Implementing EAD3: Conversion and Migration

Prepared by EAD3 Study Group 1: Conversion and Migration
Elizabeth Dunham, Arizona State University (Chair)
Christy Tomecek, Yale University
Dave Mayo, Harvard University
Sue Luftschein, University of Southern California
Michael J. Fox, Retired
Christine DeCatanzaro, Georgia Institute of Technology

Released 2017 May 23

Introduction

The Study Group chose to organize this report conceptually as opposed to examining each EAD2002 element individually in order to maximize readability and minimize repetitious analysis of similar elements and issues. We established our conceptual framework by grouping EAD2002's elements together based on their functions and their potential to be used together (or, in some cases, the requirement that they be used together). We then examined these groups in order to determine how their components could best be migrated to EAD3, including identifying necessary data remediation and potential migration issues. This document is not intended to cover every possible migration path for every element; rather, it aims to provide an overview of common strategies and to highlight the issues most likely to complicate or completely prevent migration. The strategies and issues discussed are applicable only to well-structured, DACS-compliant EAD2002 documents. In cases where the source EAD2002 does not meet these standards, the migrating repository will need to undertake pre-migration cleanup work in order to bring the guides into compliance.

In preparing this report, the Study Group made extensive use of the EAD2002 tag library (available at https://www.loc.gov/ead/tglib/index.html), the EAD3 tag library (available at https://www.loc.gov/ead/EAD3taglib/index.html), and the XSLT EAD2002 to EAD3 migration stylesheets created by the Society of American Archivists Technical Subcommittee on Encoded Archival Standards (TS-EAS; available at https://github.com/SAA-SDT/EAD2002toEAD3). As of this writing, four versions of the migration stylesheet are available. EAD2002ToEAD3dtd.xsl and EAD2002ToEAD3dtd_undeprecated.xsl migrate DTD-valid EAD2002 documents to EAD3. EAD2002ToEAD3dtd.xsl remediates elements deprecated in EAD3 while EAD2002ToEAD3dtd_undeprecated.xsl retains them. Similarly, EAD2002ToEAD3schema.xsl and EAD2002ToEAD3schema_undeprecated.xsl migrate schema-valid EAD2002 documents to EAD3 with EAD2002ToEAD3schema.xsl remediating deprecated elements and

EAD2002ToEAD3schema_undeprecated.xsl retaining them. It must be noted that in some cases, especially those where there is no obvious forward migration path, these stylesheets will remediate deprecated elements by removing them completely and inserting a note recording the field's original contents. The Study Group has not attempted to discuss all possible TS-EAS stylesheet behaviors and recommends that the migrating repository test their selected stylesheet thoroughly against local EAD2002 instances before beginning a full migration. Unless otherwise noted, "migration stylesheet" refers to either EAD2002ToEAD3dtd.xsl or EAD2002ToEAD3schema.xsl in this document.

Some of the migration paths discussed in this report require customizing migration tools and workflows. The majority of this customization can be achieved by editing a TS-EAS migration stylesheet, creating a new stylesheet to handle pre- and/or post-processing work, or scripting. If the technological expertise necessary to conduct this work is not available, the migrating repository will need to select another migration path, which may result in data loss or substantial manual remediation pre- and/or post-migration. Thus, it is strongly recommended that repositories plan and test their migration path(s) carefully prior to undertaking a full migration in order to ensure that all relevant data migrates as expected.

Repositories may also wish to consider using migration as an opportunity to restructure existing EAD2002 data. For example, this document describes several instances where there is no good forward migration path for a particular element configuration. In these cases, repositories may wish to reevaluate whether the affected information is necessary to the document and, if so, how it should be represented going forward. Should the repository decide to retain the information, it should carefully investigate how much work will be required to remediate the issues and whether a stopgap solution is available (for example, retaining a deprecated element in EAD3 until further corrective work can be completed) before committing to a full migration.

Finally, repositories should note that many of the migration paths described in this document result in EAD3 guides that are valid and useful but not as well structured as they could and likely should be. In some cases, substantial time and effort may be required to bring these guides into compliance with the repository's standards. Thus, it may be advisable to structure the migration as two projects instead of one, with the first project migrating EAD2002 guides to EAD3 and the second undertaking longer-term work to remediate the resulting EAD3 documents.

Table of Contents

Areas of Particular Concern and Opportunity

The Study Group considers the following elements and groups of elements to be most likely to impede or prevent migration from EAD2002 to EAD3 or to provide a significant opportunity to improve guides during migration. All are discussed in detail in their respective sections.

<archref> and <bibref>

Migration issues may arise surrounding these elements, which are often converted to <ref> in EAD3, in two scenarios. If <archref> or <bibref> were used to provide static citations in a configuration requiring migration to <ref>, they will have to be restructured either pre- or post-migration in order to map the affected data to a more appropriate tag. If these elements were used in some cases to provide static citation and in others to provide links, the migrating repository will need to determine what purpose each instance was intended to serve prior to migration. <archref> and <bibref> also have far fewer valid child elements in EAD3 than they did in EAD2002, which could lead to situations in which the migrated guide does not validate and has to be manually remediated. In both cases, a significant investment of time and effort, especially if use of these elements was not standardized locally, may be required.

<dao>, <ref>, and <ptr>

<dao> and its associated elements have been substantially simplified, including reducing the number of elements they can be nested in, in EAD3. When <dao> was used outside of the container list, it is likely that retaining the element will require moving it up in the hierarchical tree during migration. This relocation, however, may strip it of context provided by its original parent element. Thus, the migrating repository will need to decide whether this relocation is acceptable or if it wishes to handle the data differently. In either case, pre- and/or post-migration cleanup, which can generally be accomplished by editing the migration stylesheet or scripting, may be necessary.

Because a number of deprecated linking elements map to either <ref> or <ptr>, some migration scenarios result in guides where <ref> appears as a child of <ref>. This configuration is invalid in EAD3. If a repository employed a configuration resulting in this error regularly, either pre-migration remediation to restructure the affected information or customizing the migration stylesheet is advisable. The stylesheet tends not to handle these situations well, especially when <archref> and/or <bibref> are nested within multiple levels and migrated to <ref>.

<date> and <unitdate>

EAD3 provides substantially more options for date encoding than did EAD2002. EAD2002's <date> and <unitdate> can usually be migrated directly to comparable EAD3 elements, but repositories that routinely used @normal in their date tags (or can add it easily) may wish to explore opportunities for migrating to one of EAD3's more granular date options.

<note>

<note>, which could be used as a child of numerous EAD2002 elements, has been deprecated in EAD3. Generally speaking, it has been replaced with <descriptivenote>, <didnote> or <footnote>. In some cases, the information originally contained in <note> will have to be moved in the hierarchical tree in order to produce a valid EAD3 guide, which could strip it of important context. If <note>'s use was standardized, it may be possible for the repository to either edit the migration stylesheet or create a script to handle this tag in a manner appropriate to local usage. If <note>'s use was not standardized, the repository may need to either invest in manual cleanup or discard the affected information.

<physdesc>

While EAD2002's <physdesc> can be migrated directly to EAD3's <physdesc>, the migrating repository should consider this element's path carefully prior to committing to this mapping. EAD3 encourages the use of <physdescstructured> rather than <physdesc> and although there is no direct migration path from <physdesc> to <physdescstructured>, it may be possible to partially automate this migration or identify instances where it would be beneficial to convert <physdesc> to <physdescstructured> manually post-migration. These decisions should be made prior to the start of the migration in order to accurately estimate how much remediation work will be required.

Section 1: Container and Container List Tags

1.1: Overview

Container list tags have changed very little from EAD2002 to EAD3. Most major changes involve complications of restructured child elements, such as <archref> and <bibref>. Other possible issues relate to EAD3's standardization of common practices with elements such as <dao>. Finally, possible issues exist in cases where an element or attribute has undergone semantic/use changes but has not been renamed. This situation may require the repository to devise their own migration strategy beyond the migration stylesheet to meet local needs.

1.2: <c> and <c01-c12>

Migration Strategy and Concerns

Data within <c> and <c01-c12> tags can be migrated directly from EAD2002 to EAD3 without intervention in many cases. One of the few concerns is the use of <dao> outside of <did> within components, as <dao> tags can only be used within <did> or <daoset> tags in EAD3. As for the structure of <c> and <c01-c12> tags, there are few differences. The changes that have been made are mainly the addition of attributes @base, @lang, and @script, which provide granularity that was not previously available in EAD2002 and therefore would not be present in another form to migrate. The only attribute that is removed from EAD3 is @tpattern, which requires some amount of work to remediate.

Data Remediation Concerns

The only serious concern with data remediation arises if repositories routinely used @tpattern in their EAD2002 guides. The migration stylesheet will strip out @tpattern entirely, so repositories will have to adjust their display stylesheet to create the format they want without using any references to @tpattern. If <dao> does not appear within <did>, the migration stylesheet will move <dao> into the correct place. See the Section 9.2 for more detailed information on <dao>.

1.3: <container>

Migration Strategy and concerns

Data within <container> tags can be migrated directly from EAD2002 to EAD3 without intervention in many cases. All attributes extant in EAD2002 have been retained in some form in EAD3. The only major change is the renaming of @type to @localtype. Some attributes have been added to <container>, which offers granularity not previously available in EAD2002. The data that would belong in those fields would not have to be migrated from another field in most cases. However, there is a serious consideration when migrating if there are local uses of @label that would be more appropriate for the new attribute @containerid. This is especially worth attention as two of the most popular archival data management systems, Archivists'

Toolkit and ArchivesSpace, currently put barcode information in the @label attribute. The stylesheet does not change or remove any values in @label, as it is still valid in EAD3, so if @label information is used in the final styled guide released by the repository, the repository will have to consider whether or not they wish to continue leaving that information in the @label attribute as a matter of local practice, or to utilize @containerid and remediate data accordingly.

<extptr>, <extref>, and <linkgrp> are no longer valid as children of <container> in EAD3. EAD3 streamlined linking elements so that linking information, whether external or internal, will use the same tags. The migration stylesheet converts the first two elements to <ptr> and <ref> and removes <linkgrp>, leaving behind its child elements and data. None of the information that is present in those elements is lost. The other three child elements, <archref>, <bibref>, and <title>, are still present in EAD3 but are no longer valid within <container>. <archref> and <bibref> elements will be converted to <ref>, which will pose a problem if the data was used for static citation rather than linking. The stylesheet will strip <title> tags entirely from the guide, but the data within the tags will remain, becoming part of the text in the field. <title> is discussed further in Section 2.3; <archref> and <bibref> in Section 5.5; and <linkgrp> in Section 9.5.

Data remediation concerns

Almost no attribute values will have to be remediated for <container>. Values that are used in the @label attribute that would be properly assigned to the new @containerid require a choice by the institution. If @label was routinely used with barcodes or other identifying numbers rather than display data, it could be remediated by a local XSL stylesheet that migrates all @label attributes to @containerid. If @label is mixed between display information and identifying information, more exhaustive remediation, likely requiring hand editing, may be required. This may not be a viable option depending on the amount of available staff. However, if the repository only has a few guides that need remediation in one category, it could be manageable.

Child elements within <container> may require more remediation. Deprecated elements are easily remediated via the stylesheet offered by SAA. <archref> and <bibref>, however, will be more problematic if they're not only used for linking to another collection via a URI or @id. In that case, the information will have to be moved to a parent element it can reside in, such as <relatedmaterial> as a sibling of the <did> element that the <container> appears within, or completely removed from the inventory.

Section 2: <controlaccess> and Children

2.1: Overview

The vast majority of the changes to <controlaccess> and its children have to do with the addition of <part> in EAD3. <part> can be deployed in one of two ways: either one <part> can be used to wrap an entire data string, or multiple <part>s can be used to establish greater granularity. If a repository wishes to use multiple <part>s per data string routinely, substantially more cleanup (some of which could likely be automated if the data strings in question are well-structured) will be needed.

Additional issues arise in the case of <title>, which is not only affected by the addition of <part> but also by the simplification of linking elements in EAD3. Specifically, a number of linking attributes associated with <title> in EAD2002 are no longer valid in EAD3. Their function, however, can usually be replicated by adding a <ref> element with the relevant information to <title>/<part>.

2.2: <controlaccess>

Migration Strategies and Concerns

The only major difference between <controlaccess> in EAD2002 and <controlaccess> in EAD3 is that <address> can no longer be used as a child of <controlaccess> in EAD3. Most of the formatting associated with <address> can be preserved using <p>. For more information regarding <address>, consult Section 4.8.

Data Remediation Concerns

This element should require no remediation unless <address> was used as a child.

2.3: <title>

Migration Strategies and Concerns

Use of <title> is much more restrictive in EAD3 than in EAD2002. Specifically, <title> can no longer be used as a child of <bibliography>, <container>, <dimensions>, <emph>, <label>, <langmaterial>, <materialspec>, <origination>, <otherfindaid>, <physdesc>, <physloc>, <relatedmaterial>, <repository>, <separatedmaterial>, <unitdate>, and <unitid> in EAD3.

