User-Designed Standardized File Loader udStFiLr XML Users Guide

Version 2.12(r)

J. J. Weimer

August 25, 2007

Distribution Details
Developed with Igor Pro Version: 6.0.2
Procedure Files: udStFiLrXML and udStFiLrXML.lcl
Demos: none (uses SimpleStandardLoaderPanelDemo)
Experiments: none XOPS: none Help Files: none
Requires Packages: udFLStandardFunctions and udFLStandardStructure

Abstract

The udStFiLrXML procedure is designed to read an XML file into Igor Pro according to rules in a user-generated XSL file. Values for a given item are stored in a string wave as keyword=value sequences, in waves, or in a text matrix.

This file loader procedure is compliant with proposed standards for user designed file loaders, and it requires the procedures developed for such file loaders. It can be used directly as a "plug-in" module" for the simple file loader panel or other standards compliant file loader user interfaces.

Applications for this file loader include Igor Pro procedures that need access to databases of numeric or string values that are provided in XML format. An example XLS that reads in values (atomic number, density, electron configuration, ...) for 112 elements from an XML periodic table (obtained directly from the Web) is provided. An additional XLS that reads row-based intensity data from an x-ray diffraction system is also given.

Contents

1 Summary

The udStFiLrXML procedure is designed to read an XML file into Igor Pro according to rules in a user-generated XSL file. Values for a given item are stored in a string wave as keyword=value sequences or in waves.

Applications include Igor Pro procedures that need access to databases of numeric or string values that are otherwise provided in XML format.

An example experiment that reads in values (atomic number, density, electron configuration, ...) for 112 elements from an XML periodic table (obtained directly from the Web) is provided.

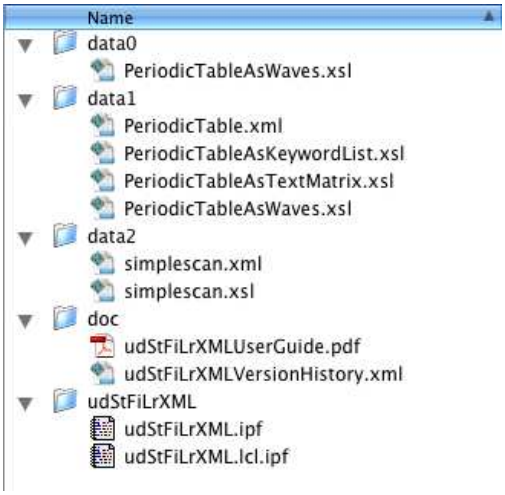
2 Setup

2.1 Requirements

This procedure has only been tested on Igor Pro 6.0 and is defined with a minimum requirement of Igor Pro 6.0. Please see the section Known Limitations in section 5 to learn how this procedure may be used with Igor Pro 5.0.

2.2 Package Contents

The procedure is provided in a ZIP archive. Unpacking the archive will reveal the primary (root) folder udStFiLrXMLN, where N is the version number. The directory structure inside this folder is shown below.



data0
This directory only contains an XSL translation document. It is to be used with the XML file in the data1 directory.

data1
This directory contains XML data from a periodic table of elements and three XSL files to show how to load the data in different formats.

data2
This directory contains XML data from an x-ray diffraction system and an XSL file to show how to load the row of intensity data.

doc
This directory contains the documentation (this document) and an XML version history.

udStFiLrXML
This directory contains two files. The file udStFiLrXML.ipf contains the code to read, parse, and store the XML data into Igor Pro. The file udStFiLrXML.lcl.ipf contains dialog and alert strings in different languages.

2.3 Installation

The procedure can be installed in one of two ways. In both cases, the entire directory `udStFiLrXML` is to be copied (or moved) to a specific location.

