THE EAD COOKBOOK

Archives and libraries implementing EAD face many choices: what software to use, what elements and attributes to employ in encoding, and how best to create print and web-accessible versions of their finding aids. During the meeting of the EAD Roundtable at the 1999 Annual Meeting of the Society of American Archivists, attendees agreed that a simple model encoding protocol with an accompanying suite of software tools for "authoring" electronic finding aids and stylesheets for "publishing" them would be very useful. This Cookbook has been developed to accomplish that end. It functions as an extension of the *EAD Tag Library* and the *EAD Application Guidelines*. To use it, one must have a basic understanding of the EAD element set.

The *EAD Cookbook* includes a markup protocol that is based on recommendations found in the *EAD Application Guidelines* and those promulgated by several EAD projects. It offers an explanation of the decisions behind the encoding recommendations, is accompanied by a suite of software tools that incorporate the model markup in various SGML/XML authoring applications, and includes step-by-step instructions for applying them. A sample finding aid for a fictional collection, encoded according to these guidelines, is included as an appendix.

One has many options as to how a finding aid that uses this markup syntax will appear on the web or in print. The tools that accompany the Cookbook produce a basic, standard presentation suitable for either environment. Its style is drawn from a number of sources. These include the format developed by the Minnesota Historical Society and that employed by the Library of Congress for its online finding aids. The Society's work on redefining the structure and format of its inventories was chronicled by Dennis Meissner in the Fall 1997 issue of the *American Archivist* (also published as *Encoded Archival Description: Context, Theory and Case Studies*). The Library of Congress was an early and active participant in EAD development and deployment. The format of its archival registers has been widely copied by other repositories in the past. However, the encoding protocol in the Cookbook does not reflect current practices of either institution exactly but is a composite from various sources, including suggestions received from reviewers. A set of stylesheets, written in the Extensible Stylesheet Language (XSL), is included to help you generate editions of your finding aids that are suitable for viewing in a web browser or as print copies. These too do not resemble any institutional prototype exactly but are a composite from several sources. Step-by-step instructions guide you through each phase of EAD implementation. Bon appetit!

Michael J. Fox
July 2000

Section 1: The Recipe

1.1 Learn about EAD. Three volumes published by SAA make a good start: *Encoded Archival Description: Context, Theory and Case Studies*, the *EAD Application Guidelines*, and the *EAD Tag Library*. The EAD Help pages maintained by the SAA EAD Roundtable contain much practical and useful information, including descriptions of many implementation projects and lists of available training opportunities. They are available at:

http://jefferson.village.virginia.edu/ead/

1.2 Determine how you will deliver your finding aids over the web. How will researchers find these documents? Many options are available. You can do any or all of the following:

a. Create links to the finding aids from other web pages.

b. Create links to finding aids from entries in your online catalog.

c. Index your finding aids on your local web site. Many now have software that performs at least keyword searching of documents found on the site. EAD-encoded documents transformed into HTML may be indexed and accessed along with other files.

d. Contribute to a consortium that hosts a central file of finding aids. Such data servers typically feature sophisticated software that provides structured and keyword searching of the full text of finding aids. A growing number of state projects are creating such services. The Research Libraries Group's Archival Resources service hosts finding aids from repositories in several countries and is not limited to RLG member institutions.

e. Buy your own data server and software. Choices range from freeware to commercial full-text search software.

f. Start you own cooperative project.

The tools for creating EAD documents that accompany the Cookbook are applicable to all of these options.

The Cookbook provides explicit directions in Sections 5 through 8 for implementing publishing options a and b above.

1.3 Determine the file format in which you will deliver the documents to the end user. The Cookbook supports the option that is most widely used today- conversion of EAD documents to HTML format for viewing in standard browsers. It does not cover the use of browser plug-ins and other supporting files required when delivering EAD documents in "classic SGML" format. The cost and inconvenience to users of acquiring and installing such software makes this approach less desirable as XML options emerge.

For further advice on these issues, consult Chapters 4 and 5 of the *EAD Application Guidelines*, read the case studies in *Encoded Archival Description: Context, Theory and Case Studies*, analyze the implementation strategies of various institutions documented at the EAD Help Pages web site, and follow the discussion or ask questions on the EAD list.

1.4 Review the general principles and specific details of the Encoding Protocol found in sections 2 and 3. Some choices have been made for you already; others you will have to determine. These sections will help you identify your options and make those decisions. The sources cited above and the *EAD Tag Library* will provide additional information.

1.5 Acquire one of the software applications for authoring finding aids supported by the Cookbook. These include XMetaL, Author/Editor, and WordPerfect 9.

1.6 Repeat step 1.4.

1.7 Install and configure the software. (Section 4)

1.8 Create EAD-encoded finding aids. (Section 4)

1.9 Convert any files in SGML into XML. (Section 5)

1.10 Select an appropriate XSL stylesheet. (Section 6)

1.11 Transform your EAD files into HTML. (Section 7)

1.12 Mount files for web access. (Section 8)

Section 2: The EAD Cookbook Design Principles

EAD offers great flexibility in how one might encode a finding aid. The options chosen may have a significant impact on the subsequent usability of the file. The particular markup choices suggested in the *EAD Cookbook* are designed to facilitate the transport of data between systems, the sharing of data through union catalogs, the reuse of data for different purposes, hyper-navigation within inventories, and the presentation of data in different environments, currently through the web and as print output. The following principles serve as the basis for the model encoding protocol.

1. EAD encoding is not a substitute for sound archival description. Before marking up a single document, consider what information you wish to record. Descriptive conventions like *Archives, Personal Papers, and Manuscripts* (APPM), the *Rules for Archival Description* (RAD), and the *General International Standard for Archival Description* (ISAD(G)) all have very useful things to say about what information ought to be included in a good archival description and how to structure that data. Finding aids do not differ fundamentally from other tools we create to assist our users in identifying and locating relevant materials in our collections. Consider using indexing terms drawn from standardized vocabularies in controlled elements to improve retrieval across metadata systems.

2. The description embodied in an EAD-encoded finding aid conveys the multilevel, hierarchical nature of the materials themselves. This approach reflects the fundamental and intrinsic characteristics of archival materials and their interrelationships, and the function of the finding aid as a surrogate for the originals and a tool for resource discovery and interpretation.

3. Description proceeds from the general to the specific, in reflection of the hierarchy just described, as a practical and flexible solution to the problem of creating intellectual and physical control over large bodies of materials, and as an aid to resource discovery by researchers.

4. Markup may use either "classic SGML" or XML syntax. If an SGML file is to be transformed into HTML using the accompanying stylesheets, it first must be converted into XML syntax.

5. The model assumes that the materials form a collection of personal papers or organizational records that is subdivided into several series. The encoding further assumes that the collection is fully processed and described. More complex or simpler collections, e.g., those with sub-groups or sub-collections or ones consisting only of a single series probably will require modifications to the stylesheets.

6. No method for managing large numbers of images or other associated non-textual files is prescribed. Repositories using finding aids to link to many external files will need to review the options outlined in Section 4.4.3 of the *EAD Application Guidelines*. The simple inclusion of an institutional logo can be handled through a single Extended Pointer element or by a stylesheet, as is the case with those that accompany the Cookbook (see Section 6.2).

7. There is a growing consensus within the archival community that there is a set of core data elements that ought to be included at a minimum in every finding aid to help users determine the applicability of a collection to their research needs and to identify specific materials for retrieval and inspection. Appendix A of the *EAD Application Guidelines* specifies four elements that should be included in every finding aid: the top level Descriptive Identification, Biography or History, Scope and Contents, and Controlled Access. Administrative Information elements such as user restrictions should be included as appropriate. The specific sub-elements within these groups are described in Section 3. Further, there are certain elements necessary for the identification of the electronic file that carries the encoded finding aid. Some of these are required by the EAD DTD, particularly in the EAD Header element.

8. The protocol detailed in Section 3 prescribes the explicit inclusion of certain data elements to facilitate transport, display in external systems, internal hyper-navigation, data reuse such as in converting EAD content into MARC records, and the future migration of data to new systems. One cannot assume that local protocols for display, such as may be specified by a local stylesheet, will always travel with the document or be applied to its presentation in other systems and contexts. The protocol specifies the inclusion of labels to provide coherent, user-friendly displays using the Head element for blocks of text and groups of related elements, the specific inclusion of default values for LABEL attributes, and the addition of encoding analogs. It includes samples of explanatory text that might be added to a finding aid to assist remote users who, for example, might not readily understand the purpose of elements such as Controlled Access terms. To minimize subsequent display problems, avoid the use of phrases written entirely in upper case.

