Fits Explorer

Technical Reference



|  |  |
| --- | --- |
| Author | Russell Archer |
| Date | 16th February, 2012 |
| Version | 1.00 Draft A |

Contents

[Application Design 3](#_Toc317171708)

[FITS Files 3](#_Toc317171709)

[Neutral data format 3](#_Toc317171710)

[Open and extensible meta-data header 3](#_Toc317171711)

[Internal Structure 3](#_Toc317171712)

[FITS Header 4](#_Toc317171713)

[FITS Image Data 4](#_Toc317171714)

[How Grayscale Images are Stored 5](#_Toc317171715)

[How Color Images are Stored 5](#_Toc317171716)

[FITS Library Support 5](#_Toc317171717)

[Windows Thumbnails and Previews 5](#_Toc317171718)

[Registry Support 6](#_Toc317171719)

[Adding a Custom Preview Handler 6](#_Toc317171720)

[Adding a Custom Thumbnail Provider 7](#_Toc317171721)

[Installing the Preview Handler and Thumbnail Provider 8](#_Toc317171722)

[Removing the Preview Handler and Thumbnail Provider 8](#_Toc317171723)

[References 8](#_Toc317171724)

# Application Design

FITS Explorer is designed for Windows Vista and Windows 7, with both 32-bit and 64-bt versions available. A Metro-style version will be available for Windows 8. Note that while the core application runs on Windows XP, some features like Windows explorer integration for previews, thumbnails, etc. does not work and is not supported.

# FITS Files

Used by both professional and amateur astronomers, FITS files are the most commonly-used format for capturing astronomical image data. The popularity of the FITS file format is due to two primary reasons:

Neutral data format  
Data, which may represent an image, spectrum, graph, or any tabular data set, is stored in a platform- and format- neutral structure. For the purposes of this document we assume the data represents an image.

One or more image data sets are stored as two-dimensional arrays of numbers. The numeric data can be unsigned 8-bit integer, signed 16- or 32-bit integer, or 32- or 64-bit floating-point format. Thus, each platform-neutral pixel can be described with a high-degree of fidelity if required.

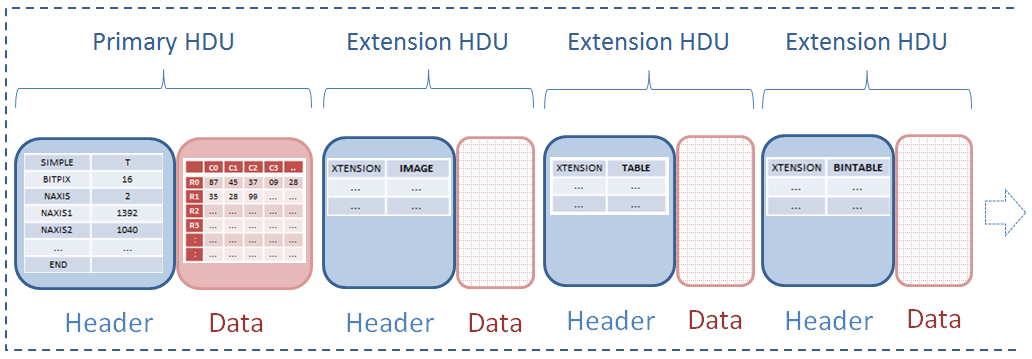
It is the responsibility of client applications to interpret data in a manner consistent with the host environment. For example, a Windows application must retrieve the raw numeric image data from the FITS file and then encode that data as a JPEG, PNG, BMP, TIFF, etc. capable of being displayed in a Windows environment

## Open and extensible meta-data header

Associated with each image data set is a meta-data header that can be used to describe the data (e.g. the number of bits used to describe each platform-neutral pixel of image data, etc.), the manner in which the data was captured (the exposure time, filter used, etc.) .