- To have the file loader available every time you start Igor Pro, move or copy the `udStFiLrXML` directory to the `Igor Procedures` directory of your local installation of Igor Pro.
- To have the file loader available only when you wish to include it, first move or copy the `udStFiLrXML` directory to the `User Procedures` directory of your local installation of Igor Pro. Then, when you wish to include the `udStFiLrXML` procedure in a specific experiment, open the Macros window while in Igor Pro and put the line

  ```
  #include "udStFiLrXML"
  ```

  somewhere directly after the `#pragma rtGlobals=1` directive that appears in this window.

This procedure can be used directly as a “plug-in” to the Simple Standard Loader Panel Demo, therefore no stand-alone experiment is provided as a user interface.

3 Use

The udStFiLrXML procedure can be used in one of two ways. It can be integrated into more complex routines as a file loader module. An example of this is where the procedure reads preference settings for a package that are stored externally in an XML database. In this case, the user is typically unaware of and has no direct interaction with the operation of the udStFiLrXML loader–it is just a functioning part of the larger package.

Alternatively, the udStFiLrXML procedure may be used as the primary method to load data in XML databases into Igor Pro for further processing. In this case, the user is typically prompted to select the data file(s) to load.

3.1 File Formats

However this procedure is used, a basic familiarity with XML and XSL is essential. The discussion below is no substitute for reading on-line or other references about the formats. In fact, I strongly recommend that anyone who uses this procedure in any way become far more familiar with XML and XSL. The upcoming versions of the udStFiLrXML procedure will rely directly on XSLT engines that are built-in to the computer OS (MacOS or WinXX). All translations will then be directly dependent on how conversant you are with XML/XSL transformations, and such discussion is well beyond the scope of this document.

3.1.1 XML File

The XML file should have the basic format shown below. The (optional) lines are part of well-structured XSL files and should be included for future compatibility. They are not recognized by Ver. 2.12 of the Igor Pro procedure, however future versions of the procedure are likely to require them. All unmarked lines are required for the current version of this procedure.

```
<?xml version="1.0"?> (optional)
<DATABASE> (optional)
  <ELEMENT>
    <VALUE1>value1</VALUE1>
    <VALUE2>value2</VALUE2>
    <VALUE3>value3</VALUE3>
    ...
    <VALUEN>valueN</VALUEN>
  </ELEMENT>
  <ELEMENT>
    ...
  </ELEMENT>
  ...
</DATABASE> (optional)
```

DATABASE
The name of the XML database. For example, the Periodic Table XML file has a DATABASE name of PeriodicTable.

ELEMENT
The name of the elements within the database. For example, all ELEMENTS within the Periodic Table XML file have the name ATOM.

VALUEj
The name of the j^{th} property of the given ELEMENT within the DATABASE. Within the Periodic Table XML file for example, ATOMS have VALUEs such as SYMBOL, NAME, ATOMIC_WEIGHT, DENSITY, and ELECTRONIC_CONFIGURATION.

valuej
The *value* (numeric or string) given to the j^{th} VALUE of the given ELEMENT. For example, for one ELEMENT in the Periodic Table XML file, a SYMBOL value Au is associated with the NAME value Gold. The *valuej* parameter is never enclosed in quotes, regardless of whether it is a string value or a numeric value.
Multilevel nesting to indicate properties of VALUEs is not supported in Ver. 2.12.

One variation of the above XML file format is permitted. The VALUEj names can include UNITS in the following manner:

```
<VALUEj UNITS="unitsj">valuej</VALUEj>
```

UNITS
The designation indicating the VALUEj has units.

unitsj
The units associated with *valuej*. The units must be enclosed in quotations!

An alternative form of data storage in an XML file is shown by the section below.

```
<?xml version="1.0"?> (optional)
<DATABASE> (optional)
  <ELEMENT>
    <NAME>name1</NAME>
    <VALUES>value1 value2 value3 ... valueN</VALUES>
  </ELEMENT>
  <ELEMENT>
    <NAME>name2</NAME>
    <VALUES>value1 value2 value3 ... valueN</VALUES>
    ...
  </ELEMENT>
  ...
</DATABASE> (optional)
```

In the above format, the data is stored as a row within one element of the XML database. The data values may be numeric or text. They may be separated by demarcations other than spaces, however this version of the procedure only recognizes spaces as separators.

