i4i the structured content company

Find & Fix Document Differences in Drug Product Labeling

Part I: Comparing Two or More Documents

> ***Editor's Note**: This document assumes basic familiarity with i4i's A4L technology and products. Details can be found at www.i4i.com.*

Background to the problem

Comparing two documents to identify differences is a core activity in any document production environment.

Compare tools highlight differences between two or more documents. This can be used to support:

1. **Collaboration**: See what others have done to their versions.
2. **Verification**: See any changes that have been introduced or errors that have been rectified.

Compare is a fundamental tool in an organization's information agreement[1] arsenal. In the world of drug product labeling, it is needed to manage the many versions, variants, and translations of documents.

Traditional compare tools have a core characteristic: they work on documents of the same format, such as Word or PDF. This is necessary because, to provide meaningful results, the compare process must compare apples with apples.

Cross-format compares most often result in ambiguous errors. For example: comparing a Word document and a WordPerfect document, or even two documents from different versions of Word, will often produce ambiguous results—because there is not 100% feature equivalency between the formats.

PDF is a popular solution to the problem, because it provides a common format to which all printable documents can be rendered. Thus, despite its limitations, PDF is a "solution" to the problem.

This paper will explore a compare process and technology that uses XML documents and challenges the cross-format limitations. In doing so, it changes the way that compare can be used to support information agreement.

What is an XML document?

An XML document contains, at its simplest, two things: *content* and *XML tags*.

Content

Content is a Unicode[2] text stream that is encapsulated by XML tags. Some XML documents will include inline multimedia as content, achieved by *base64 encoding* the binary data (i.e., presenting the content in textual form).

XML Tags

XML tags are specially constructed text strings bound by delimiters, typically the "<" and ">" characters. XML tags have two parts: a mandatory name and optional attributes, or metadata, about the name.

For instance, a tag with the *name* section can have an at- tribute such as an LOINC code, with the attribute defining the type of section according to the LOINC classification system.[3]

The problem with XML of course is that it is XML—eXtensible Markup Language—meaning that any user can, by just following some basic rules, create their own tag set.

XML product developers such as i4i Inc., Microsoft Corporation, and Adobe Systems Inc. develop XML tags specific to the characteristics of their own software tools. The tools "natively" understand their XML and may have capabilities to support other XML via custom programming. See ***Figure 1*** for an example of encoding by the different tools.

The XML examined in figure 1 is often called *presentation markup*. This term characterizes XML tags that are applied to identify the presentation characteristics of content (e.g., **bolding** a word or drawing a line under a word).

Presentation markup is "hard-wired" into word processors, desktop publishing tools, and web browsers so that no coding work is required to use it.

String "ABC" encoded in Different Tools

i4i's A4L (using x4o):

```
<p>A<bold>B</bold>C</p>
```

Microsoft's Word:

```
<w:r><w:t>A</w:t></w:r><w:bookmarkStart
w:id="0" w:name="_GoBack"/><w:r
w:rsidRPr="001B7EDD"><w:rPr><w:b/></
w:rPr><w:t>B</w:t></w:r><w:bookmarkEnd
w:id="0"/><w:r><w:t>C</w:t>
```

Adobe's InDesign:

```
<Content>A</Content><CharacterStyleRange
AppliedCharacterStyle="CharacterStyle/$ID/[No
character style]"
FillColor="Color/ubc" FontStyle="Bold"
PointSize="16"><Properties><AppliedFont
type="string">Minion Pro</AppliedFont></
Properties><Content>B</Content> </Character
StyleRange><CharacterStyleRange
AppliedCharacterStyle="CharacterStyle/$ID/[No
character style]"
FillColor="Color/ubc"
PointSize="16"><Properties><AppliedFont
type="string">Minion Pro</AppliedFont></
Properties><Content>C</Content>
```

Figure 1

Each tool has its own tag set, specific to its properties, for which there may or may not be equivalents, or even ambiguous approximations, in other tag sets. The tag set is in effect an XML representation of the proprietary binary format of each of the mentioned tools.

There is another type of XML called *semantic markup*. This term characterizes XML tags that are ap- plied to semantically describe content. For example:

```
<activeIngredient>ABC</activeIngredient>
```

OR

```
<customXML name="activeIngredient">ABC</customXML>
```

The semantics are specific to the business application that uses a specific type of document—for example, the US Food and Drug Administration's SPL (Structured Product Label) for drug product labeling, or DITA[4] for technical documentation. This markup is used to unambiguously identify to computer systems the tagged content's data processing role, within the context of the relevant business process.

