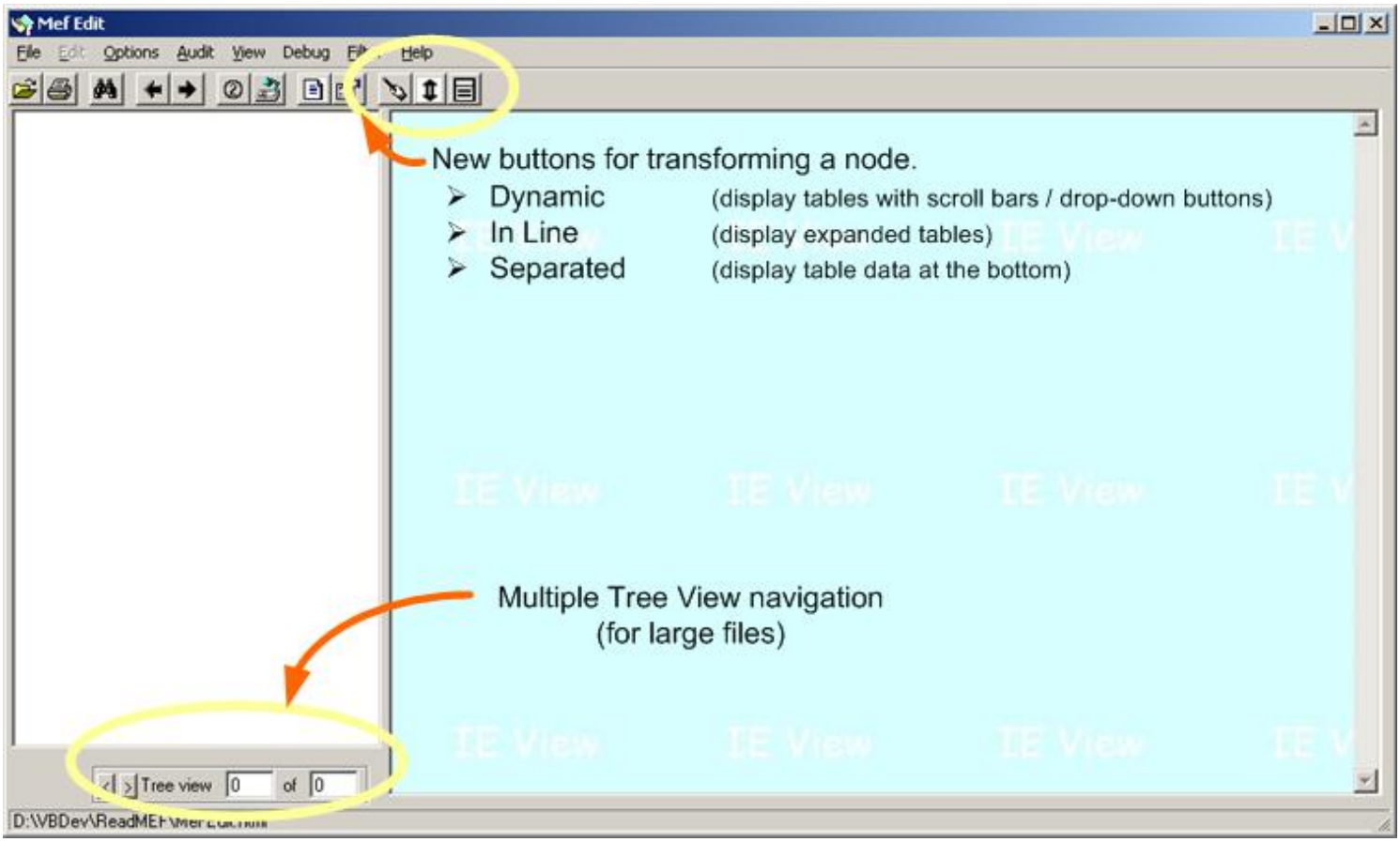
New Features and Bug Fixes

(04/03/2008)