3.1.2 XSL File

The basic XSL file should have the format shown below. The (optional) lines are part of well-structured XSL files and should be included for future compatibility. They are not recognized by Ver. 2.12 of the Igor Pro procedure. All unmarked lines are considered to be required in the format specified (unless otherwise noted).

```
<?xml version="1.0" encoding="UTF-8"?> (optional)
<?igorpro version=VERSION input-as=METHOD?> (REQUIRED!)
<xsl:stylesheet xmlns:xsl="..." version="1.0"> (optional)
  <xsl:output method="text" /> (optional)
<xsl:template match="//ELEMENT">
<xsl:value-of select="VALUE1"/>;
<xsl:value-of select="VALUE3"/>;
<xsl:value-of select="VALUE2"/>;
...
<xsl:value-of select="VALUEM"/>;
</xsl:template> (optional)
</xsl:stylesheet> (optional)
```

VERSION

A string version number. As of 2.12, this value should be `"1.03"` (see the known limitations in Section 5).

METHOD

A string that is either `"keyword-list"`, `"waves"`, or `"text-matrix"`. This defines how the *value j* values will be stored when input into Igor Pro.

ELEMENT

The same name that appears in the XML file.

VALUEj

The names of properties that you want to read into Igor Pro from the XML file. Note, in the XSL file, they are not necessarily listed in the order they appear in the XML file. Three rules apply to the `VALUEj` lines.

- The capitalization of VALUE names in the XSL should follow exactly with those given in the XML. See below for instructions on how to store values with other VALUE tags, for example to remove capitalization.
- The first VALUE appearing in the XSL file will be used as the default name of the keyword=value string wave when the values are stored as keyword-lists.
- The line must end in a semicolon ";" unless followed by a unit designation (explained below).

The basic XSL file can be modified in three ways.

- STORAGE NAME

  To indicate that a different name should be used to store a particular value, precede the `<xsl:value-of ... >;` line with the designation

  ```
  namej = <xsl:value-of select="VALUEj"/>;
  ```

  Under default conditions in keyword=value lists, the name of *valuej* will be `VALUEj`. The above coding in the XSL replaces `VALUEj=` with `namej=` as the storage method. This is used for example to store ATOMIC_WEIGHT from the Periodic Table XML file as `AM` in the keyword=value list within Igor Pro.

  The `namej` designation will also be used to name the waves that store the values. For example, the line below in the XSL

  ```
  rate = <xsl:value-of select="Reaction_Rate"/>;
  ```

  is a directive to store all values of `Reaction_Rate` from the XML file into an Igor Pro wave named "rate".

  Finally, when storing as text-matrix, this method will also change the designation of the storage from `VALUEj` to `namej`.

- UNITS

  To indicate a different set of units to use for a given *valuej*, include the designation after the `<xsl:value-of ...>;` line as

  ```
  <xsl:value-of select="VALUEj"/>;"new units"
  ```

  Under default conditions, the *units* given in the XML file will be applied. The above use will override the units in the XML file. This is used for example to designation that ATOMIC_WEIGHT from the Periodic Table XML file is to be stored in units of "kg/mol" rather than "g/mol".

  The `"new units"` designation must follow after the `value-of ...` selection and after the required semicolon. The new units must also be enclosed in quotations.

- ALGEBRAIC MANIPULATIONS

  The XSL parse routine within udStFiLrXML supports the interpretation of simple algebraic coding to indicate manipulation of numeric values is to occur before storing. Examples of such formating are provided as illustration.

  ```
  TK = 273.15 + <xsl:value-of select="temperature"/>;"K"
  ```

  This line directs the procedure to add 273.15 to every *value* of `temperature` and store them as `TK` with units of "K".

```
lnrate = ln(<xsl:value-of select="REACTION_RATE"/>);
```

This line takes the natural log of every *value* for REACTION_RATE and stores them as lnrate.

