THE EAD COOKBOOK- 2002 Edition

The first edition of the EAD Cookbook appeared in 2000 in response to a desire within the profession for practical, step-by-step assistance with the implementation of EAD. It contained a simple model encoding protocol with an accompanying suite of software tools for "authoring" electronic finding aids and stylesheets for "publishing" them on the Web. It functioned as an extension of the *EAD Tag Library* and the *EAD Application Guidelines*.

The appearance of EAD Version 2002, the shift of the EAD community from SGML to an XML environment, the appearance of new tools for creating and distributing finding aids, and the emergence of community-based encoding protocols necessitate a revision of that earlier work. While the basic EAD recipe has not changed, some of the ingredients have. As an update, this edition focuses on those aspects of implementation that have changed since 2003, specifically changes in the EAD element set, new tools for creating EAD-encoded documents, and the need to provide additional XSLT stylesheets for transforming EAD files into HTML.

The encoding protocol in the first edition was a compilation of guidelines from several sources. In this version, it is based on the "RLG Best Practice Guidelines for Encoded Archival Description," though no significant differences have resulted from that change. Regularized encoding is important for very practical reasons. The consistent use of the EAD element set makes it possible to exchange and consolidate multiple finding aids from many institutions into union databases or simply with others in one's own repository. Without standardized encoding, it is difficult to manage indexing, display, and manipulation of files. Consistency of presentation also improves user understanding of the purpose and scope of inventories just as the standardized display of library catalogs makes them comprehensible to a large and diverse audience. As testimony to the necessity and wisdom of such regularization, every EAD consortium has adopted some form of encoding protocols.

This edition provides assistance in using three applications for creating encoded finding aids- XMetaL, <oXygen/>, and Note Tab. This includes instructions for installing and modifying the applications, and auxiliary files such as templates that make them easier to use.

New stylesheets have been created that accommodate the changes in EAD 2002, offer greater flexibility and more choices for local display, and a simpler and more fully documented syntax to facilitate local modifications.

Bon appetit!

Michael J. Fox
July 2003

Section 1: The Recipe

Step 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/

Step 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 sites now have software that performs at least keyword searches of documents located there. 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 and regional projects are creating such services. The RLG's Archival Resources service hosts finding aids from repositories in several countries and is not limited to its 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 7 for implementing publishing options a and b above.

Step 3. Review the general principles and specific details of the Encoding Protocol.

These are found in sections 2 and 3. Some encoding choices are specified in the encoding protocol and built into that accompanying templates. You will need to ratify those decisions and; in other cases, decide what your local practice will be. For example, what text do you wish to appear in specific <head> elements? 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.

Step 4. Acquire a software application for authoring finding aids.

Products currently available include XMetaL, XML Spy, Oxygen, Note Tab, WordPerfect 2000, and others. Consult your favorite web search engine for current addresses and additional information about each of these products.

Step 5. Install and configure the software. (Section 4)

Step 6. Create EAD-encoded finding aids. (Sections 3 and 4)

Step 7. Adopt or modify an existing XSLT stylesheet or create your own. (Section 5)

Step 8. Transform your EAD files into HTML or print. (Section 6)

Step 9. Deliver your finding aids to your users. (Section 7)

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 are not different fundamentally from the other tools we create to help our users in identify and locate 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 or 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. While markup may use either "classic SGML" or XML syntax, virtually all current applications employ XML. This edition of the Cookbook focuses exclusively on XML implementations. If you have older SGML files, please consult the first edition.

5. The model assumes that the materials described in the finding aid form a collection of personal papers or organizational records. The accompanying stylesheets provide formatting for collections that are either complex with multiple series and perhaps sub-series or have a simpler structure consisting of multiple files in a single series. The encoding further assumes that the collection is fully processed and described.

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 5.5).

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. Information regarding 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. The "RLG Best Practice Guidelines" goes farther and specifies a larger set of elements as mandatory or recommended.

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 in Section 3 is more prescriptive than the RLG guidelines in one respect. It requires the inclusion of user-friendly displays using the Head element for specific individual elements, because, among other reasons, the text of these Head elements is needed to populate the entries in the table of contents produced by the accompanying stylesheets. It also includes recommended text for display labels used with <did> elements and samples of explanatory text that might be added to a finding aid elsewhere to assist remote users who 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 3.3.22 of the Cookbook. Internal links are specifically encoded with a Reference element.

10. Additional markup such as the encoding of personal names and subjects beyond the prescribed structural elements may be 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. They are based on the "RLG Best Practice Guidelines for Encoded Archival Description." These choices are assumed by and incorporated into the accompanying templates and other software tools described in Section 4 and the stylesheets in Section 5.

The protocol includes every major high-level element available in EAD 2002. If your repository does not wish to include a particular element, it simply may be deleted globally from the accompanying templates or removed from individual documents on a case-by-case basis.

As a display convention, EAD element names appear in the encoding protocol in their tag form, as in <eadid>. Following the conventions of XSLT, the names of attributes are differentiated by the inclusion of a preceded @ symbol. The id attribute is thus written as @id.

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 to facilitate the conversion of EAD data into MARC records. Of course, other encoding analogs, such as to the 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 @relatedencoding as “MARC21”.

3.2 EAD Header <eadheader>

Record the attribute @audience as “internal”.
Record @langencoding as “iso639-2”.
Record @countrycode as “iso3166-1”.
Record @dateencoding as “iso8601”.
Record @repositorycode as “iso15511”.
Record @scriptencoding as “iso15924”.

These are the default values defined by the EAD DTD.

3.2.1 EAD Identifier <eadid>

Record a value that is suitable as a computer file name in order to facilitate file management. Stylesheet 7 which accompany the Cookbook uses the value of <eadid> as the basis for several file names in an HTML frameset.

To ensure that the <eadid> assigned is unique among the universe of inventories, use the attributes @countrycode and @repositorycode.

Record a coded value for the location of the repository in @countrycode based on ISO Standard 3166-1. For the United States, the code is "us". In @mainagencycode, enter the code based on ISO 15511 for the institution maintaining the electronic EAD instance. In North America, these standard codes are supplied by the Library of Congress or the National Library of Canada (see the *MARC Code List for Organizations*).

If your institution does not have such a code, one can be requested online at

http://www.loc.gov/marc/organizations/orgshome.html#requests.

The electronic version of the finding aid may also be identified through the use of a public identifier, a uniform resource locator (url), or a uniform resource name (urn). The "RLG Best Practices" require the inclusion of one of these options.

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 creator of 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.

Provide 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>

Provide the name of the publishing repository in the Publisher <publisher> element.

Provide the date of the inventory in the Date <date> element.