In many cases, when migrating <title> to an EAD3 element where <title> is no longer allowed as a subelement, both <title> and its contents can be preserved by adding elements (usually <p>) between the parent element and <title> or by adding attributes to <title>. In the cases of <container>, <dimensions>, <label>, <materialspec>, <physdesc>, <physloc>, <unitdate>, and <unitid>, <title>'s contents and formatting can be be retained by replacing <title> with <emph> and using <emph>'s @render element to replicate any formatting associated with <title>.

Alternatively, <title>'s contents can be inserted into the parent element as free text and the <title> tag discarded entirely. The migration stylesheet employs this technique. There is no clear migration path for instances where <title> was used as a subelement of <origination> or <repository>. In these cases, the migrating institution will need to review the affected tags individually and restructure the data appropriately or strip both <title> and its contents during migration, thus losing the information.

<title> is also affected by the simplification of the linking structure in EAD3. Specifically, @actuate, @arcrole, @entityref, @href, @linktype, @show, @title, and @xpointer, which were used in EAD2002 to enable linking, are no longer available and have no equivalent in EAD3. Much of this functionality can, however, be replicated by employing <ptr> as a subelement of <title>/<part>. This technique is the migration stylesheet's default behavior. Changes to @authfilenumber, @role, and @type and the addition of <part> as a mandatory subelement are discussed in the following section.

Data Remediation Concerns

The amount of data remediation necessary depends extensively on how and where <title> was used in EAD2002. If the migrating repository did not use <title> as a child of either <origination> or <repository>, post-migration data cleanup should be minimal to nonexistent. If <title> appears as a child of either of these elements, however, these instances should be reviewed and restructured prior to migration in order to ensure that the affected information is represented in the resulting guide. If this structure was used frequently, substantial data cleanup may be required.

2.4: <corpname>, <famname>, <function>, <genreform>, <geogname>, <name>, <occupation>, <persname>, and <subject>

Migration Strategies and Concerns

Changes to these elements are largely limited to the addition of EAD3's <part> subelement, a required child that has no equivalent in EAD2002. <part> can be deployed in two configurations: either one <part> can wrap the element's entire contents (the migration stylesheet's default behavior) or multiple <part>s can delineate individual components of the heading. If headings were consistently delimited in EAD2002 (for example, commas appear between sections of personal names or "--" was used to separate sections of subject headings) it may be possible for the migrating repository to customize their migration tool to add multiple <part> elements to a data string automatically. Whether or not this technique can be used effectively, however, will depend on how well structured the source EAD2002 is and how much technological support is available.

Unlike EAD2002, EAD3 does not allow these elements to be used as children of either <label> or <physdesc> and forbids using <emph>, <lb>, and <ptr> as subelements. In cases where one of these elements was used as a child of <label> or <physdesc>, the element can be deleted and its contents inserted into the parent element as free text, optionally formatted using

<emph>, with no loss of data. In all other cases, the addition of <part> will resolve issues associated with the use of <emph>, <lb>, and <ptr>.

EAD3 discontinues two attributes, @authfilenumber and @role, previously associated with these elements. The removal of @role does not affect <function>, <genreform>, and <occupation>, where it was not permitted in EAD2002; @type has been discontinued for <genreform>. Migration paths for all of these attributes are simple: @authfilenumber maps to @identifier; @role maps to @relator; and @type maps to @localtype. Thus, this change should not pose a major obstacle to migration.

Data Remediation Concerns
The amount of remediation required for these elements depends primarily on what use the migrating repository wishes to make of <part>. If using one <part> to wrap an entire data string is deemed acceptable, no post-migration cleanup will be required. If the migrating repository is able to customize their migration tool to add multiple <part> elements to individual data strings at a high level of accuracy, minimal post-migration cleanup will be necessary. If, however, the migrating repository elects to add <part> elements manually or can only partially automate the addition of <part>, substantial post-migration cleanup will be required.

Section 3: Date Elements

3.1: Overview

While it's certainly true for the entirety of a guide, date elements are especially proof that the more structured the metadata and data, the easier it will be to automate migration using the EAD2002 to EAD3 stylesheet! With the introduction of date elements such as <unitdatestructured>, <daterange>, <fromdate>, and <todate>, an EAD3 guide has the potential for more granularity with its dating than EAD2002, and thus more opportunities for machine readable operability. However, for the stylesheet to optimally function in creating these structured dates, attributes such as @normal will need to be present.

Another important thing to note with EAD3 date elements is that many of the original EAD2002 elements are still available, meaning that migrating between the standards does not mean that a repository will have to exhaustively retool all of their guides to create valid EAD3 guides. While a repository may be interested in taking advantage of the new structured elements, it could very well introduce them with the creation of new guides while only doing essential remediation of data for older guides to allow them to validate. These older guides could then be edited retroactively over a longer term.

3.2: <date>

Migration strategies and concerns
While <date> was only one of two date-related elements in EAD2002, the number of date

related tags has increased dramatically in EAD3 to enable greater granularity. <date> has not been deprecated, but its use within parent tags has been limited. <date> can also be used as a child of some EAD3 elements it could not be used in with EAD2002, specifically <abstract>, <archref>, and <bibref>.

<date> is no longer a valid child element of several elements due to the fact that those parents have been deprecated in EAD3. They include <change>, <creation>, <extref>, <extrefloc>, <imprint>, <refloc> and <titlepage>. Data within <creation> and <change> are now handled in <maintenanceevent> within <maintenancehistory>, which replaced <revisiondesc>. Date information in <maintenanceevent> is present in the new child element <eventdatetime>, which migrates directly with the EAD2002 to EAD3 stylesheet. <imprint> is stripped during migration, but <date> elements remain in the data and could continue to be useful, as both <bibref> and <unittitle> in EAD3 use <date> elements. <date> within <refloc> and <extrefloc> is painless to migrate via the stylesheet as the <date> elements are preserved in the new parent element, <ref>. <extref> requires more work in migration, however. <extref> does migrate directly to <ref>, which in previous examples involving –ref and –loc elements preserved the <date> elements. In the case of <extref>, <date> tags do not make it through the migration and so are left unencoded, requiring some post-migration cleanup to insert <date> tags back into the newly minted <ref> elements. <titlepage> and its data are completely removed, along with the entirety of <frontmatter>, as all of that information is repetitive of information generally present in <control>. Therefore, any date information present in <control> would likely cover what was present in <titlepage> and little or nothing would be required to salvage information that was present in that element.

The parent tags that have entirely gotten rid of <date> that are not deprecated, <subtitle>, <title>, <label>, <legalstatus>, and <physdesc>, will have the <date> tags completely stripped out of the guide using the EAD2002 to EAD3 transformation, leaving behind the date that the tags were encoding. If dates are necessary for <subtitle>, <label>, <legalstatus>, and <title> data in the repository's local practice but do not necessarily need to be encoded, it could be left in as part of the free text in those fields without the specific encoding, with minimal cleanup possibly being needed to fix spacing and style issues. If the repository wishes to have the date continue to be encoded and <date> is valid in the same parent element as <title>, automation via scripting may be useful in positioning the element at the same sibling level with <title>. Otherwise, serious consideration will need to be made about the repository's descriptive practices in terms of possibly mapping that information to a different field, such as <unitdate> if the <date> originally in <title> would be appropriate and valid there.

<date> elements that appeared in the EAD2002 version of <physdesc> can now only appear in <physfacet> within the new <physdescstructured> element. (It also could appear in the <physfacet> element in EAD2002, which was a child of the <physdesc> element.) Therefore, there are two possibilities for migration: either the repository allows the migration stylesheet to strip <date>, leaving its contents in EAD3's <physdesc> as free text, or it maps <date> to a <physfacet> child of <physdescstructured>. For more information about migrating

<physdesc>, consult Section 4.2.

The only former parent element that would need its <date> element migrated to either of the new date-related elements, <daterange> or <datesingle>, is <chronitem>. This process can be completely automated if <date> is structured with @normal. If @normal is not present, the stylesheet automatically converts it to <datesingle>, as it cannot recognize date ranges located in the data. If @normal is missing, it would take a considerable amount of work, either in pre-migration to add @normal, or to convert any appropriate <datesingle> elements to <daterange> after the migration.

Attributes within <date> are mostly the same with some additions. The exception to this rule is @localtype, which would be labelled @type in EAD2002. That attribute switch is easily automated as it is a direct translation. The other two new attributes, @lang and @script, were not present in EAD2002, nor was the information in those tags present within <date>.

Data remediation concerns
Since <date> has been removed from so many parent elements, whether due to new date encoding elements, deprecation of elements, or differing structure between EAD2002 and EAD3, the largest concern will be with migration strategies, as the structure of <date> itself has not changed much. The largest factor in migration with the data itself will be with how well structured the <date> elements in present EAD2002 guides are. A good example of this principle is the migration of <chronitem>'s <date> to either <datesingle> or <daterange>. The migration stylesheet does correctly transform items with the attribute @normal, including adding the new child elements <fromdate> and <todate>. However, if the repository did not include @normal in their date tags, all dates will be transformed to <datesingle>, without the child tags. Thus, pre-migration work to add the attribute is required. This can be done via scripting, but additional work by hand may be necessary. If there aren't resources for scripting to be developed, it would require a great deal of work to remediate by hand, which may not be feasible depending on the amount of guides in the repository.

3.3: <daterange>, <datesingle>, <fromdate>, and <todate>

Migration strategies and concerns
<daterange> and <datesingle> are new to EAD3. These elements are used primarily to encode dates in machine-readable-only fields rather than narrative fields. The only parent element in EAD2002 that would need its <date> element migrated to either <daterange> or <datesingle> is <chronitem>. As discussed in the <date> section, this work can be automated if <date> is structured with @normal. The remaining <daterange> and <datesingle> parent elements are new to EAD3, and the few that have equivalents in EAD2002 were not places where <date> appeared.

<fromdate> and <todate> are new elements created for their parent element <daterange>. They allow more granularity in denoting date information for elements that are slanted towards

being machine-readable. These elements appear in the parent element <daterange> and therefore the bulk of the concerns regarding this migration relate to the parent element that <daterange> appears in.

A larger concern in migrating these elements is the addition of <unitdatestructured>. (There will be more discussion of <unitdatestructured> in section 3.5, but this discussion will overlap). <unitdate> is an element present in both EAD2002 and EAD3 and can still be used in expressing dates in <c> elements. However, with the ability of providing more granularity in <unitdatestructured>, many institutions would likely want to take advantage of this new element. The primary concern in that is since <unitdate> is a valid element in EAD3, the stylesheet will not migrate it to <unitdatestructured>, no matter how well structured <unitdate> is. Therefore, in order to produce the child elements <daterange> and <datesingle>, some XSL work would need to be done, similar to how <date> to <chronitem> has been created in this stylesheet. This workflow will be discussed further in the <unitdatestructured> section.

There are fewer attributes of <daterange> and <datesingle> than there are for EAD2002's <date>. Attributes that have direct mappings from <date> to either of these elements will be transformed accordingly. Any that do not have a direct mapping will be dropped. If attributes without direct mappings are needed, @localtype could be used if the needed information can be denoted properly. This approach will require choosing which value is worth retaining as only one @localtype attribute can be used per element.

Data remediation concerns
As stated in the <date> section, <date> and <unitdate> tags will migrate best when the @normal attribute is present. The only other remediation concern is making description choices about attributes present in EAD2002 <date> but not in <daterange> or <datesingle> and without direct mappings to those valid attributes. Since only one value can be mapped to @localtype, the rest of the attributes will either have to be deleted or the needed attribute will have to placed in the element. If the decision is made that none of these attributes are necessary, all of them will have to be deleted.

3.4: <dateset>

Migration strategies and concerns
<dateset> is a new EAD3 wrapper element for expressing complex date structures. Because it is a new element to provide granularity and there is no like element in EAD2002, there is no way to use the migration stylesheet to add <dateset>. If repositories would like to add <dateset> to their migrated guides, it would have to be inserted by hand or partially automated through a series of scripts. This would depend on whether there was already an appropriate set of <daterange> and/or <datesingle> elements or on how the data is structured otherwise.

Date remediation concerns
Since this is a new element, again, the only data remediation that would be useful is adding

any structured attributes to preexisting <date> elements, like @normal, if needed. This will ease the process of adding <dateset>, but unfortunately it would only achieve partial automated remediation at best. While there are certainly automated methods of adding a parent element, the complexity of the element will likely require manual remediation as the logic of the <dateset> structures is likely to be unique to that particular object of description.

3.5: <unitdatestructured>

Migration concerns and strategies

<unitdatestructured> is a new element in EAD3. It was created to give date information present in <c> levels more granularity. It uses the child elements <daterange>, <datesingle>, and <dateset> rather than standalone date expressions. While this is very useful for guides originally created in EAD3, it poses difficulties for repositories who wish to migrate <unitdate> information to <unitdatestructured> since <unitdate> remains a valid element in EAD3.

The good news is that if data is well structured with @normal in <unitdate> and the repository desires every date in the guide to be migrated to <unitdatestructured>, the work can be automated easily through additional XSL, whether added to the migration stylesheet or created in a local stylesheet that is used after the initial transformation if not all guides need to be transformed in that manner. If @normal is not present, it will require some pre-migration work via scripting to add the attribute. If there are not enough resources to accommodate the scripting work, it will require a great deal of work to add the attribute by hand. If <unitdate> is a more appropriate element in parts of the same guide, this will require more focus, whether migrating in full as above and redoing the individual elements to <unitdate> if they are sparsely present, or if the opposite is true, fixing only the <unitdatestructured> appropriate elements. It may also be possible to create a script to change certain elements/nodes and leave others alone by designated positions, but this would have to be retooled for each guide. Since <unitdate> is still a valid element, however, it is possible to leave the element as is and slowly over time convert guides in a long-range project.

