpose of defining coverage elements. References can be used with any EML element that is referred to later in the dataset. For example, if your <creator> and <contact> are the same person, you can use the tag <creator id="pat.eco23"> to open your creator description and then simply refer to your contact information as

```
<contact>
   <references>pat.eco23</references>
</contact>
```

This technique also works within and among attributes (if you have several entities that share attributes in common), with any of the <physical> file descriptors, or with <citation>, which is another important module that will be defined shortly.

## Knowing the details: Methods, protocols and citations

The <methods> module is located in the EML schema just below <contact> on the dataset level, and also within the EntityGroup and the <attribute> module. There are three element choices within this module, consisting of <methodStep>, <sampling>, and <qualityControl>. The first and most common type of method is the required <methodStep>, which contains a variety of elements that assist in describing procedures.

The only required element within the <methodStep> is a <description>. This <description> is written in paragraph format and is enclosed in the <para></para> tagset, except in cases where it is long enough to require the additional <section></section> tagset as well. There is a choice of formatting within the

## Metalog 4

Pat: I was just going to ask about that. Citations are important to provide for any other ecologist looking at my metadata. So are my methods, for that matter. Can I use references with those, too?

EMLMaven: References can be used with most modules, but at this time they are not designed for use with the methods module. The idea is that method entries are generally unique. In most cases, your methods will either apply to your entire dataset, or only to a single entity or attribute within your dataset. If your methods apply to multiple entities, then you can just place your <methods> descriptions at the dataset level, where they will apply to all entities and attributes. If it's a standardized method that you want to reference, then you can publish it as a <protocol> instead. Protocols are located within the methods module on each level of the dataset, and the protocol module does include references.

Pat: Aren't protocols and methods essentially the same thing?

Eco Informatics: Not necessarily. In EML, a protocol is a procedure that prescribes a standardized method, while a method refers to a method that was actually performed in the process of acquiring the data. A method describes what the researcher or research team did, such as "We trapped rodents at these five locations using Sherman live traps." Protocols would refer to a set standard such as instructions for using the traps and are more imperative in nature, such as "Bait the trap using 3 oz. of steamed, crimped oats."

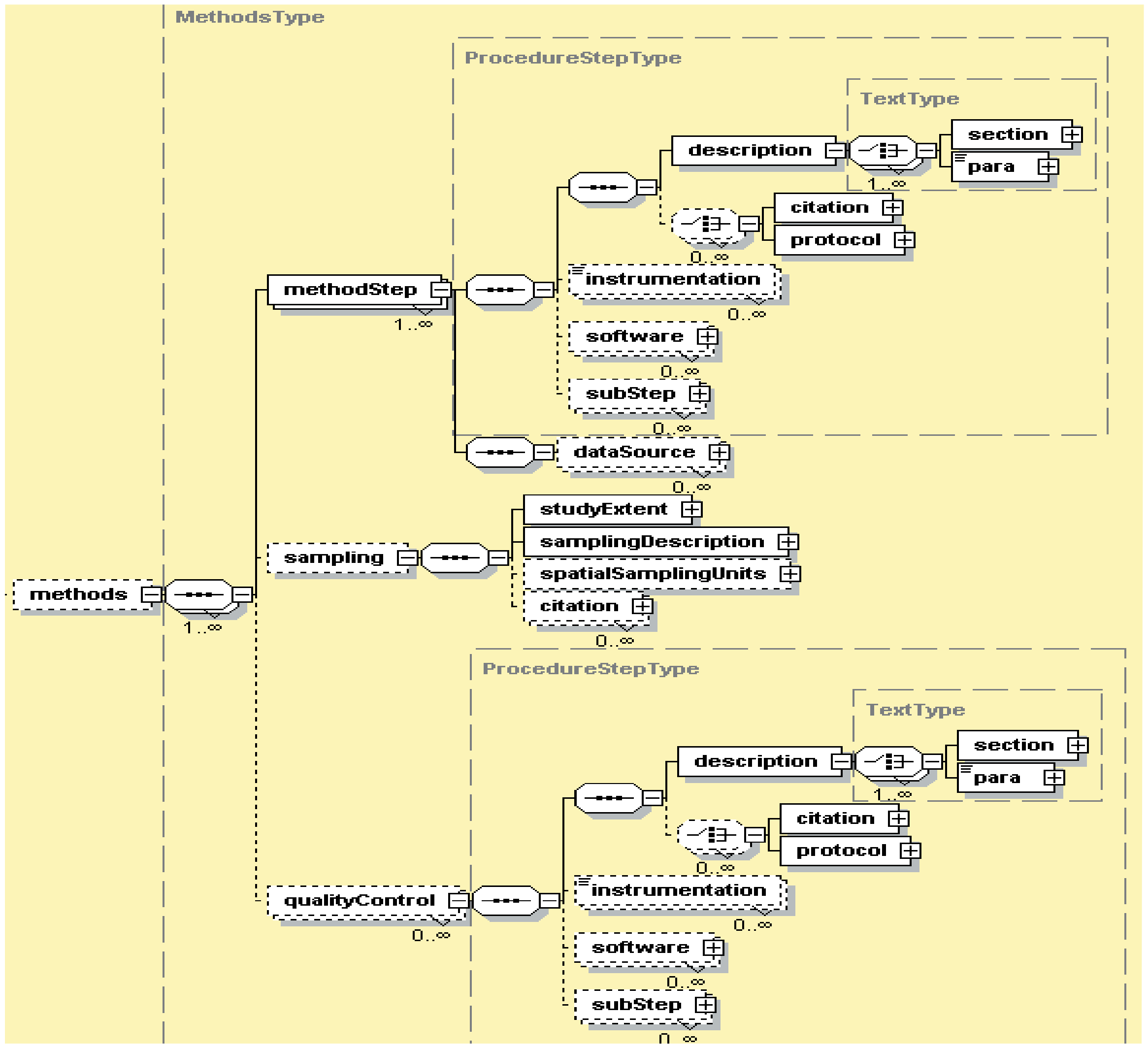
EML Maven: That is why protocols are located in the <methods> modules. In the course of describing your own methods for acquiring data, you may call on or reference previously published protocols that do not come from your own data. Citations are also located within the methods module, so we can examine all three of these elements at once.