Provide 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>

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

3.2.3.2 Language Usage <langusage>

Provide 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 in the attribute @level, the organizational level of the entire body of materials described in the finding aid. This attribute is required by the EAD DTD..

Record @type as “inventory”.

Provide 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 user 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 5.3.3 for further explanation of this process and instructions on how to alter the sequence..

3.3.1 Descriptive Identification <did>

Provide 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”.

Provide the name of the repository in <repository><corpname> even if its display is to be suppressed.

Record @encodinganalog as “852$a”.
Record @label as “Repository:”.
Provide the repository’s city and state in <addressline> to conform to ISAD(G) standards.

Provide the name of the creator of the materials in <origination>. Include a Corporate, Personal, or Family Name element as appropriate.

Record @encodinganalog for the sub-element as either “100” or “110”, depending on which is the appropriate MARC element.
Record @label as “Creator:”.

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

Provide the date of the materials in <unitdate>. Current practice prefers that unitdate be recorded separately from unittitle, though the accompanying stylesheets accommodate the encoding of unit dates either as separate elements or as a sub-element of the Unit Title element.

Whitespace and punctuation used to separate data within <unittitle> and <unitdate> should be included in the markup.

This element may be repeated if both bulk and inclusive dates are recorded.
Record @encodinganalog as either “245$f” or “260$c” as appropriate to the MARC conventions your institution employs.
Record @label as “Dates:”.

Provide a physical description in <physdesc>.
Record @encodinganalog as “300$a”.
Record @label as “Quantity:”.

Provide an abstract of the contents of the materials in <abstract>.
Record @encodinganalog as “520$a.
Record @label as “Abstract:”.

Provide in <unitid> a uniquely identifying name or number for the materials described, as required by ISAD(G).
Record @encodinganalog as “099”.
Record @label as “Identification:”.
Record @countrycode as found in ISO 3166-1. The default value in the accompanying templates is “us”.
Record in @repositorycode attribute the value found in the mainagencycode attribute in <ead>

Provide in <langmaterial> the language of the materials being described.
Record @label as "Language:".
Record@encodinganalog as "546".

Provide in <materialspec> information unique to the class of the materials being described.
Record @label as "Material Detail:".
Record @encodinganalog as 254, 255, or 256 as appropriate.

3.3.2 Biography or History <bioghist>

Record @encodinganalog as “545”.

Provide 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.

3.3.3 Scope and Contents <scopecontent>

Record @encodinganalog as “520”.

Provide 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.

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 of the materials may be included in the scope and content statement or in a separate Arrangement element.

3.3.4 Arrangement <arrangement>

Record @encodinganalog as “351$a”.

Provide a <head> that describes the content of this element.

Encode the body of the element as paragraphs of text.

Alternatively, one may describe the arrangement of the materials by listing the titles of the component series. In this case, precede 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 arrangement element appears separately or within a scope and contents element.

3.3.5 Related Material <relatedmaterial>

Record @encodinganalog as “544 1”.

Provide 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 Separated Material <separatedmaterial>

Record @encodinganalog as “544 0”.

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text or as a <list>

3.3.7 Other Finding Aid <otherfindaid>

Record @encodiganalog as “555”.

Provide a <head> element that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.8 Index <index>

Record @encodinganalog as "500".

Provide a <head> that describes the content of this element.
Encode the body of the element with paragraphs of text and the Index Entry element.

3.3.9 Bibliography <bibliography>

Record @encodinganalog as "510".

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text or Bibliographic Reference elements.

3.3.10 Controlled Access Headings <controlaccess>

Provide 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.

Provide <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 @source 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.

Record @encodinganalog 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.

3.3.11 Restrictions on Access <accessrestrict>

Record @enodinganalog as "506".

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.12 Restrictions on Use <usesrestrict>

Record @encodinganalog as "540".

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.13 Custodial History <custodhist>

Record @encodinganalog as "561".

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.14 Alternative Form Available <altformavail>

Record @encodinganalog as "530".

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.15 Preferred Citation <prefercite>

Record @encodinganalog as “524”.

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.16 Acquisition Information <acqinfo>

Record @encodinganalog as “541”.

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.17 Processing Information <processinfo>

Record @encodinganalog as “583”.

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.18 Appraisal <appraisal>

Record @encodinganalog as “583”.

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.19 Accruals <accruals>

Record @encodinganalog as “584”.

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.20 Location of Originals <originalsloc>

Record @encodinganalog as “535”.

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.21 Physical Characteristics and Technical Requirements <phystech>

Record @encodinganalog as “340” or "538" as appropriate.

Provide a <head> that describes the content of this element.
Encode the body of the element as paragraphs of text.

3.3.22 Description of Subordinate Components <dsc>

See Chapter 3 of the *EAD Application Guidelines* for a discussion of the possible structural models for the Description of Subordinate Components. The accompanying stylesheets dsc1 through dsc13 assume that encoding follows the “combined” model for marking up the <dsc> in order to present the text of the <dsc> in a single sequence. 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 5.6 for further details.

Provide 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 encoding model and the accompanying stylesheets assumes that the Component (First Level) <c01> element represents many different levels of description, depending on the nature of the collection. It may be a sub-group, a sub-collection, series, sub-series or a file. Record the following information for each <c01>.

Record @level as appropriate, using the appropriate value from the list supplied in the EAD DTD.

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. Experience shows that it is virtually impossible for a stylesheet to predict and properly insert all variations in punctuation given the number of permutations that are possible. The stylesheets will attempt to insert a space between elements whenever 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, some other type of container designation such as reel, frame or drawer, or no container information at all. The accompanying stylesheets accommodate a wide range of local practice. See Section 5.4 for a detailed discussion of the options supported. The Cookbook promotes the explicit inclusion of full container data for all components as a highly desirable practice that will facilitate future data manipulation. Fortunately, it is not difficult to insert <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 in section 4.

Record the @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 anywhere in a finding aid, including the <dsc>. 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 @colnum. Cookbook stylesheets will use this information and the @colwidth values to <tgroup> to define the table's configuration. The stylesheets require only these two attributes.

3.4.1.2 Table Group <tgroup>

Include a Table Column Specification <colspec> element for each column. Specify the width of the columns in @colwidth 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 attribute @id 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 as the value of the attribute @target of this <ref>.

The cookbook stylesheets support the use of <ref> in a very limited way. The text of a <ref> element will be transformed into an HTML <a> at the source of the link. Users of <ref> will need to modify the stylesheet code for any element that is to become the destination of a link so that it generates the proper HTML <a> at that location.

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, Oxygen, and NoteTab. 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 encoded documents, is produced by the Corel Corporation of Ottawa, Ontario. It is available through standard software retail outlets as well as directly from Corel. The software runs on various Windows platforms. The following tools were created to be compatible with versions 2.0 and higher 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 a Corel/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.