Date remediation concerns

In order to perform the smoothest migration for <unitdatestructured>, <unitdate>s present in the EAD2002 guide will need to be well-structured, especially with @normal attributes. Otherwise, any work to migrate will be severely hampered.

3.6: <unitdate>

Migration strategies and concerns

In EAD3, several of <unitdate>’s child elements have been removed and its use is confined to <did> only. With the migration stylesheet, if the <unitdate> is not a direct child of <did>, the element will be moved to that level. <unitdate> lost some child elements as well, including <archref>, <bibref>, <title>, <extptr>, <extref>, and <linkgrp>. As reported with the child elements in <date> that were valid in EAD2002 but not in EAD3, the migration stylesheet will strip the offending elements from the EAD, leaving the data alone in the <unitdate> elements.

While those elements would still be valid in EAD3 since <unitdate> is a free text element, it would likely be desirable to remove the data or map it to another field. For example, <title> may be better served within <unittitle> and <archref> and <bibref> would work better in a <bibliography> element within <c>. <extptr>, <extref>, and <linkgrp> have all been deprecated from EAD3, but <extptr> and <extref> migrate to <ptr> and <ref>, which are both valid within <unitdate>. <linkgrp> is stripped, leaving its valid child elements in place.

<unitdatestructured> is considered more desirable than <unitdate> due to its granularity and better encoding possibilities. As stated in <unitdatestructured>, it is important for @normal to be present in order to automate migration with XSL of the repository's making, whether it's added to the migration stylesheet or created as a local stylesheet. Another thing that needs to be considered are the children tags, <ptr> and <ref>, which are valid within <fromdate> and <todate>, meaning the stylesheet will need to copy that entire node and push it into the correct <fromdate> and/or <todate> elements. That being said, assuming that all <unitdates> should be migrated to <unitdatestructured> and have the structuring attributes in place, it should be relatively easy to develop local automated practices for migration.

Attributes within <unitdate> in EAD3 are identical to the ones available in EAD2002, with @type directly migrating to @unitdatetype. Two new attributes are added for granularity, @lang and @script, but since that data is not present in any other areas, there would be no data to directly migrate.

Data remediation concerns
As mentioned with all EAD2002 date-related elements, making sure <unitdate> is appropriately structured with @normal will help with all migrations. This can be achieved in an automated fashion via scripting. If there are no resources available for scripting, it will require a great deal of work by hand, which may not be feasible, depending on the amount of guides in the repository. Since <unitdate> does not require @normal, if the work does need to be done by hand, this remediation could be handled as a long term project.

Section 4: <did> Children

4.1: Overview

The two <did> children undergoing the most substantial changes between EAD2002 and EAD3 are <physdesc> and <langusage>. Because <physdesc> is much less granular in EAD2002 than is its equivalent in EAD3, this element is the most likely to cause substantial problems during migration. Thus, repositories should carefully consider their migration strategy and stringently test their migration process in order to avoid data loss or prohibitive amounts of data cleanup. Several migration strategies for this element are discussed below.

While EAD2002's <langusage> has been substantially restructured in EAD3, it should be possible to migrate data in this element relatively easily as long as the migrating repository has employed it consistently. Repositories may also wish to explore the opportunity to create new <script> subelements when migrating <langusage> either by using the value of @scriptcode or by automatically adding data in bulk. The remainder of the elements in this section are very similar in EAD2002 and EAD3 and should, with few exceptions, migrate with minimal difficulty.

It should be noted that a number of elements had child elements in EAD2002 that are no longer allowed in EAD3. Generally speaking, information in these elements can be migrated either to analogous elements in EAD3 or to a free-text field in the relevant EAD3 element. If the content of more than one child element is migrated to a free-text field, the migrating repository should ensure that appropriate formatting is added as needed in order to separate concepts.

4.2: <didnote> and <footnote>

Migration strategies and concerns

<note> has been deprecated in EAD3. To replace it are four different types of note-related elements, depending on what information needs to be conveyed: <controlnote>, <descriptivenote>, <didnote>, and <footnote>. Since <controlnote> and <descriptivenote> have very specific functions that would not be relevant in this section, the focus will be on <didnote> and <footnote>.

In EAD2002, <note> was a valid child element of almost all of <did>'s siblings. The corresponding EAD3 tags can be used in a much more limited fashion. The migration stylesheet handles <note> in one of two ways. Where <note> appears as a child of <entry>, <event>, <item>, , <archref>, <bibref>, <extref>, <extrefloc>, <ref>, or <refloc>, it is stripped of @encodinganalog and @label and converted to <footnote>. Where <note> appears as a child of <accessrestrict>, <accruals>, <acqinfo>, <altformavail>, <appraisal>, <arrangement>, <bibliography>, <bioghist>, <blockquote>, <controlaccess>, <custodhist>, <daodesc>, <dsc>, <fileplan>, <index>, <note>, <odd>, <originalsloc>, <otherfindaid>, <phystech>, <prefercite>, <processinfo>, <relatedmaterial>, <scopecontent>, <separatedmaterial>, <userestrict>, , or <titlepage>, it is stripped of @encodinganalog and @label and converted to <footnote /> .

These <note> automations may not make sense if the information in <note> is not specifically related to the field it's placed in, e.g. if it's information about the repository rather than information about the creator in <bioghist>. In those cases, it may be necessary to move that information to a <didnote> element or remove the <footnote> encoding and allow the information to stand alone in elements. The latter could be far more easily automated than the former, especially if this is a standard treatment of <note> information in a specific element, as the stylesheet could be updated to include this deletion or a local stylesheet prepared for these instances. Moving <note> to <didnote> would be more complicated, but would be

doable via scripting or a stylesheet depending on how focused the need is. If the repository does not have the resources to invest in script or XSL creation, this could be done manually, depending on the amount of guides requiring attention.

In certain cases, no version of <note> will be appropriate for the data. These exceptions will be discussed in the affected elements.

Data remediation concerns
Since all migration issues with <didnote> and <footnote> are about placement rather than structure of the data itself, there are no data remediation concerns.

4.3: <physdesc>

EAD2002 => EAD3 Migration Summary
The simplest <physdesc> migration path is from EAD2002's <physdesc> to EAD3's <physdesc>, an unstructured field used to provide a brief statement about the materials being described. This path is the migration stylesheet's default behavior. In some cases, sufficient structure may be present to partially automate migration from EAD2002's <physdesc> to EAD3's structured physical description elements, <physdescstructured> and <physdescset>. This approach requires considerably more effort because the repository will need to both create the necessary migration tool(s) and undertake substantial post-migration data cleanup. The choice between these two migration paths will be heavily informed by the amount and structure of physical description data present in the guides to be migrated and the amount of time and effort the repository is willing to devote to data cleanup.

Migration Strategies and Concerns
The majority of the migration concerns regarding <physdesc> stem from the fact that EAD3's physical description elements allow for substantially more granularity than did their counterparts in EAD2002. While <physdesc> can be structured in EAD2002, this structure is not required and is constructed in a much looser fashion than that mandated by EAD3's <phsydescstructured>. EAD3 includes a free-text <physdesc> field similar to that present in EAD2002 but strongly recommends using <phsydescstructured> and associated elements and attributes in order to facilitate consistent machine processing and data exchange.

Due to these structural differences, fully automating migration from <physdesc> to <physdescstructured> is essentially impossible. In order to automate fully, a repository would not only have had to consistently structure EAD2002 <physdesc> information using the four recommended subelements (<dimension>, <extent>, <genreform>, and <physfacet>) but must also have used @unit in <extent> to indicate the unit of measurement without including a free text description of this unit in the tag itself. Moreover, any information contained in subelements supported in EAD2002 but not EAD3 would have to be structured in such a way that the information could be migrated directly into EAD3's <descriptivenote> without requiring post-migration cleanup. Thus, while it is hypothetically possible to structure <physdesc>

extensively enough to fully automate migration to <physdescstructured>, it is unlikely that most, if any, repositories not only undertook such extensive encoding work but also structured the final <physdesc> element in a way compatible with fully automating EAD2002=>EAD3 migration.

Partially automating migration from <physdesc> to <physdescstructured> is feasible if structured data and/or standardized free text are available and the repository is willing to invest in post-migration data cleanup. If <physdesc> was structured using the recommended subelements (<dimension>, <extent>, <genreform>, and <physfacet>) in the EAD2002 guide, the migration can be partially automated by mapping these subelements to their approximate counterparts in EAD3 (<dimensions>, <quantity>, <unittype>, and <physfacet> respectively). Post-migration data cleanup will, however, be necessary in this instance.

Data Remediation Concerns

The amount of data remediation required post-migration depends largely on whether the migrating repository wishes to map physical description information to <physdesc> or to <physdescstructured>. If the repository considers <physdesc> suitable for their purposes, data cleanup should be minimal: if the EAD2002 to EAD3 stylesheet is used, <physdesc> elements in the resulting EAD3 guide will need to be reviewed to ensure that the text is formatted according to local standards and any necessary edits made, but no further remediation effort will be required. If <physdesc> use and structure was standardized in EAD2002, some of this cleanup (for example, ensuring spacing between information extracted from several subelements) can be automated either by editing the migration stylesheet or creating a local stylesheet or other tool to perform the necessary post-processing functions.

If the repository wishes to migrate to <physdescstructured>, however, data cleanup requirements will be much more substantial. If this migration is partially automated, cleanup will be necessary in the resulting <physdescstructured> tags to move any free text information and information contained in subelements supported in EAD2002 but not EAD3 into appropriate <physdescstructured> subelements. If the repository used <physdesc> sparingly and has a relatively small number of guides, this data cleanup may be manageable; if the repository has a large number of guides and/or routinely used <physdesc> at the component level, cleanup work will involve a substantial investment of time and effort that may well prove prohibitive.

If the migration from <physdesc> to <physdescstructured> is not partially automated, the repository will need to migrate from EAD2002 <physdesc> to EAD3 <physdesc> and then convert each <physdesc> to <physdescstructured> by hand. This approach entails a significant investment of staff time and effort, especially if a large number of guides are involved and/or <physdesc> was used extensively at the component level.

4.4: <langusage>

Migration Strategies and Concerns

EAD2002's <langusage> maps directly to EAD3's <languagedeclaration>. The most significant difference between <langusage> and <languagedeclaration> is that <languagedeclaration> does not allow free text content. It does, however, permit a free-text subelement, <descriptivenote>, to provide general descriptive information regarding the parent element.

Because <langusage>, in tandem with the subelement <language> and attributes @langcode and @scriptcode, was widely used in EAD2002 guides, the majority of the migration from <langusage> to <languagedeclaration> can be fully automated by mapping <langusage> to <languagedeclaration>, <language> to <language>, and @langcode to @langcode. If the repository routinely used free-text descriptions in <langusage>, it may wish to remove any elements not allowed as subelements of <p> and migrate the information to a paragraph in <descriptivenote> in order to ensure that no content is lost.

Generally speaking, the migration stylesheet will handle <langusage> in one of two ways. If <langusage> appears without a child <language> element, the stylesheet emits a <languagedeclaration> element with empty <script> and <language> children. When <language> subelements are present, each <language> will be migrated to it's own <languagedeclaration>. The stylesheet will, however, omit the source <langusage>'s @id from the resulting <languagedeclaration> elements. The @id value is not captured in a comment, so it will be lost. In both cases, the resulting <languagedeclaration> will include a child <descriptivenote>/<p> containing the original contents of <langusage>.

Although EAD3 allows <script> as a subelement of <languagedeclaration>, it is not required. If the source document includes @scriptcode as an attribute of <language>, the EAD2002 to EAD3 stylesheet processes @scriptcode to ensure capitalization, uses its value to populate <script>, and adds a comment reading "SCRIPT NAME NEEDED." This comment, however, appears to be an overstatement, as it's perfectly legal to have an empty <script> with a @scriptcode attribute. Repositories may also wish to use migration as an opportunity to insert this information, as it can easily be supplied automatically if the institution uses a limited number of scripts.

Data Remediation Concerns

If the migrating repository has employed <langusage> consistently and used the subelement <language> with the @langocde attribute, <languagedeclaration> should require only minimal post-migration remediation. Much of this work can be automated either by altering the migration stylesheet to suit local needs or by creating a local tool to complete the needed postprocessing work. Because the stylesheet's handling of <langusage> and its children is moderately complex, it is recommended that the results be examined carefully.

4.5: <langmaterial>

Migration Strategies and Concerns

The major difference between EAD2002's <langmaterial> and EAD3's <langmaterial> is that EAD2002 allows mixed content in the element while EAD3 does not. As is the case for <langaugedeclaration>, however, the free-text subelement <descriptivenote> is available to provide general descriptive information regarding the parent element if necessary.