Semantic Markup in Different Tools

To identify "ABC" as an active ingredient:

i4i's A4L (using x4o):

```
<activeIngredient>ABC</activeIngredient>
```

Microsoft's Word:

```
<customXML name="activeIngredient">ABC
</customXML>
```

Adobe's InDesign:

```
<XMLElement Self="di3i4i1"
MarkupTag="XMLTag/activeIngredient">
<Content>ABC</Content></XMLElement>
```

Figure 2

The above markup identifies the string "ABC" as an active ingredient, as opposed to an inactive ingredient—something of critical importance in the drug product labeling process.

The US FDA recognized the importance of this type of markup when it mandated the use of SPL XML for drug product labels.

As with presentation markup, there are many semantic tag sets in existence and many different ways of implementing semantic markup. See ***Figure 2*** for an example. There is a critical difference between presentation markup and semantic markup.

Presentation markup is specific to a tool and can be used in many different business processes (e.g., Word is used to write both operations manuals and drug product labels).

Semantic markup is used in specific business processes (e.g., DITA for technical manual production and SPL for drug product approval). Semantic tags set are, from the point of view of tools, unknown in advance and subject to change by forces other than the manufacturer of the tool (e.g., users of the tool). Presentation tag sets are known in advance and are part of the tool.

Sophisticated XML tools have ways of managing semantic tag sets. An example is i4i's A4L[5] software, which uses its semantic tag sets to facilitate the business processes required for drug product labeling authorization.

Comparing XML Documents

As can be understood from the previous discussion, comparing XML documents is not simple, for it requires the ability to differentiate the three parts of XML documents: the content, the presentation tag set, and the semantic tag set. Two of the parts, the content, and the semantic tag set, are part of a business process. The other part, the presentation tag set, is specific to its tool.

Scenario

Consider a typical situation where a drug-advertising brochure is developed in InDesign and the drug label is developed in Word. It is a requirement that the *Warnings* section in both documents must use the exact wording. InDesign is the "apple" and Word is the "orange."

A manual process is required to compare content from a Word document to that in an InDesign document. The traditional information agreement solution, as shown in ***Figure 3***, is to:

1. Normalize both documents to a common format such as PDF.
2. Manually review/compare the PDFs.
3. Hope for the best and manually replicate any identified differences to the sources for reconciliation.

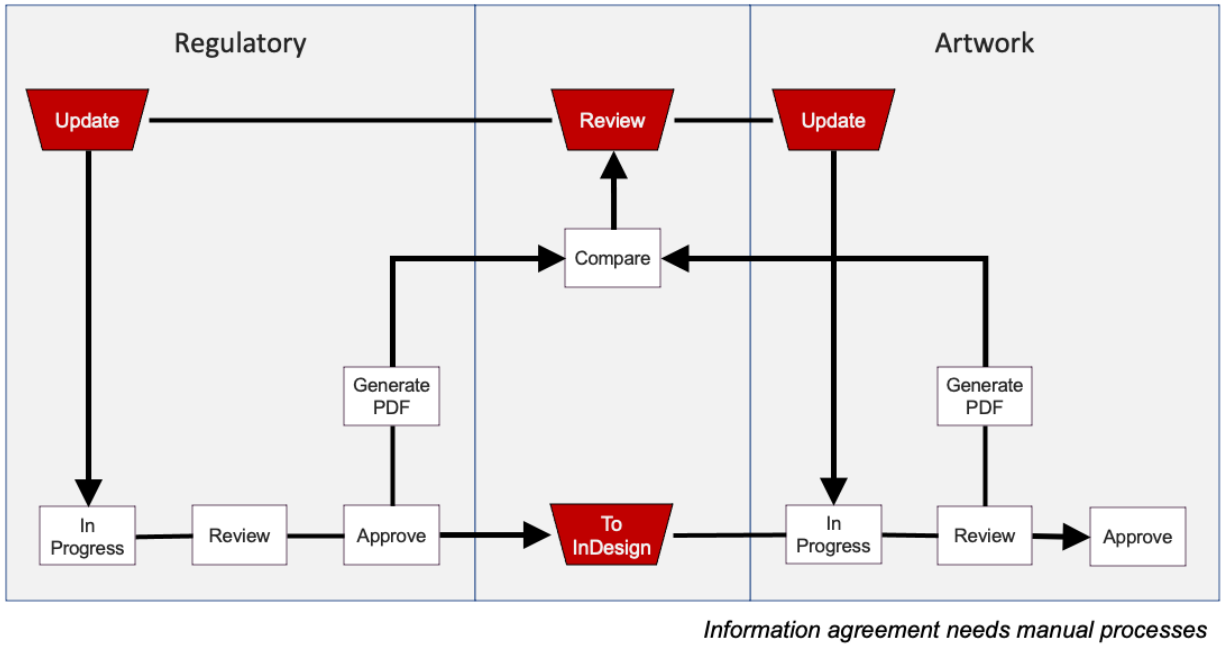

Figure 3

This process is repeated until both sides are in agreement.