If you wish to use the Page Preview function that enables you to perform XSLT transformations from XML to HTML within the XMetaL application while you are creating the document, you will need to do one or two things prior to installing XMetaL. You will need to have a copy of Microsoft Internet Explorer (IE) version 5.0 or higher installed on your computer. If you have IE 5.0 or IE 5.5 and are not using the Windows XP operating system, you will need to upgrade your version of the MSXML files. The current version is 4.0 and it is available for download from the Microsoft web site. The address for the download as of July 2003 is www.microsoft.com/xml. If you have IE 6.0 or higher installed or are using Windows XP, the proper files are already present in your computer.

Once these files have been installed or updated, you can install XMetaL using its built-in, standard Windows installation routine.

Install the helper files described 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

Download the file "ead.dtd" from the official EAD web site maintained by the Library of Congress at www.loc.gov/ead. Install this file in Rules directory within the folder in which you have installed XMetaL. The first time you create a new EAD file, you will be asked to specify the DTD you wish to use. Select the file "ead.dtd". When you do this, XMetaL will automatically compile a copy of ead.rlx and adds it to the Rules directory.

4.1.3 Templates

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

eadpersonxm.xml
eadcorporatexm.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 @mainagencycode attribute in the EAD Identifier and @repositorycode 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 may never use. Simply delete the markup from the template.
- Some elements in the template, such as repository name and address, which 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 and 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/General directory.

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 key combination. 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><unittitle>
        </unitdate></unitdate>
        <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 affect 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 NoteTab

NoteTab is a highly respected text editor from Fookes Software of Geneva, Switzerland. It has been customized for use with EAD through a series of tools that are available with the Cookbook. These routines were developed by Chris Prom from the University of Illinois. While a free version of the software is available, users are encouraged to purchase the NoteTab Pro version (currently $19.95). This will permit you to customize the application fully for your use. Discounts are available for purchases of five or more copies. The software can be downloaded from

http://www.fookes.com

4.2.1 Installation of NoteTab

The software can be downloaded either as a self-extracting executable file or in a fully zipped version. The former is simpler as one needs merely to run the NotePro_setup.exe file but some system do not permit the import of exe files and so the zip version, with an unzipping program like WinZip, should be employed.

4.2.2 Installation of NoteTab Customization for EAD

Download the self-extracting file, eadnotetab.exe, from the EAD Cookbook files on the EAD Help Pages web site. When you run the file, it will create a new directory, c:\eadcb, and install the appropriate files. Please read the accompanying file, "readme.txt", which will instruct you on which files need to be moved to other directories on your computer.

4.2.3 Templates

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

eadpersonnt.xml
eadcorporatexntxml.

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 Templates directory of the folder called eadcb.

4.3 <oXygen/>

The <oXygen/> XML Editor is a Java based application that runs in a variety of operating systems, including Windows, Mac OS X, Linux, and Solaris. Discounts are available for academic and non-profit users and for purchases of multiple copies.

It is necessary to have the Java 2 Runtime Environment (JRE) software loaded on your computer. Java version 1.3 (or higher) is required and freely available at

www.sun.com.

The <oXygen/> software itself can be downloaded from Syncro Soft at

www.oxygenxml.com

4.3.1 Installation of <oXygen/>

The <oXygen/> User Manual is available in PDF format at the site listed above and contains step by step installation for each operating system.

4.3.2 Templates

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

eadpersonoxy.xml
eadcorporateoxy.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".

Open each of these files in <oXgen/> and then execute the Save as Templates function. To employ them, go to File>>Open from Template.

Section 5: Publishing Tools: Stylesheets

Stylesheets transform encoded text from one format into another. Stylesheets written in the Extensible Stylesheet Language Transformation (XSLT) syntax are computer programs that instruct a software application called an XSLT processor how to perform the transformation.

We use stylesheets to transform our EAD-encoded finding aids from XML files into HTML documents for display in a web browser. In this case, the stylesheet specifies which elements in the XML source document are to appear and in what order. It also defines the appearance of the text- its size, weight, font, color, etc. and the position or layout of the text on the computer screen or the printed page. See Sections 6 and 7 for detailed directions on how this process occurs.

This Section has five parts. Following this introduction, there is an overview of the five new stylesheets and a summary of their key formatting features. The fourth part deals with issues relating to the Description of Subordinate Components and the section concludes with a summary of how those stylesheets for the <dsc> function.

5.1 The EAD Cookbook stylesheets

The first edition of the EAD Cookbook included four different stylesheets that converted finding aids created according to the syntax of EAD Version 1.0 and that were consistent with the encoding protocol specified in that edition of the Cookbook. They are described in detail in the first edition. A new set of stylesheets is now available as part of the second edition of the EAD Cookbook to accommodate the changes in EAD 2002. Their HTML output, however, retains the same appearance as the original stylesheets.

None of these stylesheets is a general, all-purpose tool that will process each and every valid EAD document. Simply put, there are too many possible variations in finding aid creation and encoding to make it feasible to address every possible institutional option in a single set of stylesheets. The Cookbook stylesheets are designed particularly to function with the encoding recommendations found in Section 3. However, they certainly may be modified to meet local preferences in encoding or presentation. To facilitate this, these stylesheets are relatively verbose in their code and explanatory comments. They are structured in a modular fashion that privileges readability, generalization, and ease of future modification rather than optimal performance for a specific file. Hopefully, XSLT 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 new stylesheets differ from their predecessors in several respects.

5.1.1. They include support for the new elements and attributes included in EAD 2002. They assume that the finding aid is marked up in accordance with the encoding protocol found in this edition of the Cookbook. Specifically, users need to remember that these are not all-purpose, universal stylesheets that will support any and all encoding options.

5.1.2. The internal structure of the new XSLT code follows a syntax that should be easier than the previous ones to interpret and modify locally. The stylesheets are deliberately verbose for the same reason. Since these are offered as a community resource that others might adapt as needed, the XSLT syntax strives for clarity rather than technical "elegance". The text also includes more explanatory comments that describe how each portion of the stylesheet works.

5.1.3. The new stylesheets utilize a more modular structure. Whereas the original stylesheets each formatted the entire finding aid, the code in the new stylesheets comes in two parts. One file formats the high-level elements. A second file formats the description of subordinate components, <dsc>. This has been done for several reasons. The first edition stylesheets were extremely long (over 5,000 lines) and therefore difficult to interpret and modify. They included five options for the <dsc>. Given the complexity and variations in the way the information in the <dsc> may be recorded and presented, it seemed desirable to create more options for the user and at the same time to separate them out for easier interpretation and modification. As a result, the second edition offers more choices (thirteen) for the <dsc> and formats each option as a separate file.