Due to this change, migration strategy must be informed by whether the repository used <language> as a child of <langmaterial> either with or without @scriptcode. If <language> was used in EAD2002 without @scriptcode, both <langmaterial> and <language> can be migrated directly to EAD3. If @scriptcode was used, two migration paths are available: either this information can be ignored (and thus lost) or a wrapper <languageset> element can be created in EAD3 for each <language> subelement using @script, <language> migrated as a subelement of <languageset>, and the value of @script used to populate a sibling <script> tag.

If <language> was not used as a subelement, the contents of EAD2002's <langmaterial> can be migrated to a <descriptivenote> subelement of <langmaterial> in EAD3 after all elements not permitted as subelements of <p> have been stripped.

Data Remediation Concerns

Data remediation should be minimal to nonexistent for this element. Although a <langmaterial> element with all informational content contained in <descriptivenote> is not ideal, it is valid and provides approximately the same functionality as the corresponding <langmaterial> did in EAD2002. Thus, while a repository may wish to invest in remediation work to improve this element, it is not necessary to create a valid EAD3 document.

4.6: <language>

Migration Strategies and Concerns

EAD2002 allows four child elements of <language> (<emph>, <extptr>, <lb>, and <ptr>) that are no longer valid in EAD3, which permits only free text in the element. In cases where this configuration was used, two forward migration paths are available. Either <emph>, <extptr>, and/or <ptr> and any child elements can be stripped and their contents inserted into <language> as free text or <extptr> can be converted to <ptr> and both <ptr> and <emph> migrated to a <descriptivenote>/<p> child of <language>'s parent. In either of these approaches, formatting information provided by <lb> will be lost.

Data Remediation Concerns

Unless <emph>, <extptr>, <lb>, and/or <ptr> were used as children of <language>, remediation for this element should be nonexistent. If any of these elements appear as children, the migrating repository will need to decide whether it is acceptable to insert their contents into <language> as free text or if they wish to place the elements involved into a

<descriptivenote>/<p> child of <language>'s parent. If the second path is used, each <language> tag will need to be examined to determine whether important context was lost as a result of the move and, if so, replace it. The migration stylesheet is problematic in this instance, as it will convert <extptr> to <ptr> but leaves all of these elements in place as children of <language>, which will cause the final guide not to validate.

4.7: <materialspec>

Migration Strategies and Concerns

In general, it should be possible to migrate data in <materialspec> directly from EAD2002 to EAD3 unless child <arcref>, <bibref>, <num>, or nested <materialspec> elements were used. Two attributes, @label and @type, also need to be considered during migration. Because @type in EAD2002 maps to @localtype in EAD3, it should be relatively easy to convert this attribute automatically during migration. Repositories that employed the @label attribute may wish to evaluate this use before migration in order to determine whether it should remain @label in EAD3 or be converted to something else. For example, if a repository routinely used @label rather than @type to describe the type of information being given, it may wish to convert @label to @localtype rather than to @label during migration.

Data Remediation Concerns

This element should require no remediation unless the repository routinely used two or more subelements in EAD2002 that are not approved for use as subelements in EAD3. In this case, the repository will need to identify and address any formatting concerns incurred by removing the affected subelements prior to migration.

4.8: <repository>, <address> and <addressline>

Migration Strategies and Concerns

Because <repository> and its <address> and <addressline> subelements are very similar in both form and purpose in EAD2002 and EAD3, it should be possible to migrate the affected data directly. The exception to this rule is instances where the migrating repository routinely used free text in its <repository> elements. Because free text was allowed in EAD2002 but not in EAD3, the migrating repository will need to remediate this data prior to migration. If this free text was used consistently (for example, the repository's name was given as free text and its address was encoded using <address> and <addressline>) it should be possible to automate remediation during migration.

Use of <address> and <addressline> outside of <repository> has been substantially limited in EAD3. Specifically, <address> can no longer be used as a child of <accessrestrict>, <accruals>, <acqinfo>, <altformavail>, <appraisal>, <arrangement>, <bibliography>, <bioghist>, <blockquote>, <controlaccess>, <custodhist>, <dsc>, <entry>, <event>, <fileplan>, <index>, <item>, <odd>, <originalsloc>, <otherfindaid>, <p>, <phystech>, <prefercite>, <processinfo>, <ref>, <relatedmaterial>, <scopecontent>, <separatedmaterial>, and <userestrict>. Generally speaking, the migration stylesheet handles these deprecations by

converting <address> to <p>, stripping child <addressline> elements, and placing the the contents of each <addressline> in <p> as free text with a comma and a line break separating each data string.

Data Remediation Concerns
The amount of data remediation necessary for this element depends on how the migrating repository has constructed and employed <repository> and <address> tags. If <repository> routinely contains free text other than the repository's name or was used on the item level and/or as a subelement, pre-migration testing and remediation will be necessary to ensure that the data migrates as expected. If the only free text used is the repository's name, data remediation can be automated fairly easily; if other free text is present, substantially more remediation work will be required.

In cases where <address> was used outside of <repository>, the resulting guide should be carefully checked to ensure that strippping <address> and <addressline> tags has not caused the migrated information to become unreadable, but no further intervention should be required.

4.9: <unittitle>

Migration Strategies and Concerns
In cases where <unittitle> was used as a child of <did>, the contents of <unittitle> can be migrated directly from EAD2002 to EAD3. EAD3's use of <unittitle>, however, is much more restrictive than it was in EAD2002: in EAD3, <unittitle> is only allowed as a child of <did>; in EAD2002, it could appear as a child of <archref>, <entry>, <event>, <item>, <label>, <p>, <ref>, <extref>, <extrefloc>, and <refloc>. With the exception of <label>, in cases where <unittitle> was used as a child of one of these elements it may be possible to migrate its contents to <title> rather than <unittitle>. Alternatively, <unittitle> can be removed completely and its contents inserted into the parent element as free text.

Data Remediation Concerns
This element is expected to require little to no remediation when it was used as a child of <did> in EAD2002. If the migrating repository routinely used <unittitle> as a child of another element, it should decide whether it wishes to migrate <unittitle> to <title> or remove the element entirely prior to migration in order to avoid data cleanup. If it is decided to strip <unittitle>, testing to ensure that the resulting element is formatted legibly should be undertaken as well.

4.10: <abstract>, <fileplan>, <origination>, <physloc>, <phystech>, and <unitid>

Migration Strategies and Concerns
Because these elements are very similar in EAD2002 and EAD3, it should be possible to migrate their contents directly from EAD2002 to EAD3.

Data Remediation Concerns
These elements should require little to no remediation post-migration unless the repository has employed <address> and/or <note> as subelements on a regular basis. If these elements were used, the repository should investigate its proposed migration path to ensure that the information in these fields will migrate acceptably.

Section 5: <did> Siblings

5.1: Overview

While the majority of the elements below have not changed all that much functionally and semantically from EAD2002, the changes made with linking elements will greatly affect several of them. This is especially noticeable with <archref> and <bibref>. Since their focus has narrowed to static citation rather than providing linking, the EAD2002 to EAD3 stylesheet is prepared in most cases to migrate depending on whether linking attributes are present. Therefore, repositories will have to consider how those elements are being used to handle data.

Generic elements such as <note> have been deprecated, which can provide its own set of challenges. While the stylesheet provides automated migration to <footnote>, <note> has four different possibilities depending on the semantic use of the data in the original EAD2002 guide. For the purposes of this section, the repository will have to be ready to consider its semantic usage of <note> so as to choose whether to follow the migration stylesheet's <note> to <p><footnote></footnote><p> path, if they wish to include <note>'s contents as part of the rest of the narrative text, or if they must take further steps to change <note> to <didnote>. Further discussion on this migration is available in Section 4.2.

The nesting possibilities in EAD2002 and simplifications in EAD3 can sometimes cause validation issues after migration, even if the EAD2002 guide completes the migration to EAD3. Nested linking is a particular issue—the stylesheet does not handle it well, especially when <archref> and <bibref> are nested within multiple levels and migrated to <ref>. Another validation concern is for elements that no longer belong in certain levels of the hierarchy and map to another element in EAD3 but are not in valid locations if preemptively moved within an EAD2002 guide. This will cause the migration to fail. Therefore, post-migration editing that may not be doable in an automated fashion will become necessary.

Beyond the programmatic issues, a repository may need to pay attention to general description and formatting issues due to how migration takes place. This is true of elements that are radically moved within the hierarchical tree as they may lose the context they had in the description they used to be nested in. On a more detailed note, migration of elements such as <address> to <p> may involve the removal of child elements, which may distort formatting of

the data within the tags. In those cases, the repository will need to be prepared to edit for human readability.

5.2: <accessrestrict> and <legalstatus>

Migration Strategies and Concerns

<accessrestrict> appears in the same parent elements in EAD2002 and EAD3, with the exception of the deprecated <archdescgrp> and <descgrp>. A large part of why <eadgrp> was deprecated along with child element <archdescgrp> was its lack of use. Most of its child elements remain in place, with the exception of <address>, <legalstatus>, and <note>. <address> and <note> are discussed in more detail in Sections 4.8 and 4.2 respectively.

<legalstatus> is a much more problematic element to migrate. While it could only appear in <accessrestrict> in EAD2002, it is a sibling to <accessrestrict> in EAD3. When using the migration stylesheet, <legalstatus> elements are migrated to <p> elements that remain nested in <accessrestrict>. Another change with <legalstatus> is the addition of structure. The EAD2002 version of the element allowed free text with further child elements only present as options for formatting or providing linking. However, the EAD3 version of <legalstatus> requires child elements for denoting the form of data, such as <p> or <table>. None of these required child elements in EAD3 were valid in EAD2002. Since the data in <legalstatus> in EAD2002 would likely be more appropriately put into <p> elements, it may be possible to automate migration by editing the stylesheet to move <legalstatus> up one level and wrap its data in <p> elements. If its contents need to be migrated to multiple child elements rather than to a single child element, migration work could only be partially automated by moving <legalstatus> to the sibling level with a valid child element wrapping the data and necessary additional child elements added manually. In some cases, both automatically adding a standard child element and editing manually will be employed, for example by automating the migration of data to <p> elements and then building out additional child elements manually. It may be doable to perform entirely manual post-migration clean-up if the amount of guides needing attention is relatively small. Another consideration are issues arising from data in <legalstatus> not being detailed enough to warrant its own encoding. In this case, it may be possible to leave the information within <p> elements, especially if there is enough context and detail in the majority of the description to emphasize the accessibility to the repository's satisfaction.

The attributes of both <accessrestrict> and <legalstatus> are essentially the same between EAD2002 and EAD3 and would not cause any issue with automation of those elements.

Data remediation concerns

Since migration concerns hinge solely on movement of elements regardless of how they are structured, there are no notable concerns about remediating the data itself. The only issues that may be present are formatting related. In those cases, either manual or automated procedures may need to be performed to ensure readability.

5.3: <accruals>, <altformavail>, <appraisal>, <arrangement>, <originalsloc>, <processinfo>, and <userestrict>

Migration strategies and concerns

<accruals>, <altformavail>, <appraisal>, <arrangement>, <originalsloc>, <processinfo> and <userestrict> are easily automated migrations as very little has changed between EAD2002 and EAD3. The two child elements that have been dropped, <address> and <note>, have direct, sensible migration paths and are discussed in more detail in Sections 4.8 and 4.2 respectively. Attributes have remained consistent between the two versions as well, with the addition of @lang and @script in EAD3, which would have no data to migrate from an EAD2002 guide, and the change of @type to @localtype, which is a direct migration path.

Data remediation concerns

Since <accruals>, <altformavail>, <appraisal>, <arrangement>, <originalsloc>, <processinfo>, and <userestrict> all have direct automated migration paths in their simplest, least structured forms, there are almost no concerns for data remediation.

5.4: <acqinfo>

Migration strategies and concerns

In most cases, <acqinfo> is relatively easy to migrate in an automated manner. As with several of the elements in this section, both <address> and <note> are no longer valid child elements. Another migration concern is that <acqinfo> is no longer a valid child element of <custodhist> in EAD3—it can only appear as a sibling of <did>. With the EAD2002 to EAD3 stylesheet, <acqinfo> elements within <custodhist> are migrated to <p> elements. If that information would be best left with <acqinfo> encoding, pre-migration clean up would be necessary to move all <acqinfo> data in <custodhist> to an upper level, which could be accomplished via scripting. Additional work could be undertaken to combine multiple <acqinfo> elements into one. All attribute migration can be fully automated.

Data remediation concerns

While there are concerns with <acqinfo> migration when it comes to its node placement, none of them involve the structure of the element. Therefore only formatting remediation concerns remain.

5.5: <archref> and <bibref>

Migration strategies and concerns

<archref>’s and <bibref>’s usage in EAD3 has been reduced from EAD2002 in terms of valid parent elements. Specifically, these tags can no longer be children of <abstract>, <container>, <creation>, <descrules>, <dimensions>, <emph>, <entry>, <event>, <extent>, <extref>, <item>, <label>, <langmaterial>, <langusage>, <materialspec>, <origination>, <p>, <physdesc>, <physfacet>, <physloc>, <ref>, <repository>, <unitdate>, <unitid>, and <unittitle>.