As of Vers. 2.12, algebraic manipulations are only supported using one *value* per line in the XSL file.

3.2 Storage Options

The values from the XML can be stored in Igor Pro as keyword=value string lists, as waves, or as a text-matrix. An option is also available to store row-type data as waves.

3.2.1 Keyword=Value Lists

To store the XML data in this format, include the line below within the XSL file.

```
<?xmligorpro version="1.03" input-as="keyword-list"?>
```

The name of the string waves will be obtained from the name of the first ELEMENT within the XSL file (programmers have an option to override this behavior).

The keyword=value string wave will have the format

```
name1=value1,units1; name2=value2,units2;...nameN=valueN,unitsN;
```

where each *namej* is stored in the order that it appears in the XSL file. When a *valuej* has no units, it will be stored as namej=valuej; only.

All values within the string wave are strings, even when the original value within the XML file can be considered a numeric value.

All numeric values stored within the keyword=value list are pre-processed according to the algebraic manipulation rules set in the XSL before being converted and stored in the keyword=value list.

The first value-of within the XSL file establishes the name of the string wave. For the lines below, the names of the string waves will be the NAME value of an ELEMENT.

```
<xsl:value-of select="NAME"/>; (first value-of line)
<xsl:value-of select="SYMBOL"/>;
```

By comparison, for the lines below, the names of the string waves will be the SYMBOL value of an ELEMENT.

```
<xsl:value-of select="SYMBOL"/>; (first value-of line)
<xsl:value-of select="NAME"/>;
```

By default, the first keyword=value is not stored in the string wave. To include it within the string, duplicate the line as shown below.

```
<xsl:value-of select="SYMBOL"/>; (first value-of line)
<xsl:value-of select="SYMBOL"/>; (repeated here to include SYMBOL in list)
<xsl:value-of select="NAME"/>;
```

3.2.2 Waves

To store the XML data in this format, include the line below within the XSL file.

```
<?igorpro version="1.03" input-as="waves"?>
```

Every VALUEj within the XSL will create a new wave with that name. The *valuej* values will be stored sequentially by ELEMENT within the wave.

The procedure recognizes the difference between text and numeric values. Text values will be stored in string waves, and numeric values in (numeric) waves (as floats).

Algebraic manipulation will be done prior to storing a value as demanded within the XSL file.

The units of a given VALUEj will be stored in a string variable called VALUEjUNITS. For example, the line below will store all ATOMIC_WEIGHT values in a numeric wave named AM. The values will have been pre-multiplied by 0.001 prior to storage. A string variable AMUNITS will contain the string “kg/mol”.

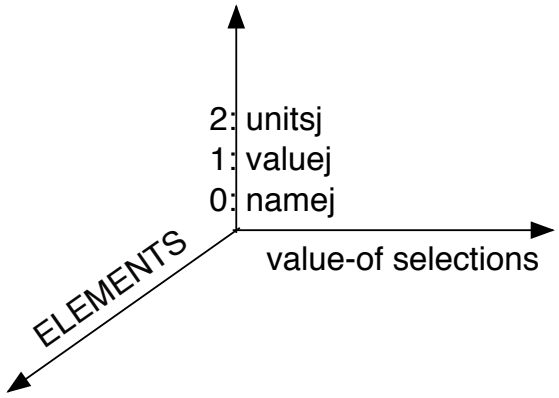
```
AM = 0.001*<xsl:value-of select="ATOMIC_WEIGHT"/>;"kg/mol"
```

3.2.3 Text-Matrix

To store the XML data in this format, include the line below within the XSL file.

```
<?igorpro version="1.03" input-as="text-matrix"?>
```

The matrix will be filled along rows, columns, and layers. Following conventions in Igor Pro, the indices along the rows, columns, and layers are zero based (the first index is zero). The matrix will have dimensions of [N][3][M], where N is the number of `ELEMENT` segments in the XML and M is the number of `value-of` lines in the XSL (including the first one). In the columns, index zero will have the name of the value stored, index one the value, and index two the units. A representation of this is shown below.