5.2 Stylesheets for the High Level Elements in EAD

The second edition includes five new stylesheets that format the high level elements of the finding aid, that is everything except the description of subordinate components <dsc>. They are imaginatively called *Style 5, Style 6, Style 7, Style 8, and Style 9.* Additional stylesheets are supplied for the description of subordinate components. They are discussed in section 5.4.

5.2.1 Style 5

This new stylesheet (eadcbs5.xsl) formats the high level elements of the finding aid, that is everything except the description of subordinate components <dsc>. It is the functional twin of Style 1 (eadcbs1.xsl) in that it creates a table of contents at the head of the document in the manner of a book.

If you use Style 5, you must do two things.

You will have to select one of the stylesheets that controls the display of the <dsc>. There are thirteen options. You will need to select the one that matches your encoding and the style of presentation you want. To help you make that selection, the following section includes a review of the issues involved and lists the features of each of the stylesheet options.

Once that selection is made, you will need to link that stylesheet to eadcbs5.xsl. If you open the eadcbs5.xsl file in a text editor, you will find an <xsl:include> statement at the very end of the document. You will need to insert the file name of the dsc stylesheet you have chosen in that statement. If you select dsc5.xsl, the <xsl:include> statement will read as follows:

```
<xsl:include href="dsc5.xsl"/>
```

5.2.2 Style 6

This new stylesheet (eadcbs6.xsl) formats the high level elements of the finding aid, that is everything except the description of subordinate components <dsc>. It is the functional twin of Style 2 (eadcbs2.xsl) in that it creates a table of contents in an HTML table along the left side of the screen.

If you use Style 6, you must do two things.

You will have to select one of the stylesheets that controls the display of the <dsc>. There are thirteen options. You will need to select the one that matches your encoding and the style of presentation you want. To help you make that selection, the following section includes a review of the issues involved and lists the features of each of the stylesheet options.

Once that selection is made, you will need to link that stylesheet to eadcbs6.xsl. If you open the eadcbs6.xsl file in a text editor, you will find <xsl:include> statement at the very end of the document. You will need to insert the file name of the dsc stylesheet you have chosen in that statement. If you select dsc5.xsl, the <xsl:include> statement will read as follows:

```
<xsl:include href="dsc5.xsl"/>
```

5.2.3 Style 7

This new stylesheet (eadcbs7.xsl) formats the high level elements of the finding aid, that is everything except the description of subordinate components <dsc>. It is the functional twin of Style 3 (eadcbs3.xsl) in that it also creates a table of contents in an HTML frame along the left side of the screen. Internal hyperlinks connect this table of contents with the various sections of the body of the finding aid. The XSL transformation process described in Section 6 generates the three files needed to populate this frameset in a single step. This stylesheet is guaranteed to work only with the Saxon XSLT processor as it employs an extension to XSLT that is not available in other transformation software, notably MSXML. You must use Instant Saxon, version 6.5.2 or higher.

If you use Style 7, you must do two things.

You will have to select one of the stylesheets that controls the display of the <dsc>. There are thirteen options. You will need to select the one that matches your encoding and the style of presentation you want. To help you make that selection, the following section includes a review of the issues involved and lists the features of each of the stylesheet options.

Once that selection is made, you will need to link that stylesheet to eadcbs7.xsl. If you open the eadcbs7.xsl file in a text editor, you will find <xsl:include> statement at the very end of the document. You will need to insert the file name of the dsc stylesheet you have chosen in that statement. If you select dsc5.xsl, the <xsl:include> statement will read as follows:

```
<xsl:include href="dsc5.xsl"/>
```

5.2.4 Style 8

This new stylesheet (eadcbs8.xsl) formats the high level elements of the finding aid, that is everything except the description of subordinate components <dsc>. It is the functional twin of Style 4 (eadcbs4.xsl) in that it does not contain either a table of contents nor internal or external links. Rather, it is intended to generate an HTML file that can be loaded directly into a word processor to create a nicely formatted print version of the finding aid.

If you use Style 8, you must do two things.

You will have to select one of the stylesheets that controls the display of the <dsc>. There are thirteen options. You will need to select the one that matches your encoding and the style of presentation you want. To help you make that selection, the following section includes a review of the issues involved and lists the features of each of the stylesheet options.

Once that selection is made, you will need to link that stylesheet to eadcbs8.xsl. If you open the eadcbs8.xsl file in a text editor, you will find <xsl:include> statement at the very end of the document. You will need to insert the file name of the dsc stylesheet you have chosen in that statement. If you select dsc5.xsl, the <xsl:include> statement will read as follows:

```
<xsl:include href="dsc5.xsl"/>
```

5.2.5 Style 9

This new stylesheet (eadcbs9.xsl) is different from the other new stylesheets in that it formats the entire finding aid in a single file. For the high-level elements, it behaves exactly like Style 6. However, it also includes the formatting for the <dsc> rather than implementing the formatting of that text by calling a separate file with an <xsl:include> statement.

It does this to accommodate a very common occurrence that many archives encounter and which cannot be handled simply by separate stylesheets. Many institutions typically have two situations with the <dsc>. In some finding aids, the <c01> element describes an entire series and not have associated container information. In others, the c01 describes a file with container information. The former situation is handled by the <dsc> stylesheets 4-7 and the latter in <dsc> stylesheets 8-11 (see section 5.4 for more detail).

Rather than have to employ two stylesheets, one for each situation, it may be desirable to employ a single stylesheet that accommodates both options. This is not technically possible with the <xsl:include> feature. So, to achieve this simplification, *Style 9* incorporates the text of eadcb6.xsl, dsc7.xsl, and dsc11.xsl in a single file. If the attribute @level="series" for <c01>, the formatting of dsc7.xsl is implemented. If @level="file" for <c01> or there is a <container> element associated with a <c01>, the formatting of dsc11.xsl is applied..

5.2.6 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.

5.3 Formatting Features in Style 5 through Style 9

This section describes some of the decisions about formatting that were incorporated into these stylesheets. Many are continuations of features found in the stylesheets for the first edition. It is worthy of mention once again that these are stylesheets cannot and do not accommodate every possible and valid EAD option.

5.3.1 <eadheader>

The <titleproper> element is the source for the text of the HTML <title> element produced by these transformations. Browsers typically display the contents of the <title> element in a line at the top of the screen and it is an element typically indexed by web search engines.