- Support for returns with a lot of documents.
  - Multiple tree view displays to handle large returns
  - Search across all tree view displays
  - Note: this feature is only partially tested. Please use caution when working with a large return and save your work in any other open windows you are using prior to opening and working with them in ReadMeF.
- Document linking for printed documents
  - ReadMeF now includes a document number for each document within a return and a "ref. by" (referenced by) document reference at the top (next to the DLN). The "ref. by" display only applies to sub-return document references (all documents are assumed to be related to the main return document)
- "Transform selected" feature re-enabled to allow for multiple document transformation to the display window
  - Right click the display window to send the transformed document to a printer
  - transforms with print option (in-line or separated) to expand table displays
- "Print selected" feature transforms one return document at a time and sends it to the printer.
  - may require tweaking the print settings if using the "Adobe" printer (see AdobeSetup.htm)
- New node selection features under "Edit"
  - Note: these selection choices don't always work as expected, please verify your selections prior to transforming or printing a group of documents.
- Developer notes
- New window controls:
- 

Features Added In Previous Releases

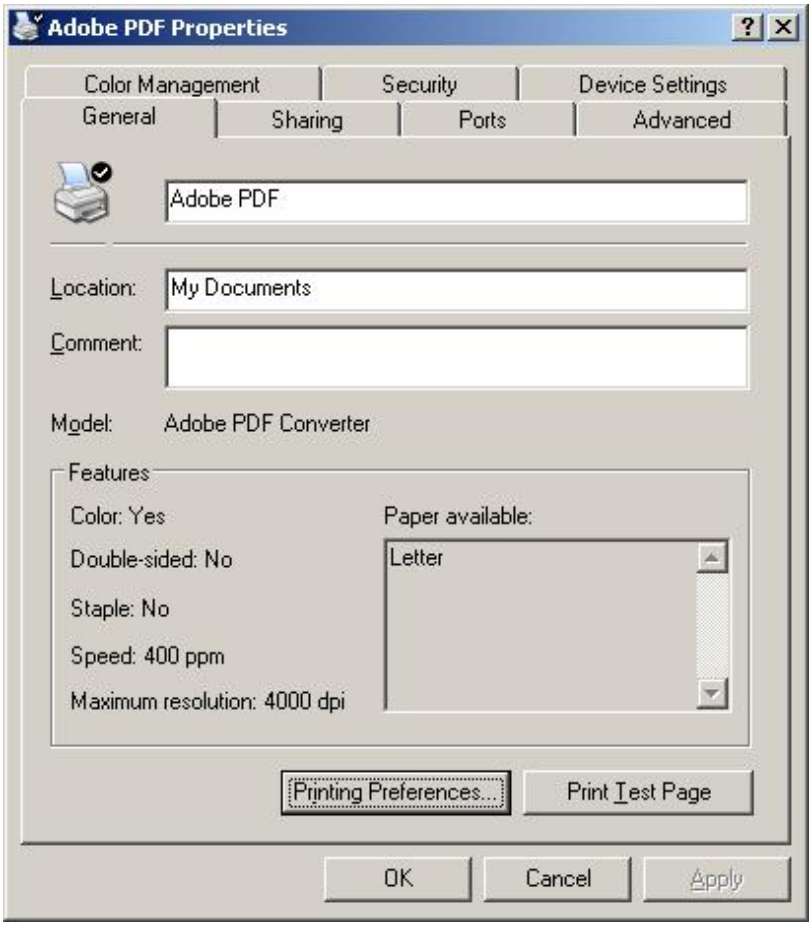
- Toolbar icon access to common features
    - Includes
        - "Back" and "Forward" navigation buttons
        - All data viewing options
- New data viewing options
    - Generic
        - Uses an XML extract built from related XSDs to build "generic" displays.
        - Contains the line reference and description elements of the XSD. (If the XSD extract is not available, it builds the display with out the line references)
        - Displays every data element (elements that don't match up to the extract are displayed in purple)
        - Built-in option extracts the line reference information from XSD files allowing for easy update as new XSD schemas are released.
        - Multi-year extracts supported
    - Filtered