The selection given in the first `value-of` line in the XSL will define the name of the text matrix. The `valuej` obtained from the XML for that selection will be the first (index = zero) value of each row.

As a specific example, the sequential `value-of ...` lines below will create the matrix named “symbol” that follows (showing only the first two elements for brevity).

```
symbol=<xsl:value-of select="SYMBOL"/>;
AM = 0.001*<xsl:value-of select="ATOMIC_WEIGHT"/>;"kg/mol"
ro = <xsl:value-of select="ATOMIC_RADIUS"/>;"Angstroms"
den = 0.001*<xsl:value-of select="DENSITY"/>;"kg/m3"
```

Third Layer	units	kg/mol	Angstroms	kg/m3	
	units	kg/mol	Angstroms	kg/m3	
Second Layer	value	0.0227		1.88	10.07
	value	0.02698154		1.43	2.7
First Layer	Ac	AM	ro	den	
	Al	AM	ro	den	

The following (string/text) results would be obtained for the contents of the matrix symbol:

```
symbol[0][0][0] = Ac
symbol[0][0][1] = value
symbol[0][0][2] = units
symbol[1][0][0] = Al
symbol[1][0][1] = value
symbol[0][1][0] = AM
symbol[0][1][1] = 0.0227
symbol[0][1][2] = kg/mol
symbol[0][2][0] = ro
symbol[0][2][2] = Angstroms
```

Any value that is blank or missing in the XML file is stored as a blank cell or cells in the matrix.

3.2.4 Row Waves

This storage format is really a translation operation rather than a new storage method. When the XML data appear as a row vector within one element of the XML database, this operation will pull out each data value as an element in a wave.

To use this translation/storage option, include the following line in the XSL:

```
<?igorpro version="1.03" input-as="row-waves"?>
```

The translation/storage is illustrated with the example below.

XML File

```
<data>
  <values>64 56 78 92 45 36 33 48</values>
</data>
```

XSL File

```
<xsl:template match="//data">
mydata = <xsl:value-of select="values"/>;
```

Igor Pro Storage

```
wave mydata
... type: numeric
... number of points: 8
```

The procedure only recognizes spaces as separators between row delimited values. The procedure recognizes the difference between text and numeric values. Text values will be stored in string waves, and numeric values in (numeric) waves (as floats).

Algebraic manipulation will be done prior to storing a value as demanded within the XSL file.

The units of a given `VALUEj` will be stored in a string variable called `VALUEjUNITS`. For example, the line below will store all `ATOMIC_WEIGHT` values in a numeric wave named `AM`. The values will have been pre-multiplied by 0.001 prior to storage.

3.3 Standard User Interface Modes

As a standardized file loader, `udStFiLrXML` has three modes of operation: Auto, Auto-XML, and Manual. Further details of these modes are presented again in the section For Programmers.

Auto Mode

In Auto mode, the procedure must be provided with all information needed to automatically load and parse the files. In particular, this includes the file names for the XSL

and XML files. This mode is typically used by programmers who wish to integrate the routine into a larger package.

Auto-XML Mode

In Auto-XML mode, the procedure will present a dialog box requesting the user to select an XSL file. The procedure then searches in the same directory for an XML file with the same name.

As an example, the following two files can be processed in this mode:

```
mydata.xsl
mydata.xml
```

At the first dialog prompt, the user should select the `mydata.xsl` file. The routine will automatically find the `mydata.xml` file and process it.

Both files must be in the same directory!

Manual Mode

In Manual mode, the procedure will present a dialog box requesting the user to select an XSL file followed by a dialog box requesting the user to select an XML file. Both files can be in separate directories.

You can test all three of these modes in the Simple Standard Loader Panel Demo using the data provided in the directories `data0`, `data1`, and `data2`.

3.4 Localization

You can change the dialog and alert texts to a different language. This is done by modifying the `udStFiLrXML.lcl.ipf` file.