5.3.2 <eadid>

Style 7, like Style 3, creates a table of contents that is embedded in an HTML frame. A display of this type requires three files in HTML, a frameset that defines the overall presentation and the size of the two frames, and two the files that populate each of the content frames. In this case, the names of all three files have the same root, an alpha-numerical string suitable as a file name. For example, they might be 2468f.html, 2468t.html, and 2468b.html. The root is the string "2468". Because Stylesheet 7 uses the value of <eadid> to form the root, the value of <eadid> must be a string that is valid as a file name. Otherwise, a value suitable for a file name must be stored elsewhere in the document and the stylesheet altered to point to that location as the source of the file name. For example, this would be necessary if the repository uses a formal public identifier or a uniform resource name as the <eadid>.

5.3.2 Table of Contents

The table of contents is generated by displaying the value of the <head> element of selected elements in a finding aid. The new stylesheets automatically generate the links between the table of contents and the body of the document. It is not necessary, as it was with the first edition, to add prescribed @id attributes to these elements to produce the hyperlinks.

5.3.3 Order of Elements

The order of elements among the high-level elements in the finding aid, such as <bioghist> and <scopecontent> is prescribed by a rule in the stylesheet that begins <xsl:template match="archdesc">. A series of statements follow that begin either with <xsl:template…..> or <xsl:call-template…>. To change the order of display of these elements, simply reorder the statements in this list.

5.3.5 <did>

In the child elements of <archdesc><did> such as <origination> and <unittitle>, the stylesheets insert the value of any text in the label attribute. If a values are not encoded in label attributes, default text, specified in the stylesheet, is inserted. If one wishes to change the default values, simply replace the existing text with that which is desired.

While the stylesheets support the display of <unitdate> elements either as children of <did> or <unittitle>, the EAD instance must include any punctuation and separating whitespace between elements. The author has despaired of attempting to anticipate and accommodate within the stylesheet every suitable option an archives might employ.

5.3.6 Restrictions on Access and Use

These elements are significant to users though the distinction between the two may be apparent to most readers. For these reasons, these two elements are displayed together under a heading, "Restrictions", that is supplied by the stylesheet.

5.3.7 Arrangement

EAD 2002 conflated the elements <origination> and <arrangement> into a single Arrangement statement so as to be consistent with both ISAD(G) and long-standing archival terminology in the United States. The stylesheet inserts the heading "Arrangement", over-riding whatever appears in the <head> element of <arrangement>.

The <arrangement> statement may consist of narrative paragraphs or a listing of the titles of component series. In the latter situation, the stylesheet will endeavor to create links automatically between the series title statements in the list and the description of the appropriate series in the <dsc>.

5.3.8 Separated and Related Material

The second edition of ISAD(G) concatenated this data into a single element on the grounds that few archivists and even fewer users could (or cared to) distinguish between the two. In that spirit, this edition continues the practice of the first . The text of <relatedmaterial> and <separatedmaterial> elements appear as a series of separate paragraphs under a common heading, Related Materials. Encoded <head> elements do not display.

5.3.9 Administrative Information

The element <admininfo> was removed from EAD 2002 as a wrapper element. For the sake of continuity, however, the new stylesheets continue to display together all the elements that formerly nested within <admininfo> under a heading, "Administrative Information" that is supplied by the stylesheet. As previously noted, this does not include the elements for Restrictions on Access and Restrictions on Use.

5.4 Format Issues Relating to the Description of Subordinate Components

Before selecting the file that will transform the description of subordinate components as you wish it to appear, it is necessary first to review several aspects of this part of a finding aid in order to match institutional practices and preferences with the appropriate stylesheet. There is a great variety from repository to repository in the way this data is recorded and the way in which various archives wish to display it. These stylesheets attempt to accommodate a wide range of practice. In order to select the format best suited to your repository, it is necessary first to understand some of the variables in encoding and presentation that are addressed in the different stylesheet files.

In order to present this data in columnar form in HTML, it is necessary to create a table which consists of a grid of columns and rows since HTML has no concept of the use of tabs so common in word processors. A large part of creating the desired display of the <dsc> is a matter of formatting the table properly in order to place each data element of the description into the proper cell in the table. Sometimes the data has to span several columns and sometimes cells need to be left empty to create the proper horizontal spacing on the screen or page. Much of the encoding in this part of the stylesheet is taken up with the problem of getting these mundane details correct. Most of the differences among the 15 stylesheets for the <dsc> are no more than variations in spacing caused by different content in different descriptions, as described in the following sections.

5.4.1. Number of Columns.

An archival description typically presents information about the packaging of files and items by citing some combination of box or folder numbers, or other housing information. This data, often shown in columnar form, may consist of either a single number, for example a box number, a folder number or a combined box and folder number, or a pair of numbers such as a box number and a folder number. In the EAD encoding, this data is found in one or more <container> elements associated with a given component. Subsequently, stylesheets for the <dsc> must accommodate a columnar display of at least one, two, or zero container numbers. There are Cookbook stylesheets for all three scenarios.

5.4.2. Display of Container Numbers

Numbers designating containers often repeat from component to component. Several folders may be in single box. Some institutions always want the box number to display.

1	1	1957-1961
1	2	1962-1965
1	3	1966
1	4	1967-1970

On the other hand, many repositories have chosen not to show the box number for a component when it repeats from file to file. One typical approach has been to display box numbers only when a new number appears. The display could appear like this.

1	1	1957-1961
	2	1962-1965
2	1	1966
	2	1967-1970
	3	1971-1973

With EAD encoding, this effect may be achieved in either of two ways. The markup may include only the container values to be displayed. In the example immediately above, only the first instance of each box number would be added to the encoding. Alternatively, and perhaps preferably, all the box numbers could be encoded and the display of those that repeat could be suppressed by the stylesheet.

Given these variations, there are three options: all containers encoded and shown, all containers encoded but some not displayed, and only some containers encoded but with all those that are there being displayed. One set of stylesheets accommodates the first and third options while another set is written to suppress repeating container values.

5.4.3. Column Headings

A repository may include labels like "Box" and "Folder" at the top of the columns listing the numbers for those containers.

Box	**Folder**	
1	1	1957-1961
1	2	1962-1965

Such text may be specifically embedded in the EAD encoding through the use of a <thead> element in this fashion.

```
<thead>
    <row>
        <entry>Box</entry>
        <entry>Folder</entry>
    </row>
<thead>
```

Some of the Cookbook stylesheets are written to interpret that markup and generate the correct display.

It is also possible to encode the data for such column headings in your EAD instance using the label attribute for specific <container> elements but the Cookbook does not address that approach.

On the other hand, a repository may wish to avoid the overhead of such encoding and instead generate the labels "on the fly", that is as the XSLT transformation occurs. While one may instruct the stylesheet always to include standard, default text such as the words "Box" and "Folder", greater flexibility is possible if the stylesheet instead displays text specific to the individual finding aid.

