One Doc per XML File

Overview

This is an example of using the File Connector to load an XML file that contains a single document. See Loading XML Content for other examples.

View incoming links. **>>**

Loading XML Content , Multiple Docs per Small XML File

File Connector, Single Document per File

Since our sample file contains only one document, we we'll use a File Connector and the **xmlIngest** workflow to load the document. The exercise involves the following steps:

1. Edit the project's **schema.xml** file to add some new fields, as described in a following section.
2. Create a File Connector (**OneDocPerFileConnector**) that opens the **book.xml** file in **c:\documents,** and turns the contents into a single IngestDocument. Send this IngestDocument to the standard **xmlIngest** workflow.
3. It will be necessary to configure the **xPathExtractor** component of the **xmlIngest** workflow to copy values from the XML elements to IngestDocument fields.
4. Run the **OneDocPerFileConnector** and inspect the resulting records in SAIL. Verify that we got a single book, and that the content fields are as expected.

The next few subsections demonstration how to use the AIE Administrator to accomplish these steps.

Setting Up the Demonstration

We need to create a project and make some modifications in it before running AIE.

Create the Project

Create a new project that includes the **demo** group of AIE Modules. (None of the modules in the **demo** group is necessary to XML ingestion.)

```
aie-exec createProject -n xmlbooks -g demo -o c:\attivio-projects
```

Do not start AIE at this time. We need to edit the project's schema file first.

Mapping XML Elements to AIE Index Fields

We will be using a workflow transformer to map XML elements into AIE schema fields. This creates an issue. The schema for AIE's index is very robust, but it does not contain every imaginable XML element. We'll have to compare the schema to the XML document to see:

- How to map our XML elements to existing index fields; or
- How to add new fields to the schema to accommodate the XML elements.

There are five attribute elements in a <book> description. In this case, it turns out that the AIE Schema contains fields that exactly, or at least adequately, match most of the elements of the XML document:

XML Elements	AIE Schema Fields
<title>	title

<author>	author
<yearpublished>	creationdate
<description>	text
<location>	location

However, we are going to need two additional fields. The XML contains **firstname** and **lastname** elements that are not in the AIE schema.

<project-dir>\conf\schema.xml

```
    <fields>
      <!--existing field descriptions...-->
          <!-- add these new fields to handle the new fields from the xml elements -->
      <field name="firstname" type="string" indexed="true" stored="true" facet="false"/>
      <field name="lastname" type="string" indexed="true" stored="true" facet="false"/>
    </fields>
```

Setting Up the Target File

To experiment with XML ingestion, we need to create a directory and put a XML source file in it.

In this exercise we used the following source directory:

```
c:\documents
```

The file **book.xml** is our example of a file that contains a single document.

book.xml

```
<?xml version='1.0'?>
  <book id="100">
    <title>Adventures of Tom Sawyer</title>
    <author>
      <firstname>Mark</firstname>
      <lastname>Twain</lastname>
    </author>
    <yearpublished>1876</yearpublished>
    <description>Mark Twain's classic about growing up in Missouri.  Tom Sawyer paints a fence, attends his own
funeral,
    and gets himself locked in a dark cave.
    </description>
    <location>U.S.A.</location>
  </book>
```

Note that the document id number is coded as an attribute of the <book> element. This lets us demonstrate how to extract attribute values as well as element values.

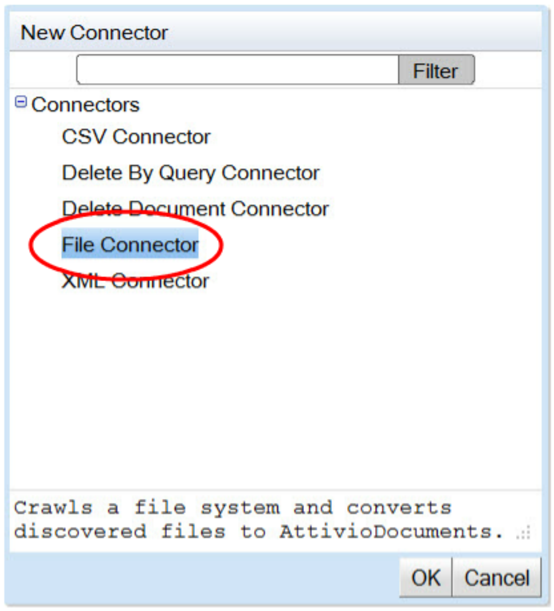
We placed a copy of this file in our source directory.

Start AIE

Start AIE with the new project. Use the **start-aie-local.bat** file (Windows) or the **start-aie-local.sh** file (Linux) in the project directory.