<para></para> tagset that, while not required, is useful if the method description is written in list format rather than simple paragraph format.

Formatting choices within <para></para> include <itemizedlist>, <orderedlist>, <emphasis>, <subscript>, <superscript>, and <literalLayout>. Most of these markup tags are fairly self-explanatory, although it should be noted that <itemizedlist> is for use with bulleted lists, while <orderedlist> corresponds to numbered lists. The <literalLayout> element is one that specifies that none of the text structure (whitespace) within the tagset text should be altered.

If a <methodStep> description is to be organized into an itemized list, the entire list should be enclosed in the <itemizedlist> tagset, with each separate item as its own <listitem> entry, which, in turn, has its own <para></para> tagset within it. The EML schema for this is shown in Figure 13.

*Figure 13*



In EML, the code fragment for this type of list would resemble the following example:

```
<methods>
   <methodStep>
      <description>
         <para>
            <itemizedlist>
               <listitem>
                  <para>Traps were set at a locality for four trap-nights.</para>
               </listitem>
               <listitem>
                  <para> Each trap was baited with a handful of steamed, crimped oats tossed
into the trap after it was placed on the ground; a few oats were left outside the trap
entrance to entice passers-by.</para>
               </listitem>
               <listitem>
                  <para> During expectedly cold nights, and during all of the fall period, a
handful of cotton batting was also placed to the rear of each trap to help keep captures
warm overnight.</para>
               </listitem>
            </itemizedlist>
         </para>
      </description>
   </methodStep>
</methods>
```