Some of the Cookbook stylesheets take this approach and display the content of the attribute @type for the respective <container> elements as a heading. For example, if the encoding is <container type="Box">, the word "Box" appears as the column heading. Specifically, these stylesheets insert new column headings whenever the value of the first container (typically the box number) is not the same as the previous one or when the value of @type changes, as from "Box" to "Drawer". You may wish to begin the attribute value with an uppercase letter for display purposes. The EAD version 1.0 to EAD 2002 stylesheet automatically converts the initial letter of @type to uppercase.

5.4.4. Level of the Component

For some finding aids, the component level <c01> may represent a series description. The four stylesheets from the first edition, Style 1 through Style 4, all assume that <c01> is a series description without associated container numbers and indent the text accordingly. New stylesheets dsc2 through dsc7 all operate in the same way. They

assume that a <c01> contains a description of a series, subseries, subgroup, or subcollection that does not have container information associated with it dierectly.

Another common situation occurs when the description of subordinate components is strictly a file listing with container numbers associated with every level. The stylesheets dsc8 through dsc15 are optimized for that scenario.

For complex collections, <c01>, <c02>, and even <c03> may represents sub-groups, sub-collections, series, or subseries that do not have associated container values at those levels. The stylesheets dsc14 and dsc15 format such descriptions. The number of options for the proper placement of data becomes almost impossibly complex when container numbers are displayed along the left side of the page or screen and the descriptions in <c02> and <c03> sometimes do and sometimes don't have associated container data. To simplify the display options and to de-emphasize the packaging aspect of description, these two stylesheets place the container data on the right side of the screen or page.

5.5 Formatting the <dsc>: Applying the Analysis

Multiple permutations in display are possible with the four variables discussed in the previous section. The following table indicates how each <dsc> stylesheet accommodates them. The first column shows the number of columns in the output for the display of container numbers. The second column indicates whether all encoded container numbers display or if the first listed container for a component is suppressed if it has the same value as a preceding one. The third column indicates whether columns heads are supplied by the explicit encoding of a <thead> element or if they are to be derived by the stylesheet from the @type attribute of the container.

A detailed description of the behavior of each of these stylesheets follows. This same information also appears as comments at the beginning of each stylesheet file.

5.5.1 Overview

5.5.1.1 The stylesheet dsc1.xsl formats component lists that have no container numbers given.

5.5.1.2 Files dsc2.xsl, dsc3.xsl, dsc8.xsl, and dsc9.xsl format components with a single container number, e.g. only a box number or a folder number or a combined box/folder number. The options dsc2.xsl and dsc8.xsl insert the value of <thead> elements whereas dsc3.xsl and dsc9.xsl insert, as headings, the contents of container @type.

5.5.1.3 The files dsc4.xsl through dsc7.xsl and dsc10.xsl through dsc15.xsl all accommodate some variation of component lists with two container numbers. These are

subdivided into two groups.

5.5.1.4 The files dsc4.xsl through dsc7.xsl assume that a <c01> is a subgroup subcollection, series, or subseries description that will be displayed differently from other components. The identification, origination, title, date, and physical description elements appear on a separate line from other <did> elements. If the <c01> has a level attribute of "series", "subseries", "subgrp", or "subcollection", this data appears in bold face type, the title and date are displayed in the table of contents, and a link is created between that entry and the appropriate location in the <dsc>. Them same applies to any <c02> with one of these @level types. The insertion of column headings and the display of repeating first container values may be affected as well. When converted to HTML, each <c01> is embedded in a separate table.

5.5.1.5 In many finding aids, the <c01> level component describes a filing unit with associated container numbers, rather than an entire series. In this case, the files dsc8.xsl through dsc13.xsl will be preferred to the corresponding files dsc2.xsl through dsc7.xsl. Here, the <c01> <did> elements do not appear in bold, no entry is created for it in the table of contents, and space is left for container numbers. These stylesheets will produce more compact HTML files upon transformation as the entire <dsc> forms a single HTML table. Paradoxically, browsers may find it difficult to process very large finding aids done this way if the resulting table becomes too large. Users are advised to test the output of their transformations.

5.5.1.6 The final two <dsc> stylesheets, dsc14.xsl and dsc 15.xsl, are structured very differently from the previous thirteen. They are designed for very large and complex collections where the <c01>, <c02>, and <c03> levels may represent subgroups, sub-collections, series, or subseries. Formatting is adjusted to present more levels in the descriptive hierarchy that do not have associated containers, as may be the case with sub-groups, sub-collections, series, and sub-series. Moreover, containers are displayed on the right side of the screen or page so as not to emphasize storage locations at the expense of making clear the hierarchical structure of the arrangement of the materials. All encoded container values are presented. Finally, both <c01> and <c02> elements are added to the table of contents if they are encoded with a @level of subgrp, subcollection, series, or sub-series.

In dsc14.xsl, the headings for the columns displaying container numbers are supplied by use of the <thead> element. In dsc15.xsl, the stylesheet inserts headings for the first component that has a container value, before every seventh one, and every time the container @type value changes.

5.5.2 Tabular summary of dsc stylesheets

Stylesheet	Column Number	Container display	Column headings
dsc1.xsl	0	NA	NA
dsc2.xsl	1	Display all	Encoded
dsc3.xsl	1	Display all	Derived
dsc4.xsl	2	Display all	Derived
dsc5.xsl	2	Display all	Encoded
dsc6.xsl	2	Suppress repeats	Derived
dsc7.xsl	2	Suppress repeats	Encoded
dsc8.xsl	1	Display all	Encoded
dsc9.xsl	1	Display all	Derived
dsc10.xsl	2	Display all	Derived
dsc11.xsl	2	Display all	Encoded
dsc12.xsl	2	Suppress repeats	Derived
dsc13.xsl	2	Suppress repeats	Encoded
dsc14.xsl	2	Display all	Derived
dsc15.xsl	2	Display all	Encoded

5.5.3 Formatting Features in Stylesheets dsc1.xsl through dsc13.xsl

This section describes some of the general decisions about formatting that were incorporated into the stylesheets for the <dsc>.

5.5.3.1 <did>

The stylesheets accommodates all of the <did> elements, with a select group appearing on a first separate line (see section 5.5.1.4 for details). The remaining <did> elements each appears on a separate line, indented two columns from the title and date statements.

5.5.3.2 Other Supported Elements

Because of wide range of data elements possible within the <dsc>, the stylesheets only support those most likely to appear. These include <scopecontent>, <bioghist>, <accessrestrict>, <userestrict>, <note>, <odd>, <arrangement>, <descgrp>, <processinfo>, <acqinfo>, <custodhist>, and <controllaccess>. Each of these elements and each of its children appears on a separate line, indented two columns from the title and date statements. Any <head> element appears in bold.