Create the OneDocPerFileConnector

Open the AIE Administrator and navigate to **System Management** > **Connectors**. Click the **New** link. This opens the **New Connector** dialog box. Select **File Connector** and click OK.


New Connector

Filter

Connectors
- CSV Connector
- Delete By Query Connector
- Delete Document Connector
- File Connector
- XML Connector

```
Crawls a file system and converts
discovered files to AttivioDocuments.
```

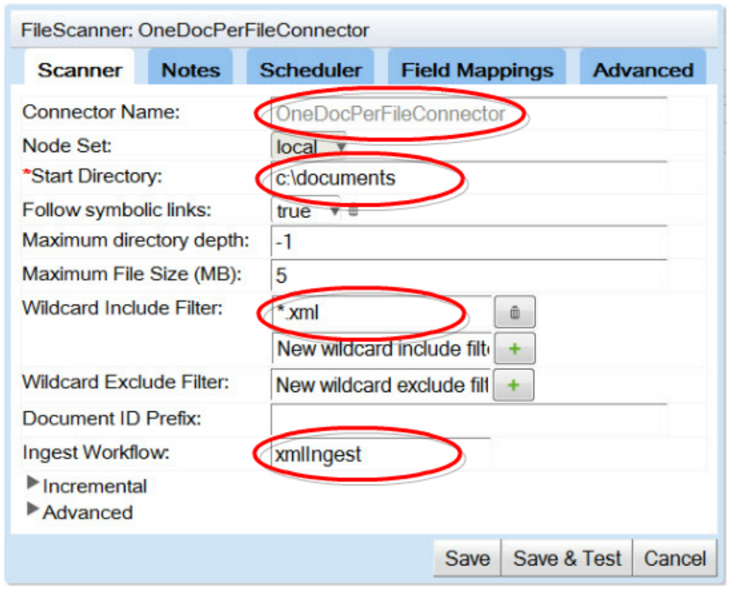
OK Cancel

On the **Scanner** tab of the **New Connector** editor, fill in the following values:

- The name of the connector. We called this one **OneDocPerFileConnector**.
- The directory that contains the source file.
- Adjust the include/exclude filters until the connector will look for ***.xml** files only. (You can use an explicit file name if you wish.)
- Designate the **xmlIngest** workflow as the destination of the new IngestDocuments.

UNC Paths

File connectors support the Uniform Naming Convention (UNC) path format used to designate Windows network shares. However, UNC paths are not supported for other path specifications in AIE for example the location of AIE logs or indexes. It is also possible to use a mapped network drive to specify a Windows file share as if it were a local drive. Note that scanners running on Linux hosts cannot access file content via UNC paths or local Windows paths - these scanners must run on Windows hosts.


FileScanner: OneDocPerFileConnector

Scanner | Notes | Scheduler | Field Mappings | Advanced

Connector Name:	OneDocPerFileConnector
Node Set:	local
*Start Directory:	c:\documents
Follow symbolic links:	true
Maximum directory depth:	-1
Maximum File Size (MB):	5
Wildcard Include Filter:	*.xml
	New wildcard include filt
Wildcard Exclude Filter:	New wildcard exclude fil
Document ID Prefix:	
Ingest Workflow:	xmlIngest

▶ Incremental
▶ Advanced

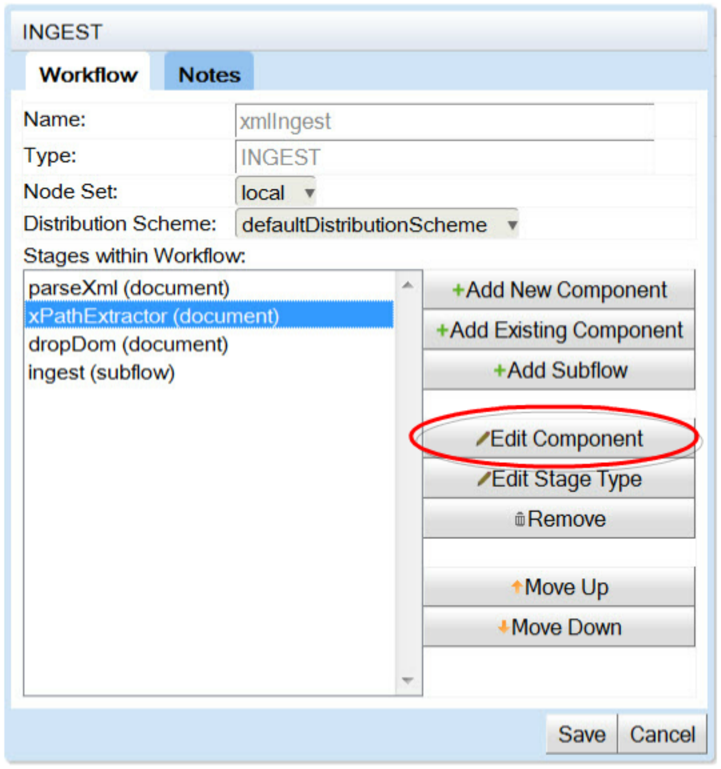
Save | Save & Test | Cancel

Click **Save** to store the new connector.