The file starts with a compiler directive that is a definition of the current language. Below this are the language specific alert and dialog strings. A subset of the file is shown below.

```
#define ENGLISH

#ifdef ENGLISH
StrConstant alert00 = "..."
StrConstant alert01 = "..."
StrConstant alert01a = "..."
StrConstant alert02 = "..."
```

```
...
#endif
```

To change the language, first confirm that a command block `#ifdef ... #endif` exists for the language you want to use. Then, change the language after the `#define` directive to that language. Please use ALL UPPER CASE LETTERS when doing this!

The line given below changes the language from English to German, assuming the block of code `#ifdef DEUTSCH ... #endif` exists to define the strings in German.

```
#define DEUTSCH
```

To create a new language localization, copy the entire existing block of code found between `#ifdef ENGLISH ... #endif`. Paste this at the bottom of the localization file. Change the `#ifdef ENGLISH` to reflect the language you wish to define. Then, change all of the string text (between quotation marks) to reflect the proper translation of the alert and dialog strings. Do not change the names of the alert or dialog string constants when doing this.

I encourage anyone who develops a new language localization for this procedure to forward it to me so that I can post it for others to use.

3.5 For Programmers

Programmers who wish to use the procedure directly in their own Igor Pro routines are encouraged to review the guidelines below. Only one point of top-level access exists.

Function udStFiLrXML(udFL)

This function is accessed according to the rules defined in the proposed standard document. The relevant parameters are outlined below.

Descriptors

```
udFL.mimetype = "TEXT"
udFL.extensions = ".xsl;.xml;"
udFL.procModes = "Auto;Auto XML;Manual;"
udFL.itemsType = 14
udFL.itemsDim[0]=0
udFL.setNames=1
```

When queried with `udFL.itemsType` set, the procedure also returns the following:

```
2
  udFL.Dim[0] = 1
4
  udFL.Dim[0] = 99
8
  udFL.Dim[0] = -1
  udFL.Dim[1] = -1
  udFL.Dim[2] = -1
  udFL.Dim[3] = 0
```

Inputs

The inputs will control the loader in the following manner:

```
udFL.userCtrl ... sets processing mode
udFL.userData ... used during programming (see below)
udFL.reportCtrl ... used only sparingly
```

The “Silent” mode of control and the “Normal” mode of control are the same. The “Verbose” mode of control keeps the parsing notebook resident after it has been used and reports progress of the loading, parsing, and saving operations throughout.

Localizers

The inputs will control the loader in the following manner:

```
udFL.pathStr ... according to standards
udFL.fileList ... according to standards
udFL.returnCtrl ... currently unused
```

In “Auto” mode, the values of `udFL.pathStr` and `udFL.fileList` can be passed in one of two ways. In the first case, `udFL.pathStr` should contain the full path to two files in the same directory. Correspondingly, `udFL.fileList` should contain just the names of the two files, with the XSL file first and the XML file second in the list. In the second method, the value of `udFL.pathStr` should be passed empty (as `""`), and `udFL.fileList` should contain the full path to the XSL file followed by the full path to the XML file.

Returns

The file loader will return -1 on a fatal error. It currently has its a mixture of its own internal alerts and standardized alerts via `udFL.errCode` and `udFL.errMsg`.

4 Package Contents

4.1 Folder

The folder for this package is called `XML2IgorPro` (I tend to follow a philosophy of avoiding liberal names in setting up package folders within Igor Pro).

The package installs itself as root:Packages:XML2IgorPro (following conventions suggested by WaveMetrics).

The package folder contains two parameters and any number of sub-folders.

4.1.1 Parameters

Parameter	Type	Use
currXMLData	string	the (compressed) name of the current XML data file
currXSLData	string	the (compressed) name of the current XSL data file

4.1.2 XSL Sub-Folders

Every set of XSL/XML database values is stored in Igor Pro in its own folder located directly from the root level. For every one of these folders, a corresponding folder is created in the `XML2IgorPro` folder. The name of that folder is `dbnameXSL` where *dbname* is the name of the Igor Pro folder where the data is stored. For example, when values are stored in root:PeriodicTable, the parameters in the Igor Pro folder root:Packages:XML2IgorPro:PeriodicTableXSL define how they were derived.

