JATS-CON 2012: How Well Do You Know Your Data? Converting an Archive of Proprietary Markup Schemes to JATS: A Case Study

Faye Krawitz, Jennifer McAndrews, and Richard O'Keeffe
Content Management Group, American Institute of Physics

Abstract

Converting data should be a straightforward project involving document analysis, markup mappings, developing conversion tools, running the transforms, quality control and...*voila*...a nice new collection of data! The American Institute of Physics (AIP) has recently completed converting its archival content collection to JATS. The content consisted of approximately 750,000 records in multiple generations of two base DTDs, one derived from ISO 12083 SGML, and one in proprietary XML. It was found that when facing content in multiple DTDs and created over the course of decades, tailored for specific repurposing, and checked by various automated processes, the data may still hold surprises. Quality control, staff, business requirements, markup — even within the same DTD — all change over time. As a result, your data may not be what you think it is. And, even if it is, its modern-day reuse scenarios significantly outnumber that of 10 years ago, let alone 20–25 years ago. This case study describes the project, the obstacles encountered, and the eventual success of the endeavor. The challenges included how to handle generated text, how to best represent metadata, and how much customization to add, if any, to the JATS implementation. But markup was not the only consideration. The importance of preparation, training resources, and the availability and application of XML technologies such as XSLT, XPATH and Schematron to facilitate the process also played an important role. For AIP, an organization with a history of successfully "doing it yourself" with homegrown tools, the paradigm shift to using standards-based tools was as much a challenge as the actual tag mappings.

I. Introduction

A. AIP: Who we are

The American Institute of Physics (AIP) is an organization of 10 physical science societies, representing more than 135,000 scientists, engineers, and educators. As one of the world's largest publishers of scientific information in physics, AIP employs innovative publishing technologies and offers publishing services for its Member Societies. AIP's suite of publications includes 15 journals, three of which are published in partnership with other organizations; magazines, including its flagship publication *Physics Today*; and the AIP Conference Proceedings series. Through its Physics Resources Center, AIP also delivers valuable services and expertise in education and student programs, science communications, government relations, career services for science and engineering professionals, statistical research, industrial outreach, and the history of physics and other sciences.

B. The AIP ecosystem

Historically, AIP relied on its own ISO 12083-derived SGML formats, one for short, abstract records, one for full-text records (since 1995), and a full-text XML tag set deployed in 2004. The markup served two critical functions:

1. as the source for the in-house Xyvision composition system; and more importantly;
2. as the source for AIP's own Scitation® hosting platform.

For years this content philosophy worked well, successfully serving the needs of our customers, in-house production unit, and online hosting of over 100 publications. However, business trends were rapidly changing with the focus no longer on specific products, but on the maintenance and reuse of the content itself. All new business proposals required NLM XML, distribution of deliverables in multiple formats, less emphasis on "print" format, more online functionality, and more rapid online publication schedules. This paired with the economies of scale brought about by the globalization of production operation services indicated that the AIP's approach was not sustainable long term.

What was the problem?

- the proprietary XML markup was “AIP-centric,” designed for specific AIP products, in particular the typeset page;
- it lacked true extensibility, obsolescing rapidly in today’s dynamic online publishing environment;
- data recipients and consumers wanted JATS/NLM XML, the *de facto* accepted industry markup.

As expected, the above had a detrimental effect on content production, archiving, and syndication costs. It was “Business 101”, adapt or be left behind by the STM community’s adoption of NLM XML, a trend that has led to its eventual adoption as the NISO JATS 1.0 standard.

Therefore, the decision to move to JATS XML for AIP production and archival content was the easiest part of the transition. Revising the entire AIP content philosophy and ecosystem proved to be the more difficult challenge.

Our task was to make JATS XML the core of AIP’s revised archival content strategy.

Beginning in summer 2010, AIP performed an in-depth review of its content strategy and future usage aspirations. The results were not surprising:

- Recognition of content as the most important asset by adopting a true non-product-centric data model for the AIP archival collection
- Invest in an up-to-date content management system to manage the data
- Standardization: adoption of a “standard”, i.e., better integrate with the STM community and “speak their language”
- Flexibility/agility: adopt XML that permitted greater data portability, more product options, and wider partnering opportunities
- Access to a wider array of XML resources wherein use of “standard” XML markup opens the door to a vast community of users for support and brainstorming. This applies not only to the tag set but also to XML technologies such as XSLT and Schematron.
- Retire AIP’s standard Physics and Astronomy Classification Scheme (PACS) in favor of a comprehensive thesaurus for richer semantic enrichment of the content
- Reorganize the AIP departments which handled content to more efficiently execute a unified strategy
- Eliminate costly proprietary dependencies:
    - Replace three archival formats with JATS XML and eliminate the associated data conversions and production time costs (See Fig. 1).
    - In conjunction with one XML format, eliminate the cost and support heavy infrastructure dependent on the multiple markup formats.
    - Streamline recursive markup modifications, particularly for post-publication corrections
    - Eliminate less extensible, custom-coded quality control checks in favor of using Schematron technology
    - Minimize specialized programming support for operational tasks by using XSLT to convert content. In effect, develop a well-rounded content management support staff trained to maintain content using standardized XML technologies, i.e., maximize staff potential and efficiency.