Configure the xPathExtractor Transformer

AIE includes a default **xPathExtractor** component, which is configured in **<install-dir>\conf\core-app\attivio-components.xml.**That component, in turn, is an instance of ExtractXPaths. Although the xPathExtractor comes preconfigured in AIE, its default mappings are unlikely to match your XML. Fortunately, it is easy to configure a new xPathExtractor containing a new set of mappings.

Navigate to **System Management** > **Workflows** > **Ingest** and select **xmlIngest** from the list. Click on the name of the workflow. This opens a new Workflow Editor.



INGEST

Name:	xmlIngest
Type:	INGEST
Node Set:	local
Distribution Scheme:	defaultDistributionScheme

Stages within Workflow:

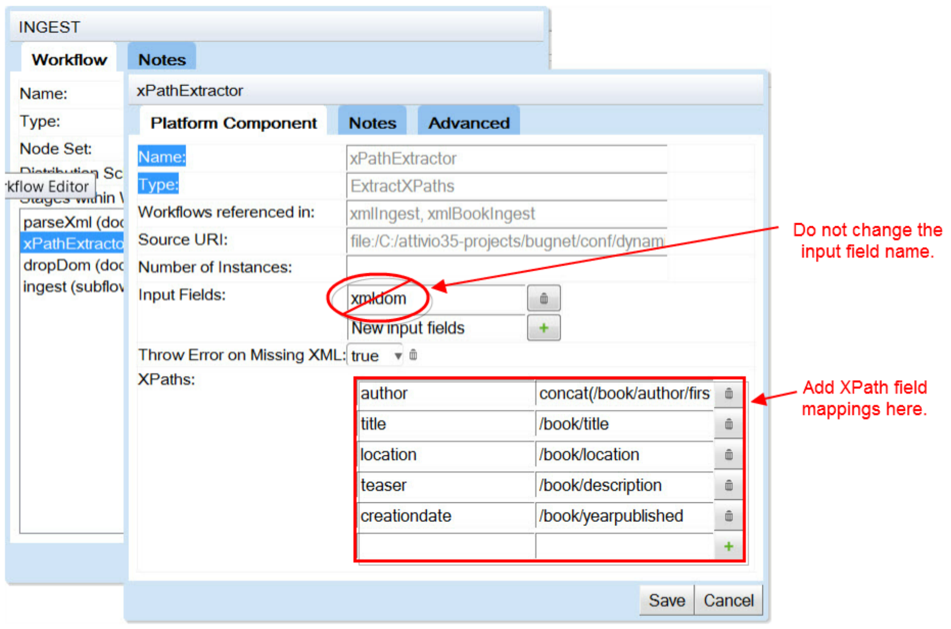
- parseXml (document)
- xPathExtractor (document)
- dropDom (document)
- ingest (subflow)

Select the **xPathExtractor** component and click the **Edit Component** button.

In this editor, enter the list of AIE Schema fields (such as **creationdate**) that you intend to use, paired with XPath expressions that tell AIE how to find the values for each field.
The XML elements are identified using XPath notation, which means we can take advantage of xPath's string operators to concatenate the authors' first and last names for a more graceful display:

xPath concat combines first and last names.

```
concat(/book/author/firstname," ",/book/author/lastname)
```



xPathExtractor

Platform Component | Notes | Advanced

Name:	xPathExtractor
Type:	ExtractXPaths
Workflows referenced in:	xmlIngest, xmlBookIngest
Source URI:	file:/C:/attivio35-projects/bugnet/conf/dynam
Number of Instances:	
Input Fields:	xmldom
	New input fields
Throw Error on Missing XML:	true
XPaths:	

author	concat(/book/author/firs
title	/book/title
location	/book/location
teaser	/book/description
creationdate	/book/yearpublished

xPath can also be used to perform a variety of equality and inequality tests on element values.

Click the **Save** button to store the modified component. This exposes the Workflow Editor again. Click the **Save** button on that editor, too.