They also no longer can be used as child elements within each other (e.g. no <bibref> within <archref>). In many of these former parent elements, <archref> and <bibref> are converted to <ref> with the migration stylesheet. As the EAD2002 versions of <archref> and <bibref> were meant to both provide linking and static citation, the logic of <archref>'s and/or <bibref>'s appearance(s) in many of the previously valid parent elements has to do with linking capabilities. Therefore, the repository will have to weigh how <archref> and/or <bibref> was/were used in that particular instance in that guide. If the attributes for <archref> and/or <bibref> are set up for linking, the element could remain in place as <ref>, and therefore the migration could be entirely automated. If the data is not meant for linking and serves solely as citation, as could be possible with parent elements like <abstract>, either the element will have to be mapped to another element entirely, such as within <bibliography>, or it will have to lose the encoding, leaving the data alone in the narrative. The former method could be accomplished via pre-migration scripting to clone the information, append the information to the new node, and then delete the original. For the latter method, clean-up would also have to be done pre-migration, either by stylesheet or manually.

An additional cause for concern regarding the <archref> and/or <bibref> to <ref> migration emerges when these elements were used as children of <extref> in the EAD2002 guide. <archref>, <bibref>, and <extref> will all migrate to <ref> in the EAD3 guide. In this case, the resulting guide will not validate since <ref> cannot be a child of other <ref> elements and the repository will have to choose which element (or all!) will need to be removed in order to perform any kind of migration. An example would be to remove the <archref> encoding and let the <extref> data migrate to <ref> to retain that linking if <archref> was solely used as a citation element. In those cases, the migration would be incredibly difficult to automate.

Within some parent elements, <archref> is not migrated to <ref>. Instead, the migration stylesheet strips it out in some form. When <archref> is used within <creation> and <bibref>, the elements are stripped, but the data is retained. Within <langmaterial>, <origination>, and <repository>, both the elements and the data are entirely stripped out. Migration strategies for these elements are best considered through alternative linking methods. For instance, since <langmaterial>, <origination>, and <repository> are structured to mandatorily use controlled access point elements in EAD3, it might be worthwhile to consider linking from those access points, including possibly creating new ones for the previous <archref> and/or <bibref> information. <creation> has been deprecated in EAD3 and its data migrated to various elements within <maintenanceevent>. The data within <archref> and/or <bibref> migrates to the child element <eventdescription> in combination with other descriptive text in <creation>. In that case, the repository would want to consider what information needs to be present and if it's worth placing that data into another element such as <relatedmaterial> or <custodhist>. Finally, within the opposite <archref> or <bibref>, rather than nesting <archref> within <bibref> or vice versa, it would be wise to bring the nested element to the sibling level. This work could be accomplished via scripting.

While <archref> and <bibref> are valid children of <bibliography> in both EAD2002 and EAD3,

there can be issues with <archref> and/or <bibref> migrating correctly if <bibliography> is structured with <list> and <item> elements. Since <archref> and <bibref> were valid children of <item> in EAD2002 and not so in EAD3, any <archref> or <bibref> within <item> will be migrated to <ref> using the stylesheet, whether or not it has linking attributes. If there are no linking attributes for that instance of <archref> or <bibref>, a decision will have to be made whether to remove <archref> or <bibref> encoding and leave the data in <item> elements as free text, or if the entirety of <bibliography> will need to be restructured so as to no longer use <list>. The former could be automated by creating a local stylesheet to remove <archref> and <bibref> elements from <item> elements. The latter, however, would be a much larger investment in time and labor as there is not a good way to automate such restructuring, especially if this was a common way to structure <bibliography> in a repository's guides.

Linking attributes in <archref> and <bibref> migrate in an automated fashion, whether they remain the same element or migrate to <ref>. If <archref> and <bibref> are remaining the same and have linking information in the attributes, the stylesheet will migrate the linking attributes into a new <ref> child element and insert the relevant data. If either of them are migrated to <ref> due to parent elements, the linking attributes will migrate to reflect valid linking attributes in those <ref> elements. Non-linking elements carry over appropriately. Both <archref> and <bibref> have @lang and @script attributes added in EAD3.

Child elements have changed considerably for both <archref> and <bibref>. <archref> has lost 15 child elements in EAD3. When the majority of this data is migrated via the stylesheet, the elements themselves are dropped while the data within is retained. This happens for <unittitle>, <unitid>, <abstract>, <container>, <physdesc>, <physloc>, and <repository>. In those cases, decisions will need to be made as to whether to keep the information that was present in those elements at all, if it can stand on its own in the <archref> element without further encoding, or if it should be mapped to other fields. It should be noted that child elements of <repository>, such as <corpname> and <address>, will be retained in the migration even if <repository> itself is stripped and therefore the guide will not validate! If the information is kept, some post migration cleanup will need to happen to fix any formatting errors incurred by elements being removed. If it is not, a more exhaustive effort, either pre- or post-migration, will have to be undertaken to remove the information, either through additional stylesheet work or manually. If the choice is made to map elsewhere, work has to be done via scripting or through work with a local stylesheet to move the information to a different element.

In five cases, <extref>, <extptr>, <note>, <unitdate>, and <address>, the elements are mapped to other elements: <extref> to <ref>, <extptr> to <ptr>, <note> to <footnote>, <unitdate> to <date>, and <address> to <p>. The former four are valid within <archref> in EAD3. <p> is not, however, and the guide will not validate. Therefore, removing address information from <archref> would be required. Finally, in two cases, <dao> and <daogrp>, the elements and their data are entirely removed. For more information on <dao> and <daogrp>, please refer to Section 9.2.

<bibref> only has five child elements that are no longer valid: <bibseries>, <edition>, <imprint>, <extptr> and <extref>. After migration with the EAD2002 to EAD3 stylesheet, <bibseries>, <edition>, and <imprint> are stripped from the guide, leaving behind the data originally contained in these tags. If the repository decides that it's important to encode this information, using <title> and <part> elements would be the most logical way to map this data, with the @localtype attribute of <part> set to designate what the data is. It would require scripting to accomplish this migration if the repository wishes to automate the work. As in <archref>, <extptr> and <extref> are migrated to <ptr> and <ref>, which are valid in EAD3.

Data remediation concerns
In some (or many) cases, the repository may have to decide whether it wants <archref> and <bibref> to serve as linking or static citation elements. Depending on such decisions, the data may have to be cleaned up to add linking attributes prior to migration or remove attributes either in preparation for migration to another element or to give precedence to another linking element, such as in reconciling migrations with <archref> within <extref>.

A major stumbling block in terms of data will be remediating child elements in <archref> and <bibref> that are no longer valid in EAD3. Especially of concern will be elements that do not validate after migration, such as migrating <address> to <p> under certain conditions. In those cases, the repository would have to perform a great deal of pre- or post-migration clean-up in order to get their guides to properly validate at the end. Also, since many of these child elements are removed but their data retained, there would need to be a great deal of remediation performed on these elements in order to have them mapped appropriately to new elements, such as mapping <bibseries> to something like <title><part localtype="bibseries">Example</part></title>.

5.6: <bibliography>

Migration strategies and concerns
<bibliography> has much of the same functionality in EAD3 as it did in EAD2002. It lost some of its previous child elements, including <address>, <extref>, <linkgrp>, <note>, and <title>. <address> and <note> are discussed in more detail in Sections 4.8 and 4.2 respectively. Using the stylesheet, <title> elements that are not children of other valid child elements such as <bibref> are retained and wrapped in <p> elements. <extref> and <linkgrp> have both been deprecated and therefore are migrated to <ref> elements. These <ref> elements are wrapped in <p> elements to keep them valid within <bibliography>. In short, these migrations automate smoothly, unless information in <address> absolutely must be encoded as <address> data. In that case, the repository will need to decide if there is a better field to map the address information to.

One major concern with migrating <bibliography> arises if it contains <archref> and <bibref> within a <list> element structure. If <archref> and <bibref> are contained in <item> elements,

they will be migrated to <ref> elements. This mapping is a problem if <archref> and/or <bibref> information is meant to be used as static citation rather than to establish linkages. If this is so, a decision will have to be made whether to remove the <archref> or <bibref> encodings or to restructure the note in order to remove the <list> structure.

Attributes for <bibliography> are almost identical between EAD2002 and EAD3. @id and @script were added but do not map to any information that would be present in an EAD2002 guide. @type became @localtype, which is a direct migration path.

Data remediation concerns
<bibliography> itself will migrate smoothly without any serious regard to data structure. However, attention must be paid to possible child elements <archref> and <bibref> if they are being used within <list> elements. If children <archref> and <bibref> are used as static citations, this structure will require the repository to decide if <list> is needed or if <archref> and <bibref> data need to be encoded as such. This decision will likely depend on the amount of guides that have these specific issues in migration.

5.7: <bioghist>

Migration strategies and concerns
<bioghist> has not changed too much between EAD2002 and EAD3. The four elements that are no longer valid as children, <address>, <note>, <dao>, and <daodesc>, either map to different fields or have been moved to other areas within the guide. <address>, <note>, and <dao> are discussed in more detail in Sections 4.8, 4.2, and 9.2 respectively. Very little will need to be considered in terms of attributes, as none have been removed, and the one that has changed, @localtype, is analogous to EAD2002's @type.

Data remediation concerns
While the structure of <bioghist> and its child elements themselves will not require much attention, the move of <dao> and <daogrp> out of <bioghist> will likely require some editing of the descriptive data.

5.8: <custodhist>

Migration strategies and concerns
<custodhist> has lost three child elements, <acqinfo>, <address>, and <note>. In both <acqinfo>'s and <address>'s cases, the migration stylesheet maps the original elements to <p> elements. <note> is mapped to <footnote>, which is wrapped within <p> parent elements. Work will have to be done for <acqinfo> for its encoding to stand if the repository wishes to do so. If the repository would like to keep the data within <acqinfo> encoded as such, the repository will want to move the <acqinfo> into a direct child level of <c> or <archdesc> and let the information stand on its own. This could be accomplished in an automated fashion by pre-migration scripting or through a pre-migration local stylesheet, with the task becoming more complicated if there are multiple <custodhist>s nested within the other as the repository will

need to address each level in the hierarchy the node will need to climb. Attributes have not radically changed, with two additions of @lang and @script, and the change of @type to @localtype, which is a direct mapping and easily automated.

Data remediation concerns
<acqinfo> can be moved to a different level in the guide via automated methods, but since it will lose the context from the information in the former <custodhist> parent element, the data within <acqinfo> may have to be edited to maintain clarity. Also, depending on how the data within elements migrate and the repository's decision to keep things in certain elements, the repository may have to perform post-migration formatting clean-up to remove or add whitespace where needed, etc.

5.9: <index>, <indexentry>, <namegrp>, <ptrgrp>

Migration strategies and concerns
<index>, <indexentry>, <namegrp>, and <ptrgrp> have very few serious changes as they all appear in the same parent elements and are structurally the same in EAD2002 and EAD3. <index> has lost child elements <address> and <note>; <namegrp> has lost <note>. <address> and <note> are discussed in more detail in Sections 4.8 and 4.2 respectively. <note>'s standard migration path allows for an automated migration in this instance, but it will likely not retain the semantic meaning of the original <note> element within <index> as they were mainly intended for subdividing. Since additional <indexentry> elements can be used to create nesting and therefore subdivision, in this instance it would be useful to map <note> to <indexentry> prior to migration. This may be automated via a local stylesheet. Otherwise, the repository will have to invest some amount of manpower to manually update the elements. This could be doable depending on the amount of guides needing this attention. If the number is relatively few, it may make more sense to have staff manually update the guides instead of investing in creating a stylesheet if there aren't resources for on-demand creation of XSL.

Data remediation concerns
Since <note> serves a different semantic purpose in <index>, <indexentry>, and <namegrp> than it does in most parent elements in EAD, there will have to be pre-migration editing to prepare <note> elements as additional <indexentry> elements.

5.10: <odd>

Migration strategies and concerns
<odd> has lost four child elements: <address>, <dao>, <daogrp>, and <note>. <address>, <note>, and <dao> are discussed in more detail in Sections 4.8, 4.2, and 9.2 respectively. No attributes have been dropped from <odd>: @lang and @script have been added, which does not affect any of the data that could be present in <odd>, and @type migrates to @localtype.

Data remediation concerns
<dao> and <daogrp>/<daoset> migrate in an automated fashion, but because they are

repositioned higher up the hierarchical tree in the process, descriptive information about the digital object may require expansion and editing in order to provide appropriate context since it will no longer have said context from the <odd> parent element.

5.11: <otherfindaid>

Migration strategies and concerns
<otherfindaid> lost a few child elements, half of them deprecated linking elements, as well as <address> and <note>. <address> and <note> are discussed in more detail in Sections 4.8 and 4.2 respectively.

The deprecated linking elements, <extref> and <linkgrp>, have automated migration paths, but the end product may require formatting. Using the migration stylesheet, <extref> migrates to <ref> elements wrapped in <p> parent elements. <linkgrp> is broken down more due to the simplification of linking elements in EAD3. It maps to <p> elements. Most of its child elements migrate to <ref> elements, while <arc> and <resource> are completely stripped from the guide, along with their data. Because these elements are exclusively relevant to XLink structures that aren't used in EAD3, this deletion is irrelevant. The migration's result is a list of links.