Each XSL folder has the parameters listed below, as obtained from parsing the XSL.

Parameter	Type	Use
eKey	string	the KEY obtained from match="//KEY"
firstKey	string	the KEY in the first `<xsl:value-of select=...>` line
sunits	text wave	the ordered list of any units for all KEYS
scmd	text wave	the ordered list of commands to EXECUTE for each KEY
sname	text wave	the ordered list of names to store all KEYs
ename	text wave	the ordered list of names that define all KEYs

The text waves will have as many points as the number of elements that are directed by the XSL file to be input from the XML database.

4.2 Structures

The procedure uses one "global" structure and one static structure.

- *udFL* - (STRUCTURE)
  This is a reference to the global standard structure `udFLStandardStructure`
- *xip* - (STRUCTURE)
  This is a reference to a static structure `xml2igorproglobals` that contains

```
xip.version ... the version of the routines
xip.xslversionsupported ... the version of XSL supported
```

4.3 Procedures

The procedure contains the functions listed below. Those that can be called from outside the procedure are listed with ←.

Function udStFiLrXML(udFL) ←
This is the primary point of entry to the file loader.

Example:
`udStFiLrXML(udFL)`

Static Function udFLXMLInitialize(udFL)
This is called when the file loader is initialized.

Static Function udFLXMLQuery(udFL)
This is called when the file loader is queried.

Static Function initParseXML()
This function is called the first time the procedure `inputXSLFile(...)` is invoked to set up the package folder and globals.

Static Function readXSLFile(udFL)
This function reads the XSL file into a (hidden) notebook.

Static Function parseXSLFile(udFL)
This function parses the XSL file (from within the notebook).

Static Function readXMLFile(udFL)
This function reads the XML file into a (hidden) notebook. It passes `udFL.userData` as the optional `KEY` for keyword=value parsing.

Static Function parseXMLFile(udFL)
This function parses the XML file (from within the notebook). It receives the `udFL.userData` as the optional `KEY` for keyword=value parsing.

Static Function/S parseXMLValue(lstr,xhow)
This function parses the lines in the notebook.

- *lstr* This is the string line to be parsed.
- *xhow* This is a variable defining how to parse the line. A value 0 means to parse for the `UNITS` (if any), a value 1 means to return the value after collapsing all spaces, a value 2 means to return the value with spaces, and a value 3 means to return `lstr` after replacing all spaces with semicolons `";"`.

---

5 Known Limitations

- The procedures can likely be used with Igor Pro 5.0 by making the following modifications:
  - First, put the udStFiLrXML folder in the `User Procedures` folder of Igor Pro. This modification will NOT work when the udStFiLrXML folder is in the `Igor Procedures` folder!
  - Open the udStFiLrXML.ipf file within Igor Pro.
  - Change the `#pragma IgorVersion=6.0` to `#pragma IgorVersion=5.0`
  - Change the line that states `#include ":udStFiLrXML.lcl"` to read exactly as `#include "udStFiLrXML.lcl"` (remove the colon before the file name).

  - Save the procedure file.

- To use the stand-alone procedure when the `udStFiLrXML` folder is placed in the `Igor Procedures` directory, comment out the line `#include "udStFiLrXML"` in the procedure window of the stand-alone experiment.
- Please pay attention to the `igorpro` version number you give in your XSL. This will assure that your data is handled by the correct version of the `udStFiLrXML` procedure file.
- Simple algebraic manipulations on one value are supported. To perform complicated manipulations, load the data as waves and work directly within Igor Pro.
- The procedure works carefully to remove spaces in the XSL and XML file names (to avoid the need for liberal string quoting in Igor Pro). It may or may not be able to read KEY names with spaces - no testing of this has been done as of vers. 2.12.
- The procedure parses the XSL and XML by reading them into a notebook. Alternative methods using XSLT engines are to be included in future versions.