Several generic formatting elements- <table>, <list>, and <chronlist> are also supported within components but must be embedded within <p> elements to assure their proper indention.

5.6 Transformation Details for Particular <dsc> Styleheets

5.6.1 dsc1.xsl

This stylesheet formats the dsc portion of a finding aid where components have no container element of any type.

5.6.2 dsc2.xsl

This stylesheet formats the dsc portion of a finding aid where components have a single container element of any type.

It assumes that c01 and optionally <c02> is a high-level description such as a series, subseries, subgroup, or subcollection and does not have container elements associated with it. It differs from dsc8.xsl only in this respect.

The position and text of column headings are determined by the presence of <thead> elements encoded in the description.

The content of any and all container elements is always displayed.

5.6.3 dsc3.xsl

This stylesheet formats the dsc portion of a finding aid where components have a single container element of any type.

It assumes that c01 and optionally <c02> is a high-level description such as a series, subseries, subgroup or subcollection and does not have container elements associated with it. It differs from dsc9.xsl only in this respect.

Column headings for containers are inserted whenever the content of a container's type attribute differs from that of the container in the preceding component.

The text of a column heading is taken from the type attribute of the container element.

The content of any and all container elements is always displayed.

5.6.4 dsc4.xsl

This stylesheet formats the dsc portion of a finding aid where components that have 2 container elements of any type.

It assumes that c01 and optionally <c02> is a high-level description such as a series, subseries, subgroup or subcollection and does not have container elements associated with it. However, it does accommodate situations where there a <c01> that is a file is occasionally interspersed. However, if <c01> is always a file, use dsc10.xsl instead.

Column headings for containers are displayed when either the content or the @type of a component's first container differs from that of the comparable container in the preceding component. The text of column headings are taken from the @type attribute of the container elements.

The content of any and all container elements is always displayed.

5.6.5 dsc5.xsl

This stylesheet formats the dsc portion of a finding aid where components that have 2 container elements of any type.

It assumes that c01 and optionally <c02> is a high-level description such as a series, subseries, subgroup or subcollection and does not have container elements associated with it. It does accommodate situations where there a <c01> that is a file is occasionally interspersed. However, if <c01> is always a file, use dsc11.xsl instead.

The position and text of column headings are determined by the presence of <thead> elements encoded in the description.

The content of any and all container elements is always displayed.

5.6.6 dsc6.xsl

This stylesheet formats the dsc portion of a finding aid where components that have 2 container elements of any type.

It assumes that c01 and optionally <c02> is a high-level description such as a series, subgroup or subcollection and does not have container elements associated with it. However, it does accommodate situations where there a <c01> that describes a file is occasionally interspersed. However, if <c01> is always a file, use dsc10.xsl instead.

Column headings for containers are displayed when either the content or the @type of a component's first container differs from that of the comparable container in the preceding component.

The text of column headings is taken from the @type attribute of the container elements.

The content of the initial container element for a component is displayed whenever a new column head is inserted.

5.6.7 dsc7.xsl

This stylesheet formats the dsc portion of a finding aid where components that have 2 container elements of any type.

It assumes that c01 and optionally <c02> is a high-level description such as a series, subgroup or subcollection and does not have container elements associated with it. It does accommodate situations where there a <c01> that is a file is occasionally interspersed. However, if <c01> is always a file, use dsc11.xsl instead.

The position and text of column headings are determined by the presence of <thead> elements encoded in the description.

The content of initial container elements is displayed when either the content or the @type of a component's first container differs from that of the comparable container in the preceding component.

5.6.8 dsc8.xsl

This stylesheet formats the dsc portion of a finding aid where components have a single

container element of any type.

It differs from dsc2.xsl only in that it assumes that <c01> describes a file that has an associated container rather than a series or other high level unit. This difference affects the placement of text in the proper column in the display table.

The position and text of column headings are determined by the presence of <thead> elements encoded in the finding aid.

The content of any and all container elements is always displayed.

5.6.9 dsc9.xsl

This stylesheet formats the dsc portion of a finding aid where components have a single container element of any type.

It differs from dsc3.xsl only in that it assumes that <c01> describes a file that has an associated container rather than a series or other high level unit. This difference affects the placement of text in the proper column in the display table.

Column headings are inserted for the first component, for every seventh one, and whenever a container's type is not the same as that of the container of the preceding component.

The text of the column head is taken from the container's @type.

The content of any and all container elements is always displayed.

5.6.10 dsc10.xsl

This stylesheet formats the dsc portion of a finding aid where components have 2 container elements of any type.

It assumes that <c01> contains a description of a file.

Column headings for containers are displayed when either the value or the @type of a component's first container differs from that of the comparable container in the preceding component.

The text of column headings are taken from the @type attribute of the container elements.

The content of any and all initial container elements is displayed whenever a new column head is inserted.

5.6.11 dsc11.xsl

This stylesheet formats the dsc portion of a finding aid where components have 2 container elements of any type.

It assumes that <c01> contains a description of a file.

The position and text of column headings are determined by the presence of <thead> elements encoded in the description.

The value of the initial container element for a component is displayed when either the content or the @type of a component's first container differs from that of the comparable container in the preceding component.

5.6.12 dsc12.xsl

This stylesheet formats the dsc portion of a finding aid where components have 2 container elements of any type.

It assumes that <c01> contains a description of a file.

Column headings for containers are displayed when either the content or the @type of a component's first container differs from that of the comparable container in the preceding component.

The text of column headings is taken from the @type attribute of the container elements.

The content of the initial container element for a component is displayed whenever a new column head is inserted.

5.6.13 dsc13.xsl

This stylesheet formats the dsc portion of a finding aid where components have 2 container elements of any type.

It assumes that <c01> contains a description of a file.

The position and text of column headings are determined by the presence of <thead> elements encoded in the description.

The content of initial container elements is displayed when either the content or the @type of a component's first container differs from that of the comparable container in the preceding component.

5.6.14 dsc14.xsl

This stylesheet formats the dsc portion of a finding aid where components that have 2 container elements of any type.

It assumes that c01 and optionally <c02> and <c03 contain a description of a series, subseries, subgroup or subcollection.

Column headings for containers are displayed when either the content or the @type of a component's first container differs from that of the comparable container in the preceding component.

The text of column headings are taken from the @type attribute of the container elements.

The content of any and all container elements is always displayed and appears along the right side of the screen or page.

5.6.15 dsc15.xsl

This stylesheet formats the dsc portion of a finding aid where components that have 2

container elements of any type.

It assumes that c01 and optionally <c02> and <c03> contain a description of a series, subseries, subgroup or subcollection.

The position and text of column headings are determined by the presence of <thead> elements encoded in the description.

The content of any and all container elements is always displayed and appears along the right side of the screen or page..