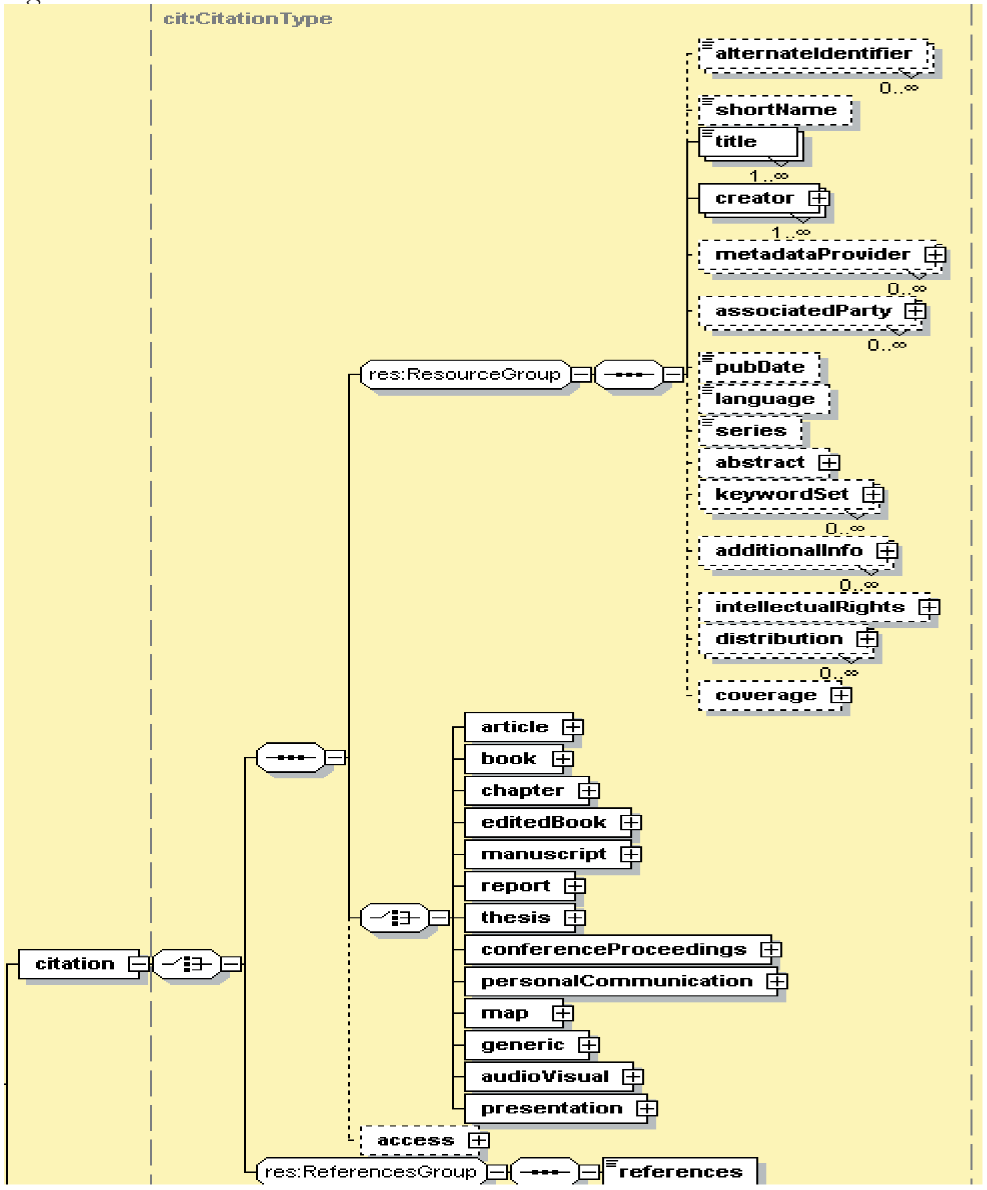
When used with an xslt script, the above EML fragment returns a list with a format that looks like this:

- Traps were set at a locality for four trap-nights.
- Each trap was baited with a handful of steamed, crimped oats tossed into the trap after it is placed on the ground; a few oats were left outside the trap entrance to entice passers-by.
- During expectedly cold nights, and during all of the fall period, a handful of cotton batting was also placed to the rear of each trap to help keep captures warm overnight.

Aside from the <description>, <methodStep> also includes an option of using the <citation> and <protocol> modules to reference additional information.

When expanded, the <citation> module opens into two separate sections (see Figure 14), one of which is the ResourceGroup module described on page XX. Elements from the ResourceGroup module that are used with most citations include <title>, <creator> (author), and <pubDate>, while the second section provides the means to describe the type of work that is being cited.

*Figure 14*



There is a large selection of citation types available, from <article>, <book>, <editedBook>, and <manuscript> elements to <conferenceProceedings>, <personalCommunication>, and <audioVisual> materials, each of which contain any number of required and optional descriptors. Most of these elements are very straightforward, such as those within <book>, which contains the required elements <publisher> and <organizationName>. Other optional, but helpful, elements within <book> include <publicationPlace>, <edition>, and <volume>.

Similarly, <article> requires the elements <journal>, <volume>, and <pageRange>, while <issue>, <publisher>, <publicationPlace>, and <ISSN> are optional.