It should be noted that <ref> cannot be used as a direct child of <otherfindaid> in EAD3. When the guide undergoes migration with the EAD2002 to EAD3 stylesheet, the elements and the data remain but it are wrapped in <p> elements to allow it to stay valid in formatting. <archref> and <bibref> also go through selective transformations depending on whether linking attributes are present. If they are, the data migrates to <ref> elements wrapped in <archref> or <bibref> depending on the original element. If they are not, they remain the same.

There are no changes to attributes, aside from the addition of @lang and @script.

Data migration concerns
Elements related to linking in either EAD2002 or EAD3 may not require a great deal of remediation for structure or content, but may require a small amount of post-migration formatting clean-up so as to display properly in the final product.

5.12: <prefercite>

Migration strategies and concerns
Very little has changed between EAD2002 and EAD3 for <prefercite>. The two child elements that have been dropped, <address> and <note>, have direct, sensible migration paths. Attributes have remained consistent between the two versions as well, with the addition of @lang, @script, and @localtype to EAD3, which would have no data to migrate from an EAD2002 guide.

Data remediation concerns
Since <prefercite> has direct automated migration paths in its simplest, least structured form,

there are no concerns for data remediation, aside from possible formatting editing needed for data to read well, such as spacing between words.

5.13: <relatedmaterial> and <separatedmaterial>

Migration Strategies and Concerns
<relatedmaterial> and <separatedmaterial> lost several child elements, including <address>, <extref>, <linkgrp>, <note>, <ref>, and <title>. Using the EAD2002 to EAD3 stylesheet, <address> migrates to <p>, with its children <addressline> being stripped away, leaving the data behind. <extref>, as well as <archref> and <bibref> with linking attributes, have their information shifted to <ref>. The newly created <ref> elements from <extref> are wrapped in <p> elements while <ref> data from <archref> and <bibref> are nested as children of their original elements.

<note>, <ref>, and <title> also are migrated within <p> parent elements in different ways. <note> is migrated to <footnote> wrapped in the <p> elements. <ref> retains its element, but is wrapped within <p> elements. <title> also retains its element and is wrapped in a <p> element. Both <ref> and <title> are easy to automate without any issues concerning data or validation.

The migration stylesheet maps <linkgrp> to <p> as well. Its children are then mapped to <ref> elements, allowing it to remain valid. It should be noted that if <archref> and <bibref> are deployed in <linkgrp> as children of other deprecated linking elements, they will migrate to <ref> regardless of the presence of linking attributes, causing the guide to not validate. In those cases, it will become necessary to move <archref> and <bibref> information since they were meant as static citation. The work will almost entirely have to be done manually since moving <archref> and <bibref> to their correct EAD3 positions in EAD2002 invalidates the guide and causes the initial migration to fail.

Data remediation concerns
Some data may require post-migration clean-up of formatting for whitespace issues, etc. Otherwise, the majority of concerns in migrating <relatedmaterial> and <separatedmaterial> are not with the structure of its data, but rather with positioning and positioning of child elements.

5.14: <scopecontent>

Migration strategies and concerns
<scopecontent> lost five child elements: <address>, <arrangement>, <dao>, <daogrp>, and <note>. <address>, <note>, and <dao> are discussed in more detail in Sections 4.8, 4.2, and 9.2 respectively. <arrangement> elements are stripped from <scopecontent>, leaving the data behind in <p> elements. In this case, the repository could choose to do one of two things. They could either choose to move the <arrangement> element to make it a sibling of <did>, or they can choose to leave the data in <scopecontent> within <p> elements. If the repository wishes to move <arrangement> up to the appropriate level, this could be accomplished by editing the

EAD2002 to EAD3 migration stylesheet to automatically execute this move or creating a local pre-migration stylesheet to automate that move. The repository could also choose to merge multiple <arrangement> elements into the same element via XSL. Whether the repository chooses to move the elements, or leave the information in <p> elements, the data itself may require some cleanup for formatting.

Data remediation concerns
The majority of data remediation is related to formatting and description choices rather than any structural issues. Any <arrangement> data and descriptive data within <dao> and/or <daogrp>/<daoset> will likely require editing and expansion in order for the repository to provide context that is lost from removing it from <scopecontent>, especially if there is no <descriptivenote> for <dao> or <daoset>.

5.15: <userestrict>

Migration strategies and concerns
Almost all of <userestrict>'s parent elements remain the same. The same is true of its child elements, with the exception of <address> and <note>, which are discussed in more detail in Sections 4.8 and 4.2 respectively.

The attributes of <userestrict> are essentially the same between EAD2002 and EAD3 and would not cause any issue with automation. The new attributes, @lang and @script, have no values that would need to be migrated from elsewhere and @localtype directly migrates from @type.

Data remediation concerns
While <address> migrates in an automated fashion to <p> without any concern with structure, the repository may need to perform some minor post-migration clean-up to fix formatting issues, e.g. whitespace. Otherwise, migration hinges on positioning rather than the structure of EAD2002 <userestrict> and child elements.

Section 6: <eadheader> to <control>

6.1: Overview

The majority of the changes to elements involved in the transition from <eadheader> to <control> have to do with semantic changes rather than major restructurings. There are, however, several scenarios that may create migration issues. Depending on how <descrules> was constructed, @id may be either stripped or duplicated. Similarly, if <date> and/or <num> were used as children of <titleproper> or <subtitle> in <controlnote>, the migration stylesheet will strip them and the associated information will be lost. Several elements require setting xsl:params in the migration stylesheet, but this work will require minor edits only after the

repository decides what values it wishes to use. Thus, while it is advisable to examine these elements carefully during pre-migration testing in order to ensure that all data transfers as anticipated, it is unlikely that they will pose substantial challenges to a full migration.

6.2: <eadheader>

Migration strategies and concerns
With the exception of @findaidstatus, EAD2002's <eadheader> and its associated attributes migrate directly to EAD3's <eadheader>. @findaidstatus is replaced with /ead/filedesc/localcontrol/term[@localtype="findaidstatus"]. This change makes the element more expressive and does not cause any data loss.

Data remediation concerns
Because the only major change to this element will not cause loss or garbling of information, it should require little to no remediation.

6.3: <eadid>

Migration strategies and concerns
EAD2002's <eadid> maps to EAD3's <recordid>/<otherrecordid>. <eadid> is highly structured and, in general, is thoroughly mapped onto <recordid> and <otherrecordid localtype="old_eadid_attr_name"> elements. One <otherrecordid> is created per <eadid> using @identifier, @publicid, @urn, and/or @url. @mainagency and @countrycode are mapped onto the new <maintenanceagency> element, which is discussed in more detail in a separate section.

Data remediation concerns
The only potential "gotcha" is that the migration stylesheet does not carry @encodinganalog to <otherrecordid> elements created from @identifier, @publicid, @urn, or @url, but only to the <recordid> element that directly replaces <eadid>. Systems depending on @encodinganalog to map any of these values to another schema must process both the relevant <otherrecordid> and <recordid>, or additional processing will need to be done to produce correct local mappings for <otherrecordid> values.

6.4: <archdesc>

Migration strategies and concerns
The only change made to <archdesc> between EAD2002 and EAD3 is the transliteration of @type to @localtype.

Data remediation concerns
Because the transition from @type to @localtype is a simple transliteration, <archdesc> is expected to require no pre- or post-migration remediation.

6.5: <descrules>

Migration strategies and concerns

EAD2002's <descrules> maps to <conventiondeclaration> and <citation> in EAD3. When processing the <descrules> element into <conventiondeclaration> elements, the migration stylesheet allows for two possible outcomes, which depend on whether <descrules> contains a <title> child:

If <descrules> DOES contain one or more <title>s, each title will become a separate <conventiondeclaration>, with a <citation> child containing the text of the <title> and a <descriptivenote> child containing an EAD3 processed version of the contents of <descrules> (this will result in repetition of contents, so any ids in <descrules> child elements will be duplicated!)

Most attributes on <descrules> will be copied from <descrules> into the new <conventiondeclaration> elements, but if more than one <title> is present, @id attributes will be dropped entirely, and are not preserved in another element.

If <descrules> does NOT contain any <titles>s, a single <conventiondeclaration> will be created with an empty citation and all attributes of <descrules>, including @id, and a <descriptivenote> with the contents of <descrules>, wrapped in a <p>.

Data remediation concerns

The EAD2002 standard seems to be fairly permissive within <descrules>, and caution is called for during migration. Most information is captured in the <descriptivenote> element(s), but @id elements attached to <descrules> are subject to loss with some codings and not with others, and @id elements attached to titles will be duplicated, which violates standards and can interfere with linking.

6.6: <filedesc>, <titlestmt>, <editionstmt>, <publicationstmt>, <seriesstmt>, and <notestmt>

Migration strategies and concerns

<filedesc>'s attributes and their contents are template-processed and copied. <filedesc> has five possible child elements in EAD3: <titlestmt> (required), <editionstmt> (optional), <publicationstmt> (optional), <seriesstmt> (optional), and <notestmt> (optional). Changes to the majority of these elements between EAD2002 and EAD3 are minimal. Two child elements of <titlestmt>, <titleproper> and <subtitle>, can no longer use <date> or <num> as subelements. The EAD2002 to EAD3 stylesheet will strip these tags and retain their content as free text. <publicationstmt> undergoes no changes unless it has a child <publisher> element. In this case, <publisher> is left in situ and text contents of all <publisher> children of <publicationstmt> are concatenated with ", " and placed in

/ead/control/maintenanceagency/agencyname. @show, @actuate, and @label can no longer be used as attributes of <notestmt>'s required child, <controlnote>, in EAD3. The migration stylesheet will preserve this data in a comment but does not retain the linking capabilities these elements provide in EAD2002. <editionstmt> undergoes no changes. In several cases, the value of @type will be migrated directly to @localtype.

Data remediation concerns
These elements are expected to require little to no remediation unless the repository wishes to preserve linking provided by attributes of <controlnote> or stripping <date> and/or <num> when they were used as children of <titleproper> or <subtitle> will prove problematic. In these cases, the repository will need to create a custom migration solution.

6.7: <localcontrol>/<term>

Migration strategies and concerns
<localcontrol>/<term> is generated from the contents of eadheader/@findaidstatus.
If @findaidstatus was not used in EAD2002, the corresponding elements are not created in EAD3.

Data remediation concerns
Because both of these elements are new to EAD3 and are created using data present in EAD2002, not data loss is possible during the migration. Thus, these elements should require no remediation.

6.8: <maintenanceagency>, <agencycode>, <agencyname>, and <otheragencycode>

Migration strategies and concerns
The migration stylesheet constructs <maintenanceagency> and its children from attributes on <eadid>, an optional xsl:param ($agencynameValue), and/or the comma-concatenated text of filedesc/publicationstmt/publisher. Though the new construction is strictly more expressive than the first two sources, internal tagging from <publisher>s is not carried over into <agencyname>, although it is retained in its original location. @countrycode is carried over directly from <eadid>.

<agencyname> is an entirely new element. The migration stylesheet will provide its content from one of several sources. If the xsl:param $agencynameValue is set, its contents will be presented. If not, and <publisher>s exist in filedesc/publicationstmt, their text will be concatenated with a comma and space between each value. If neither of these sources are present, the string '[Agency Name]' will be inserted. <agencycode> is also a new element in EAD3. The stylesheet creates this element and populates it with the value of eadid@mainagencycode if the attribute was used in the source EAD2002. The stylesheet cannot produce <otheragencycode> elements because <eadid> cannot represent multiple agency codes.

Data remediation concerns
Repositories should decide whether they wish to set xsl:param $agencynameValue or rely on the comma-concatenated values of <publisher> prior to beginning migration. If these preparations are carried out, these elements should require no post-migration remediation.

6.9: <maintenancehistory>

Migration strategies and concerns
EAD3's <maintenancehistory> is a new element, which the EAD2002 to EAD3 stylesheet populates using a generated event to represent the guide's conversion from EAD2002 to EAD3 as well as the contents of EAD2002's <revisiondesc> and <profiledesc>/<creation>. The children of the generated event can be adjusted using xsl:params if desired.

In many cases, the attributes and contents of the affected elements are copied directly from EAD2002 into analogous elements in EAD3. <revisiondesc>'s attributes are copied into <maintenancehistory>; the value of <profiledesc>/<creation> is copied into <eventdescription>; the value of <change>/<date> is copied into <maintenanceevent>/<eventdatetime>; and the value of <change>/<item> and/or <list>/<item> is copied into <maintenanceevent>/<eventdescription>. <change>, which migrates to <maintenanceevent>, has a somewhat more convoluted migration path. The stylesheet copies all of relevant attributes, creates an <eventtype> child with @value 'unknown,' and creates empty <agent> and an <agentype> elements with contents 'unknown.'

Migration of <profiledesc>/<creation> is also somewhat convoluted. <creation> itself is fairly straightforward: <creation> is converted to <maintenanceevent>, all attributes are copied, and an <eventtype> child with @value='created' is generated. <creation>/<date> is somewhat more complicated. If @calendar, @era, @certainty, or @type were used, they and their contents will be stripped and the information lost. @normal is converted to @standarddatetime. If @normal represents a range, the latter half of the range is processed out; otherwise the whole @normal value is used. The contents of the element itself are copied verbatim.

