MediaMan 3 Data Source Open Specification

This document describes the specification of the alternative data source import feature in MediaMan 3. This document contains no specific copyrights and is released to Public Domain.

Document release information:

Version:	1.0
Release Date:	9-23-2008
MediaMan Version:	3.0 (build 1020)
Author:	He Shiming (heshiming@imediaman.com)
URL:	www.imediaman.com

Overview

MediaMan 3 features importing from alternative data sources in addition to Amazon International websites. These alternative data sources are typically online stores that provide product information similar to Amazon. Commonly, online stores feature searching, item detail lookup, and image retrieval. MediaMan read and parse data from these stores using these features and eventually form a record inside the collection file.

A data source can be defined with an XML file. All data source definition XML files should be put into C:\Program Files\MediaMan\DataSource . There is no special limitation on the name of the file. But it's recommended that you prefix the file name with your nicknames to avoid duplicate problems among other developers. The XML file doesn't have a particular schema or DTD, but it must be defined according to the rules and specification in this document. In a nutshell, this XML file contains information about the URL of the website, and a series of regular expressions to extract meaningful data.

Knowledge required to develop a data source definition: HTML, some HTTP knowledge, and fluent regular expression. Tools required: Notepad or other editing tool, Regular Expression Designer (www.radsoftware.com.au).

Understanding Alternative Data Sources in MediaMan 3

MediaMan 3 will check the DataSource directory upon program start. It'll read all XML files and parse them. It will then keep an internal list of successfully parsed ones. In the import mode, a user can click the "Amazon U.S." link to access an import site menu. The parsed data sources will be included in "Other Sites".

When the user chooses to run an import from an alternative data source in MediaMan 3, the internal import manager will prepare a 4 stage process:

1. Item Search
2. Item Detail Lookup
3. Item Image Page Lookup

4. Item Image Retrieval

The stage is designed to fit the common process of item look up in e-commerce sites. A user will first find the "search page" and enter some keywords of the product to look for. Then choose an item from the results to look for details. And click on the image to get a larger view, finally, see the larger image.

Each stage is associated with a URL, a regular expression and a list of field IDs to specify where to store the parsed data. When the task is being executed, the program retrieves the content for each stage and parses data for each stage. Currently, MediaMan will only look for 5 items from the search results.

Evaluating and Beginning Development

To develop an XML data source definition, the target website should be should be well-formed. This means if the website is user-friendly and good-looking, the development will be much easier. You will also need to check if the information on the website can be fetched with the 4 stage mentioned above.

Let's take a first look at the definition file for Barnes and Noble. It's included in MediaMan 3's installation. So you can find it at C:\Program Files\MediaMan\DataSource\demo.xml . It's designed to retrieve content from www.barnesandnoble.com .

```
<?xml version="1.0" encoding="UTF-8"?>
<MediaManDataSource>
   <Name>Barnes and Noble</Name>
   <Version>2008-07-01</Version>
   <Copyright>(C) 2008. He Shiming.</Copyright>
   <Encoding>ANSI</Encoding>
   <Nature>HTML</Nature>
   <Definition>
      <Stage>
         <Type>SearchResult</Type>
         <URL>http://search.barnesandnoble.com/booksearch/results.asp?WRD=[SearchKeyword]</URL>
         <RegularExpressionPreParse><![CDATA[Sort by:(.*)Sort by:]]></RegularExpressionPreParse>
         <RegularExpressionT1><![CDATA[<div class="bc-wrapper">.*?<div class="bc-desc"><h2><a
         href="(.*?)">(.*?)</a>.*?ISBN-13.*?</strong>(.*?)</li>.*?Add to wish list]]></RegularExpressionT1>
         <ResultMappingT1>x1000000e;x10000006;x10000010</ResultMappingT1>
      </Stage>
      <Stage>
         <Type>RecordPage</Type>
         <URL>http://search.barnesandnoble.com/booksearch/pfp.asp?ean=[x10000010]</URL>
         <RegularExpressionT1><![CDATA[<img border="0" src="(.*?)" alt="Cover Image".*?<h2>(.*?)<em
         class="nl">(.*?)</em>.*?<p class="format">(.*?)</p>.*?Customer Rating for this product is (.*?) out of
         5"> \((.*?) ratings.*?Publisher: (.*?)</li>.*?Pub\. Date:(.*?)</li>.*?ISBN-13: (.*?)</li>.*?Sales Rank:
         (.*?)</li>.*?([0-9]+?) pp.*?</table>(<h4>.*?)<table width="100%" cellpadding="0" cellspacing="0"
         border="0">]]></RegularExpressionT1>

         <ResultMappingT1>x10000011;x10000006;x10000025;x1000000a;x1000002f;x10000030;x1000000b;x10
         000022;x10000010;x1000004f;x10000023;x10000014</ResultMappingT1>
      </Stage>
      <Stage>
         <Type>ImagePage</Type>
         <URL>http://search.barnesandnoble.com/booksearch/imageviewer.asp?ean=[x10000010]</URL>
         <RegularExpressionT1><![CDATA[<img src="([^"]*?)" alt="Cover Image"]]></RegularExpressionT1>
         <ResultMappingT1>x10000011</ResultMappingT1>
      </Stage>
      <Stage>
         <Type>Image</Type>
         <URL>[x10000011]</URL>
         <ResultMappingT1>1</ResultMappingT1>
      </Stage>
   </Definition>
</MediaManDataSource>
```