It is somewhat excessive to describe every element within each citation component, as most of them are descriptive enough not to need detailed explanation. For those elements that are not self-explanatory, a complete breakdown of the <citation> schema and its various components is available in the EML technical specifications manual (see Appendix XX)

The important thing to remember about the <citation> module as a whole is that any ResourceGroup elements used in the citation description must be defined first, before the actual citation type is defined. An EML fragment for a typical <book> type of citation would resemble the following example:

```
<citation>
  <title>Limnological Methods</title>
  <creator>
    <individualName>
      <givenName>R.G.</givenName>
      <surName>Wetzel</surName>
    </individualName>
  </creator>
  <creator>
    <individualName>
      <givenName>G.E.</givenName>
      <surName>Likens</surName>
    </individualName>
  </creator>
  <pubDate>1979</pubDate>
  <book>
    <publisher>
      <organizationName>Saunders</organizationName>
    </publisher>
    <publicationPlace>Philadelphia</publicationPlace>
  </book>
</citation>
```

Notice that the title, name, and publication date elements all occur in the EML code before the <book> type is specified, along with the <book> elements of <publisher>, <organizationName>, and <publicationPlace>. This hierarchy within the <citation> module is the same for all elements within it and it occurs in the same form wherever the <citation> module is found within the EML structure.

The <protocol> module follows <citation> within <methodStep>, although it, too, is found in various other places within the EML structure. As discussed earlier, <protocol> is designed for use with standardized series of instructions, especially those that are referenced from outside the particular dataset in which they occur.

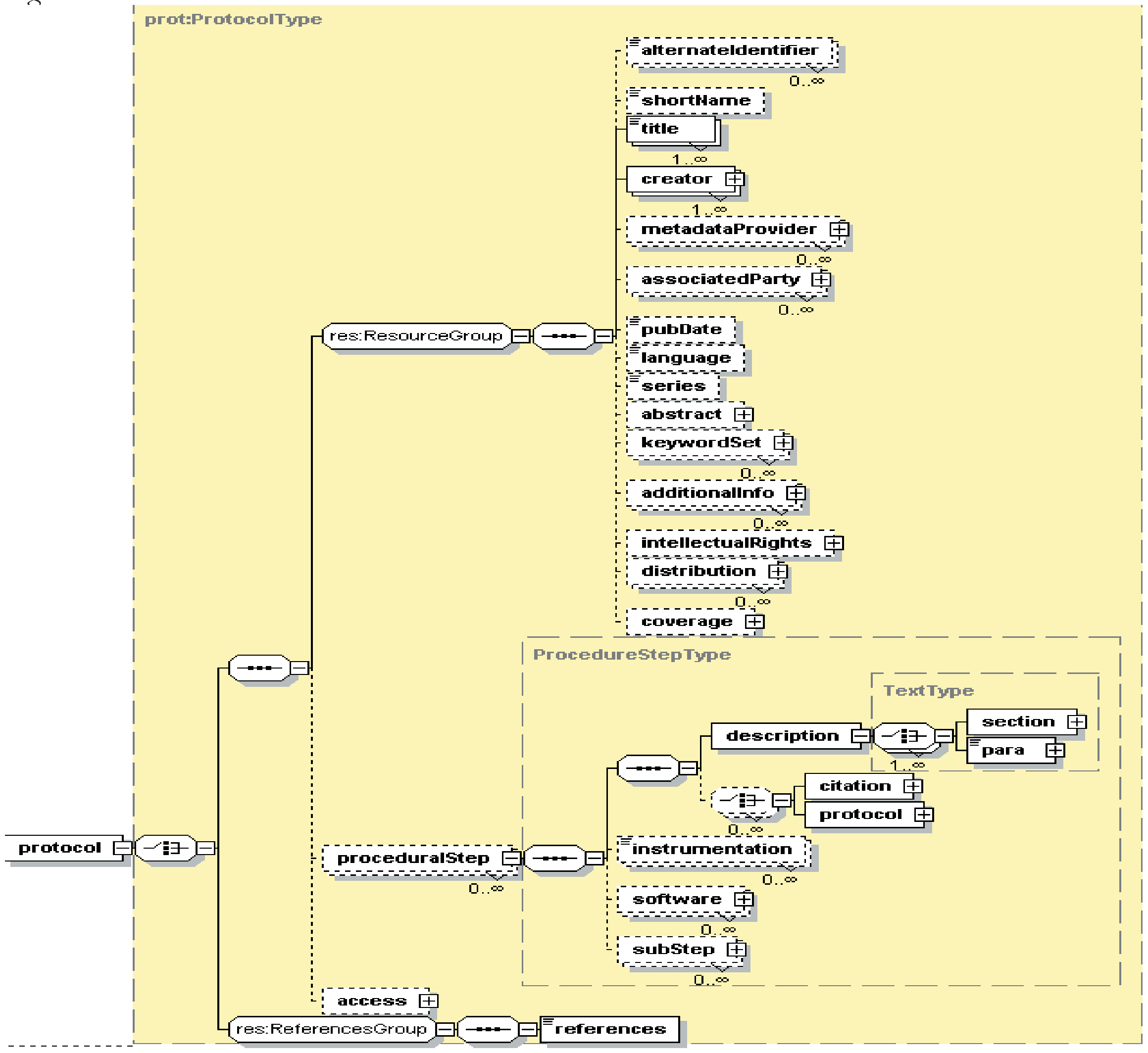
Like <citation>, <protocol> also expands into the ResourceGroup module, allowing the <title> and <creator> of the protocol to be identified (see Figure 15). The <protocol> module also contains an optional <proceduralStep> element, which requires a <description> in the aforementioned <para></para> format of the standardized procedure being described, and an optional <access> component in the same form as the one described on page XX.