## Internal Structure

The internal structure of FITS files is comprised of one or more *Header/Data Units* (HDU). The primary HDU must be present, and may, optionally, be followed by one or more *extensions*. The FITS standard documents three types of extension: *IMAGE* (an 2-dimensional array of pixel values), *TABLE* (text-based tabular data) and *BINTABLE* (binary representation of tabular data):



Thus, a single FITS file may contain multiples ‘images’ if required. A fourth type of extension (RANDOM GROUPS) is used in radio astronomy applications, and is not considered here.

## FITS Header

Each header is composed of any number of ASCII text strings with the following format:



Most header keywords are optional. However, a certain number of required keywords/values should always be present. For example, the header for an HDU containing an image should always have the following:

SIMPLE = T / **flags that this is a FITS file (T = true, F = false)**  
BITPIX = 16 / **the number of bits used to represent each image data pixel**  
NAXIS = 2 / **the number of data axes (2 (height/width) for grayscale images, 3 for color)**  
NAXIS1 = 1392 / **length of data axis 1 (width of image)**  
NAXIS2 = 1040 / **length of data axis 2 (height of image)**

The end of the header is marked by an *END* keyword. Note that the header is always a multiple of 2880 bytes in length, and is padded with ‘blanks’ or ASCII NULL values as required.

A full list of standard FITS keywords is available from: <http://heasarc.gsfc.nasa.gov/docs/fcg/standard_dict.html>

## FITS Image Data

The *BITPIX* header value signifies the number of bits used to represent each pixel:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bits per pixel | BITPIX  value | Signed/  Unsigned | Numeric Type | Min  value | Max  value | C#  Type | .NET  Type |
| 8 | 8 | Unsigned | Integer | 0 | 255 | byte | System.Byte |
| 16 | 16 | Signed | Integer | -32,768 | 32,767 | short | System.Int16 |
| 32 | 32 | Signed | Integer | -2,147,483,648 | 2,147,483,647 | int | System.Int32 |
| 32 | -32 | - | Real | -3.4 × 1038 | 3.4 × 1038 | float | System.Single |
| 64 | -64 | - | Real | ±5.0 × 10−324 | ±1.7 × 10308 | double | System.Double |

The most commonly used values are signed 16-bit integer and 32-bit floating point.

Note that many amateur CCD imaging cameras generate raw data in *unsigned* 16-bit format, with a range of 0 (black) to 65,535 (white). The FITS provides a standardized solution for such cases via the *BZERO* and *BSCALE* keywords. This allows the data stored in the FITS file to be scaled and/or offset as required.

For example, the Starlight Xpress SXVR-H9 camera generates unsigned 16-bit raw data (0...65,535). When using an industry-standard imaging package such as MaxIm DL, a value of 32,768 is *subtracted* from each pixel of raw data when it is saved to a FITS file. This results in 16-bit signed data with the required range of -32,768...32,767.

If required, an offset may be applied to raw camera data to “shift” the data range. The *BZERO* FITS value would then be used to indicate the amount by which raw has been offset.

Any software system (including *FITSExplorer*) that loads FITS files must inspect the *BITPIX*, *BZERO* and *BSCALE* values to determine how to treat stored image data.

For example:

BITPIX = 16  
BZERO = 1  
BSCALE = 32768

***Transformation applied to each pixel value before encoding the data as an image:***

BZERO + (BSCALE \* FITSPixelValue)

Note that compressed FITS files are not supported.

### How Grayscale Images are Stored

### How Color Images are Stored

NAXIS = 3 (always set to 3 – Red image stored in the primary HDU, with the Green and Blue images stored in extensions 1 and 2 respectively. )

# FITS Library Support

A number of FITS-related software libraries exist, including the NASA-maintained [*CFITSIO*](http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html) library of C and Fortran routines. There is also [*CSharpFITS*](http://vo.iucaa.ernet.in/~voi/CSharpFITS.html), which provides a .NET/C# wrapper for CFITSIO.

FITS Explorer does *not* make use of any commercial or open-source FITS library for a number of reasons, paramount of which is *performance*. During early prototyping, it was found that none of the available libraries performed as required. Thus, a simple, lightweight framework designed to support the reading of FITS header and image data was developed. Written in a combination of managed (C#) and unmanaged (C++) code, this framework provides the desired performance to meet the design goal of presenting a preview image to user in under a second.