Data remediation concerns
Data remediation for these elements should be nonexistent unless the migrating repository wishes to either populate <maintenanceevent>/<eventtype @value>, <maintenanceevent>/<agentype>, or <maintenanceevent>/<agentype> with values other than "unknown" or retain information present in <creation>'s @calendar, @era, @certainty, or @type attributes. In any of these cases, the repository will need to establish a customized migration path by either editing the migration stylesheet, creating an internal stylesheet, or scripting. Because these elements generally occur in only one section of the guide, it may be possible to complete the necessary editing work by hand if a relatively small number of guides are involved.

6.10: <maintenancestatus>

Migration strategies and concerns
EAD3's <maintenancestatus> is a new required element and thus has no exact equivalent in EAD2002. The EAD2002 to EAD3 stylesheet populates the element using an xsl:param. A

value of “revised” is inserted by default, but repositories may edit the stylesheet to provide “deleted”, “new,” “deletedsplit,” “deletedmerged,” “deletedreplaced,” “cancelled,” or “derived” instead.

Data remediation concerns
If the migrating repository decides what value it wishes to provide for this element and edits the stylesheet accordingly prior to migration, no remediation should be necessary.

6.11: <citation>, <localtypedeclaration>, <publicationstatus>, <representation>, and <sources>

Migration strategies and concerns
These elements are all new in EAD3 and cannot be created by the migration stylesheet with the exception of <publicationstatus>, which can be set using an optional xsl:param.

Section 7: Generic and Wrapper Elements

7.1: Overview

The element undergoing the most change between EAD2002 and EAD3 in this section is <p>. Use of this element has been limited in EAD3 in that it can be a child of fewer elements and fewer elements are valid as its children than in EAD2002. In most cases, minimal work is required to accommodate these changes. The migration stylesheet will, however, strip certain constructions of <p> without copying the data to another element, thus causing it to be lost. Additionally, there is no obvious forward migration path for <table> and <chronlist> when they were used as children of <p>. Depending on whether these elements are valid children of <p>’s parent element, substantial manual remediation may be required to preserve the affected information.

The only other element in this section that could hinder migration is <did>, and that hindrance will only occur if one <did> was nested in another. Thus, it is likely that <p> will be the only element to pose a substantial problem.

7.2: <descriptivenote>

Migration strategies and concerns
In EAD3, <descriptivenote> replaces <daodesc> in all contexts and <resource> elements in <daogrp>. It is also used as a wrapper for contents in <langusage>, <langmaterial>, and <conventiondeclaration> elements.

Data remediation concerns
Because migration issues associated with <descriptivenote> vary widely depending on the context in which the tag appears, issues are discussed in the section dealing with the relevant parent element.

7.3: <dsc>

Migration strategies and concerns
Generally speaking, EAD2002's <dsc> migrates directly to EAD3's <dsc>. The exception to this rule occurs where <did> structures have been nested. In this case, the EAD2002 to EAD3 stylesheet will ignore the structure and insert a comment warning that it is too complex to be converted. The stylesheet will handle <dsc>'s @type (which maps to EAD3's @dsctype) in one of two ways. If the value of @type is "othertype", @dsctype is set to "otherdsctype" and @otherdsctype is set to the value of @type. Otherwise, @dsctype's value is set to the value of @type.

Data remediation concerns
This element is expected to require little to no remediation unless the migrating repository nested <dsc> structures. In this case, a custom migration tool will be required.

7.4: <p>

Migration strategies and concerns
<p> is very similar in EAD2002 and EAD3, although its use has been limited. Specifically, <p> can no longer be used as a child of and can no longer contain <address>, <blockquote>, <chronlist>, <origination>, <repository>, and <table>. Some of these issues are reasonably easy to circumvent: <blockquote> can be easily converted to <quote> (the stylesheet's default behavior), <address> can be reformatted using <p>, and <origination> and <repository> can be stripped and their contents, likely including any child elements, can be inserted into <p>.

The EAD 2002 to EAD3 stylesheet will drop several constructions if found inside <p> elements, namely <chronlist>, <blockquote>/<chronlist>, <table>, <blockquote>/<table>, <blockquote>/<list>, <unitdate>/<title>, <origination>, <repository>, and <blockquote>/<p>.

There is no obvious forward migration path for <chronlist> and <table> when used as children of <p> in EAD2002. These elements are discussed in more detail in Sections 10 and 11.

Data remediation concerns
Data remediation for this element should be minimal unless <table> or <chronlist> were used as children of <p>. In these cases, the migrating repository should examine the affected guides to determine whether the information should be retained and, if so, how it should be structured. If a substantial number of guides are involved, a significant investment in remediation work may be required to salvage the affected information.

Additional issues may arise if the migrating repository is employing the EAD2002 to EAD3 migration stylesheet and needs to retain one of the constructions that the stylesheet drops. In this instance, the repository will need to either edit the stylesheet or create a custom stylesheet. Depending on how many alterations need to be made, significant work may be required.

7.5: <ead>, <expan>, and <num>

Migration strategies and concerns
These elements undergo only minor changes between EAD2002 and EAD3. The EAD2002 to EAD3 stylesheet converts <num>'s @type attribute to @localtype; provides no special handling for <expan>; and impacts <ead> only with a parameter-based setting to decide between DTD and XSD schemas.

Data remediation concerns
Because changes to these elements between EAD2002 and EAD3 are either slight or nonexistent, they are expected to require little to no remediation.

Section 8: Formatting and Labeling Elements

Section 8.1: Overview

Changes to these elements between EAD2002 and EAD3 are minimal and most can be accommodated with little effort. The most serious concern is that the migration stylesheet will strip structures associated with <head> and <blockquote> in some instances. Thus, although the migrating repository should take care to ensure that all necessary data is retained, it is unlikely that any of these elements will pose a major barrier to migration.

Section 8.2: <blockquote>

Migration strategies and concerns
The most significant change to <blockquote> between EAD2002 and EAD3 is that <blockquote> can no longer be used as a child of <event>, <item>, <p>, or <ref> in EAD3. Because all four of these elements allow the new <quote> to be used as a subelement, this issue can be resolved by simply converting <blockquote> to <quote>. The EAD2002 to EAD3 stylesheet employs this migration path, but will drop many structures beneath <blockquote>.

Data remediation concerns
If the migrating repository employed <blockquote> regularly, it should test its migration strategy carefully to ensure that all needed structures are retained. It may be necessary to create a custom migration path, which will involve editing the EAD2002 to EAD3 stylesheet, creating a custom stylesheet, or scripting.

Section 8.3: <head>

Migration strategies and concerns
<head> is virtually identical in EAD2002 and EAD3. The EAD2002 to EAD3 stylesheet will, however, strip <head> and its contents when it appears as a child of <scopecontent>/<arrangement> or <acqinfo>/<custodhist>.

Data remediation concerns
If the loss of data occasioned by the stylesheet's removal of <head> is deemed to be unacceptable, the migrating repository will need to create a custom migration path to retain this information, most likely by editing the stylesheet.

Section 8.4: <abbr>, <emph>, <label>, <lb>, <listhead>, <head01>, <head02>, and <head03>

Migration strategies and concerns
With the exception of <head03>, which is new in EAD3, these elements are virtually identical in EAD2002 and EAD3 and can be migrated without any special handling.

Data remediation concerns
Unless <title> was used as a child of either <emph> or <label>, no data remediation should be required. For more information about <title>'s migration, consult Section 2.3.

Section 9: Linking Elements

9.1: Overview

Linking elements have been substantially simplified and XLink discontinued in EAD3. <dao> and its associated elements, <daoset> and <descriptivenote>, can now only appear as children of <did>. This change may require <dao> to be moved up in the hierarchical tree during migration if it is to be preserved in this form. Where <dao> was used in the container list, this move will cause minimal disruption as <dao> will remain associated with the relevant component element. Where <dao> was used outside of the container list, however, this move may strip it of context supplied by the parent element. In these cases, the migrating repository will need to examine the affected guides and remediate the description associated with <dao> accordingly.

Extended reference elements (<extptr>, <extptrloc>, <extref>, and <extrefloc>) have been deprecated in EAD3. Generally speaking, these elements migrate to either <ref> or <ptr> with minimal difficulty. Use of <ptr> and <ref>, however, has been limited in EAD3 in that both elements can be used as children of far fewer elements that they could be in EAD2002. Thus, they may need to be restructured or moved into other child elements order to create a valid

EAD3 guide. Additional issues arise if migration creates a scenario where <ref> is used as a child of <ref>, which is invalid in EAD3. If only a few guides are affected, the migrating repository may be able to conduct the necessary pre- or post-migration remediation by hand; if numerous guides are affected, the repository may need to edit the migration stylesheet or create a custom migration tool in order to establish a serviceable migration path.

9.2: <dao>, <daogrp>, <daodesc>, and <daoloc>

Migration Strategies and Concerns

The changes surrounding <dao> and <daogrp> (now <daoset>) in EAD3 stem primarily from the substantial simplifications made to the elements' structures and allowable placements. EAD3 permits <dao> only as a child of <did> or <daoset> and <daoset> only as a child of <did>. Thus, neither element can be embedded in <archdesc>, <archdescgrp>, <archref>, <bioghist>, <c>, <c01>-<c12>, <odd>, or <scopecontent> as they could in EAD2002.

Where <dao> or <daogrp> were used as children or grandchildren of <archdesc> or at the component level, this change can be accommodated by placing <dao> within the parent or grandparent element's <did> during migration, which is the migration stylesheet's default behavior. Making <dao> or <daogrp> siblings of their former parents, however, may strip them of valuable contextual information and thus necessitate manual work to either transfer the relevant information or create new description. The most complicated migration scenario occurs when <dao> or <daogrp> appears as a child of <archref>. In this instance, the elements and their contents cannot be migrated directly to EAD3 and the stylesheet will delete both. <dao>'s functionality can, however, can be partially preserved using EAD3's <ptr>, but this technique will require either manual editing or creation of a custom migration tool.

Additional issues are occasioned by <daogrp>'s conversion to <daoset>. In EAD2002, <daogrp> could contain <arc>, <daodesc>, <daoloc>, <extptrloc>, <extrefloc>, <ptrloc>, <refloc>, or <resource> and <dao> was not a required subelement. In EAD3, all of these children have been deprecated and <dao> is a required subelement. Generally speaking, the contents of the deprecated elements and attributes can be retained by transferring them to a new <dao> element. Specifically, attributes associated with <daoloc>, <extptrloc>, <extrefloc>, <ptrloc>, and <refloc> can be mapped directly to attributes associated with <dao>. If more than one of these elements occurs as a child of <daogrp>, multiple <dao> tags will be required. Because EAD3 does not use the XLink structure present in EAD2002, the contents of <arc> and <resource> are irrelevant and can be discarded during migration.

Migration of <daodesc>, which maps to <descriptivenote> in EAD3, is slightly more complicated. The element itself can be migrated easily and directly from EAD2002's <daodesc> (available in <dao>, <daogrp>, or <daoloc>) to <descriptivenote> (available in <dao> and <daoset>). <descriptivenote>, however, allows fewer children than did <daodesc>. Specifically, <address>, <blockquote>, <chronlist>, <head>, <list>, <note>, and <table> can no longer be used. In the case of <list>, the issue can be resolved by adding a parent <p>.

<note> can either be mapped to <footnote> or the tag can be stripped and its contents inserted into a new <p> element. Similarly, both <blockquote> and <head> can be stripped and their contents inserted into a new <p> in order to preserve the affected information. The only migration path for <address>, <chronlist>, and <table> is to strip the tags and use either <p>/<list> or multiple <p> tags to structure the affected content. In these cases, repositories will need to examine the guides where this structure is used and decide whether the information should be retained and if so, how it should be encoded.

Finally, a new mandatory attribute, @daotype, has been added to <dao> in EAD3. Because @daotype, which has no equivalent in EAD2002, allows a value of "unknown," this value can be assigned automatically in order to facilitate migration.

Data Remediation Concerns
The amount of data remediation necessary for these elements depends largely on where and how <dao> and <daodesc> were employed. If <dao> was used routinely outside of container lists, the migrating repository must decide whether making <dao> a sibling of its former parent is an acceptable migration path. If it is not, the repository will need to decide whether the information should be retained and if so, how it should be represented. Similarly, substantial remediation will be required if the repository used <address>, <chronlist>, <list>, or <table> as children of <daodesc>. In both cases, if the information is retained the majority of the necessary remediation work will need to be done by hand and will prove extremely labor-intensive if a large number of guides are involved.

Additional remediation work will be necessary if the migrating repository wishes to populate @daotype with a value other than "unknown." This work could be partially automated if specific guides or sets of guides where all materials described with @daotype require the same value can be identified. It is possible, however, that the repository will need to review each <dao> and assign a value manually to accomplish this goal. If <dao> was used frequently, substantial time and effort will be required.

9.3: <ptr>, <extptr>, <ptrloc>, and <extptrloc>

Migration Strategies and Concerns
While <ptr> serves a very similar function in EAD 2002 and EAD3, its use in EAD3 has been standardized and slightly limited. Specifically, <ptr> can no longer be used as a child of <corpname>, <famname>, <function>, <genreform>, <geogname>, <langmaterial>, <language>, <langusage>, <legalstatus>, <name>, <occupation>, <origination>, <persname>, <subject>, <repository>, and <title> in EAD3.