9. When encoding inventories, other conventions also apply. Observe the recommendations regarding whitespace and internal punctuation found in sections 4.3.5.1 and 4.3.5.2 of the *EAD Application Guidelines* and Section 3.3.10 of the Cookbook. Internal links are specifically encoded with a Reference element. Specific ID attributes are given to facilitate internal hyper-links, especially between the Table of Contents and the body of the finding aid. Most of these features are supplied as "default" values through the software templates used to create a new inventory and therefore add no additional overhead to encoding operations.

10. Additional markup such as the encoding of personal names and subjects beyond the prescribed structural elements seems cost-justifiable only if it supports enhanced comprehension or retrieval through indexing or display. The protocol markup scheme does not provide any special support, such as keyboard macros for data entry or distinctive styling for display through the stylesheets, for nominal content markup such as <persname> within elements for narrative text such as paragraphs. However, this certainly does not preclude in any way their inclusion. This topic is discussed in several of the EAD case studies that appeared in *Encoded Archival Description: Context, Theory and Case Studies.*

11. Markup follows the "combined" model for the Description of Subordinate Components. The resulting information may be presented either in the sequence in which it is encoded or divided for presentation into separate "analytic overview" and "in-depth" views of the contents of the inventory by a stylesheet. The use of stylesheets to produce either of these outputs from a single set of data eliminates the need for duplicate entry of information common to the two views.

12. No conceptual significance is implied by the sequence in which elements appear in the encoding specifications or the templates except insofar as the EAD DTD enforces a given order. With the use of the stylesheets written in the XSL Transformation (XSLT) language that accompany the Cookbook, the arrangement of information upon output to the web is largely independent of the order of data in the source document. While the protocol makes few assumptions about input sequence, the stylesheets do enforce a predefined order at publication. However, publication techniques other than XSL stylesheets often are not so flexible and one may have little or no ability to reorder elements. In those scenarios, the order of data input may be directly and absolutely related to the order in which data is published.

Section 3: EAD Cookbook Encoding Protocol

This section describes the Cookbook's key encoding recommendations and the rationale behind them. These choices are assumed by and incorporated into the accompanying templates and other software tools described in Section 4 and the publication tools in Sections 6 and 7.

The protocol includes every major high-level element available in EAD. If your repository does not wish to include a particular element, it may be simply deleted globally from the accompanying templates or removed from individual documents on a case-by-case basis.

Arbitrary values are assigned to ID attributes to simplify the creation of a table of contents and other internal hyper-links. Any changes to these values will require a partial rewriting of the stylesheets. If you wish to include additional elements, such as multiple instances of <odd>, and link them from a table of contents, additional unique ID attributes must be assigned. Any values not already used in the templates may be applied. If an element such as <bioghist> is used recursively, the value of the ID attribute specified in this section should be applied only to the parent element. SGML requires each ID attribute have a unique value.

Text is suggested for certain elements, particularly Head and the EAD Header elements. You may certainly alter the language according to local practice. For a detailed description of the form and usage of each element, consult the *EAD Tag Library*.

Encoding analogs are given for many elements in order to correlate EAD elements with equivalent data fields in the MARC 21 cataloging format. They are included only to facilitate the conversion of EAD data into MARC records. If you do not intend to use your finding aids in this way, these attributes may be safely ignored or deleted. Other encoding analogs, such as to ISAD(G) elements certainly may be substituted.

The Cookbook does not utilize the <frontmatter> element either by specifying encoding practice for it or supporting its display in the accompanying stylesheets. Many institutions find that all the information necessary for a presentation that resembles a formal title page for the finding aid may be extracted from the <eadheader>.

3.1 EAD <ead>

Record the RELATEDENCODING attribute as "MARC21".

3.2 EAD Header <eadheader>

Record the AUDIENCE attribute as "internal".
Record the LANGENCODING attribute as "ISO 639-2".
Record the FINDAIDSTATUS attribute as "EDITED-FULL-DRAFT".
Record the ID attribute as "a0".

3.2.1 EAD Identifier <eadid>

Record a value that is suitable as a computer file name in order to facilitate file management. Stylesheet 3 which accompanies the Cookbook (see Section 6.1.3) uses the value of <eadid> as the basis for several file names in an HTML frameset.

Record a coded value for the repository in the SYSTEMID attribute to ensure that the <eadid> assigned is unique among the universe of inventories. Use a MARC repository code supplied by the Library of Congress or the National Library of Canada (*MARC Code List for Organizations*), or other code assigned by OCLC, by the Research Libraries Group, or by another national authority.

Record in the SOURCE attribute a code that identifies the body that issued the code in the SYSTEMID attribute. For example, the codes for the institutions listed above are DLC, NLC, OCoLC, and CStRLIN respectively.

Record the ENCODINGANALOG attribute as "852".

3.2.2 File Description <filedesc>

3.2.2.1 Title Statement <titlestmt>

Record the name of the creator of the collection [as opposed to the finding aid] in the Title Proper <titleproper> element.
Record in the Subtitle <subtitle> element a statement in the following syntax- An Inventory of *His/Her/Its Characterization of the materials taken from the Unit title* at the *Name of the Repository.*

For example-
<titleproper>William Fonds Provenance: </titleproper><subtitle>An Inventory of His Papers at the Cupcake Corners Historical Society.</subtitle>

This formula creates a statement that contains four critical informational elements- the name of the creator of the materials, the type of document (an inventory), the nature of the records, and the repository's name. This data is used as the <title> element in the HTML output for display in the browser and serves as metadata for web search engines.

Record the name of the creator of the inventory in the Author <author> element. This provides useful information for tracking the history of the inventory.

3.2.2.2 Publication Statement <publicationstmt>

Record the name of the publishing repository in the Publisher <publisher> element.
Record the date of the inventory in the Date <date> element.
Record the repository's location in the Address <address> element. For long-term maintenance, it may be preferable to record data that is subject to change such as street addresses, phone numbers, etc. in an external file accessed through an entity reference or supply it as default text with a stylesheet.

3.2.3 Profile Description <profiledesc>

3.2.3.1 Creation <creation>

Record the name of the individual responsible for the encoding. Include the date in the Date <date> element.

3.2.3.2 Language Usage <langusage>

Record a statement about the language of the finding aid in <langusage>.
Note that names of the specific languages involved are recorded in the Language <language> sub-element.

3.3 Archival Description <archdesc>

Record the organizational level of the entire body of materials described in the LEVEL attribute as appropriate.
Record the language(s) of the collection in the LANGMATERIAL attribute as appropriate. The default value given in the accompanying templates is “eng”.
Record the TYPE attribute as “inventory”.

Record content for at least the following four elements: Descriptive Identification, Biography or History, Scope and Content, and Controlled Access, as recommended in Appendix A of the *EAD Application Guidelines.* Enter details of Administrative Information such as restrictions as applicable. The accompanying templates support the inclusion of all other high-level elements. The EAD DTD requires that the Descriptive Identification <did> element come first. Further order is a matter of local preference though the accompanying stylesheets will produce an output file that presents elements in a predefined order when they transform an EAD file from XML into HTML syntax. See Section 6 for further explanation of this process.

3.3.1 Descriptive Identification <did>

Record the ID attribute as “a1”.
Record a Head element that characterizes <did> and its sub-elements. The accompanying templates use the expression “Overview of the Records” or “Overview of the Collection”.

Record the name of the repository in <repository> even if its display is to be suppressed.
Record the ENCODINGANALOG attribute as “852$a”.
Record the LABEL attribute as “Repository:”.
Record the repository’s city and state in <addressline> to conform to ISAD(G) standards.

Record the name of the creator of the materials in <origination>. Include a Corporate, Personal, or Family Name element as appropriate.
Record the ENCODINGANALOG attribute for the sub-element as either “100” or “110”, depending on which is the appropriate MARC element.
Record the LABEL attribute as “Creator:”.