How might an XML-compare help reduce such complexity?

Comparing is a Process

The first thing to recognize is that compare is a process with two mandatory steps and one optional step.

The first step is to identify the objects to compare and the role of each object: base or variant.

The conventional PDF solution relies on comparing printouts of the InDesign and Word documents, *which is in fact not what the user wants*. The user wants to compare only the content of the *Warnings* section in the InDesign document (the variant) against the content of the *Warnings* section in the Word document (the base).

The appropriate application of semantic XML in both documents allows the XML-compare to select and limit the compare to exact objects (e.g., *Highlights* and *Dosage* are not compared because that is not what the user asked for).

This is particularly important when the compare is exercised in situations where the organizational model of the two document types may be different.

For example, Word is a lineal stream while InDesign is a collection of "stories" (text groupings) laid out over several pages. See ***Figure 4***.

The second step is to identify the specifics of the required comparison. In this situation, the user wants to compare the wording.

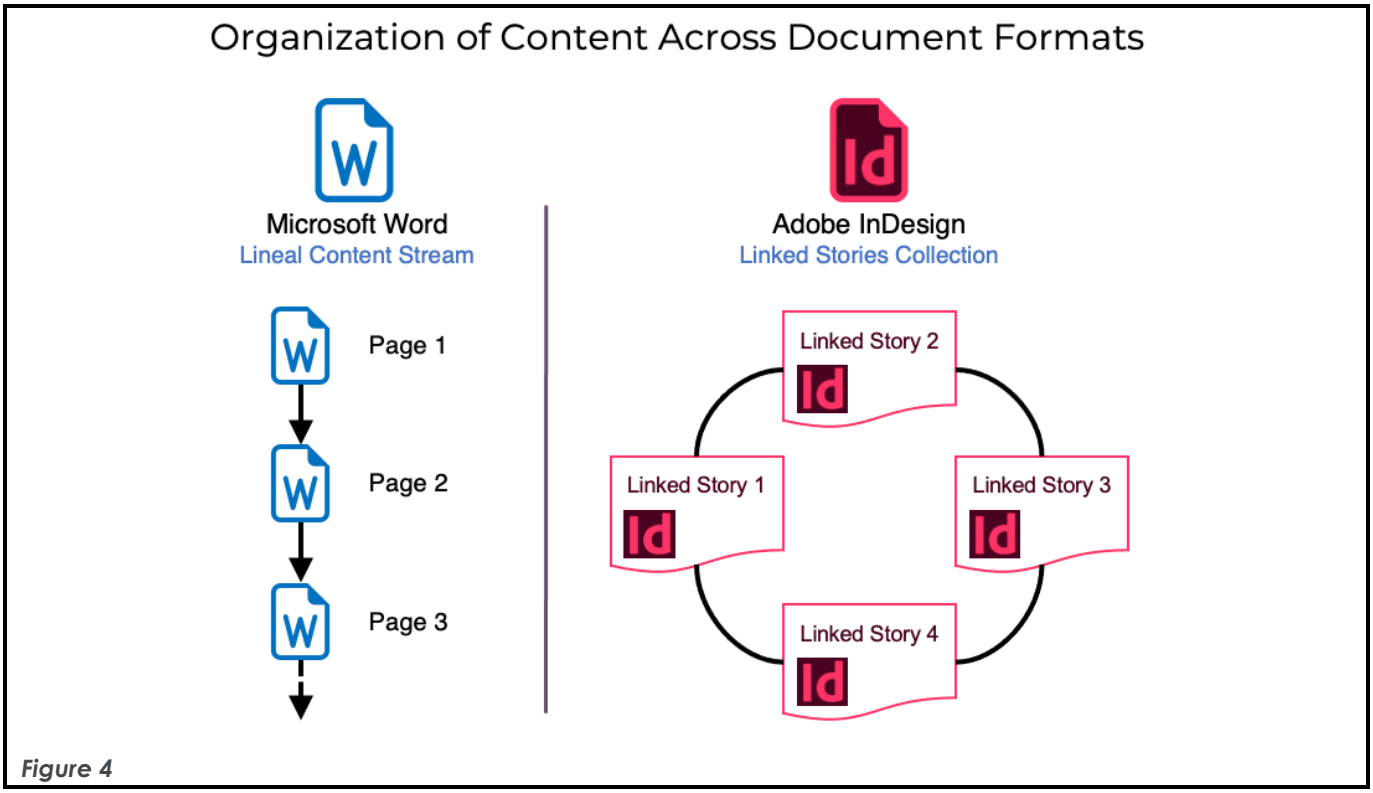

Figure 4

The XML-compare therefore removes all XML markup in each of the selected objects, leaving only the content. The content objects are then compared to identify insertions and deletions of content relative to the base, in this case the Word content.