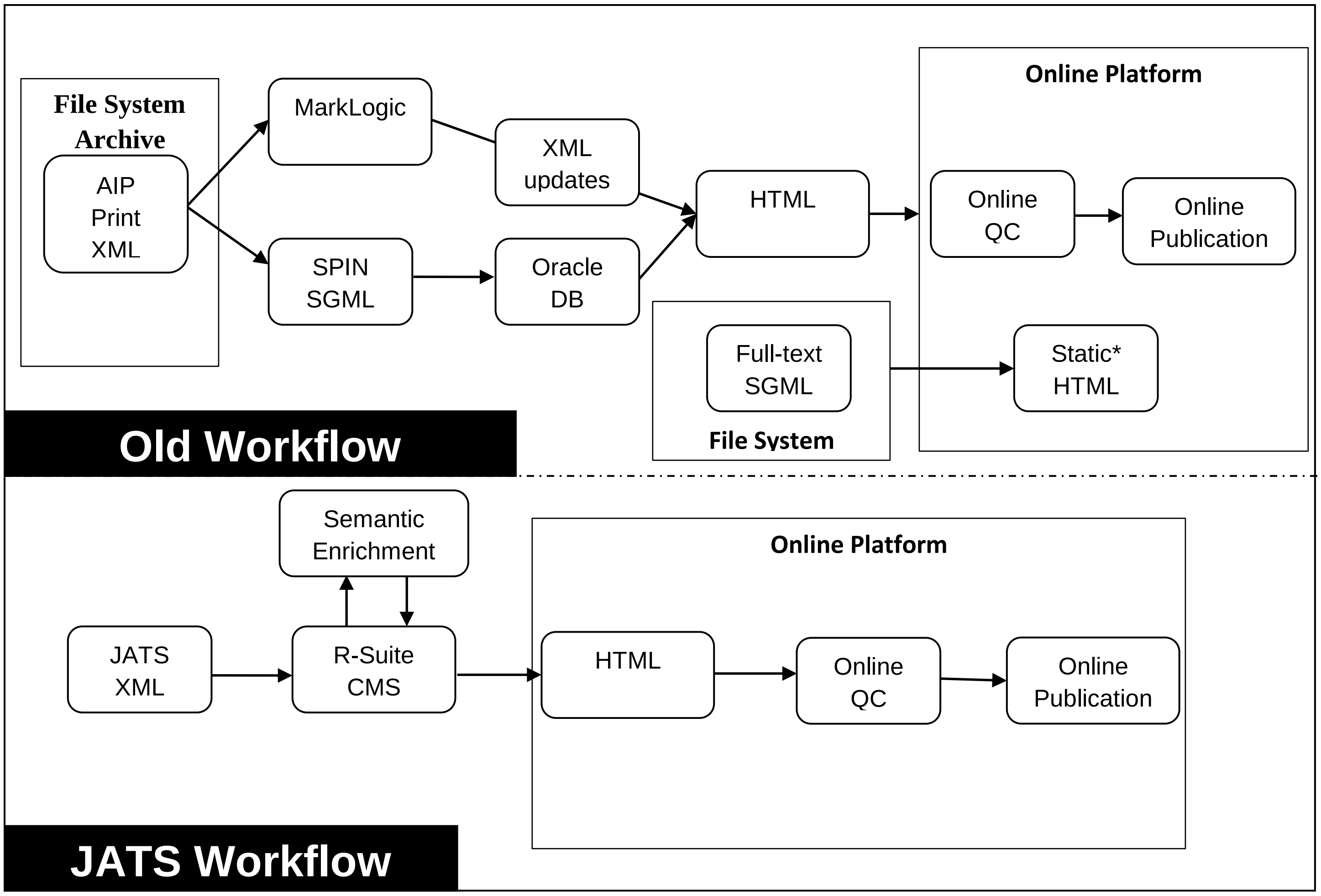


Fig. 1. AIP Workflows. *In some cases despite accurate XML, corrections or data upgrades were needed because of the additional conversions and database needs driving the production/online system. In other cases updates were needed in both the XML and SGML records, creating an environment where it was possible to have the content out of sync. Adopting the single JATS format for both full-text and abstract records eliminated this danger. Using a sophisticated content management system to manage the repurposing of the content enables the source XML to be properly managed.*

II. The Challenge

A. Make the plan known

Although the need to revamp the AIP content strategy was self-evident, the ramifications were monumental. The whole content production infrastructure needed to be revised. Every aspect from receipt of original material to final deliverable distribution had to be evaluated and rebuilt and this was just the system, also needed were training for internal and external production staff and partners to learn the new tools and JATS XML.

Considering the magnitude, there was little if any internal resistance. Really? How do you change everything and not have any significant resistance? After all, it's human nature to dislike substantial change and keep things as they are… especially in the workplace!

Well, how about communication?

Frequent updates and "State of AIP" staff meetings were conducted by the CEO, CIO, and AIP Publisher's Office to explain the reasons behind the changes and their importance to the organization's continued success and mission. Not everyone may have been happy but there was no doubt as to the "why" driving the tasks. It was globally understood that AIP's greatest asset was its content.

B. Set up your organization for success

With the premise for change accepted, it was important to restructure the organization to promote success. The existing organizational product-oriented paradigm created an environment where the needs of an online development group, a composition support group, and a production group occasionally competed resulting in conflicting content interpretations. This was resolved by uniting all into a single content technology group, establishing a clear chain of content responsibility which enabled sharper focus on AIP's new content philosophy and imbued a clear sense of ownership on the staff (see Fig. 2).

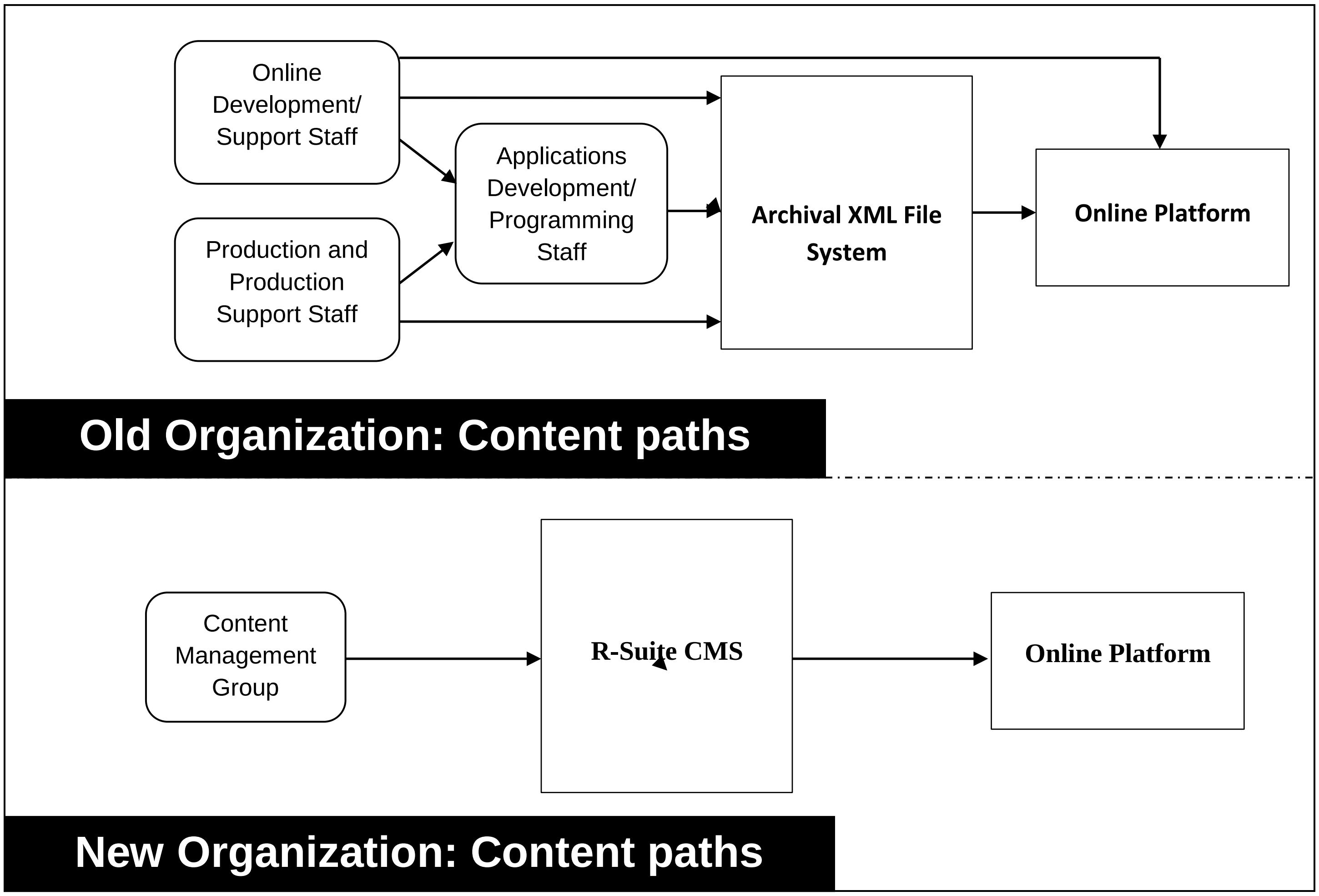

Online Development/ Support Staff
Applications Development/ Programming Staff
Production and Production Support Staff
Archival XML File System
Online Platform
Old Organization: Content paths
Content Management Group
R-Suite CMS
Online Platform
New Organization: Content paths

Fig. 2. Old Organization. *This illustrates the possible paths data could be passed to the archive file system and the online platform. Despite checks and balances, procedures, and protocols, it is clear that too many hands were in the mix. Inconsistencies could easily be introduced and version control was compromised.* **New Organization**. *The revised structure removes the multiple input streams, placing a singular governor on the process which now sets content and archival updating policy. No longer do self-contained unit silos unilaterally affect the content.*

C. The existing content

So what did the content strategy review and selection of JATS actually mean for the AIP archival content transformation?