Record the title of the materials in <unittitle>.
Record the ENCODINGANALOG attribute as “245$a”.
Record the LABEL attribute as “Title:”.
Do not include separating punctuation between the Unit Title and the Unit Date as the stylesheets will display them on separate lines.

Record the date of the materials in <unitdate>. The accompanying stylesheets accommodate the encoding of unit dates either as separate elements or as a sub-element of the Unit Title element.
This element may be repeated if both bulk and inclusive dates are recorded.
Record the type of date in the TYPE attribute.
Record the ENCODINGANALOG attribute as either "245$f" or "260$c" as appropriate to the MARC conventions your institution employs.
Record the LABEL attribute as "Dates:".

Record a physical description in <physdesc>.
Record the ENCODINGANALOG attribute as "300$a".
Record the LABEL attribute as "Quantity:".

Record an abstract of the contents of the materials in <abstract>.
Record the ENCODINGANALOG attribute as "520$a.
Record the LABEL attribute as "Abstract:".

Record in <unitid> a uniquely identifying name or number for the materials described, as required by ISAD(G).
Record the ENCODINGANALOG attribute as "099".
Record the LABEL attribute as "Identification:".
Record the COUNTRYCODE attribute as found in ISO 3166. The default value in the accompanying templates is "US".
Record in the REPOSITORYCODE attribute the value found in the SYSTEMID attribute in <eadid>.

3.3.2 Biography or History <bioghist>

Record the ENCODINGANALOG as "545".
Record the ID attribute as "a2".
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text <p>, a chronology list <chronlist>, or both.
Nested <bioghist> elements within the parent element (recursion) may be helpful to divide the text of the biography or history narrative into sections such as when the collection includes materials from several individuals in a family. This technique is supported by the stylesheets. Recursive <bioghist> elements must not have the same ID attribute value as the parent <bioghist> as SGML requires each ID attribute to have a unique value.

3.3.3 Scope and Contents <scopecontent>

Record the ENCODINGANALOG as “520”.
Record the ID attribute as “a3”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text <p>.
Nested <scopecontent> elements within the parent unit (recursion) may be helpful to divide the text of the scope and content narrative into separate sections such as when the collection includes materials from several individuals in a family. Recursive <scopecontent> elements must not have the same ID attribute value as the parent <scopecontent> as SGML requires each ID attribute to have a unique value.

If you wish to create a hyperlink from a section of the text in the scope statement to the part of the Description of Subordinate Components <dsc> that describes the same material in detail, you must explicitly encoded those links within the narrative using the Reference <ref> element. (See Section 3.5)

Statements about the arrangement and organization of the materials may be included here rather than appearing as separate elements.

3.3.4 Organization <organization>

Record the ENCODINGANALOG as “351$a”.
Record the ID attribute as “a4”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

Alternatively, one may describe the organization of the materials by giving a list of the titles of the component series. In this case, preceed the list with a short explanatory statement characterizing its contents. The accompanying stylesheets are designed to automatically generate hyperlinks between an individual series title in such a list and the location in the description of subordinate components where that series is described in detail. The links need not be specifically encoded here. However, the series titles must be enumerated in the list in the same sequence as they appear in the description of components. This linking works whether the organization element appears separately or within a scope and contents element.

3.3.5 Arrangement <arrangement>

Record the ENCODINGANALOG as “351$b”.
Record the ID attribute as “a5”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text, lists, or tables.

3.3.6 Adjunct Descriptive Data <add>

The sub-elements of <add> are described below. To increase flexibility in authoring, it suggested that each of these elements be nested inside a separate <add> element rather than all nested within a single <add> parent element.

3.3.6.1 Related Material <relatedmaterial>

Record the ENCODINGANALOG as “544 1”.
Record the ID attribute as “a6”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text or as a <list>.

3.3.6.2 Separated Material <separatedmaterial>

Record the ENCODINGANALOG as “544 0”.
Record the ID attribute as “a7”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text or as a <list>

3.3.6.3 Other Finding Aid <otherfindaid>

Record the ENCODINGANALOG as “555”.
Record the ID attribute as “a8”.
Record a <head> element that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.6.4 Index <index>

No ENCODINGANALOG applies.
Record the ID attribute as “a9”.
Record a <head> that describes the content of this element.
Encode the body of the element with paragraphs of text and the Index Entry element.

3.3.6.5 Bibliography <bibliography>

No ENCODINGANALOG applies.
Record the ID attribute as “a10”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text or Bibliographic Reference elements.

3.3.7 Other Descriptive Data <odd>

Record the ENCODINGANALOG as “500”.
Record the ID attribute as “a11”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.
This element should be used only if another, more specific, element does not apply.

3.3.8 Controlled Access Headings <controlaccess>

Record the ID attribute as “a12”.
Record a <head> that describes the content of this element. Include an introductory paragraph that explains the origins and uses of the headings found in this section.

Record <persname>, <corpname>, <famname>, <geogname>, <subject>, <genreform>, <title>, <occupation>, and <function> terms in the form determined by an appropriate authority such as the Library of Congress Name Authority file or the Art and Architecture Thesaurus. A list of possible sources is included in the Attributes section of the *EAD Tag Library.*

Optionally, record in the SOURCE attribute the name of the authority file from which the heading was derived.

Sub-elements may be organized recursively into groups by type of access term using nested <controlaccess> elements with explanatory <head> elements. Recursive <controlaccess> elements must not have the same ID attribute value as the parent <controlaccess> as SGML requires each ID attribute to have a unique value.

Record ENCODINGANALOG attribute values for the appropriate MARC 21 fields to facilitate reuse of the information in catalog records and to assist in the creation of Dublin Core metadata in the HTML output. For example, appearance of the value "600" in this attribute will cause a personal name to appear in a Subject element in a <meta> tag in the HTML output generated by the stylesheets whereas if the value is given as "700", the name will appear in a <meta> tag as a Contributor element. The values shown below are included by default in the templates that accompany the Cookbook. In situations where more than one ENCODINGANALOG value is possible, the default is listed below. If you do not plan to transform your EAD data into MARC records or if the default values are sufficiently precise for your uses, simply retain the default values and ignore the following chart.

The appropriate MARC values are

<persname>	(600 as a subject or 700 as a creator; 700 is the template default)
<corpname>	(610 as a subject or 710 as a creator; 710 is the template default)
<famname>	(600 as a subject)
<geogname>	(651)
<subject>	(650)
<genreform>	(655)
<title>	(630 as a subject or 730 as part of a collection; 630 is the template default)
<occupation>	(657)
<function>	(656)

Default values are included in the accompanying templates and macros.

3.3.9 Administrative Information <admininfo>

The sub-elements of <admininfo> are described below. To increase flexibility in authoring, it suggested that each of these elements be nested inside a separate <admininfo> element rather than all nested within a single <admininfo> parent element.

3.3.9.1 Restrictions on Access <accessrestrict>

Record the ENCODINGANALOG as “506”.
Record the ID attribute as “a14”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.9.2 Restrictions on Use <usesrestrict>

Record the ENCODINGANALOG as “540”.
Record the ID attribute as “a15”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.9.3 Custodial History <custodhist>

Record the ENCODINGANALOG as “561”.
Record the ID attribute as “a16”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.9.4 Alternative Form Available <altformavail>

Record the ENCODINGANALOG as “530”.
Record the ID attribute as “a17”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.9.5 Preferred Citation <prefercite>

Record the ENCODINGANALOG as “524”.
Record the ID attribute as “a18”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.9.6 Acquisition Information <acqinfo>

Record the ENCODINGANALOG as “541”.
Record the ID attribute as “a19”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.9.7 Processing Information <processinfo>

Record the ENCODINGANALOG as “583”.
Record the ID attribute as “a20”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.9.8 Appraisal <appraisal>

Record the ENCODINGANALOG as “583”.
Record the ID attribute as “a21”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.9.9 Accruals <accruals>

Record the ENCODINGANALOG as “584”.
Record the ID attribute as “a22”.
Record a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.10 Description of Subordinate Components <dsc>

The "combined" model is used for encoding the <dsc>. See Chapter 3 of the *EAD Application Guidelines* for a discussion of the possible structural models for the Description of Subordinate Components. Accompanying stylesheets 1 through 3 present the text of the <dsc> in a single sequence. Stylesheet 4 presents the text as a separate "Summary Description of the Records" and a "Detailed Description of the Records" section for print output. The stylesheets all use HTML tables to produce an indented, tabular display. The table structure is derived from the content of the encoding; no explicit encoding of tables in EAD is required to generate this tabular output. See Section 6.1 for further details.