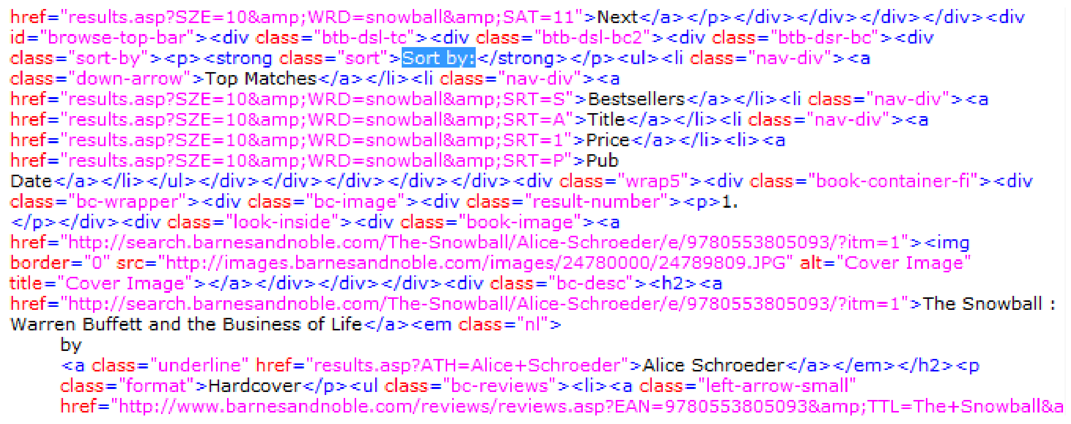
The definition is very straightforward. First, the XML definition itself must be in UTF-8 encoding. The root element is named "MediaManDataSource". Inside which, there are nodes that define the basics. You can put the information you like into "**Name**", "**Version**", "**Copyright**", but be aware that "**Name**" is used to display the entry in the menu.

"**Encoding**" specifies the encoding of the target website. Currently, only two values are supported "UTF-8" or "ANSI". If the website is not in UTF-8 encoding, you need to use ANSI and allow local processing of encodings. "**Nature**" specifies the type of the website. Currently only "HTML" is supported. These nodes must exist.

Then there is the "**Definition**" node, it's a big node containing the stage definition. The 4 stages mentioned earlier are defined here.