The result is returned as a single stream of content. This stream can then be evaluated by the user to determine whether to exercise the optional final update step.

The update places identified content insertions and deletions at the appropriate locations, in the original InDesign and Word sections. The inserts and deletions have target-specific presentation markup so that they are visually distinct in the target applications.

Both the InDesign user and the Word user are now aware of the wording differences in the context of their tool and can work to reconcile. They are in effect collaborating.

Other situations arise when the objective is to compare differences in the presentation. Without the creation of impossibly complex equivalency tables that allow Word's XML to be mapped to InDesign's XML, or to SVG XML (used by Adobe Illustrator), cross-format presentation compare is not possible.

For this purpose, PDF or some other neutral format remains a viable option.

However, when the compare is within an XML format (e.g., Word to Word, InDesign to InDesign, or Illustrator to Illustrator), XML-compare is the preferred option as it deals with originals, not an interpretation of the original.

As with the content compare, the first step is to identify the objects to compare. Once the objects are selected, the XML-compare strips all the content and any semantic markup from the objects, leaving only presentation markup.

The presentation markup is then compared, and a result document is streamed back. If required, the differences are inserted into the source documents, marked as differentiated text objects, to be evaluated by an expert in the target tool.

The precision of this level of compare will exceed that of using an intermediate form, such as PDF, because the compare is working with source data—not a representation of the output that is limited by the quality of output technology.

Finally, semantic XML can be compared within a format or across formats. The same process of object selection is used. After the objects are selected, content and presentation markup are stripped from the objects. The semantic markup is then normalized and compared.

The result stream is returned and evaluated. If required, the source documents are updated with XML objects that identify any differences.

How XML-Compare can be Used

The XML-compare process can be fully automated and invoked at any time in a document's life cycle. In systems like i4i's A4L, which supports complex variant relations between documents, it can be used as an information agreement management tool to catch unacceptable drift between variants.

For instance, whenever a variant document is checked-in, the check-in process automatically invokes the XML-compare process to immediately identify any differences—which may represent unacceptable drift from the source documents. Rather than waiting for specific life cycle stages to review and reconcile compound differences, problems can be identified and addressed immediately.

A typical A4L cycle, as shown in ***Figure 5***:

1. An automated process moves content from a Word document to an InDesign document.
2. To ensure information agreement between regulatory and artwork, the Word and InDesign documents are compared and manually reviewed.
3. Updates are automatically made to the regulatory and artwork documents, and the process repeats until both sides are in agreement.

This solution:

- Replaces three manual process steps with automation.
- Optionally eliminates all manual process steps.

- Eliminates two automated steps.
- Reduces the number of documents the system must manage.