Record the ID attribute as "a23".
Record the TYPE attribute as "combined".

Record a <head> that describes the content of this element. The accompanying templates use the default expression "Detailed Description of the Records" or "Detailed Description of the Collection."
Include an introductory Paragraph that explains this section of the inventory.

Component <c>

This model assumes that the Component (First Level) element represents a series. Record the following information for each series.

Record the ID attribute as "series*nn*", where *nn* is the sequential number of the series, e.g. series1, series2, series3, etc.
Record the LEVEL attribute as "series".

For all component levels, record the appropriate <did> elements. You may include <unitdate> either within <unittitle> or separately. Contrary to the recommendation in Chapter 4 of the *EAD Application Guidelines*, do include separating punctuation between the individual sub-elements of component <did> elements, such as between title and date. During testing of the Cookbook, it was determined that it is impossible for a stylesheet to predict and properly insert all variations in punctuation. The stylesheets will insert a space between elements as appropriate.

Record box and folder or other container information according to institutional practice. Possibilities include showing box and folder numbers, only box numbers, only folder numbers, a combined box/folder number, or no container information at all. The accompanying stylesheets accommodate all five possibilities. When container numbers are recorded, however, they must be recorded explicitly for each component even if an entry duplicates information in a previous component. The accompanying stylesheets suppress the public display of repetitive container numbers. See Section 6 for further, specific details. The Cookbook promotes the explicit inclusion of container data for all components as a highly desirable practice that will facilitate future data manipulation.

You may wish to include container information for most components but not for certain others, for example one that simply represents a filing unit such as "Subject files organized by date." In this case, include the container element but without any textual or numeric content. For programming reasons, the stylesheets determine the display of container information based on the appearance of the container element as well as the presence or absence of text within it. Simply put, if you include container elements for some components but not for others, the Cookbook stylesheets will mess up your displays. Fortunately, it is not difficult to insert the <container> tags since the keyboard macros for the accompanying templates automatically supply them.

The adjustments that need to be made to the templates and macros to accommodate the type of containers you record are described separately for XMetaL, Author/Editor, and WordPerfect in the appropriate parts of Section 4.

Record the appropriate TYPE attribute for all Container elements.

Record other elements such as appropriate. The accompanying stylesheets specifically support the presentation of <scopecontent>, <bioghist>, <note>, <odd>, <userestrict>, and <accessrestrict> within components.

3.4 Tables

The EAD Cookbook stylesheets support the inclusion of simple tables for the presentation of data in two or more columns within paragraphs in narrative elements such as Scope and Content, Biography or History, and Note. Tables may include any number of columns. For further information on the encoding of tables, see the Overview of EAD section and the descriptions and examples of individual table elements and attributes in the *EAD Tag Library*.

3.4.1 Table <table>

Record the number of columns in the table in the COLNUM attribute.

3.4.1.2 Table Group <tgroup>

Include a Table Column Specification <colspec> element for each column. Specify the width of the column in the COLWIDTH attribute. Width may be expressed as an absolute number of pixels or points or as a relative percentage.

3.4.1.3 Table Head <thead>

Record a heading for each column as an <entry> element within a <row>.

3.4.1.4 Table Body <tbody>

Record the body of the table as a series of "data cells" that are defined horizontally by <row> elements with each individual piece of data in an <entry> element.

3.5 References

The Reference <ref> element provides a mechanism for creating internal hyperlinks between sections of a single finding aid. All such links contain two parts- a source and a target or destination. The target is created by adding a unique value as the ID attribute of the element that is the destination of the link.

The source is created by embedding the text that forms the link within a <ref> element. Record the value of the target element's ID attribute as the value of the TARGET attribute of this <ref>.

The cookbook stylesheets support the use of <ref> in a limited numbed of situations. Links may be made to <c01> elements (series titles) in the description of components section and to paragraph elements anywhere in a finding aid.

Section 4: Authoring Tools

The Cookbook provides a variety of tools to facilitate the encoding of new inventories. These include templates, macros and other customization files. Parallel files are available for three applications: XMetaL, Author/Editor, and WordPerfect 9. Aids for other software packages may be added in the future. Directions follow for installing, customizing, and implementing each application and its associated files. The files that are listed in the following sections are available for downloading on the EAD Help Pages.

4.1 XMetaL

XMetaL, a software application for creating EAD-encoded finding aids, is available from SoftQuad Ltd. 161 Eglington Avenue East, Suite 400, Toronto, ON, Canada M4P 1J5. It runs on various Windows platforms. The following tools were created to be compatible with version 1.2 of the software.

While XMetaL may create either SGML or XML output, it is configured in the Cookbook to produce XML files.

4.1.1 Installation

The default installation of XMetaL places the application in an XMetaL sub-directory of the Program Files folder on the C: drive. Review the user's manual to familiarize yourself with the key features of the software, especially the Tags On and Plain Text displays and the use of the Insert Element and Attribute Inspector features for markup.

The installer program also will suggest that you add the accompanying version of Microsoft Internet Explorer browser (IE version 4.0 or 5.0 being required by the software). You may chose to install the browser or update it at this time if you have an older version.

When you are asked during the installation process or upon first opening the application if you wish to preserve whitespace, select that option that indicates you do not wish to do so.

If you will be using XMetaL to generate your HTML output (see Section 7), you will need to install Microsoft's version of a Java Virtual Machine which they refer to as the Microsoft Virtual Machine. To do so, select a Customized installation for the browser and pick the optional Microsoft Virtual Machine choice as one of the components to install.

Install the helper files described in in the next two sections.

4.1.2 Rules

XMetaL uses a binary version of the EAD DTD called a "rules" file for enforcing compliance with the DTD during document creation and final validation. This file is called

ead.rlx

Install this file in the Rules sub-directory of the XMetaL folder.

4.1.3 Templates

Templates provide a standardized file that can be reused over and over to provide the default markup and "boilerplate" text that you wish to include in every document. Two templates are provided with the Cookbook. They are named

eadperson.xml
eadcorporate.xml.

There are only minor variations between the two. The first is geared towards inventories of collections of personal papers, the other to organizational or governmental records. In the former, the Origination element contains a Personal Name sub-element, in the latter a Corporate Name. The only other variations are in the suggested text of Head elements. For example, in one case the text reads "Scope and Content of the Collection" and in the other "Scope and Content of the Records".

Install the two files listed above in the General sub-directory of the Templates directory of the folder in which you have installed XMetaL.

4.1.3.1 To Create a New Document

Open XMetaL, chose **File** on the menu bar, and then select **New** from the pull-down menu. You will be presented with a list of templates including these files. When you select the appropriate template, the default tags, attributes and text will appear for you to fill in and add to. XMetaL includes a special feature wherein grayed-out text appears within the element as a prompt to the data entry operator, the contents of which disappears when something is entered into that element.

The order of elements in the templates follows the sequence found in Section 3. The EAD DTD requires that the <eadheader> sub-elements follow this order and that the <did> element comes first within <archdesc>.

4.1.3.2 Customize Your Templates

You may wish to customize the template files for various reasons:

- Certain attributes are repository specific. You will need to supply locally valid values for the SOURCE and SYSTEMID attributes in the EAD Identifier and the REPOSITORYCODE attribute in the ID of the Unit for the collection as a whole, i.e. in the Descriptive Identification for the Archival Description.

- Default text for EAD Header elements, explanatory paragraphs, and Head elements may be modified. For example, you may wish the Head element for the Biography or History <bioghist> to read "Biographical Sketch" rather than "Biography of….".

- The template may include elements such as appraisal or accruals which you never use. Simply delete the markup from the template.

- Some elements in the template, such as repository name and address, that initially appear as prompted data ought to be converted to default data so that they do not need to be rekeyed for each inventory. Simply type over the existing prompt to convert it to "boilerplate" data.

- The default text of data entry prompts may be customized. XMetaL uses a proprietary application of the XML processing-instruction element to supply these prompts. They are visible and may be edited in the Plain Text view. The processing instruction begins with the code- <?xm-replace_text- which is followed by the text of the prompt in curly brackets. Simply type over the existing text in the brackets to create a new prompt.