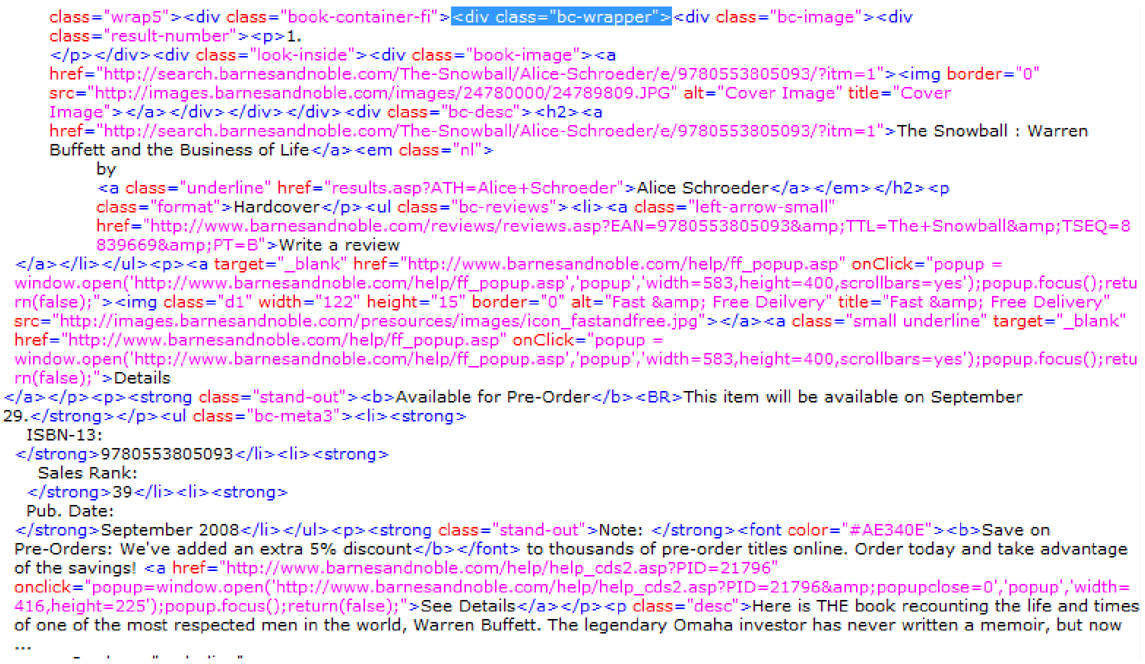
Let's take a look at the first stage. It has a "**Type**" node. Currently, only 4 values are supported in this node, "SearchResult", "RecordPage", "ImagePage", "Image". They are referring to "Item Search Result", "Item Detail Page", "Large Image Page", "Image". The next node is "**URL**", it specifies what URL to retrieve at this stage. Many websites support an HTTP/GET based search. In the example, "[SearchKeyword]" will be replaced by the keyword the user has entered. You don't need to worry about URL escaping. MediaMan handle this automatically.

"**RegularExpressionPreParse**" node can only be used with "SearchResult". Its purpose is to extract the area containing the result before actually parsing the result entries one by one. This is required because most search result pages contains not only the results, but also some other information, and/or styling page header and bottom. The item result entries can use similar HTML tags to styling page header, such as TABLE or TR. That's why we need to extract the part of the actual results and improve matching results. In this example, the regular expression is "Sort by:(.*)Sort by:" (Note: All regular expressions must be contained by a CDATA node because they can contain special characters not allowed in XML.). Which means, MediaMan will first look for "Sort by:" in the result page, and extract the content until the next "Sort by:". Note that the brackets marked the elements to be extracted. If no brackets are in the expression, nothing will be extracted. Let's take brief look at the page source code and find out how this works. The following content is part of the page:
http://search.barnesandnoble.com/booksearch/results.asp?WRD=snowball .


```
href="results.asp?SZE=10&amp;WRD=snowball&amp;SAT=11">Next</a></p></div></div></div></div><div
id="browse-top-bar"><div class="btb-dsl-tc"><div class="btb-dsl-bc2"><div class="btb-dsr-bc"><div
class="sort-by"><p><strong class="sort">Sort by:</strong></p><ul><li class="nav-div"><a
class="down-arrow">Top Matches</a></li><li class="nav-div"><a
href="results.asp?SZE=10&amp;WRD=snowball&amp;SRT=S">Bestsellers</a></li><li class="nav-div"><a
href="results.asp?SZE=10&amp;WRD=snowball&amp;SRT=A">Title</a></li><li class="nav-div"><a
href="results.asp?SZE=10&amp;WRD=snowball&amp;SRT=1">Price</a></li><li><a
href="results.asp?SZE=10&amp;WRD=snowball&amp;SRT=P">Pub
Date</a></li></ul></div></div></div></div></div><div class="wrap5"><div class="book-container-fi"><div
class="bc-wrapper"><div class="bc-image"><div class="result-number"><p>1.
</p></div><div class="look-inside"><div class="book-image"><a
href="http://search.barnesandnoble.com/The-Snowball/Alice-Schroeder/e/9780553805093/?itm=1"><img
border="0" src="http://images.barnesandnoble.com/images/24780000/24789809.JPG" alt="Cover Image"
title="Cover Image"></a></div></div></div><div class="bc-desc"><h2><a
href="http://search.barnesandnoble.com/The-Snowball/Alice-Schroeder/e/9780553805093/?itm=1">The Snowball :
Warren Buffett and the Business of Life</a><em class="nl">
        by
        <a class="underline" href="results.asp?ATH=Alice+Schroeder">Alice Schroeder</a></em></h2><p
        class="format">Hardcover</p><ul class="bc-reviews"><li><a class="left-arrow-small"
        href="http://www.barnesandnoble.com/reviews/reviews.asp?EAN=9780553805093&amp;TTL=The+Snowball&a
```

The highlighted "Sort by:" marks the beginning of the match. Note that it doesn't matter whether the extracted part is well-formed in terms of HTML. You just need to make sure they contain enough information.