- **Objective content interpretation:** First and foremost, it was liberating to shift the approach of our content review: the data was now considered the primary asset not just a means to an end. We
    - no longer needed to tag content based on print constraints;
    - no longer needed to tag content based on production workflow concerns;
    - no longer needed to tag based on internal conversion and online display constraints;
    - the large selection of JATS elements and attributes provided a wealth of markup with which to accurately tag content.
- **Source content format:** Three markup conventions to convert to JATS
    - Header SGML (based on ISO 12083 SGML);
    - Full-text SGML (based on ISO 12083 SGML);
    - Full-text XML (proprietary AIP markup inspired by ISO 12083 SGML and later by NLM XML).

  Of these, AIP made a conscious business decision to not include a full-text SGML transformation. It will be revisited later. Reason is that the bulk of archival content is in header SGML and each publication had at least five to six years of full-text XML. Therefore, the project consisted of the two transformations of header SGML and full-text XML.
- **Transformation:** Prepare two synchronized conversion mappings for XSLT transformation. This included identifying older markup that could be improved by the XSLT, identifying which content “problems” needed to be addressed in the source markup, e.g., inconsistencies, “placeholder tags”, etc., and flagging content which needed to be reviewed on a case-by-case basis.
- **Test and adapt:** Test XSLT and update based on “hidden” content permutations
- **Quality Control:** Write and develop quality control tools for three versions: migrated, production, and online export
- **Enrichment:** apply the AIP thesaurus to the content to semantically enrich the content
- **Document:** Prepare documentation for internal production staff, vendor partners, syndication recipients
- **Train:** Train internal production staff and external publishing partners to familiarize with AIP’s implementation of JATS XML

From the XML markup perspective, the most difficult transition was learning XPATH to properly define the mappings. For the first time, a semantic description was not sufficient. The past process of describing a specific tag-to-tag association of XyAscii or SGML to XML was no longer viable. The mapping was now a data “tree-to-tree” mapping with a thorough accounting of all possibilities. With a wealth of mixed content models and “or” statements, accounting for all tag combinations in the AIP tag sets required significant research! Each scenario needed to be explained and mapped using XPATH conventions. The good news: the scenarios could be mapped. The bad news: the data did not always reflect the scenarios expected! The bad data interpretation or markup was disappointing but it also highlighted opportunities to improve both the content and supporting infrastructure. The following sections describe this experience in greater detail.

III. The Process (The How)

A. Document analysis

There is no way the importance of a thorough document analysis can be overstated. This is the best opportunity to dig through the data, looking for consistencies, inconsistencies, and surprises. The more thorough the analysis, the less likely a data transform will grind to a halt due to unexpected data.

Because this step is so critical, AIP opted to perform all data analysis within house. While outsourced consultants may be well versed in analysis, it was felt that no one would care as deeply about the data or understand it as thoroughly as existing staff. For instance, those who deal daily with the data can readily identify instances of generated text and understand the important distinction between "Sec. A" in one journal, and "Section 1" in another, and would preserve that distinction rather than create a rule for standardizing.

By the same measure, a thorough analysis with an eye to converting an entire data set does provide an ideal opportunity to normalize certain styles and correct anomalies and inconsistencies in the data. Should both "References" title and "REFERENCES" be maintained? Or should these be standardized? These are decision to be made at the time of analysis. (And just for information, AIP opted to standardize on " REFERENCES.")

Perhaps the most important outcome of this analysis is the creation of a document map, or specification ("spec"), enumerating the translation requirements from one XML tag to another for the entirety of the existing input tag set. This document would be used to instruct and guide the individual writing the XSLT transform as well as all staff working on testing and quality assurance throughout the conversion process. The more robust this document is at the outset, the smoother (ideally!) the project would progress.

Ironically, with physics as the subject matter, it would be expected that the complex mathematics and tables would present the most trouble. After all, they were the most troublesome areas for composition! However, because AIP XML already used the MathML and OASIS XML standards, this complex content was simply passed through with little or no problem. It is a great illustration of the long-term benefit of adopting standards!

B. Institutional memory

When digging into any archive, expect to stumble upon data for which there is no written documentation. AIP counts itself fortunate to have been able to tap the collective memory of several employees, from both the production and the programming departments. Their years of service with AIP gave them the ability to provide answers to such mysterious questions as: what exactly does this string of numbers mean? what’s an `<mf>` tag? and is there anyone left alive who cares? Tapping into the long memory of AIP staff, we were able to learn that the mysterious string of numbers was a catalog number, and MF denoted “microfiche”. As to whether there is anyone still alive who uses microfiche, investigations are ongoing. In the end, the need to carry forward such old-school metadata as microfiche codes or Nuclear Science abstract numbers has been obviated by the easy accessibility of PDFs and the ubiquitous DOI, but you need to know what the strange things are before you can decide to ignore them.

C. Tagging principles

One of the most critical aspects of analysis is the determination of which tags from the JATS set to utilize, and how to apply them. In order to help make those decisions, AIP set itself a series of requirements:

- strict delineation of element vs. attribute and adherence to tagging (see Table 1)
- `@specific-use=”metadata”` reserved for indication of content outside of article-meta areas to be suppressed in print, e.g., ISSN and non-displaying DOIs in references.

- article-meta would be treated as database-like since this material is most likely to be atomized and/or have information shared with various library and database organizations. The end result was the front matter more closely followed the “blue” DTD style even though AIP chose to use the “green” DTD.
- All content starting with the `<body>` and including references would be considered as text with the appropriate markup added. In particular, using the `<label>` element and not generating any text.
- customized content models and modules would be avoided — starting from, essentially, the ground floor, AIP moved forward on the belief that the JATS tag set was robust enough to accommodate all conditions. If, at some date in the future, data requirements necessitate customization, so be it, but the data in our archive should be consistent enough and predictable enough that all material should be handled using existing elements
- `<x>` markup would be avoided under the same principles as customization and reserved for use as an internal means of identifying areas requiring further review; and
- `<named-content>` would be reserved for semantic enrichment markup — with an eye to being able to search any given file and obtain a reliable/complete/concise list of expressions, names, keywords.

Table 1. Content Tagging Principles

• Elements: Use an element to tag what is “seen”, i.e., visible content.
• Attributes: Use an attribute to describe data of the element or about publishing.
• Define attribute values consistently: all lowercase, hyphens instead of underscores
• All true content that is rendered should be present and visible in the XML.
• Stick with the facts…“Say what you know.” ○ What are the facts you know about the content. Do not manipulate content with “filler” information to suit a process. For example, do not set a bogus “day” in a date if there is no day as commonly found with cover dates.
• Always strive to name the content for what it is. ○ That is, use the markup that best identifies the content. For example, a running head is a print artifact and the shortened title is really, from a content perspective, a “short title” not a running head.

D. Correct known ambiguities via preprocessing or XSLT

Any thorough analysis of data is likely to turn up areas for improvement. Perhaps in the past decisions were made to push through the system a particular piece of data presumed to be a one-off, only to later learn that what was thought to be a one-off was in fact a new structure requiring a long term solution. This resulted in unexpected source tagging for similar structures. Perhaps a tag will reveal itself to be used in ambiguous ways. Conversion to a new tag set is one of the best times to address these instances. In AIP’s case, a trio of tags appeared in our data set that nicely illustrates the need for precise/expressive tagging. Our data included the non-specific tags `<extra1>`, `<extra2>`, and `<extra3>`. Any examination by an outsider would — hopefully — reveal that extra tags contained author suffix information such as Jr. or author degree information such as Ph.D. But nothing in the text would guarantee the revelation of the *difference* between extras 1, 2, and 3. The move to JATS provided the opportunity to remove ambiguities such as these.