5.7 Transformation Performance

Tests with both MSXML and Saxon reveal that the process of transformation from XML to HTML takes considerably longer when a stylesheet is asked to suppress container values or to supply column headings. The more options one incorporates, the slower the transformation. For most finding aids, this is probably not an issue to be concerned with but it may be a problem for very large finding aids. When the size of the EAD XML file approaches one megabyte, processing is notably slower. On the other hand, one might question the usability and transportability of an HTML file of this size. Institutions may find it preferable to break up very large finding aids into separate files for user convenience and speed of transport and display over the Internet.

Section 6: Transforming XML Files

Your EAD files in XML encoding can be delivered to your patrons in two ways. They may be transformed into HTML so that they can be viewed on the web by standard browsers. They can also be transformed directly into print form.

6.1 Transforming XML files into HTML.

The stylesheets described in the previous section 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". While the newest generation of web browsers can perform transformations on the user's computer, the Cookbook describes server-based transformation, currently the option of choice in the XML community.

This section describes three widely used conversion programs. All are free. Saxon was created and is maintained by Michael Kay, one of the developers of XSLT. For many, it is the "gold standard" of transformation software. MSXML from Microsoft is very widely employed and is reputedly the "fastest" of the group. Xalan from Apache is often employed because of its connection with its popular web server software.

6.2 Batch-mode Transformation

Transformation into HTML may be done in advance of a user's request.

6.2.1 Saxon

This tool is a freeware product for non-commercial use. The current recommended version is 6.5.2 (later versions are experimental at this time). It comes in two flavors- Instant Saxon 6.5.2 is a small application suitable for Windows users, especially those with the Microsoft Virtual Machine version of Java (Windows operating systems prior to XP). If you have Windows XP or wish much faster performance, select the regular edition of 6.5.2 which is much larger set of files but which includes the Sun Java Runtime Environment. Either package is available at

saxon.sourceforge.com

6.2.1.2 Installation

Either version of Saxon will require a tool to unzip the files such as WinZip or PKZip.

6.2.1.3 Running Saxon

Saxon may be invoked from the DOS command line or the Windows Run command. While other options are available and documented at the site listed above, the basic command structure to perform a transformation follows this format-

Saxon *source file name stylesheet file name* –O *output file name*

6.2.2 MSXML

Microsoft distributes a free XSLT engine called MSXML with its Internet Explorer 6.0 browser software and with the Windows XP operating system. If you have IE 5.0 or 5.5, you will need to upgrade selected software files. See section 4.1.1 for details. The MSXML software can be invoked in four ways.

6.2.2.1. Within the browser.

Internet Explorer will transform an XML file when it loads the file into the browser if the XML document includes an XML processing instruction that specifies the name and computer location of a stylesheet. This processing instruction is inserted at the beginning of the EAD document directly after the XML declaration. The first two lines of the file will read like the following example, where @href specifies the file name of the stylesheet and the path to it.

```
<?xml version 1.0?>
<?xml-stylesheet type="text/xml" href="feith.xsl"?>
```

6.2.2.2 msxsl.exe

To facilitate use of the transformation engine separately from the browser, Microsoft distributes a small utility called msxsl.exe that enable the user to perform the transformation from the DOS command line (or the Run feature in more recent version of Windows). This utility and instructions for its use may be downloaded from

www.microsoft.com/xml.

6.2.2.3 MSXML within XMetaL

The XMetaL software can perform a transformation directly within the application using the MSXML software. This requires the presence of version 5.0 or higher of the Microsoft Internet Explorer browser and the proper version of MSXML. See section 4.1.1. for details on how to install the proper files.

This transformation feature is part of the **Print Preview** option. To execute a transformation, select **Page Preview** from the **View** option from the menu bar. The

transformation will occur and the resulting file will be displayed in a browser that will open up within the XMetaL application. Each time XMetaL performs a transformation, the resulting HTML file is saved in the Documents folder under a system assigned name. That HTML file can be subsequently renamed and used for public access. XMetaL cannot be used with Style 7 which create frames using extensions to XSLT that are not supported by MSXML

To use MSXML in this way, you will need to specify to XMetaL the name and location of the stylesheet you wish to use for the transformation. This information is stored in the file called "ead.mcr" that also contains the code for the keyboard macro files previously described. The default location for the stylesheet file itself is the Display folder for XMetaL.

The default value in the "ead.mcr" file distributed with Cookbook is the eadcbs6.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 in 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\\eadcbs6.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.

Each time you wish to use a different stylesheet, you will have to edit the ead.mcr file to change the name of the stylesheet to be invoked.

6.2.3 Xalan

An XSL processing suite for the Apache web server, Xalan is available from The Apache XML Project. Like Saxon and MSXSL, it can be run from the command line. For further information, consult

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

6.3. Transformation on the server

All three of these applications, MSXML, Saxon, and Xalan, can also be used to perform tranformations on a web server, as the user requests a particular EAD file. This so-called server-side transformation requires specialized programming skills. You will need to work with your systems or web services administrator to configure the server to perform this conversion.

6.4 Transforming EAD Files into Print.

Any of four methods may be employed for creating print output from your EAD-encoded files. Each of these is described briefly below though the technical requirements to accomplish these are beyond the scope of the Cookbook

6.4.1 Transform the file to HTML and import the HTML into a word processor.

Style 8 is designed to create an HTML file that has neither a table of contents nor any linking data. The HTML output file may be loaded into a word processor and manipulated to produce print output.

6.4.2 Generate a Postscript Data File (pdf)

A stylesheet written in the syntax of XSL Format Objects (XSL-FO) can transform an EAD file into an XML file with XSL-FO formatting codes. This XML file can then be processed by a specialized piece of software called an FO processor which transforms the XML file into PDF format for printing.

6.4.3. Generate an Rich Text Format (RTF) file

A stylesheet written in the syntax of XSL Format Objects (XSL-FO) can transform an EAD file into an XML file with XSL-FO formatting codes. This XML file can then be processed by a specialized piece of software called an FO processor which transforms the XML file into RTF format for printing.

6.4.4 Generate a Microsoft Word file in XML.

Microsoft has created an XML schema that represents the syntax of Word documents. An XSL stylesheet can transform a finding aid in XML from the EAD format into another XML file in the Word format. It appears that Microsoft Office 2003 will be able to read the resulting XML file directly in Word as if it were a native Word document.

Section 7: 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.

7.1 Links from Web Pages

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

- Embed an HTML pointer to the finding aid file in the source document. This requires a basic knowledge of HTML coding. Typically, one uses the HTML anchor <a> element with an HREF attribute to accomplish this hyperlinking.
- 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.

7.2 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 whose underlying software that can generate 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.

7.3 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 including:

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