Generally speaking, issues caused by this change are easy to resolve. In the cases of <langmaterial> and <langusage>, <ptr> and its contents can be inserted into child <descriptivenote>/<p> elements. In the case of <legalstatus>, a parent <p> element can be added to <ptr>. As discussed in Section 2.2, <part> resolves <ptr> issues for <corpname>,

<famname>, <function>, <genreform>, <geogname>, <name>, <occupation>, <persname>, <subject>, and <title>.

There is no clear migration path forward for <ptr> where it was used as a child of <origination> or <repository>. In these cases, the migrating repository will need to review their use of <ptr> carefully to determine whether they wish to retain the affected information and if so, how they wish to structure it.

<extptr>, <ptrloc>, and <extptrloc> have all been deprecated in EAD3; generally speaking, information previously associated with these elements can be migrated to <ptr> with minimal changes. Four attributes, @role, @title, @linktype, and @label, can no longer be used with these elements in EAD3. Two of these attributes, @role and @title, were simply renamed as @linkrole and @linktitle respectively. Because @linktype simply identifies a link as XLink compatible and this standard is not used in EAD3, it and its contents can be discarded. Finally, @label's function is very similar to that of @linktitle, so in cases where @label is used without @title, @label's contents can be migrated to @linktitle.

Data Remediation Concerns

If <ptr> was not used routinely as a subelement of <origination> or <repository>, data remediation for this element should be minimal to nonexistent. When <ptr> was employed in this capacity, the affected guides will need to be reviewed and the relevant data either removed or relocated appropriately. If numerous guides are affected, the data cleanup requirements may be substantial, especially if this configuration was not applied consistently.

If both @title and @label were used in a single deprecated element, additional cleanup may be required. If the contents of the two attributes are similar, the repository can remove one during migration and map the other to @linktitle. If use of these two attributes was standardized, the repository may be able to automate this process. If, however, the content of these two attributes is not similar or they were not applied in a standardized manner, it may be necessary to remediate each instance by hand.

9.4: <ref> and <extref>

Migration Strategies and Concerns

As in the case of <ptr>, <ref> serves a very similar function in EAD2002 and EAD3, but its use in EAD3 has been standardized and slightly limited. Specifically, <ref> can no longer be used as a child of <bibliography>, <origination>, <otherfindaid>, <relatedmaterial>, or <separatedmaterial>. In the cases of <otherfindaid>, <relatedmaterial>, and <separatedmaterial>, <ref> and its contents can be retained by nesting them in a parent <p>. In the case of <origination> and <repository>, <ref> can be nested in an appropriate child element (<corpname>, <famname>, <name>, <persname>, or <address/addressline>). Finally, in the case of <bibliography>, <ref> can be moved to a relevant child <bibref> element or a parent <p> can be added.

<extref> has been deprecated in EAD3. Generally speaking, its contents can be migrated directly to <ref>. The exception to this rule is when converting <extref> to <ref> will result in the new <ref> element being a child of <ref> or itself having child <ref> elements. In this instance, two migration paths are available. If the <ref> element in question was used only to provide a URI in EAD2002, it can be converted from <ref> to <ptr> and maintained as a child of <ref>; otherwise, because the <ref> element is repeatable, all of the <ref> elements involved can be made siblings.

Three attributes, @role, @title, and @linktype can no longer be used with these elements in EAD3. The implications of this change are discussed in Section 9.2.

Data Remediation Concerns
Generally speaking, data cleanup for these elements should be minimal to nonexistent unless converting <extref> to <ref> causes the resulting element to itself be a child of <ref>. If <extref> was used consistently, much of the necessary cleanup work can be automated; otherwise, the repository will need to examine the instances in which this structure occurs in order to determine the most appropriate migration path. If a substantial number of guides are affected, considerable pre- and/or post-migration data remediation may be needed.

9.5: <linkgrp> and <extrefloc>

Migration Strategies and Concerns
<extrefloc> has been deprecated in EAD3. In EAD2002, <extrefloc> could be used as a child of <daogrp> and <linkgrp>. Where <extrefloc> was used as a child of <daogrp> (which becomes <daoset> in EAD3), it can be converted to <dao>. All of <extrefloc>'s linking attributes (other than @linktype, which, as discussed in Section 9.2, is irrelevant) map directly to <dao>'s linking attributes. With the exceptions of <address>, <blockquote>, <chronlist>, <note>, <origination>, <repository>, <table>, <unitdate>, and <unittitle>, <extrefloc>'s children can be added included in a child <descriptivenote>/<p> in <dao>. If <unitdate> and <unittitle> are mapped to <date> and <title> respectively, they can be nested in <descriptivenote>/<p> as well. Two migration paths are possible for <origination>. The simplest approach is to strip <origination> and any child tags not permitted as children of <p> and include the relevant data in <descriptivenote>/<p>. If the repository wishes to preserve <origination> in this circumstance, the element can be moved up the hierarchical tree to become a child of <did> and thus a sibling of its original parent or grandparent. This approach, however, may strip <origination> of its context, so the migrating repository will need to manually review all guides created using this approach to determine whether information has been lost and, if so, reinsert it. Potential migration paths for <address>, <blockquote>, <chronlist>, <head>, <list>, <note>, and <table> are discussed in Section 9.1; and potential migration paths for <repository> are discussed in Section 4.6.

<linkgrp> has also been deprecated in EAD3. In EAD2002, <linkgrp> could contain child elements <arc>, <extptrloc>, <extrefloc>, <ptrloc>, <refloc>, or <resource>. As discussed elsewhere, <extptrloc>, <extrefloc>, <ptrloc>, and <refloc> all migrate to either <ref> or <ptr>. The contents of both <arc> and <resource> are irrelevant because EAD2002 used them to construct links using XLink, which is not present in EAD3. Thus, their contents can be discarded.

With the exceptions of <bibliography>, <langusage> (which maps to <languagedeclaration>), <langmaterial>, <origination>, <otherfindaid>, <relatedmaterial>, <repository>, and <separatedmaterial>, EAD3 allows <ref> and <ptr> as children of the elements that can use <linkgrp> as a child in EAD2002. In the cases of <bibliography>, <otherfindaid>, <relatedmaterial>, and <separatedmaterial>, wrapping <ref> and/or <ptr> in a parent <p> tag will remedy the issue. For <languagedeclaration> and <langmaterial>, <ref> and/or <ptr> can be migrated to a child <descriptivenote>/<p>. <origination> and <repository> present a larger challenge. In EAD3, these elements require <address> (in the case of <repository>), <corpname>, <famname>,<name>, or <persname> to be used as children. Thus, <ref> and/or <ptr> can be preserved by embedding them in one of these child elements, but manual remediation will likely be required to determine which child element should be used.

Data Remediation Concerns
Data remediation for <extrefloc> should be minimal to nonexistent unless <origination> was used as a child. In this case, the repository should carefully examine whether the information in <origination> should be retained and, if so, how it should be structured. If <origination>'s use was standardized, it should be possible to partially automate data cleanup using either scripting or XSL. If use was not standardized, guides will most likely need to be examined individually and <origination> reformatted appropriately.

Similarly, remediation for <linkgrp> should be simple unless <origination> or <repository> were used as children. In this case, if the repository wishes to retain the affected information, it will need to undertake pre-migration work to embed it, with or without the <origination> element, into an appropriate child element. It is unlikely that this work can be automated, so a significant investment of staff time may be necessary if numerous guides are involved.

Section 10: List Elements

10.1: Overview

The biggest migration issue associated with list elements in EAD3 is that date ranges associated with <chronitem> are required to be encoded using either <daterange> or <datesingle> rather than EAD2002's simpler <date>. If the repository used @normal, this change can be accommodated fairly easily; otherwise, either a stopgap solution or substantial remediation will be required. List elements can also be used as children of fewer elements in

EAD3 than they could be in EAD2002. Thus, substantial remediation work may be required if they often appear in configurations not valid once migrated to EAD3.

10.2: <chronlist>, <chronitem>, and <chronitemset>

Migration Strategies and Concerns

Most migration issues associated with these elements are occasioned by the fact that EAD3 requires the dates associated with <chronitem> to be contained in either <daterange> or <datesingle> rather than in EAD2002's less specific <date>. If the migrating repository routinely used @normal as part of its date elements, the stylesheet is able to use this information to create either <daterange> or <datesingle> elements as needed. If @normal does not appear, <date> can be mapped to either <daterange> or <datesingle> depending on the repository's preference, but this procedure will either cause single dates to be encoded with <daterange> or date ranges to be encoded with <datesingle> (the stylesheet's default behavior). Thus, while the resulting EAD3 guide will be valid, either <daterange> or <datesingle> will be used improperly.

EAD3 also does not allow <chronlist> as a subelement of <event> or <item>, meaning that <chronlist> cannot be nested in <chronlist> or <list>. Repositories that have used <chronlist> in this fashion should consider carefully whether they wish to retain <chronlist>'s contents and if so, how.

Data Remediation Concerns

The amount of remediation required for this element depends on the frequency of <chronlist>'s use, appearance of @normal, and the migrating repository's tolerance for <datesingle> and/or <daterange> being used improperly in some instances. If <chronlist> was used infrequently, @normal was used consistently, and/or the repository is willing to accept improper but valid use of <datesingle> or <daterange>, necessary remediation should be minimal. If, however, <chronlist> was used frequently and/or the repository did not use @normal but wishes to deploy <datesingle> and <daterange> correctly, remediation will be more involved.

10.3: <list>, <item>, and <defitem>

Migration Strategies and Concerns

Because the structure and function of <list> is very similar in EAD2002 and EAD3, it should generally be possible to migrate <list> and its subelements <item> and <defitem> directly. EAD3, however, does not allow several parent-child relationships permitted in EAD2002. Specifically, <list> can no longer be used as a child of <ref> and <address>, <archref>, <bibref>, <blockquote>, <chronlist>, <note>, <origination>, <repository>, <table>, <unitdate>, and <unittitle> can no longer be used as children of <item>. If the migrating repository has employed any of these relationships regularly, it should carefully consider its migration path in order to ensure that all affected data migrates acceptably.

Repositories should also be aware that <list>'s @listtype (@type in EAD2002), @numeration, and @mark attributes have changed slightly between EAD2002 and EAD3. Neither "simple" nor "marked" are valid values for @listtype in EAD3; however, both of these types can be mapped to "unordered." Because both @numeration and @mark have more values in EAD3 than in EAD2002, repositories may wish to consider implementing a new value at the time of migration. Finally, repositories may wish to consider adding the optional @script and @lang attributes to <list>, <item>, and <defitem> if they can be assigned in bulk.

Data Remediation Concerns
These elements should require minimal remediation unless one of the parent-child configurations described above applies. In these cases, the migrating repository should carefully consider whether it wishes to preserve the affected information and, if so, whether it wishes to migrate it to a new subelement or simply insert it into the parent element as free text. Generally speaking, it should be possible to automate the vast majority of this remediation if the problematic structure was employed consistently.

Section 11: Table Elements

11.1: Overview

In terms of structure, <table> and its associated elements are nearly identical in EAD2002 and EAD3. <table>'s use is, however, more restricted in EAD3 than it was in EAD2002. In cases where <table> was used as a child of an element in EAD2002 whose EAD3 counterpart will not accept it, the migrating repository will need to carefully consider whether the data needs to be retained and, if so, how it should be structured.

11.2: <table>

Migration Strategies and Concerns
Structurally speaking, EAD2002's <table> is virtually identical to EAD3's <table> with the exception that tables can be nested in EAD2002 but not in EAD3. EAD3 is slightly more restrictive regarding <table>'s placement, no longer permitting it to be used as a subelement of <event>, <item>, <p>, or <ref>. Because these elements can be used in a wide variety of circumstances, there is no one clear forward migration path for <table> in these circumstances.

Data Remediation Concerns
<table> should require no remediation unless it was used as a child of <event>, <item>, <p>, or <ref>. In these cases, the migrating repository will need to examine each affected guide to determine whether the information needs to be retained and if so, how it should be structured. Unfortunately, it will most likely prove impossible to automate this work.

11.3: <entry>

Migration Strategies and Concerns

<entry> can only be used as a child of <row> in both EAD2002 and EAD3. EAD3, however, no longer allows <address>, <archref>, <bibref>, <repository>, <unitdate>, and <unittitle> to be used as children of <entry>. Generally speaking, issues surrounding these changes are easy to resolve. <archref> and <bibref> can be retained by converting them to <ref>; <unitdate> can be retained by converting it to <date>; and <unittitle> can be retained by converting it to <title>. Because EAD3 allows neither <address> nor <p> as children of <entry>, either <address> can be stripped and the relevant information placed in <entry> as free text or the information can be discarded. Similarly, <repository> can be stripped and the affected information placed in <entry> as free text.

Data Remediation Concerns

This element should require little to no remediation unless <address> or <repository> were used as child elements. In this case, the repository should review <entry> to determine whether these elements and their associated information should be retained and, if so, ensure that it is clearly formatted. If these structures were used frequently, substantial time and effort may be required.

11.4: <tgroup>, <tbody>, <thead>, <colspec>, <row>

Migration Strategies and Concerns

Because these elements are essentially identical in EAD2002 and EAD3, it should be possible to migrate them directly from EAD2002 to EAD3.

Data Remediation Concerns

These elements are not expected to require remediation before or after migration.