        - Simple data filter to display any element with a tag name that matches tags in a customizable list
- External parser use enabled
    - Useful for locating XSL / parser compatibility problems (using a different parser may result in a clearer error message)
- Working push-pin buttons
- Data transform using appropriate tax year specific XSL (option)
    - "On" by default, uses one of the following ReturnHeader elements to extract the return's tax year:
        - TaxPeriod
        - TaxPeriodEndDate
        - TaxYear
    - Searches the XSL directory name for "20nn", where "nn" is numeric, (2004, 2005, etc.), and replaces that part of the path with return tax year.
    - If the directory is not found, then displays a message and uses your default XSL directory.
    - Displays a status line message showing the XSL path used for each transformation
- Changes made to facilitate the use of XSL on shared drives.
    - Changed the location of the "work" html file to "C:\TEMP"
        - localizes the HTML work file
    - Changed the name of the "work" XSL file to "Temp" + documentRoot + "RRD" + hhmmss + ".XSL" (for example "TempIRS1120RRD153227.XSL")
        - makes the temporary XSL file name unique
    - Added automatic cleanup of the temporary XSL file (unless user checks menu items "Debug", "Do not delete XSL work file")
        - Note: The XSL for a document is "converted" for ReadMef and stored as temporary XSL in the XSL directory. This temporary XSL is used for the XML to HTML conversion and is only needed for as long as that takes (seconds or less)
- Added capability of including an element value in Return, ParentReturn and SubsidiaryReturn tree view nodes
    - This would allow the value of ReturnHeader/Filer/Name/BusinessNameLine1 to be included on the node (for example) - see "Key Data Extract" in help file for setup instructions.
    - node values are searchable, so a subsidiary listed on a large IRS-1120 could be searched by business name
- Fixed an indexing problem that caused some return data to be ignored
    - To see if this applies to data you've already reviewed, look for "000000000000" ending values for any element named in the index file (The index file has the same name as the data file but with the added extension