6 Acknowledgments

Many thanks go to Shaun Roe who was instrumental in establishing the proper format for the XSL file in the transition from 1.12 to 2.0.

Thanks to Luc Ortega for sending the XRD data as a motivation to create the row-wave method of input.

7 Contact

Suggestions, bug reports, and feature requests should be posted on the Igor Exchange Web site. I will be using it exclusively to track this project.

8 Legalize

This software is free to use as per the terms of any other publicly released software. Enjoy!

9 Version History

An XML file outlining the version history is provided in the docs folder.

2.12r (2007.08.25)
- changed name from xml2igorpro to udStFiLr XML to indicate standard compliance (version 2.04r is technically the last version of xml2igorpro)
- included udFL.setNames=1 parameter in initialization
- fixed an incorrect test of `V_flag` instead of `S_filenames` after Open dialog box

---

2.11r (2007.08.22)
- verbose mode prints status at key points during processing
- file loader returns the items list and number of items
- re-distributed coding in the initialize and query segments
- fixed a missing alert code problem that prevented compiling
- open dialogs now check only for file extensions exclusive of mimetype
- itemsType now conform to BIT-wise values

2.10r (2007.08.20)
- rewrote the code to conform to the working proposal on standardizing file loaders
- checks for proper values of the optional KEY
- added row-wave as storage option
- require <?igorpro version =1.03 (to read as row-wave)

2.04r (2007.06.05)
- changed <?xmligorpro version to <?igorpro version (in preparation for use of xslt engines)
-added localization file to have dialogs and alerts in different languages
- put file and localization file in a directory (director structure changed accordingly)
- added text-matrix as storage option
- require <?igorpro version =1.02 (to read as text-matrix)
- request (not require) that all lines in XSL end in ";" or in text (units string) (in preparation for use of xslt engines)

2.03r (2007.05.07)

- added optional `filepath` parameters to `inputXSLFile(...)` and `inputXMLFile(...)`
- added procedure information to the Users Guide
- updated Known Limitations information in the Users Guide

2.02b0 (2007.05.03)

- significant code changes to improve speed and reliability of parsing XML
- moved coding for how=1 in parseXML to its proper location
- missing <KEY></KEY> lines in XML are now stored as "keyword=" or as blank values in wave (you must use version `"1.01"` in the `<?xmligorpro ...>` header!)
- return 0 cases in parseXML now kill notebook and reset datafolder
- added Package Folder information to the Users Guide
- added Known Limitations information to the Users Guide

2.01b1 (2007.05.02)

- fixed ordering of lines in XSL and clarified format requirements in the Users Guide

---

2.01b0 (2007.04.30) (not released)

- used an `<?xmligorpro version="..." input-as="..."?>` directive to control versioning of the xml parsing and input directives to the parser
- removed the `<xsl:parameter ...>` directive as a way to define the input method
- procedure now checks for compatible XML version information in the XSL

2.00b0 (2007.04.29) (not released)

- major change in format of XSL file (therefore the major version number change)
- cleaned up hidden notebook and other junk after abort of XSL or XML input
- can now input XML data to waves
- only one keyword=value input type supported
- can now define pre-processing of numeric values using algebraic formula in XSL
- unit designations stored as <xsl:value-of ...>; “UNITS”
- bug fixes and code revisions

1.12b0 (2007.04.24)
- changed versioning to include letter (a, b, r) and minor number
- made pragma rtGlobals=1
- put Static Function states on internal functions
- can use KEY as name for storage wave (default is “name”)
- has how=0, how=1, and how=2 storage states
- reads UNITS in XLM file
- code changes and revisions

1.11
- changed names of files and folder from ParseXML to XML2IgorPro
- first public posting

1.10
- added "how" switch to inputXMLFile() function for future use (currently unused option) (not posted)

1.00 - first development (not posted)