Returning now to the remainder of the elements within <methodStep>, there are several optional elements located after the <protocol> module (refer to Figure 13). These include <instrumentation>, <software>, and <subStep>, as well as <dataSource>.

The <instrumentation> element provides space to describe any instruments that were used in the process of acquiring data, such as "Sherman live trap" or "Turner Designs 10-Au-005-CE fluorometer." There are no additional description tags within the element, so <instrumentation>Sherman live trap</instrumentation> is all the code that would be required.

The <software> module is for use mainly with custom software that was used in the process of documenting the dataset. It is not meant for use in describing commercial applications, such as Excel or Access, and will most likely not be used by the average researcher during metadata documentation. The technical specifications manual will provide an explanation for those who wish to explore this module further.

*Figure 15*



A <subStep> contains the same form as its parent element <methodStep> (or <proceduralStep> if it is located within a <protocol>), and is used only when a minute breakdown of the method steps is necessary. The <dataSource> element has the same format as the larger <dataset> module, including entities, attributes and everything in between. This element is used only when the submitted data is based on synthetic research drawn from several external datasets, and it allows these separate datasets to then be documented individually within the larger context of the new, synthesized dataset.

The two remaining <methods> elements we will discuss are the optional <sampling> and <qualityControl>, both of which provide more detailed information about study procedures. Within <sampling> are the required elements <studyExtent> and <samplingDescription>, and the optional <spatialSamplingUnits>, as well as another optional <citation> module. Most of these units are composed of elements that we have seen before; it is only their direct relationship to explicit sampling methods that is slightly different.

For example, <studyExtent> offers a choice of the two familiar elements of <coverage> and <description>, which are used here to specify both the sampling area and the sampling frequency. As there is a choice of descriptors, either the <coverage> module, with full geographic, temporal, and taxonomic coverage can be used, or the extent of the study can be described simply in <para></para> format under <description>. Note that <coverage> can also be referenced from an earlier <coverage> module if so desired.

The <samplingDescription> element is also presented in simple paragraph format, and it is designed for describing exact sampling methods used in the course of a research study.

The <spatialSamplingUnits> refer to specific geographical areas that are being sampled. Once again, the <coverage> module comes in handy for describing these areas; though in this instance the <coverage> module only contains <geographicCoverage> elements. A <referencedEntityId> option is also available, which operates like a regular reference, except that it refers to the "id" of a specific entity in which the metadata for the dataset that contains the actual geographic reference values is located.

The final <methods> element is <qualityControl>, which has the exact same schema as <methodStep>, with the exception of the <dataSource> element. The idea behind <qualityControl> is to provide a space for the assessment of the control procedures taken to ensure the quality of the methods being documented in a particular <methodStep>.

## Constraining data tables

The <dataTable> module contains several other optional elements that are helpful additions to an EML metadata document. The <caseSensitive> and <numberOfRecords> elements are simple fields that require only a short string to complete. 'Yes' or 'no' are the only values required by the <caseSensitive> field, while a numerical value indicating the number of records contained in the data table is the only requirement for <numberOfRecords>.