- Certain elements automatically appear when a component such as a <c02> is inserted. You may wish to change the defaults, for example, removing the <container type="folder"> element if you never include folder numbers in your inventories. These are defined in the file ead.ctm whose customization is described in Section 4.1.5.

- The order of elements may not be optimal for local data entry. One may rearrange the elements within the limits of the DTD. Remember, the order of the elements at data entry is independent of the sequence in which they display later.

To customize the template, open a new document using the template, insert, delete, or type over the relevant elements and attributes as described below, select **Save As** on the **File** menu, and save the file in the Templates sub-directory under its original name.

4.1.4 Macros

Macros are a feature of many software applications, including word processors, that permit one to execute repeatedly an oft-used set of keystrokes by typing a short two or three letter combination of keys. The following keystrokes can be used to automatically insert elements and text. The file

ead.mcr

executes the following functions. Install this file in the Macros sub-directory of the folder in which you have installed XMetaL.

CTRL + ALT + c	Opens a Chronology List Item with a Date and an Event
CTRL + ALT + s	Opens a Scope and Content and Paragraph
CTRL + ALT + u	Opens a Unit Identification
CTRL + ALT + 1	Opens a Component (Level One)
CTRL + ALT + 2	Opens a Component (Level Two)
CTRL + ALT + 3	Opens a Component (Level Three)
CTRL + ALT + 4	Opens a Component (Level Four)
CTRL + ALT + 5	Opens a Component (Level Five)
CTRL + ALT + 6	Opens a Component (Level Six)
CTRL + ALT + 7	Opens a Component (Level Seven)

4.1.5 XMetaL Customization File

This file customizes certain aspects of XMetaL functionality. It is named

ead.ctm

Install this file in the Rules sub-directory of the folder in which you have installed XMetaL.

The customization file can specify certain display features such as the color of start-tag and end-tag pairs. More significantly, it can specify which elements, attributes, and prompts are automatically supplied when a given element is inserted. For example, in the file supplied with the Cookbook, the default is to include the following element string whenever a <c02> element is inserted:

<did>
<container type="box"></container>
<container type="folder"> </container>
<unittitle><unitdate></unitdate></unittitle>
<physdesc></physdesc>
</did>.

While the macro file can be used with a keyboard short-cut to insert a <c02>, the customization file determines which elements automatically appear whenever a <c02> is inserted.

To edit the customization file, open a new EAD document. From the **Tools** menu, select **Customizations.** In a frame along the left side of the dialog box that opens is a list of elements. When you highlight a particular element on that list, any default markup or text appears in a window in the center of the box. Simply add to or delete the appropriate text and select **Apply**.

4.1.6 Display File

XMetaL uses the specifications of the Cascading Style Sheet (CSS) standard to control the appearance of marked up text in the Tags On view. This does not effect the display of the document in any other context. It defines such aspects of appearance as font size, weight, and color, element indention and tabs, and line spacing. The specifications for display are stored in a file called

ead.css

Install this file in the Display sub-directory of the folder in which you have installed XMetaL.

To edit the display file, open a new document. From **Tools** on the menu bar select **Editor Display Styles**. A box will open in the middle of the screen with several tabbed entries for editing the appearance of the document in the Tags On view. A knowledge of the Cascading Style Sheet (CSS) protocol will be helpful in making such modifications.

4.2 Author/Editor

Panorama Author/Editor is a software application for creating EAD-encoded finding aids in "classic SGML" syntax. Initially created by SoftQuad, it is now marketed by Interleaf, 62 Fourth Avenue Waltham, MA 02116.

If you use Author/Editor to encode your finding aids, you will need to take several additional steps to convert them to XML syntax in order to make use of the publishing tools that accompany the Cookbook. Section 5 describes two ways to make that conversion. The following tools were created to be compatible with the Windows version 3.5 of Author/Editor.

4.2.1 Installation

The default installation for Author/Editor places the application in a sub-directory of the C: drive called AE. Consult the user's manual to familiarize yourself with the key features of the software, particularly the inserting of elements and attributes and the difference between opening and importing and saving and exporting files.

Install also the following helper files.

4.2.2 Rules

Author/Editor uses a binary version of the EAD DTD called a "rules" file for enforcing compliance with the DTD during document creation and final validation. This file is called

ead.rls

Install this file in the Rules sub-directory of the AE folder.

4.2.3 Templates

Templates provide a standardized file that can be reused over and over to provide the default markup and "boilerplate" data you wish to include in every document. Two templates are provided with the Cookbook. They are named

_eadperson.ae
_eadcorporate.ae

There are only minor variations between the two. The first is geared towards inventories of collections of personal papers, the other to organizational or governmental records. In the former, the Origination element contains a Personal Name sub-element, in the latter a Corporate Name. The only other variations are in the text of Head elements. For example, in one case the text reads "Scope and Content of the Collection" and in the other "Scope and Content of the Records".

Install these two files in the Templates sub-directory of the folder in which you have installed Author/Editor.

To create a new document, open Author/Editor, select **Open Template** under the **File** option on the menu bar, and you will be presented with a list of templates including these files. Select the appropriate template. The default tags, attributes and text will appear for you to fill in and add to.

The order of elements in the templates follows the same sequence found in Section 3. The EAD DTD requires that the <eadheader> sub-elements follow this order and that the <did> element comes first within <archdesc>.

4.2.3.1 Customize Your Templates

You may wish to customize the template files for several reasons. To do so, open a new document using the template, insert, delete, or type over the relevant elements and attributes, select Save As and save the file in the Templates sub-directory under its original name.

- Certain attributes are repository specific. You will need to supply locally valid values for the SOURCE and SYSTEMID attributes in the EAD Identifier and the REPOSITORYCODE attribute in the ID of the Unit for the collection as a whole, i.e. in the Descriptive Identification for the Archival Description.

- Default text for explanatory paragraphs and for Head elements may be modified. For example, you may wish the Head to read "Biographical Sketch" rather than "Biography of….".

- The template may include elements such as appraisal or accruals which you never use. Simply delete the markup from the template.

- Certain elements automatically appear when a component such as a <c02> is inserted. You may wish to change the defaults, for example, removing the <container type="folder"> element if you never include folder numbers in your inventories. These are defined in the file ead.mcr whose customization is described in Section 4.2.4.

- The order of elements may not be optimal for local data entry. One may rearrange the elements within the limits of the DTD. Remember, the order of the elements at data entry is independent of the sequence in which they may display later.

4.2.4 Macros

Macros are a feature of many software applications, including word processors, that permit one to execute repeatedly an oft-used set of keystrokes by typing a short two or three letter combination of keys. The following keystrokes can be used to automatically insert elements and text.

CTRL + ALT + c	Opens a Chronology List Item with a Date and an Event
CTRL + ALT + s	Opens a Scope and Content and Paragraph
CTRL + ALT + u	Opens a Unit Identification
CTRL + ALT + 1	Opens a Component (Level One)
CTRL + ALT + 2	Opens a Component (Level Two)
CTRL + ALT + 3	Opens a Component (Level Three)
CTRL + ALT + 4	Opens a Component (Level Four)
CTRL + ALT + 5	Opens a Component (Level Five)
CTRL + ALT + 6	Opens a Component (Level Six)
CTRL + ALT + 7	Opens a Component (Level Seven)

These are defined in the file

ead.mcr

Install this file in the Macros sub-directory of the folder in which you have installed Author/Editor. The first time you wish to use the macro file, go to the **Special** option on the toolbar and select **Load macros**. Highlight the file ead.mcr and select **Open**. Once you have done so, this file will be the default macro for all EAD documents until another macro is loaded.

When a Component element is opened, the macro file also automatically inserts certain sub-elements. For example, when a <c02> element is inserted, the macro also adds the following elements:

<did>
<container type="box"></container>
<container type="folder"></container>
<unittitle><unitdate></unitdate></unittitle>
<physdesc></physdesc>
</did>

4.2.4.1 Customize Your Macros

If you wish to modify the list of elements that automatically appear within component when a keyboard macro is invoked, it will be necessary to rerecord the macro as it is not possible to edit the text of Author/Editor macros directly. To do so, go to **Special** on the menu bar and select **Record macro.** Perform all the steps on the keyboard that you wish to incorporate into the macro. When you are finished, go to **Special** and select **Stop recording.** In the dialog box that opens, assign an appropriate name and keyboard sequence to the macro you have just created.