Unfortunately, not all ambiguities were even recognizable. The AIP SGML short record (bibliographic record) collection contained the troublesome tag `<prevau>`. `<prevau>` — representative of "previous author" — was the authorial equivalent of its long-standing print counterpart *ibid.* — which also appeared in short, bibliographic records. In analyzing occurrences of `<prevau>`, however, it was discovered that somewhere through the years, vendors and staff had stopped including the actual author and publication information in conjunction with *ibid.*-type markup, and instead simply inserted `<prevau>` and/or `<ibid>` as needed to indicate to the reader that the attribution for a particular reference could be found in the reference immediately preceding. While this approach worked for print, it created an electronic information void — leading to difficulties in accurately determining how many times a particular author's paper was cited and conflicts with DOI lookups because the author names are not present. Conversion was the perfect time to address these. After much consideration of whether to correct for this issue using a preprocess or to include the correction in the conversion, it was decided to implement the correction for this through complex XSLT (see Table 2).

Table 2. Rules for substituting full author content

SPIN SGML	Rule
reflist/refitem/ /prevau	collect and add `<biaugrp>` content from the last inclusion within a `<refitem>` in the same `<reflist>`
reflist/refitem/ /prevau	If no previous `<refitem>` in the same `<reflist>`; flag as error condition
ref/ /prevau	collect and add `<biaugrp>` content from the previous `<citation>`/`<ref>` in `<biblist>` If no usage of `<biaugrp>` in the previous `<citation>`/`<ref>`; flag as an exception for content review. If previous citation is a `<reflist>`: `<citation>`/`<reflist>`; flag as an exception for content review
Error Condition output	For exceptions, output content as `<x specific-use="database">`prevau`</x>`

This decision and implementation resulted in a more thorough, better defined data set.

Still one more ambiguous area was to be found in appendix heading and section markup. Working with a variety of different journals with different editorial styles resulted in a mixed-bag of section and appendix designators. It was compounded by AIP's open DTD content model and forgiving composition formatting rules which permitted a variety of permutations to produce the same "visible" display. The result was what should have been a straightforward transformation of section titles and ID values for labels, was anything but. Any rule for appendix section, subsections, their `@id` values, and child `<sectitle>` tags needed to avoid conflicting with usage within the `<body>`. There were special instructions for title handling. And then there was a list of "exceptions." This variety of situation-dependent requirements in appendices gave rise the newly notorious "crazy appendix mode" (see Table 3):

Table 3. Sampling of "crazy appendix mode" ID and label rules and considerations

IDs:

a. Change prefix of @id from "x" or "x" to "app" in `<app>`, and any child `<section>` and `<subsect#>` tags for all instances except coden JPCRBU which uses "s" section prefix; this rule applies to the appendix and all sections and subsects in the appendix.

b. If AIP `<appendix>` has no `@id` and is not equal to coden JPCRBU: Take the value of the first `<section>` `@id`. If no `@id`, do not generate

LABELS

***Appendix:** Do not generate text "Appendix:"*

Section & Subsection label numbering:

*a. **If child of** <appendix> **has** `@id`: generate JATS <label> as "1.", "2."… based on sequential order within that appendix*

*b. **If grandchild of** `<appendix>` **has** `@id`**:** generate JATS `<label>` as "a.", "b."… based on sequential order within that appendix/section.*

c. If no `@id` do not generate label

d. When JATS `<app>` has borrowed `@id` from first section child do not generate label for section even though it has one.

Finally, we needed a way to track material the XSLT was passing straight through without special tagging, places where previously placeholders for untranslated content appeared (marked as `<atother>`@qL`</atother>`) and places such as lists where no label style for ordered lists was given. To facilitate later location of these problem areas, we selected a tagging scheme of `<x>` and `<x><strike></strike></x>`. Because those tags are not used in our conversion, any archive search for `<x>` or `<x><strike></strike></x>` will generate a list of data requiring manual attention.

Overall, the decision of whether to correct the source or address in the XSLT transform was based on the how pervasive the problem was in the corpus of data. If logic could be applied for a finite set of scenarios, then the XSLT was the solution. If the problem was a small subset of files, which could be updated in a short period of time, the source was fixed — especially, if an XSLT solution ran the risk of introducing complexity and potential conflict with perfectly functioning existing rules.

IV. Expected Trouble Spots

Generated text. Coming out of analysis, there were several situations we knew would present a challenge. First and foremost, in fact noted before any official analysis had begun, was the problem of generated text. After evaluation, the option chosen was to leverage an existing tool from AIP's decommissioned composition workflow. The Xyvision composition system allowed Perl script plug-ins (XyPerl) and AIP had used such an internal Perl script to output generated text into the typeset files.

For example, where once a reference could be tagged as `<citeref rids="c1">` and a XyPerl script could be relied upon to output and superscript the "1" in text, moving into the JATS model meant finding a method for replacing all the markup that previously generated text. That is, where once `<citeref rids="c1">` would be all the reference to reference 1 to be found in the XML, the JATS approach required a tagging of `<xref ref-type="bibr" rid="c1"><sup>`1`</sup></xref>`. The need to replace generated text markers with actual text also occurred in section tags, where the word "Section" was generated, in figure reference tagging to generate "Fig." "Figure" or "FIG" depending upon a specific journal's style, and within list labels, where once again specific and differing journal styles needed to be taken into account.

Rather than writing complex XSLT to read the ID of the cross-reference and output the appropriately-styled content, AIP was able to leverage its existing Perl-generated-text scripts (and once again tap institutional memory to understand the limits and strengths of those scripts) to generate JATS `<xref>` content in a two-step process. By running the slightly altered Perl against existing AIP XML files and populating those files with the generated text, all labels, cite numbers, section head and so forth would be in place for conversion to JATS.

Mixed content models. Different journals maintained different styles, carried different copyrights, organized article sections differently. Each of these variants needed to be accounted for. One of the most challenging in this category was the identification and punctuation of cumulative references. Because of style differences, the division between references in a list varied between periods, to semi-colons, sometimes inside tags other times outside. Each permutation need to be accounted for, with punctuation and untagged spacing on the input moving inside JATS `<mixed-citation>` tags. Again, some complex XSLT was required to accommodate these differences:

AIP source:

```
    <citation id="c5">
          <ref id="c5a"><jcite>…</jcite></ref>
          ;
          <ref id="c5b" type="ibid"><jcite>…</jcite></ref>
          ;
          <ref id="c5c"><jcite>…</jcite></ref>
    </citation>
```

Target JATS:

```
    <ref id="c5">
    <label>5.</label>
       <mixed-citation id="c5a" publication-type="journal" >…; </mixed-citation>
       <mixed-citation id="c5b" publication-type="journal">…; </mixed-citation>
       <mixed-citation id="c5c" publication-type="journal">….</mixed-citation>
    </ref>
```

***SGML inclusions*.** A further problem we realized we would face once analysis began was the finite list of available tags in our SGML as compared to our XML. The XML tag set being much richer meant some of the refinements/precision common to the XML tagging was lost on earlier translations to SGML, with several diverse XML tags all converting to one/same SGML tag. This was readily apparent in our reference section, where much of the detail of our XML was captured in a catch-all `<othinfo>` tag. Mapping for conversion from SGML to JATS meant accounting for each of the `<othinfo>` scenarios and constructing precise XPATH for each potential situation so no data was lost (see Fig. 3).