The <constraint> module, however, is more complex and requires a basic understanding of the concept of keys. Keys are special fields that are used to establish relationships within and between data tables. A <primaryKey> is used to identify a unique record within a single table, while a <foreignKey> establishes the relationship between two or more tables. These keys help to create levels of integrity between the data tables if they are to be maintained in a relational database system.

For example, a data table consists of a list of several species of plants, with a code name assigned to each species:

Table 3: Plant Species Survey

| SPECIES ID | GENUS | SPECIES | COMMON NAME |
|---|---|---|---|
| JUMO | Juniperus | monosperma | One-seed Juniper |
| LATR2 | Larrea | tridentata | Creosote bush |
| ORHY | Oryzopsis | hymenoides | Indian Rice Grass |
| PIED | Pinus | edulis | Pinyon Pine |
| SPCO4 | Sporobolus | contractus | Spike dropseed |

In this example, the Species ID column contains the primary key that uniquely identifies each record in the Plant Species Survey. Compare this with the following Table 4, which uses the Species ID column from Table 3 to establish a relationship between the data in the two tables.

Table 4: Site Survey

| SITE ID | SPECIES ID | SITE NAME |
|---|---|---|
| BB | LATR2 | Black Butte |
| FP | LATR2, PIED | Five Points |
| GD | ORHY, SPCO4 | Goat Draw |
| RS | SPCO4, JUMO, ORHY | Rio Salado |

Here, the Site ID column contains the primary key for the Site Survey, while the Species ID column from the Plant Species Survey is now the foreign key.

In EML, the <primaryKey> contains the required fields <constraintName> and <attributeReference>, with <constraintDescription> optional. The <constraintName> element consists of a simple, understandable name for the constraint, such as "PrimaryKey_PlantSpeciesSurvey." If the <contraintDescription> was used here, it would describe the composition of that primary key, for example, "The primary key of the Plant Species Survey table contains the attribute called Species ID."

The next required element, <attributeReference>, specifies the attributes to which the defined constraint applies. In the case of a primary key, the attributes specified should be unique within the entity (table), such as "Species ID, Genus, and Species." If the key is foreign, however, the attributes specified will exist in a different entity (or table).

The <foreignKey> module has much the same structure as that of the <primaryKey>, however, it also contains an <entityReference> element, which allows one to specify another entity that contains the primary key. Using the examples in Tables 3 and 4, the entity referenced by the Site Survey is the "Plant Species Survey."

The <entityReference> can also be an ID such as "pss.03.94" if an <alternateIdentifier> for the entity was defined previously. The <attributeReference> within the <key> would then consist of "Species ID."

There are several additional modules within the larger <constraint> module that function in numerous ways to help define the relationships among various entities and attributes within an overall <dataTable>.

The <uniqueKey> module functions much like the <primaryKey> module, except that this module does not contain the reference to a foreign key that is implied by the use of the <primaryKey> module. The <joinCondition> module then can be used to define a relationship between any two tables that are not linked by a primary/foreign key relationship.

The <checkConstraint> element allows a data manager to specify an entity to which a conditional clause applies. After defining the <constraintName>, the <constraintDescription> field is used to describe a check condition, such as "must be greater than 10 but less than 250." The <checkCondition> element is then used to define a statement for the condition that applies to the constraint, such as "year > 1950 and year < 2000." These statements usually are formulated in SQL or some other implementation language and they help define constraints both within and among entities.

The final type of constraint element is the <notNullConstraint>, which simply defines attributes within an entity that should not have null values. The <key> element here then specifies the <attributeReference> that should not contain this type of value.

## Some final notes on getting help with EML 2.0.0

The EML schemas described in these pages are designed to help facilitate the easiest use of this metadata language for the greatest number of users. This manual should allow the average ecologist "in the field" to understand the process of creating EML-compliant metadata documents to document his or her research.

EML, however, is still a language "in process." The creators of EML, Ecoinformatics.org, a subsidiary of the Knowledge Network for Biocomplexity, welcome comments and feedback via email at eml-dev@ecoinformatics.org. Any bugs found in the EML program should be submitted to the EML bug tracking system at http://bugzilla.ecoinformatics.org/. This is the preferred way to submit problems or feature requests.

The EML technical specifications can be found online at http://knb.ecoinformatics.org/software/eml/eml-2.0.0/index.html or downloaded from http://knb.ecoinformatics.org/software/download.html#eml.

# Index