4.2.5 Style File

Author/Editor uses an internal "styles" function to specify how the document will be displayed when the Show Tags feature is turned on. This includes the way particular tags are indented and the display of attribute values. This specifications are stored in a file called

ead.stl

Install this file in Styles sub-directory of the folder in which you have installed Author/Editor.

4.3 WordPerfect 9

WordPerfect 9 is a component of Corel's WordPerfect2000 Office Suite. It is available through regular software retail channels or from Corel Corporation (www.corel.com), 1600 Carling Avenue, Ottawa, Ontario, Canada K1Z 8R7.

WordPerfect 9 bundles SGML and XML document creation and editing functionality directly into the WordPerfect editing environment. Version 9 creates XML output as its default, though documents may be readily imported and exported as SGML as well. The following templates and macros were created by Dennis Meissner of the Minnesota Historical Society.

4.3.1 Installation

The default installation of WordPerfect 9 places the application in a Corel sub-directory of the Program Files directory of the C: drive. Consult the user's manual to familiarize yourself with the key features of the software, particularly the conventions for inserting elements, editing attributes, and validating documents. The standard presentation in the editing window displays the document page with start and end tags represented as labeled pointers. There is no "plain-text" view (*c.f.* XMetaL) in which one can view the XML documents as simple ASCII text, but one can open a "Reveal Codes" window that may be set to display in a graphical manner all of the text existing within angle brackets. In addition to the document editing window, a user also can open an XML tree display as a further navigational tool.

Install also the following helper files.

4.3.2 EAD|DTD

The following two files must be copied into the directory:

"…\Corel\WordPerfect Office 2000\XML\Dtd" folder.

ead.dtd
eadbase.ent

4.3.3 Templates

Templates provide a standardized file that can be reused over and over to provide the default markup and "boilerplate" data you wish to include in every document. Two templates are provided with the Cookbook. They are named

eadperson.wpt
eadcorporate.wpt

There are only minor variations between the two. The first is geared towards inventories of collections of personal papers, the other to organizational or governmental records. In the former, the Origination element contains a Personal Name sub-element, in the latter a Corporate Name. The only other variations are in the text of Head elements. For example, in one case the text reads "Scope and Content of the Collection" and in the other "Scope and Content of the Records".

Install these two files in the directory:

"…\Corel\WordPerfect Office 2000\Template\Custom WP Templates\Xml"

To create a new document, open the **File** menu and select **New XML Document**. A dialogue box containing a window labeled **Categories/Projects** will pop up. Highlight either of these two templates and **Select**. A new document editing window will open which will contain all the default elements, attributes and text of the selected template.

The order of elements in the templates follows the same sequence found in Section 3. The EAD DTD requires that the <eadheader> sub-elements follow this order and that the <did> element comes first within <archdesc>.

4.3.3.1 Customize the Templates

You may wish to customize the template files in the following ways.

- Certain attributes are repository specific. You will need to supply locally valid values for the SOURCE and SYSTEMID attributes in the EAD Identifier and the REPOSITORYCODE attribute in the ID of the Unit for the collection as a whole, i.e. in the Descriptive Identification for the Archival Description.

- Default text for explanatory paragraphs and for Head elements may be modified. For example, you may wish the Head to read “Biographical Sketch” rather than “Biography of….”.

- The template may include elements such as appraisal or accruals which you never use. Simply delete the markup from the template.

- Certain elements automatically appear when a component such as a <c02> is inserted. You may wish to change the defaults, for example, removing the <container type=”folder”> element if you never include folder numbers in your inventories. These are defined in the file WordPerfect templates.

To modify the templates, select **New From Project** for the **File** option on the menu bar. Highlight the appropriate template, select the **Options** button and select **Edit WP Template** from the pull-down menu. Once the template has opened, type over any elements, text or attributes that you wish to become the new default.

To edit macros which are stored inside the template file, select **Tools**, **Template Macro**, and then **Edit**. Highlight the macro you wish to edit and select **Edit.** You may add or delete macro instructions with the appropriate syntax. When finished, select **Save & Compile**.

The templates contain several document layout instructions to make data entry and document editing easier. These include the display of all <head> elements as 14-point, bold, centered text and the use of indention and line breaks between certain elements. These customizations were made to simplify navigation within the editing window and do not affect the output document. Further customization can be made in the templates by clicking **Format** on the menu bar and selecting **Edit Layout**.

4.3.4 Macros

Macros are a feature of many software applications, including word processors, that permit one to execute repeatedly an oft-used set of keystrokes by typing a short two or three letter combination of keys. The following keystrokes can be used to automatically insert elements and text.

CTRL + ALT + c	Opens a Chronology List Item with a Date and an Event
CTRL + ALT + s	Opens a Scope and Content and Paragraph
CTRL + ALT + u	Opens a Unit Identification
CTRL + ALT + 1	Opens a Component (Level One)
CTRL + ALT + 2	Opens a Component (Level Two)
CTRL + ALT + 3	Opens a Component (Level Three)
CTRL + ALT + 4	Opens a Component (Level Four)
CTRL + ALT + 5	Opens a Component (Level Five)
CTRL + ALT + 6	Opens a Component (Level Six)
CTRL + ALT + 7	Opens a Component (Level Seven)

These macros are embedded in both of the EAD templates. The user can also select them via a drop down menu (EAD Macros) that appears on the menu bar to the right of the "Help" menu.

4.3.4.1 Customizing the Macros

A further customization is incorporated into the macros for the seven component levels. Required elements and those that are typically included within each component are automatically embedded by the appropriate macro. For example, when a Component element is opened, the macro file also automatically inserts certain sub-elements. For example, when a <c02> element is inserted, the macro also adds the following elements:

```
<did>
<container type="box"></container>
<container type="folder"></container>
<unittitle><unitdate></unitdate></unittitle>
<physdesc></physdesc>
</did>
```

The macros are embedded in the WordPefect template and are modified in the template as described in section 4.3.3.2.

4.3.5 Validating Your Document

Every valid XML document must begin with an XML declaration and a DOCTYPE declaration that specifies the relevant DTD. WordPerfect 9 will insert the XML declaration automatically at the beginning of your document when you save it. However, it does not supply the DOCTYPE declaration and is unhappy when you include it in the template file yourself. This is a small problem that is readily resolved. The templates include the DOCTYPE declaration. You simply need to do two things to work around WordPefect's unhappiness on this subject.

When you validate your document, the software will tell you that it doesn't like any text to appear before the root element, <ead>. Simply ignore this message and proceed with the validation which will report any real subsequent errors. Similarly, ignore any related error messages when you save the document.

When the file is saved, WordPerfect will insert the XML declaration and leave your DOCTYPE declaration in the file, intact save for one alternation. It will replace the angle bracket at the beginning of the DOCTYPE declaration with its character value- < Simply open the document in a text editor like Windows Notepad and replace this string of characters with the left-angle bracket (less-than symbol), and save the file.

Section 5: Converting SGML Files into XML

If you choose to create your EAD documents in "classic SGML" syntax using an application like Author/Editor, and you wish to the use the XSL publishing tools described in this Cookbook, you must first convert your SGML files into XML syntax.

5.1 EAD as XML

There are five ways in which an EAD instance in XML differs from one in "classic SGML" syntax.

1. The inclusion of an XML declaration at the beginning of the document.
2. The required inclusion of a system identifier that points to an instance of the EAD DTD (optional in SGML) at the end of the DOCTYPE declaration which is found at the beginning of the document.
3. All element and attribute names in lower case. (SGML is not case sensitive in this respect).
4. Elements that are defined as EMPTY (lb, extptr, ptr, ptrloc, extptrloc) must be expressed in empty-element-tag syntax, for example as <lb/> rather than as <lb></lb>. The *Cookbook* does not employ any of these elements.
5. A slight modification to the file "ead.dtd" is required. A copy of the XML version of the "ead.dtd" file accompanies the Cookbook.

For more information on the differences between the SGML and XML syntax of EAD and the necessary modification to the DTD, see Section 4.3.2 of the *EAD Application Guidelines.*

This section describes two ways to convert an SGML file into XML syntax.