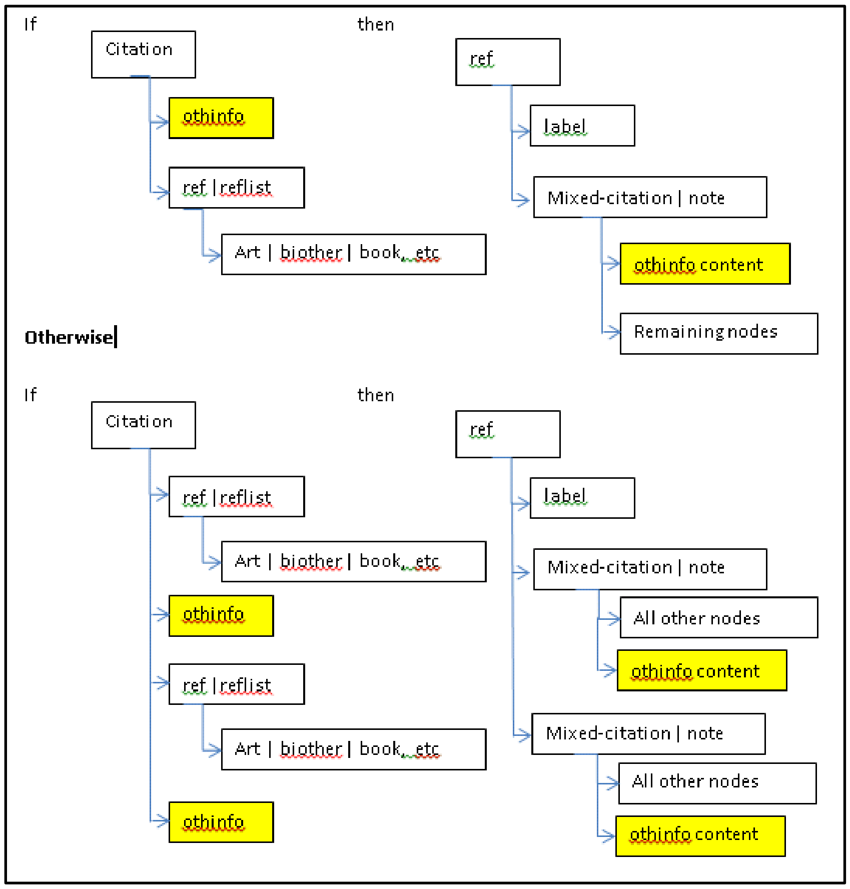
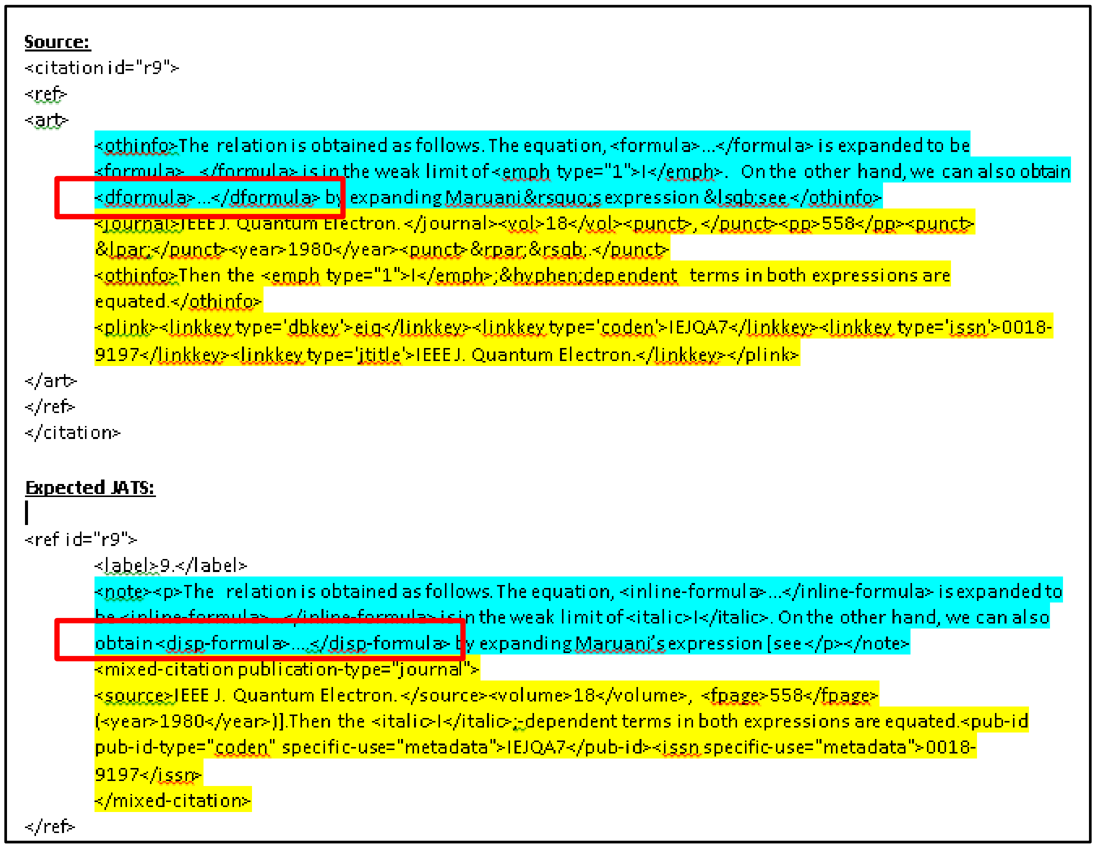


Fig. 3. <othinfo> variations. *Decisions needed to be made as to where to place the "other information" wherever it appeared.*

In some instances the `<othinfo>` needed to have its content moved and then the tag stripped, but in others the `<othinfo>` tag was simply stripped, while in still others the `<othinfo>` tag needed to transform to `<note><p>` if a displayed formulas was present within the reference (see Fig. 4). As a result, not only did the mapping require precision, the order of execution in the XSLT needed careful attention as well.



Source:

```
<citation id="r9">
<ref>
<art>
        <othinfo>The relation is obtained as follows. The equation, <formula>...</formula> is expanded to be
        <formula>...</formula> is in the weak limit of <emph type="1">I</emph>.  On the other hand, we can also obtain
        <dformula>...</dformula> by expanding Maruani&rsquo;s expression &lsqb;see </othinfo>
        <journal>IEEE J. Quantum Electron. </journal><vol>18</vol><punct>, </punct><pp>558</pp><punct>
        &lpar;</punct><year>1980</year><punct>&rpar;&rsqb;.</punct>
        <othinfo>Then the <emph type="1">I</emph>;&hyphen;dependent  terms in both expressions are
        equated.</othinfo>
        <plink><linkkey type='dbkey'>eiq</linkkey><linkkey type='coden'>IEJQA7</linkkey><linkkey type='issn'>0018-
        9197</linkkey><linkkey type='jtitle'>IEEE J. Quantum Electron.</linkkey></plink>
</art>
</ref>
</citation>
```

Expected JATS:

```
<ref id="r9">
        <label>9.</label>
        <note><p>The  relation is obtained as follows. The equation, <inline-formula>...</inline-formula> is expanded to
        be <inline-formula>...</inline-formula> is in the weak limit of <italic>I</italic>. On the other hand, we can also
        obtain <disp-formula>....</disp-formula> by expanding Maruani's expression [see </p></note>
        <mixed-citation publication-type="journal">
        <source>IEEE J. Quantum Electron. </source><volume>18</volume>,  <fpage>558</fpage>
        (<year>1980</year>)].Then the <italic>I</italic>;-dependent terms in both expressions are equated.<pub-id
        pub-id-type="coden" specific-use="metadata">IEJQA7</pub-id><issn specific-use="metadata">0018-
        9197</issn>
        </mixed-citation>
</ref>
```

Fig. 4. <othinfo> with displayed formula. *To create valid JATS, references with displayed formulas needed to transform in either a <note>, or if mixed with citable content, a <note> and <mixed-citation>.*

Multimedia. We're looking at an archive that developed along with technology, and over the course of that development, as technology changed, so did our multimedia solutions, progressing from handling "video" as supplementary material to embedding the media file directly in the XML / web page. Each step along that progression resulted in variations in our tagging.

Time. All activities — analysis, transformations, testing, etc. — would need to take place in addition to existing daily maintenance of workflows, support calls, meetings, emails, and all the many and sundry tasks staff was already responsible for. Support of management would be critical in allowing content management team to realign priorities to allow JATS conversion work to take precedence.