Then, let's take a look at "**RegularExpressionT1**" node. It contains a regular expression to parse actual item information from the extracted part above. Its content is also pretty straightforward, "<div class="bc-wrapper">.*?<div class="bc-desc"><h2><a href="(.*?)">(.*?)</a>.*?ISBN-13.*?</strong>(.*?)</li>.*?Add to wish list". It contains 3 brackets to extract 3 pieces of information. The following part of HTML is the content this expression tries to match:



```
      class="wrap5"><div class="book-container-fi"><div class="bc-wrapper"><div class="bc-image"><div
      class="result-number"><p>1.
      </p></div><div class="look-inside"><div class="book-image"><a
      href="http://search.barnesandnoble.com/The-Snowball/Alice-Schroeder/e/9780553805093/?itm=1"><img border="0"
      src="http://images.barnesandnoble.com/images/24780000/24789809.JPG" alt="Cover Image" title="Cover
      Image"></a></div></div></div><div class="bc-desc"><h2><a
      href="http://search.barnesandnoble.com/The-Snowball/Alice-Schroeder/e/9780553805093/?itm=1">The Snowball : Warren
      Buffett and the Business of Life</a><em class="nl">
            by
            <a class="underline" href="results.asp?ATH=Alice+Schroeder">Alice Schroeder</a></em></h2><p
            class="format">Hardcover</p><ul class="bc-reviews"><li><a class="left-arrow-small"
            href="http://www.barnesandnoble.com/reviews/reviews.asp?EAN=9780553805093&amp;TTL=The+Snowball&amp;TSEQ=8
            839669&amp;PT=B">Write a review
  </a></li></ul><p><a target="_blank" href="http://www.barnesandnoble.com/help/ff_popup.asp" onClick="popup =
  window.open('http://www.barnesandnoble.com/help/ff_popup.asp','popup','width=583,height=400,scrollbars=yes');popup.focus();retu
  rn(false);"><img class="d1" width="122" height="15" border="0" alt="Fast &amp; Free Deilvery" title="Fast &amp; Free Delivery"
  src="http://images.barnesandnoble.com/presources/images/icon_fastandfree.jpg"></a><a class="small underline" target="_blank"
  href="http://www.barnesandnoble.com/help/ff_popup.asp" onClick="popup =
  window.open('http://www.barnesandnoble.com/help/ff_popup.asp','popup','width=583,height=400,scrollbars=yes');popup.focus();retu
  rn(false);">Details
 </a></p><p><strong class="stand-out"><b>Available for Pre-Order</b><BR>This item will be available on September
 29.</strong></p><ul class="bc-meta3"><li><strong>
   ISBN-13:
  </strong>9780553805093</li><li><strong>
    Sales Rank:
   </strong>39</li><li><strong>
   Pub. Date:
  </strong>September 2008</li></ul><p><strong class="stand-out">Note: </strong><font color="#AE340E"><b>Save on
  Pre-Orders: We've added an extra 5% discount</b></font> to thousands of pre-order titles online. Order today and take advantage
  of the savings! <a href="http://www.barnesandnoble.com/help/help_cds2.asp?PID=21796"
  onclick="popup=window.open('http://www.barnesandnoble.com/help/help_cds2.asp?PID=21796&amp;popupclose=0','popup','width=
  416,height=225');popup.focus();return(false);">See Details</a></p><p class="desc">Here is THE book recounting the life and times
  of one of the most respected men in the world, Warren Buffett. The legendary Omaha investor has never written a memoir, but now
  ...
```

Again, pay attention to the highlighted part that says "<div class="bc-wrapper">", which is the same as the start of the expression. Then, you'll see that the 3 pieces of information extracted are: the URL of the title, the name of the title, and its ISBN-13 code.

And therefore, in the "**ResultMappingT1**" node, we have "x1000000e;x10000006;x10000010", which is the ID for these three fields delimited by semicolon. Check the appendix section for a list of available fields.

The next stage "**RecordPage**" will be executed right after the search finishes. Note the URL node, we didn't actually used the URL parsed in the previous stage. Instead, we figured out a better way to target items by ISBN-13. The "x10000010" in the square brackets will be replaced by the ISBN-13 code parsed in the previous stage. You can use any field as the retrieval URL. But make sure these fields are set in result mapping previously.

Typically, the "**RecordPage**" must also parse the URL for images or a page containing images. You can store them temporarily in field "x10000011", "x10000012", or "x10000013". It's recommended that you put the image page URL into “x10000012” or “x10000013”, and the actual URL to JPEG or GIF files in “x10000011”.