5.2 SX

SX is a freeware tool that is available for non-commercial use from James Clark as part of his suite of SGML/XML parsing software called SP. Information about SP may be found at http://www.jclark.com/sp.

5.2.1 To install the tool-

- Download and unzip the SP software from http://www.jclark.com/sp/howtoget.htm.

- Copy the five basic EAD files into the sub-directory sp/bin in the folder in which SP is installed. These are ead.dtd, eadbase.ent, eadchars.ent, eadnotat.ent and eadsgml.dcl.

  These files are available at the official EAD website
  http://www.loc.gov/ead/.

- Copy the SGML catalog file that accompanies the Cookbook into the same sp/bin sub-directory. The file is named "**catalog**".

- Copy the ISO character entity files that accompany the Cookbook into the sp/bin sub-directory.

- Add the following environmental variable statements to your autoexec.bat file (in the root directory of your C: drive) using a text editor such as Windows Notepad. The following statements assume that SP has been installed in a folder at c:\sp.

```
SET SP_CHARSET_FIXED=NO
SET SP_ENCODING=XML
SET SGML_CATALOG_FILES=c:\sp\pubtext\xml.soc
```

5.2.2 To use the tool to convert a file-

- Place the file to be converted in the bin sub-directory of the sp folder where SP has been installed.

- Go to the command line (that's right- the DOS prompt) in the directory where the file **sx.exe** is located, the sp/bin folder.

- Type the following syntax to execute the conversion

sx -xnotation –xndata –xlower –xempty –xno-lb-in-tag
name_of_source_sgm_file>name_ of_ target_ file

for example `sx –xnotation –xndata –xlower –xempty –xno-lb-in-tag fonds.sgm>fonds.xml`

This will convert the file fonds.sgm to the file fonds.xml with elements and attributes in lower case and place it in the same directory as the source file. The SX documentation describes parameters for specifying other source and target directories.

SX outputs a well-formed document but does not include the DOCTYPE declaration. You will have to insert the following statement immediately after the XML declaration (<?xml version="1.0"?>) which appears at the head of the document.

<!DOCTYPE ead PUBLIC "-//Society of American Archivists//DTD ead.dtd (Encoded Archival Description (EAD) Version 1.0)//EN" "ead.dtd">

More information about how to use SX is available at the following address:
http://www.jclark.com/sp/sx.htm

5.3 Microsoft Word

A macro file for Microsoft Word 95 is available from the EAD Help Pages site which converts all the tags in an EAD document in SGML syntax into lower case.

http://jefferson.village.virginia.edu/pub/ead/resources/Word/wordmac.html

After the macro has run, one must add the XML declaration to the beginning of the document and append a system identifier giving the path to the EAD DTD file. The beginning of the file should then look like this, assuming the file "ead.dtd" is in the same directory as the finding aid-

```
<?xml version="1.0"?>
<!DOCTYPE ead PUBLIC "-//Society of American Archivists//DTD ead.dtd
(Encoded Archival Description (EAD) Version 1.0)//EN" "ead.dtd">
```

The syntax of any empty elements will have to be changed manually or through a global Search and Replace command in the word processor. Simply save the file as ASCII text with the extension .xml

Section 6: Publishing Tools: Stylesheets

Several stylesheets accompany the Cookbook that will convert EAD finding aids that have been marked up according to the encoding protocol in Section 3 from XML files into HTML syntax for viewing in a standard web browser. These stylesheets are not general, all-purpose tools that will process each and every valid EAD document. They are designed particularly to function with the Cookbook. However, they certainly may be modified to meet local preferences in encoding or presentation. The conversion process uses the language of the XSL Transformation (XSLT) protocol, a Recommended Practice for the web developed by the World Wide Web Consortium (W3C). A software application that performs transformations using XSLT reads the text of an XML document, applies the syntax described in a specified stylesheet, and creates an HTML output file. Several such applications are available as "freeware" from Microsoft, IBM, Oracle and others. The stylesheets accompanying the Cookbook have been tested against the XT package available from James Clark. This product is quite complete in its features, and is fast and easy to install. It may be run as a batch application or in real-time on a server as an HTTP servlet. (See Sections 7 and 8 for detailed directions.)

The accompanying stylesheets are relatively verbose in their code and explanatory comments. Their structure is designed in a modular fashion that is optimized for readability, generalization, and future modification rather than for optimal performance for a specific file. However, XSL features known to affect performance adversely have been avoided. The code contains many instances of conditional processing to accommodate variations across individual finding aids and institutions.

The stylesheets support five types of markup for container information, i.e. box and folder numbers. One may include only box numbers, box and folders numbers, only folder numbers, a combined box/folder number, or no container data at all. If any container information is given, the stylesheets assume that it will be given for every component except the <c01> which, as the description of an entire series, would not normally have a particular container number associated with it. As described in Section 3.4, the container element must be present for each component but need not always contain data. The stylesheets are designed to suppress the appearance of repetitive container information. For example, if one lists box and folder numbers or only box numbers, a box number that has already been displayed in a given <c01> will not display again for other components in the same box. Similarly, where only folder numbers or box/folder numbers are given, the folder or box/folder number respectively will not repeat within the same series.

The accompanying templates and keyboard macros may insert elements into the document that you do not use. While there may be other good reasons for deleting empty tags from our document, it is not necessary to do so for the stylesheets. In all situations where a problem might arise from an empty element, the stylesheets test for the presence of actual content as well as the presence of the tags.

6.1 Styles

Four XSL stylesheet files are supplied with the Cookbook. The basic presentation of the inventory is identical in most respects in all four. Styles 1, 2, and 3 differ only in the placement of the table of contents within the finding aid.

6.1.1 *Style 1* produces the standard display of the inventory with a table of contents at the beginning of the document in the manner of a book. Internal hyperlinks connect this table of contents with the various sections of the finding aid. The stylesheet file is named **EADCBS1.xsl.** An example that applies this style maybe seen at

http://www.mnhs.org/library/findaids/2468a.html

6.1.2 *Style 2* produces the standard display of the inventory with a table of contents appearing in a window along the left side of the screen when viewed in a browser. Internal hyperlinks connect this table of contents with the various sections of the finding aid. This style uses an HTML table to create the left side table of contents. It may be useful for institutions whose web policies discourage or disallow the use of HTML frames. The stylesheet file is named **EADCBS2.xsl.** An example that applies this style maybe seen at

http://www.mnhs.org/library/findaids/2468.html

6.1.3 *Style 3* also produces the standard display of the inventory with a table of contents appearing in a window along the left side of the screen when viewed in a browser. Internal hyperlinks connect this table of contents with the various sections of the finding aid. This style uses the HTML frames feature to populate the table of contents. The XSL Transformation process described in Section 7 can generate the four files needed to populate this frameset in a single step. The stylesheet file is named **EADCBS3.xsl.** An example that applies this style maybe seen at

http://www.mnhs.org/library/findaids/2468f.html

6.1.4 *Style 4* produces a variation of the standard display that is intended to be used to create a print version of the inventory. It differs from the previous three styles in three respects. Intended for print output, it lacks any internal hyper-navigation links. The document begins with a fuller description of the collection in a manner that more closely resembles the title page of a book. Finally, the description of subordinate components is divided into two sections, beginning with a summary of each series which then is followed by a second section which provides the detailed information given for the collection to the series, file and item levels, as provided. This approach provides the reader a quick overview of the collection without having to jump through the text of the description of components to find series titles, dates and abstracts at scattered locations throughout the document. It more closely reflects the linear reading of a print document as opposed to the hyper-navigation of an online file. This "two-view" display is not included in the web version as users find it confusing and redundant in that environment.

The print copy may be produced by processing an EAD inventory against this stylesheet. The resulting HTML file is then imported into a word processor for final editing such as the insertion of head and footers and page breaks. This stylesheet has been tested with Microsoft Word 97 which is capable of converting an HTML file into Word format while preserving the physical formatting of the HTML file. Tests with Word 2000 indicate that even more sophisticated features such a hanging indention can be supported during conversions.

The stylesheet file is named **EADCBS4.xsl.** An example that applies this style maybe seen at

http://www.mnhs.org/library/findaids/2468prt.html

6.2 Modifications

Each of these stylesheets includes an HTML <img> tag near the beginning of the file that inserts an institutional logo at the top of the document. This file appears in the SRC attribute with the fictional name "yourlogo.gif". You will need to either alter the file name or delete the element altogether, depending on whether or not you intend to include your organizational logo.