V. Unexpected Trouble Spots

Having completed a thorough analysis and identified areas we knew would be problematic, it should have been smooth sailing, right? Of course wrong.

Hurdle #1: language. Over the years the content management team had become accustomed to expressing content paths in simplified terms for communicating with the Xyvision formatting staff. With the mapping specifics for XML / SGML to JATS, and the resultant XSLT translation, the lack of fluency in the more rigorous XPATH presented a problem communicating conversion requirements in early iterations of the specification.

Hurdle #2: (rotten) Easter Eggs. Any data set as large as the one AIP dealt with was bound to contain surprises. Over the years and through various data preparation vendors, a number of unseen errors had wormed their way into the data, errors not flagged by the QC tools in place.

Examples:

- a reference in which *ibid.* was used for display purposes should also have rightly contained all the specifics of the repeated (*ibid.*) information. Several cases were found where *ibid.* was used in lieu of, not in addition to, journal title and author detail.
- vendors had not been including electronic identifier information in articles. Since the electronic identifier was historically database information and never displayed, its absence went unnoticed until the conversion to JATS revealed the oversight.
- Conversion rules written to handle `<p content-type="leadpara">` yielded zero results. Why? Rather than use meaningful attribute of "`leadpara`" to generate required bolding on our displays, vendor simply tagged lead paragraphs as plain bold. This necessitated correction to input files before conversion was run guaranteeing the material was properly tagged in JATS and thus allowing for accurate online display and reuse of information.
- Feedback from online platform hosts led to inclusion of previously unused elements, e.g., `<publisher-name specific-use="short-name">`, `<self-uri>`. These new-to-conversion elements needed to be mapped on the specification and added to the XSLT.
- Server time needed to be reserved due to the large blocks of time needed to transform the data set via XSLT. The collection of bibliographic records alone required four weeks to complete, with full text conversion time topping out at 10 days.

VI. Quality Control and Testing

A. Recommended prerequisite

A valid or well-structured file does not always equate to the desired output. The testing and QC of files is a critical piece and should not wait until the completion of the XSLT but rather start early in the process. As a prerequisite to the conversion project AIP chose to expand the knowledge and skills of the entire QC and content management team. The skills included:

- **Training for NLM DTD construction and module customization**: this is necessary for understanding the in-depth tagging and customization possibilities in keeping with the NISO standards.

- **Professional two-day training class which included**

  - **Understanding XPATH:** XPATH is the foundation for both XSLT and Schematron development. This knowledge gave us the ability to write clearer and more concise mapping instructions for the XSLT developer. For example, rather than write the instruction map AIP `<graphic>` to JATS `<graphic>`, an XPATH expression and combination of semantic description was written:

    "Convert AIP tag `<graphic>` to JATS `<graphic>` if it's a child of dformula/artwork | entry/artwork | figure else convert to `<inline-graphic>`*."*

The proper use of “if,” “and,” “or,” “else,” and “otherwise” statements along with precise axis and node identification significantly enhanced the communication with the developer.

- **XSLT development:** Taught enough skills to understand the limits and strengths of XSLT programming and the ability to read and write simple to moderate code.

- **Schematron development:** Gave our staff a way to create our own QC rules in order to check specific usage of AIP’s JATS tagging. This new skill allowed the content staff to not rely on programmers and have full control of what is to be checked in files.

B. Quality control and testing: Content and tagging checks

Once the specification was complete and programming underway, the initial thought at this time was that the hard work was completed, after all how many links can a front matter portion of document really have? While there is extreme satisfaction seeing the first converted valid files the reality soon set in that no matter how hard one tries to document every scenario, it is impossible when dealing with 750,000 files converted over 20+ years. It’s not that your data is necessarily bad, it’s that business rules, publishing styles, requirements, and technology change over the years. The good news at this point is that the knowledge and skills are in place to check and report any findings.

Step 1: Preliminary testing. This step was going on simultaneously while the XSLT was being written and was performed by the document analyst. The programmer would report when a block of items were completed and the writer would run files by hand checking that the programmer understood the requirements and the XSLT functioned as needed. Daily team meetings were held to discuss any new findings or clarify instructions.
Note: It became apparent in this step that our specification document was too simple. For example, an instruction written as map AIP tag `<media-object>` to JATS `<media>` and copy over the `@id` required more detail. What we meant to say was to do this mapping but only when the AIP tag `<media-object>` had the `@version=”original”`. Incorporating our recent XSLT and XPATH class came in handy and the instruction was changed to an “if, “then”, “else” statement.

```
if media[@id]/media-object[@version=”original”] then media[@id]
else do not generate JATS @id
```

A step back was taken at this point to rewrite items that were not yet programmed. Taking the extra time to write using XPATH logic turned out to be a time saver and forced us to think of all scenarios of a given tag.

Step 2: Batch processing. When the XSLT was completed, the first task was to run a batch of approximately 200 files from a representative sample of journals through the XSLT conversion. The processor was set to log any files that were not valid or did not make it through the conversion. Reports were sent to the content management group for investigation. The error reporting allowed the content management group to find hidden problems and areas of new rules which needed to be incorporated into the XSLT.

Hidden problems required decisions in determining whether it’s easier to fix the source files or make XSLT modifications. There were some instances where it was impossible to capture the desired JATS tagging. For example, one trouble spot was with our multimedia tagging. The tagging we have today evolved from very basic tagging, some of our earlier files had the tagging but the multimedia file was missing, or we found multimedia files and tagging missing. Digging even deeper led us to discover that PDF links to the multimedia were also incorrect. We soon had a small project in itself to fix the source files.

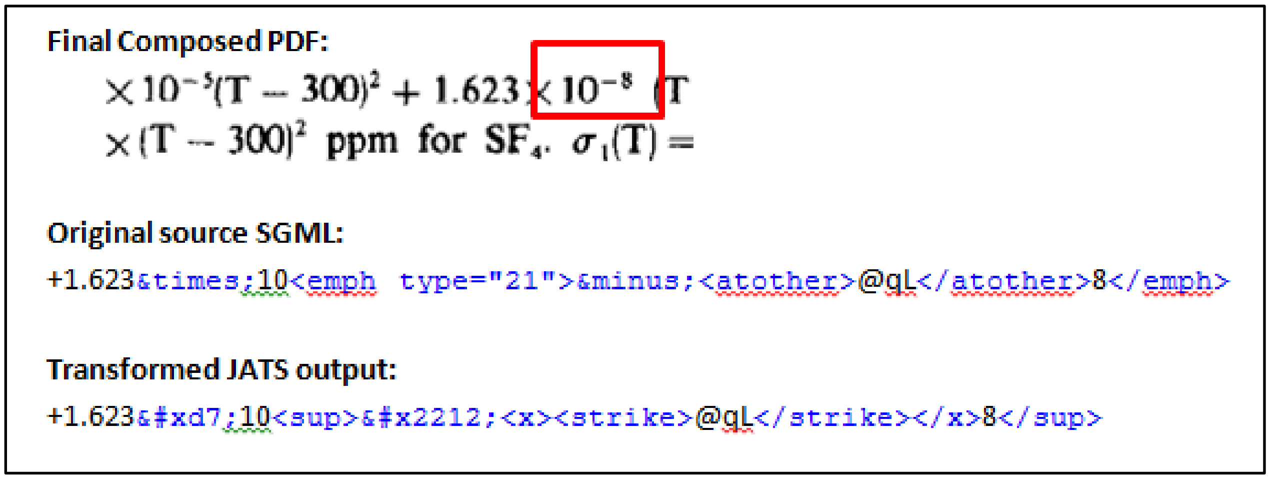
Step 3: Group testing. When it was determined that files were converting cleanly without parsing errors the next step was to focus more on the content. This meant checking there wasn't dropped text, JATS tags for all our different article types were in place and Schematron was run. This step was a group effort where our entire team performed the QC. Again approximately 200 files were run from several different journals spanning several years, with different article types. Each team member checked the "same group" of files and the cycle began pointing out hidden problems.