".ix")

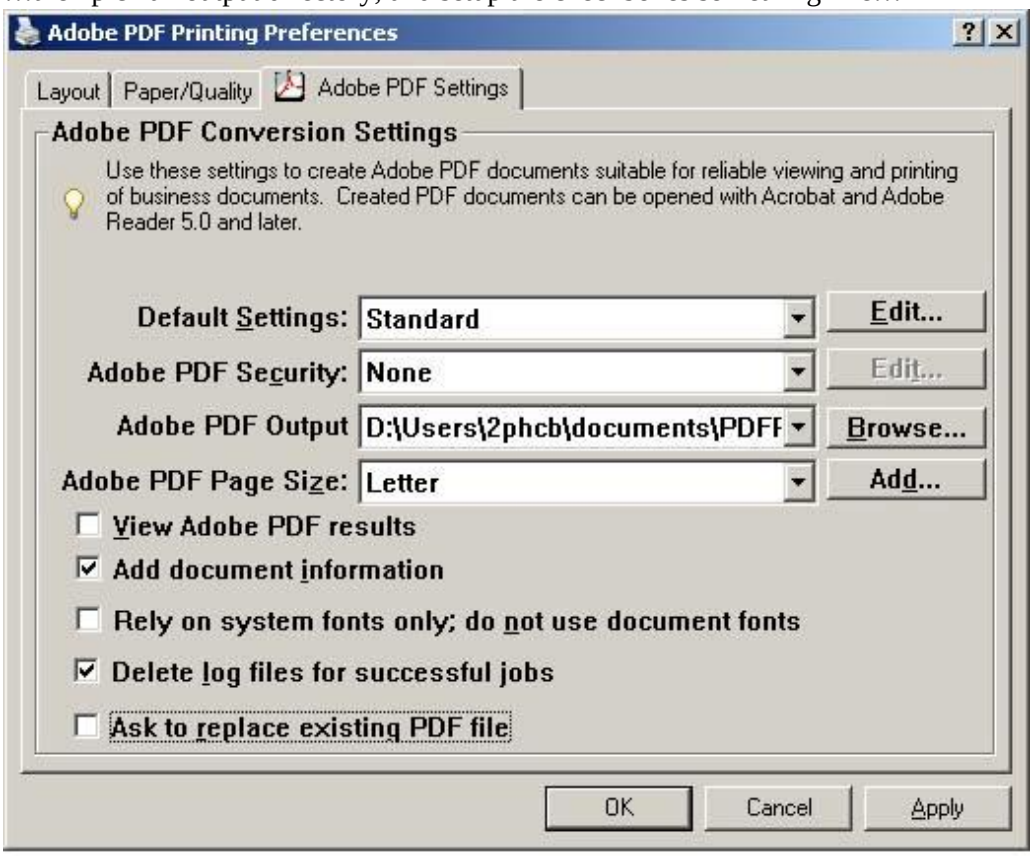
- Multiple document transformation
  - Select documents for transformation and/or print with checkbox mouse clicks
- Print handles "overflow" table data
  - Choose "separated" or "inline" display of overflow data items
    - Choose "separated" to make overflow data appear at the bottom (in a separate table)
    - Choose "inline" to expand the table to fit the data
- Print Preview
- Tree search
  - Find tree node values to locate forms or attachments
- Faster parsing of data
  - Internally uses Microsoft's XML parser for faster parsing (previous version used external batch process)
- Zoom to 150%
  - View transformed data at 100%, 125%, or 150% to help eliminate the need to print return data.
- Automatic transform
  - Optional setting to transform "on-click" if XSL stylesheet is available.
- Meaningful titles for return attachments
  - Utilizes XML file for document title to document root tag relationship (may be edited)
  - Allows addition of future documents with out requiring a new program version.

Setting up Adobe to work with ReadMeF

Get to the "Adobe" printer properties window (shown below) Via "Control Panel", "Printers", then right click "Adobe" and choose properties.



Click Printing Preferences
…then pick an output directory; and setup the checkboxes something like…



Note: **View Adobe PDF results** is unchecked because ReadMeF sends each return document to the printer in a separate file. You can combine these files in a binder with Adobe

Ask to replace existing PDF file is also unchecked to save you the bother of responding each time this happens. Because ReadMef sends one return document at a time to the "printer", you'll get asked this a lot if you happen to print the same documents twice.

Development notes

MeF puts a "wrapper" around the XML prior to using XSL to transform it to HTML for display. This wrapper is used to contain application defaults, user preferences, and parameters that contain information added by IRS (the DLN (document locator number), changes to the DLN and primary TIN (taxpayer ID number), etc. To simulate this wrapper, ReadMeF uses XmissionParams.xml found in the application's directory (defaulting to "C:\Program Files\ReadMef"), and replaces the "ReturnHeaderGoesHere" and "ReturnDocGoesHere" nodes with the appropriate ReturnHeader and return document nodes of the taxpayer data. It then finds the XSL for the transformation based on the return document's root element name. For example the XSL used to transform the multi-page IRS1120 (with document node "IRS1120") is "IRS1120.XSL".

Most of the XSL files reference a file containing templates and CSS style elements specific to the document in a file named *root*Style.xsl where root is the document node tag name.