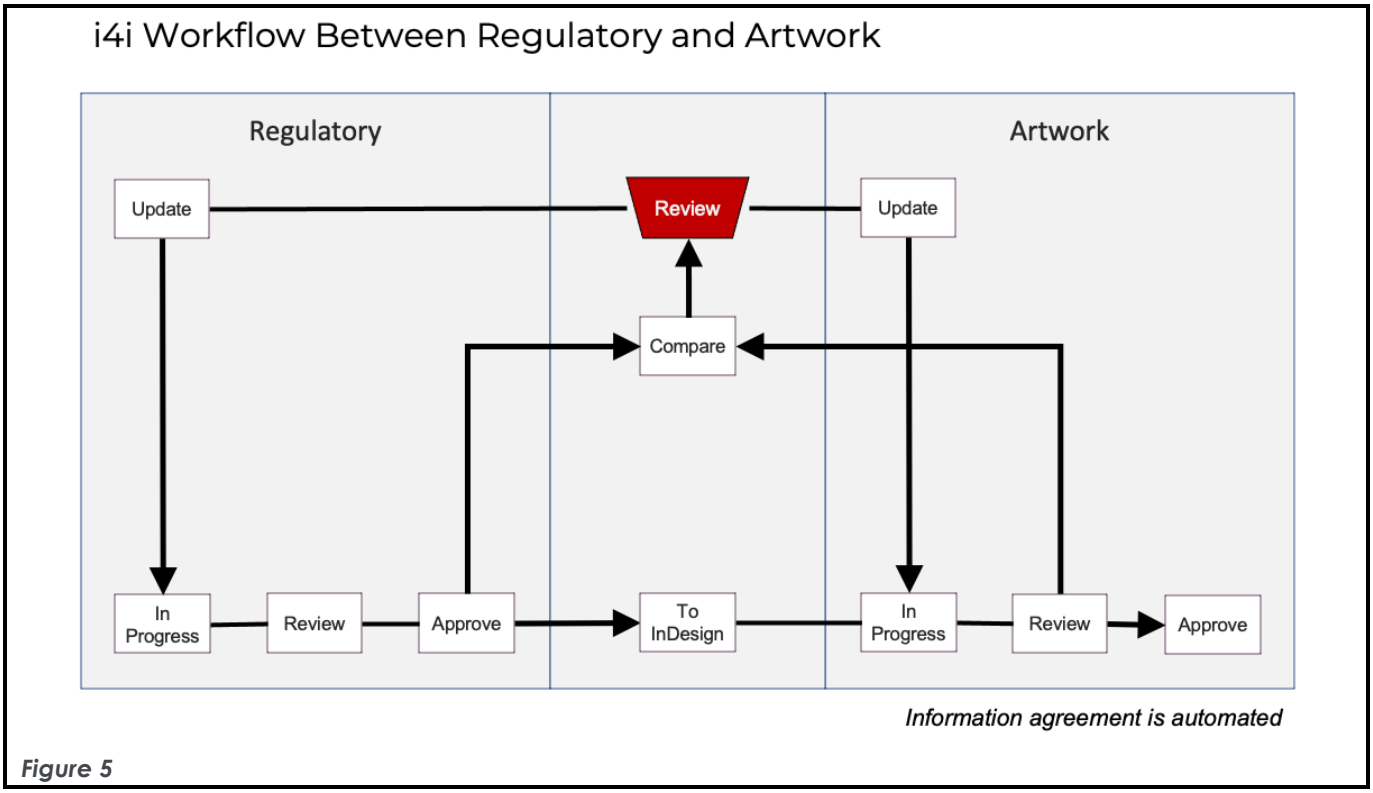

Figure 5

It is usually more effective to solve a problem when it happens, rather than wait until it has become the foundation of other problems.

Alternatively, XML-compare can be invoked at specific stages in the life cycle of a document collection. Since it is automated and does not rely on the creation and lifecycle management of secondary documents, the process is cleaner and more efficient.

There is More: Automated Notification of Change

One of A4L's web services tracks the use of XML elements in documents. All documents that are sub- mitted to this element tracker are examined for predefined XML elements (e.g., <indications>). For each occurrence of this element in the document, a database record is made of its unique identity and content.

If the XML element is already being tracked, a comparison of its current and prior content is automatically performed. If there is a difference and A4L has been configured for email-notification of changes, a message is generated to specified users informing them that a change has occurred.

Summary

We don't send a letter when an email is sufficient. Neither do we visit the library just to look up information, when we have Internet access.

XML technology has reached the stage of email and the Internet. Not only has it reached wide- spread adoption—it is seen as the only way of achieving many critical business processes. The key is to adopt tools that provide the most efficient use of XML for the required process.

The decision as to how to use XML-compare in managing information agreement should be driven by the needs and objectives of the business strategy, not the constraints of tools. With i4i's A4L tool, any constraints have been removed.

From creating and managing labels for drug product authorization to integration with artwork management tools, or other potential consumers of labeling information, XML is being used to remove layers of duplication and inefficiency.

The resulting cost savings and shortened time-to-market are further entrenching XML in companies' application infrastructures. Thus, it should be part of any company's application planning process.

Notes:

1. Information agreement: where what is said in one document is the same, or materially the same, as that said in another document—and, if not, the reason for the difference is captured.
2. The Unicode Standard is the universal character-encoding standard used for representation of text for computer processing.
3. Logical Observation Identifiers Names and Codes (LOINC) is a standard and database that provides a set of universal names and ID codes for identifying laboratory and clinical test results.
4. Darwin Information Typing Architecture: an XML-based architecture for authoring, producing, and delivering information.
5. ALiCE (**A**uthoring **Li**fecycle **C**ollaboration **E**nvironment) for Labeling.

Contacting i4i

For assistance using i4i products and tools, or for more information on technical support options, please contact:

Infrastructures for Information Inc.
720 King Street West, Suite 805
Toronto, ON Canada
M5V 2T3

Phone: +1 416.504.0141
Fax: +1 416.504.1785
E-mail: support@i4i.com

https://www.i4i.com