Note that the final stage of the workflow sends the transformed IngestDocuments to the standard **ingest** workflow for linguistic processing and, ultimately, indexing.

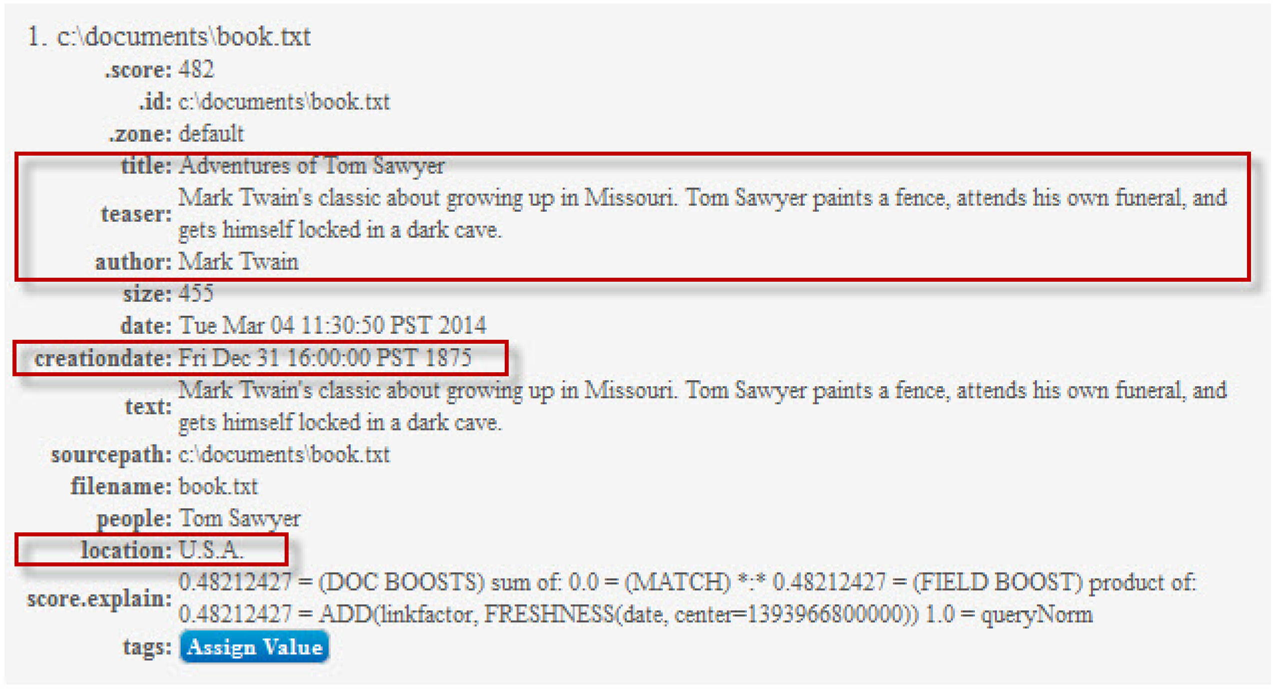
Testing the Configuration

Erasing the Index

While testing a new connector, you will frequently need to empty the index and try again. Four methods of deleting the index are described here.

To test the configuration, use the **OneDocPerFileConnector** to load the file that contains the book definition.

Then open SAIL and search for "**:**" (asterisk-colon-asterick). This retrieves all records in the index. If all has gone well, there should be one book record. Open the SAIL **Search Options** dialog and check the **Debug** checkbox.



```
1. c:\documents\book.txt
         .score: 482
            .id: c:\documents\book.txt
          .zone: default
          title: Adventures of Tom Sawyer
         teaser: Mark Twain's classic about growing up in Missouri. Tom Sawyer paints a fence, attends his own funeral, and
                 gets himself locked in a dark cave.
         author: Mark Twain
           size: 455
           date: Tue Mar 04 11:30:50 PST 2014
   creationdate: Fri Dec 31 16:00:00 PST 1875
           text: Mark Twain's classic about growing up in Missouri. Tom Sawyer paints a fence, attends his own funeral, and
                 gets himself locked in a dark cave.
     sourcepath: c:\documents\book.txt
       filename: book.txt
         people: Tom Sawyer
       location: U.S.A.
  score.explain: 0.48212427 = (DOC BOOSTS) sum of: 0.0 = (MATCH) *:* 0.48212427 = (FIELD BOOST) product of:
                 0.48212427 = ADD(linkfactor, FRESHNESS(date, center=1393966800000)) 1.0 = queryNorm
           tags: Assign Value
```

A close looks shows that all five of the extracted fields are populated correctly, including the concatenated first and last names in the **author** field.