Included or sub-XSL references usually include

- CommonPathRef.xsl
- PopulateTemplate.xsl
- AddHeader.xsl
- AddOnTable.xsl

Some key wrapper elements are

- "Location"
  - The most common values for this are
    - "RET" – meaning the document is relative to the return
    - "PAR" - ...relative to the "parent return"
    - "SUB" - ...relative to a subsidiary return
  - The appropriate return header elements are populated based on the return header type specified by this element.
  - These values are interrogated in "PopulateTemplate.xsl"
  - For a detailed description of values and meanings for this element, please see "Return Header Notes"

- "SubmissionVersion"
  - The value of this element is referenced in CommonPathRef.xsl for the "xsl:param" (a variable) called "TaxYear". The "TaxYear" parameter is referenced by other "xsl:param" variables to specify tax year specific path names.
  - Note: for TY2004, element "ReturnVersion" was used
- "FormHeaderOriginalData"
  - This value will appear in the "efile GRAPHIC print" bar

Push-Pins

I got the push-pins to work by adding a new java script to FormDisplay.js (called SendRef) and changing the template in PopulateTemplate.xsl called "SetFormLinkInLine" to use the SendRef java script function. When a push-pin is clicked, the java script directs the browser to xfer.html The ReadMef program notices whenever the browser location changes and uses the URL (which includes the ReferenceDocumentId value) to find the document with a matching ID and transform it. (If there's more than one Id (separated by spaces), it builds a temporary selection page for the user to select a display)

Document access speed trick

To speed up the program (and accommodate some of the larger 1120 returns), the XML is "indexed" prior to building the table of content nodes (actually a VB TREEVIEW control). When a user selects a document, a direct read of the bytes for that document is performed. The process that builds the index was "tricked" for speed by using an old-school mainframe style that mimics COBOL BLOCKS. It reads large chunks of data and then parses it. There is some somewhat goofy code to save the end of a block and add it to the beginning of the next one, but doing it this way is much faster than resorting to a BYTE read of the data. The index is stored in the same directory as the data file and is named the same as the data file, but with the addition of ".ix" appended to it. I think that using a SAX parser might be effective, but that I've found good results doing it this way.

Generic data display

The process of using the related XSD file to build an object that defines the line reference and description for an element given a specific XPath is slowed down by XSD complexity. The XSD for IRS990 for example is heavy with referenced or sub types that are found not only in the IRS990.XSD, but in included XSD documents. If the generic display is deemed useful, I think this process could be improved by building "expanded" XSD files (where referenced definitions are replaced with incorporated details) and using these for building the object.

A partial explanation of return headers in MeF

It used to be that we have 3 different Return Header data structures in the XML Data:
(Shown in Location Code - Return Header format)
RET - Consolidated Return Header
PAR - Parent Return Header
SUB - Subsidiary Return Header

Then 1120x form family added more complexity by introducing different Sub-Consolidations within it's form family. The 1120x form family also introduced Eliminations and Adjustments to the Location Code types. This resulted in multiple Location Codes and multiple variations of the Return Header best illustrated by this table:

	Consolidated or Top Level	1120 Sub-Consolidation	1120 L Sub-Consolidation	1120 PC Sub-Consolidation
Documents in the root level or documents in a sub-consolidation	**RET**	**SSC**	**LSC**	**PSC**
Parent	**PAR**	**PAR**		
Subsidiary	**SUB**	**SUB**	**LSB**	**PSB**
Eliminations	**ELM**	**SEL**	**LEL**	**PEL**
Adjustments	**ADJ**	**SAJ**	**LAJ**	**PAJ**

The XML header data for any of the non-red location codes is very similar to the layout of a Subsidiary Return Header. Therefore for any non-red location codes, it was decided that RRD would place the Return Header information in the Subsidiary Return Header section so that Stylesheets would not have to handle a dozen new sections in PopulateTemplate.

From a stylesheet perspective:

If the location code is RET, the stylesheets will pull data from the Return Header Section.

If the location code is PAR, the stylesheets will pull data from the Parent Return Header section.

If the location code is anything else, the stylesheets will pull data from the Subsidiary Return Header section. In this catch-all case, the fields that can be pulled from the Subsidiary Return Header section will depend on the value of the Location Parameter (for example, only some of the Location types contain the tax year begin date element)