Step 4: Bulk processing. The entire corpus of content was run through the XSLTs and any remaining errors were addressed. At this stage errors resulted from bad source outliers. Simply correcting the source and rerunning corrected the files.

Overall, the XSLT transforms had well over a 99% success rate. However, with 750,000+ files, even this small a percentage of errors meant a large number of files needed to be manually inspected.

Step 5: Final cleanup — Analyze flagged data. Decisions were made early in the project planning process to include the tags `<x>` or `<strike>` for situations that could not be mapped in the XSLT. An example of a tag mapped to the `<strike>` would be the AIP `<atother>` tag primarily used to mark unknown characters. These tags were set with the knowledge that a manual decision would be required once the conversion was complete. At this step a list of articles is supplied to the content team by the programming staff. Each item needs to be investigated and fixed accordingly.

See Fig. 5 for an illustration of the `<atother>` dilemma. Unfortunately, it could be an unknown visible character, or as in this example, a mystery as there is no visible character!



Final Composed PDF:

$$\times 10^{-5}(T-300)^2 + 1.623 \boxed{\times 10^{-8}} (T$$
$$\times (T-300)^2 \text{ ppm for } SF_4. \ \sigma_1(T) =$$

Original source SGML:

```
+1.623&times;10<emph type="21">&minus;<atother>@qL</atother>8</emph>
```

Transformed JATS output:

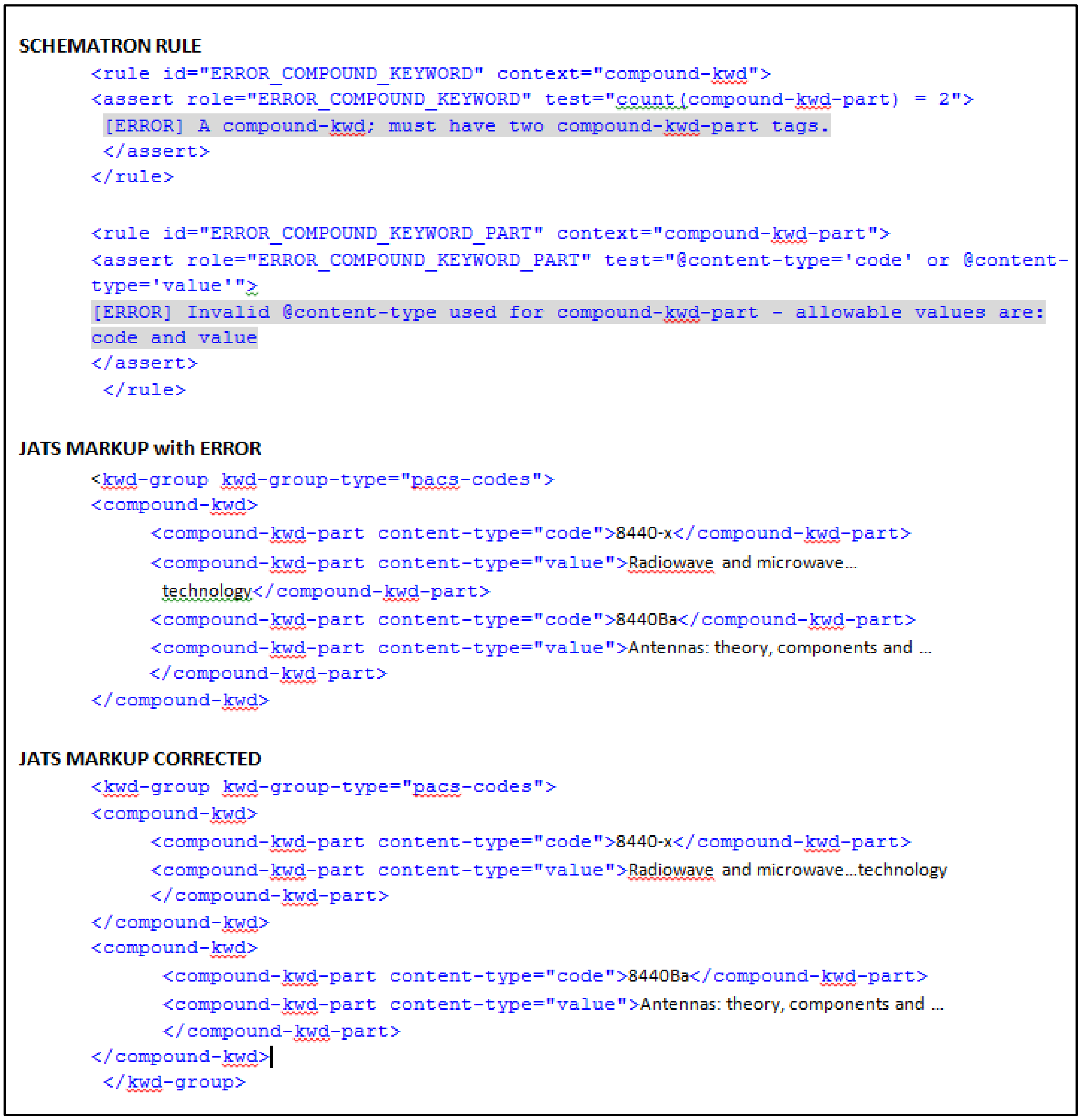
```
+1.623&#xd7;10<sup>&#x2212;<x><strike>@qL</strike></x>8</sup>
```

Fig. 5. <atother> flagged data.

C. Incorporating Schematron

The creation of the Schematron was done in parallel while the XSLT was being written and has now become the central piece in our QC process. It is run on valid files and tracks ERRORs and WARNINGs specific to our data. As discussed, valid files don't always equate to correct files. For example, after running Schematron on an early converted batch an error was detected where compound keywords, although valid, were not tagged as pairs of compound keyword parts. The files were fixed, rerun and checked again.

See Fig. 6 which illustrates the Schematron errors for `<compound-kwd>` and `<compound-kwd-part>` tags. The gray shading is the error that will be reported to the user if the error exists.



SCHEMATRON RULE

```
<rule id="ERROR_COMPOUND_KEYWORD" context="compound-kwd">
<assert role="ERROR_COMPOUND_KEYWORD" test="count(compound-kwd-part) = 2">
 [ERROR] A compound-kwd; must have two compound-kwd-part tags.
 </assert>
</rule>

<rule id="ERROR_COMPOUND_KEYWORD_PART" context="compound-kwd-part">
<assert role="ERROR_COMPOUND_KEYWORD_PART" test="@content-type='code' or @content-
type='value'">
[ERROR] Invalid @content-type used for compound-kwd-part - allowable values are:
code and value
</assert>
 </rule>
```

JATS MARKUP with ERROR

```
<kwd-group kwd-group-type="pacs-codes">
<compound-kwd>
     <compound-kwd-part content-type="code">8440-x</compound-kwd-part>
     <compound-kwd-part content-type="value">Radiowave and microwave…
      technology</compound-kwd-part>
     <compound-kwd-part content-type="code">8440Ba</compound-kwd-part>
     <compound-kwd-part content-type="value">Antennas: theory, components and …
     </compound-kwd-part>
</compound-kwd>
```

JATS MARKUP CORRECTED

```
<kwd-group kwd-group-type="pacs-codes">
<compound-kwd>
     <compound-kwd-part content-type="code">8440-x</compound-kwd-part>
     <compound-kwd-part content-type="value">Radiowave and microwave…technology
     </compound-kwd-part>
</compound-kwd>
<compound-kwd>
      <compound-kwd-part content-type="code">8440Ba</compound-kwd-part>
      <compound-kwd-part content-type="value">Antennas: theory, components and …
      </compound-kwd-part>
</compound-kwd>
 </kwd-group>
```