However, in the example, again the “**ImagePage**” doesn’t really use the parsed URL. It uses the ISBN-13 code to construct the image page. Thanks to the designers at Barnes and Noble, this really made things simple. So after some parsing, we have our actual image URL mapped to “x10000011”.

So eventually, in the “**Image**” stage, we only put [x10000011] as URL indicating we would like to retrieve it as the cover image. Please note that “**Image**” cannot contain a regular expression node, and “**ResultMappingT1**” node must contain only the character “1”.

And voila! The process is complete.

Final Checks

Before trying with MediaMan, use services like http://www.xml.com/pub/a/tools/ruwf/check.html to check the well-formness of your XML file.

MediaMan uses CRLF (“\r\n”) internally for fields such as artist, author, and actor to separate between multiple values. But often times, a parsing process cannot convert a common delimiter, such as comma or semicolon to CRLF. MediaMan will try to perform this conversion internally, but can’t guarantee graceful output.

Also note that “**SearchResult**”, “**RecordPage**”, and “**ImagePage**” can actually contain more than one regular expression. Other than “RegularExpressionT1”, you can put “RegularExpressionT2”, and “RegularExpressionT3”. The result mapping node thus becomes “ResultMappingT2”, or “ResultMappingT3”. This is a fallback design, and its purpose is simple. If T1 fails, MediaMan tries T2 with the same content. If all expressions failed to match, MediaMan will give up and prompt user that no match is found.

Appendix: Field IDs

FIELD_TITLE	x10000006	FIELD_DESCRIPTION	x10000014
FIELD_ASIN	x10000008	FIELD_ASPECTRATIO	x10000015
FIELD_NUMMEDIA	x10000009	FIELD_DVDLAYERS	x10000016
FIELD_MEDIA	x1000000a	FIELD_DVDSIDES	x10000017
FIELD_PUBLISHER	x1000000b	FIELD_PICTUREFORMAT	x10000018
FIELD_RELEASEDATE	x1000000c	FIELD_REGION	x10000019
FIELD_GROUP	x1000000d	FIELD_RUNNINGTIME	x1000001a
FIELD_DETAILURL	x1000000e	FIELD_STUDIO	x1000001b
FIELD_UPC	x1000000f	FIELD_THEATRICALRELEASE	x1000001c
FIELD_EAN	x10000010	FIELD_AUDIENCERATING	x1000001d
FIELD_IMAGEURLLARGE	x10000011	FIELD_ACTORS	x1000001e
FIELD_IMAGEURLSMALL	x10000012	FIELD_DIRECTORS	x1000001f
FIELD_IMAGEURLMEDIUM	x10000013	FIELD_FORMAT	x10000020

FIELD_LANGUAGE	x10000021
FIELD_PUBLICATIONDATE	x10000022
FIELD_NUMPAGES	x10000023
FIELD_ORIGINALTITLE	x10000024
FIELD_AUTHORS	x10000025
FIELD_TRANSLATORS	x10000026
FIELD_ISBN	x10000027
FIELD_LABEL	x10000028
FIELD_NUMTRACKS	x10000029
FIELD_ARTISTS	x1000002a
FIELD_TRACKS	x1000002b
FIELD_ESRBRATING	x1000002c
FIELD_FEATURES	x1000002d
FIELD_PLATFORMS	x1000002e
FIELD_AVGCUSTOMERRATING	x1000002f
FIELD_TOTALCUSTOMERREVIEW	x10000030
FIELD_CRRATING	x10000031
FIELD_CRSUMMARY	x10000032
FIELD_CRCOMMENT	x10000033
FIELD_CUSTOMFIELDS	x10000034
FIELD_DATEADDED	x10000038
FIELD_LISTPRICE	x10000039
FIELD_NOTES	x1000003e
FIELD_SUBJECTS	x10000040
FIELD_GENRE	x10000041
FIELD_AUDIOFORMAT	x10000042
FIELD_CATALOGNO	x10000043
FIELD_SERIES	x1000004c
FIELD_VOLUME	x1000004d
FIELD_AWARDS	x1000004e
FIELD_SALESRANK	x1000004f
FIELD_MPN	x10000050
FIELD_CREATOR	x10000051
FIELD_PACKAGEDIMENSIONS	x10000052
FIELD_DEWEYDECIMALNUMBER	x10000053
FIELD_IMDBRATING	x10000054
FIELD_FILMLOCATION	x10000055
FIELD_PLOT	x10000056