Section 7: Transforming XML Files into HTML

The stylesheets described in the previous section can convert your XML files into HTML. This transformation may occur either in advance of any user request, i.e. in "batch mode" or as each user accesses the file on a server, i.e., in "real-time.". With the newest generation of web browsers, it is possible for this transformation to occur on the user's computer rather than the server. However, the Cookbook describes server-based transformation. This will likely remain the preferred method until such XML-enabled browsers are used more widely. Conversion done in advance- batch mode- is described in this section. Real-time transformation is described in Section 8 along with other server issues.

Two conversion methods are described. The first involves the use of a free software program called XT, an application that can convert your files one at a time or an entire directory at once. The second method involves the XMetaL software which, with version 1.2, includes an embedded version of XT that can effect the transformation as an integral part of the authoring process.

7.1 XT

This tool is a freeware product for non-commercial use. It is available at

http://www.jclark.com/xml/xt.html

It can run in any Java-enabled environment. The following installation is for the Windows platform.

7.1.1 To Install the Tool-

- Create a directory in which to create your file transformation.
- Install (download and unzip) the windows version of the XT application from the web site listed above.
- Install the required EAD files- eadbase.ent and a version of the ead.dtd file that has been modified for XML use as described in Section 4.3.2.1 of the *EAD Application Guidelines.*
- Install a copy of the Microsoft version of the Java Virtual Machine which they call the Microsoft Virtual Machine. It is available as part of a customized installation of the Internet Explorer browser or from the Microsoft web site (currently at http://www.microsoft.com/java/vm/dl_vm32.htm)

7.1.2 To Use the Tool to Transform a File-

- Go to the command line in the directory where XT is located. That's right- the DOS prompt.

- Type the following syntax to execute the conversion

  xt *name_of_xml_source_file name_of_xsl_stylesheet_file name_of_output_file*

  for example `xt fonds.xml eadcbs3.xsl fonds.html`

  The processor will parse the DTD, parse your document, parse your stylesheet, and then output the file or supply an error message if there is a syntax problem with any of the files.

7.1.3 XT may also convert all the files in a directory with a single command.

- Type the following syntax to execute the conversion

  xt name_of_source_directory name_of_xsl_stylesheet name_of_result_directory

  for example `xt c:\xmldocs c:\styles\eadcbs3.xsl c:\htmldocs`

- The stylesheet should not be located in the source directory.

7.2 XMetaL

The XMetaL software includes a built-in C++ version of the XT XSL Transformation engine as of version 1.2. This processor requires the presence of version 5.0 of the Microsoft Internet Explorer browser. This transformation feature appears as part of the **Print Preview** option on the **View** menu. The stylesheet for Cookbook style 3 uses an extension to XSL called xt:document to produce the four files required for the HTML frame-based display it produces and cannot be used with version 2.1 of XMetaL as described below. With that stylesheet, use the XT application described in Section 7.1.

7.2.1 To Install the Tool-

- The configuration of the XMetaL software necessary to use XT is included in the ead.mcr file distributed with the *Cookbook*. The default setting looks for the stylesheet in the Display folder of the XMetaL directory. Install a copy of the stylesheet into that folder.

- The default value in the ead.mcr file is the eadcbs2.xsl file. If you wish to use a different stylesheet, you will have to edit the ead.mcr file. To do so,

    - Open the file a text editor such as Windows Notepad. About two-thirds of the way through the file you will find the following text:

```
// Path to XSLT stylesheet is hard codedvar xslPath =
Application.Path + "\\Display\\eadcbs2.xsl";
```

    - Change the file name at the end of the second line to the appropriate value. Save the file in the appropriate Macros sub-directory.

7.2.2 To Use the Tool

From the **View** option from the menu bar, select **Page Preview**. The transformation will occur and the resulting file will be displayed in a browser which will open up within the XMetaL application. You can save the file that is created since it is stored in the same directory as the stylesheet. XMetaL assigns the file a random but distinctive name to the output file which you will no doubt wish to change.

Section 8: Delivery of EAD on the Web

This section describes actions that may need to be carried out on your web server or with related files as the final step in the "publication" process- delivering documents to the end user. Assistance will probably be needed from your webmaster or system or network administrator.

8.1 Links from Web Pages

If you are creating links to your finding aids from other web pages, you will need to-

- Embed in the source document an HTML pointer to the finding aid file. This requires an introductory knowledge of HTML coding. Typically one uses the HTML anchor <a> element with an HREF attribute to accomplish this.
- Store the inventory files in an appropriate, publicly accessible location on your web server. This requires access to the file server and a knowledge of its file structure. Consult your web support staff.

8.2 Links from Online Catalogs

If you are planning to create links from MARC records in your catalog to finding aids, several steps will be required beyond EAD encoding. This method assumes, of course, that you have a web-enabled online catalog that generates links to external files from appropriate fields in the catalog record.

- Create a linking field in the MARC record using a MARC editing tool, typically part of your Integrated Library System (ILS) software. Changes have recently been made to the MARC 21 format that will provide additional subfield options and directions for MARC encoding in the future. Currently you have two options. You may create an 856 field, Electronic Location and Access, with a pointer to the finding aid. The note might look something like this.

  856 42 $ 3 An electronic version of the inventory for this collection is available at $u http://www.mnhs.org/library/findaids/00054.html

  When your ILS vendor supports the newly-authorized addition of subfield u in the 555 field, Cumulative Index/Finding Aids Note, you may create a link like this instead.

  555 0 $ a An inventory that provides additional detailed information about this collection is available at $u http://www.mnhs.org/library/findaids/00054.html

- Store the finding aid file in an appropriate, publicly accessible location on your web server. This requires access to the file server and a knowledge of its file structure. Consult your web support staff.

8.3 Access from Data Servers

A number of products are available if you wish to provide keyword or structured searching of the full text of your inventories. This obviously is a very complex and technical issue, well beyond the scope of the Cookbook. Several applications are available currently and new ones appear regularly. More detailed information may be found in Chapter 5 of the *EAD Application Guidelines* and other sources:

The XSL list and archives:

http:www.mulberrytech.com/xsl/xsl-list

The XML Cover Pages (the definitive source of links to SGML/XML information)

http://www.oasis-open.org/cover/

Steve Pepper's Whirlwind Guide to SGML and XML Tools and Vendors

http:www.infotek.no/sgmltool/guide.html

8.4 Transformation to HTML on the Server

Section 7 described how you might convert your files in advance of user access. Several tools are now available for various HTTP servers that enable real-time conversion from XML to HTML each time a user requests the file. The Cookbook does not describe their implementation in detail. It is a technically complex process that will require advanced computing skills. You will need to consult with your data services department. At least six major products are available as of April 2000.

8.4.1 XT

The same application we considered in Section 7, XT also may run as a servlet on an http server. For further information consult the XSL list archives (see above) under the thread "XT as servlet". Basic information is available at

http:www.jclark.com/xsl.

8.4.2 SAXON

An XSL processor that may run as an applet or a servlet. A "cut-down" version of the full processor, called Instant SAXON, may be run as a batch mode processor from the command line in the same manner as described for XT in Section 7. Information about SAXON is available at

http://users.iclway.co.uk/mhkay/saxon/

8.4.3 Microsoft

An XSLT processor is available from Microsoft as an extension to the ISAPI for use with its Internet Information Servers. At the time this section of the Cookbook was written, this application has not yet implemented some of the features found in the XSLT standard, including ones used in the accompanying stylesheets. Microsoft has indicated that it plans be fully compliant and has been issuing new releases regularly. Information is available at

http://www.microsoft.com/downloads/webtechnology/xml/xslisapi.asp

8.4.4 Xalan

An XSL processing suite for the Apache web server, Xalan is available from The Apache XML Project. Consult

http://www.apache.org/xalan/index.html

8.4.5 Cocoon

Another XSL publishing environment developed for the Apache web server by the Apache XML Project. Consult

http://www.apache.org

8.4.6 LotusXSL

An XSL processor from IBM/Alphaworks. One of many freeware products in the Alphaworks XML/XSL tool kit. For information, consult

http://www.alphaworks.ibm.com/LotusXSL