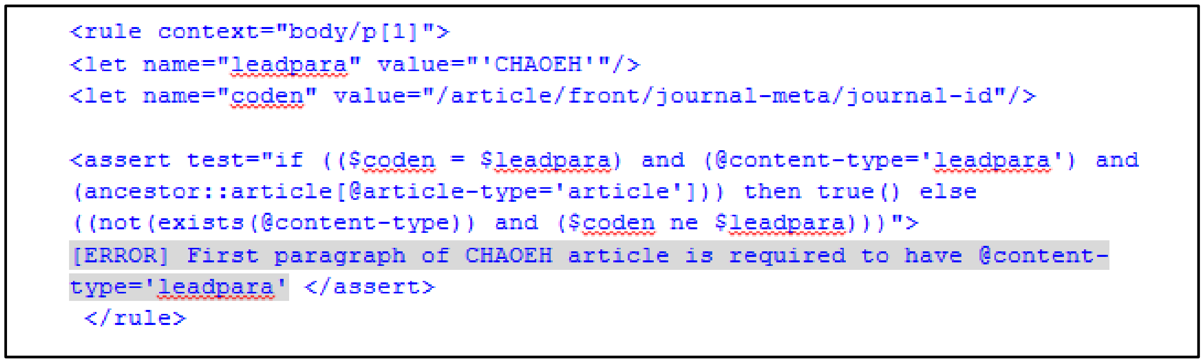
Fig. 6. <compound-kwd> and Schematron.

D. Quality control and testing: Online displays

This final phase involved several departments viewing the files online. In addition to our content team, online publishing group and random testers throughout our organization participated. Everyone was assigned a list of journals with a checklist of items to follow during the QC. The assumption at this point was that the data was parsed and all our AIP rules followed. We had an additional challenge here because AIP has recently switched over to a new online host so any oddities had to be determined if they were a result of our new host platform or JATS markup.

Although the errors decrease along the way, more items were reported at this phase which is more apparent when viewing. The error previously discussed in this paper regarding the missing `@content-type="leadpara"` in our *Chaos* journal was discovered at this point. The tag `<p content-type="leadpara">` was required in the first paragraph in the body. This paragraph was missing from the online abstract page do to this missing attribute.

As expected, a Schematron check was right away put in place specifically for the check. See in Fig. 7 the shaded error that will come out if the attribute is missing.



```
<rule context="body/p[1]">
<let name="leadpara" value="'CHAOEH'"/>
<let name="coden" value="/article/front/journal-meta/journal-id"/>

<assert test="if (($coden = $leadpara) and (@content-type='leadpara') and
(ancestor::article[@article-type='article'])) then true() else
((not(exists(@content-type)) and ($coden ne $leadpara)))">
[ERROR] First paragraph of CHAOEH article is required to have @content-
type='leadpara' </assert>
 </rule>
```

Fig. 7. Lead paragraph checking with Schematron.

VII. Lessons Learned and General Conclusions

The task of designing the conversion mappings, and analyzing our data over the past 20+ years was an extremely big challenge and far surpassed anything we imagined. The expected problem areas and those that actually manifested in the data were quite surprising. However, the experience really provided an excellent foundation for the development of more rigid quality control and data preparation measures. Our XSLT together with our Schematron are now the key, robust tools that not only have led to a solid conversion and QC process for our AIP to JATS conversion, but will serve us well into the future. Taking the time to receive training in advance was extremely beneficial. It taught us that if logic can be expressed as an XPATH statement, then it can be converted. It is surprising how many seemingly complex "editorial" variations can be handled by simple logic when viewed from data tree or node perspective.

Other lessons:

- Document analysis, document analysis, document analysis…what the data is supposed to be and what it is can be two different things. Composition templates and online style sheets may be masking underlying markup errors.
- Take advantage of the opportunity to correct the data; it's not very often that such a comprehensive review of content is factored into schedules.
- However, do not spend time trying to write XSLT code for bad data, take a step back and fix the source material. It will most likely be quicker and can avoid any unnecessary problems in the XSLT.
- Analyze the problem and decide if it's easier to fix the source XML or wait for the JATS conversion. This may very well be more a business decision than a technical decision.
- Document every decision that's made. Ensure the mapping specification is up-to-date. It is the historical record of the transformation and complements the XSLT.

- Working as a team, preparing schedules and having daily meetings while the specification is being written helped keep us on track.
- This was a tremendous project and requires the knowledge of everyone on the team. Technical staff, content staff, and editorial staff all have valuable input to offer.

We chose to use pre-existing JATS DTD elements and avoid any JATS module customization. The stock NISO JATS was more than sufficient to accommodate AIP's tagging needs. We were able apply our tagging principles and remain true to our business rules. We have achieved the XML quality we were aiming towards.

VIII. Note on Project Resources and Timeline

What started out as a seemingly straightforward tag-to-tag mapping table with a redesign of our metadata evolved into an intense 11 month project. Within AIP's reconfigured Content Technology team the three member content analyst subgroup prepared the mapping specification working with one dedicated XSLT developer and part time assistance from another developer for file processing and error list generation. As noted earlier, while this was the primary task, it was done in parallel with existing operational support duties. The project began in August 2011 with preliminary discussions on the tag/node mappings. Because the bulk of the archival content was in short SGML record format, this was the first scheduled transformation. The initial goals were to

- Complete the mapping of short SGML records to JATS XML by October 31, 2011; complete the XSLT by December 31, 2011
- Complete the mappings of AIP full-text XML to JATS XML by October 31, 2011; complete the XSLT by January 31, 2012

While the overall mappings were 95% complete within the scheduled time period, the questions of handling generated text, markup for semantic enrichment, multimedia still needed additional discussions and resolution. Once the XSLT modifications were underway and content tested, the original shortcomings, i.e., the lack of thorough XPATH, within the specifications manifested. In March the first bulk tests were run, updates implemented and tested. In late April the complete collection was converted. Again, more areas to improve were identified, incorporated into data or XSLT as needed, and tested. In June, the entire collection was rerun. The result was that through constant refinement and testing, the fine tuning of the mappings and successful transformation of all content finally completed by July 2012.

IX. The Future with JATS

The irony of this project is that the end result was the establishment of a solid archival foundation for our content assets. The real work is just beginning. When presented with the opportunity to perform an in-depth content review, identifying anomalies, and areas for improvement, it cannot be wasted. The lessons outlined need to be integrated into current processes, documented, and rigorously applied. This means

- tight control over content revisions,
- strict versioning and maintenance of QC tools,
- consistent application of tagging principles within new or updated XSLTs,
- detailed manifests identifying the tools (and version) applied to any piece of archival content
- constant awareness of updates to and releases of NISO JATS

The adoption of the standard allows AIP to more easily adopt future initiatives such as NISO multimedia initiative, ORCID, FundRef, etc. Now, instead of shoehorning *ad hoc* applications into proprietary XML, the experience and ideas of a community of users can be tapped to ensure a seamless integration into the best practices of the STM industry.
AIP can now also contribute to the JATS community, contributions based on real-world usage and not theory. It's a nice position to be in and we look forward to sharing with you all!

Our descendants in the AIP world will be forever grateful!

Acknowledgments

We would like to thank the following for their contributions: The AIP Applications Development Staff for the fantastic programming support, with a special thanks to Hu Wang for the XSLT development and Matt Stratton for batch processing and data corrections; Evan Owens for the expert guidance on the interpretation and historical evolution of the NLM/JATS XML; Portico for sharing the XSLTs used to convert AIP content; Wendell Piez of Mulberry Technologies for the great XSLT/Schematron training; and Anne LoPresti for the documentation and overall expertise on AIP editorial styles.