# Windows Explorer Integration

FITS Explorer provides Windows explorer thumbnails and previews via *shell extensions*. Thumbnails are supported by an unmanaged C++ library (*FITSThumbnailProvider*), and previews are generated by C# libraries (*PreviewHandlerFramework* and *FITSPreviewHandler*).

Note that the older COM interfaces (e.g. *IExtractImage*) used by Windows XP are **not** supported by FITS Explorer. Starting with Windows Vista, a new set of shell extension interfaces (*IPreviewHandler* and *IThumbnailProvider*) provide improved preview and thumbnail features. Thus, FITS Explorer image previews and thumbnails do not function on systems prior Windows Vista.

## Registry Support

Windows shell uses the registry to look-up the mapping between a particular file type (extension) and an associated preview/thumbnail provider.

In Windows XP and earlier, the *IExtractImage* COM interface was used to support both preview and thumbnail images. For example, MaxIm DL uses *IExtractImage* to produce previews and thumbnails.

With Windows Vista, the *IPreviewImage* and *IThumbnailProvider* interfaces were introduced to support improved file previews and thumbnails. The older-style *IExtractImage* interface continues to be supported in Windows Vista and Windows 7.

The following table summarizes the registry GUIDs used to denote particular functions:

**GUID COM Interface Platform**E357FCCD-A995-4576-B01F-234630154E96 *IThumnailProvider* Windows Vista/7  
8895b1c6-b41f-4c1c-a562-0d564250836f *IPreviewHandler* Windows Vista/7  
BB2E617C-0920-11d1-9A0B-00C04FC2D6C1 *IExtractImage* Windows XP/2000

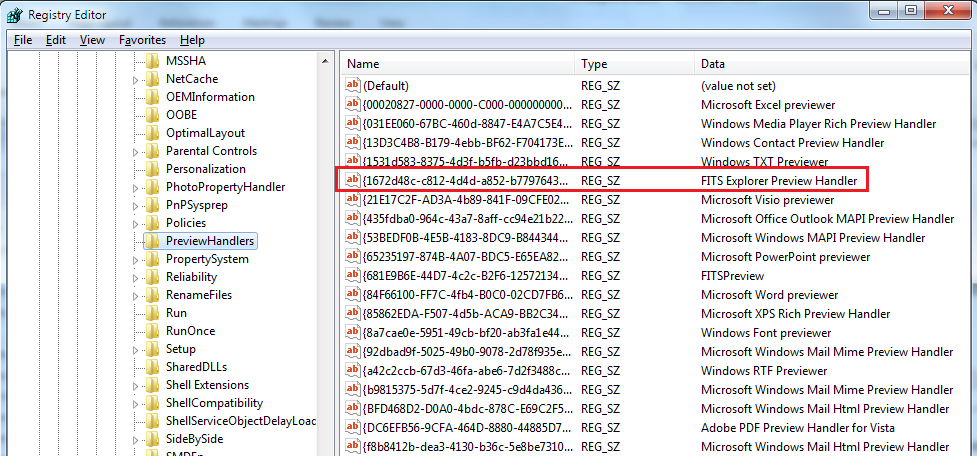
Note that FITS Explorer uses *unmanaged* code for the thumbnail provider shell extension. Microsoft does not recommend using managed code in this scenario. This is because extensions are loaded into the shell’s process, leading to potential .NET framework versioning conflicts, when various shell extensions require different framework versions (only one version of the .NET CLR can be loaded into a process at one time). Starting with .NET 4, this single version limitation has been removed, but Microsoft still explicitly recommend against managed code extensions to the shell (explorer.exe).

However, a *managed* code solution *can* be used for the custom preview handler. Preview handlers are implemented as COM components, but they are hosted outside the shell’s process by a “surrogate host” (prevhost.exe).

### Adding a Custom Preview Handler

Three changes need to be made to the registry when adding a custom preview handler. In the case of FITS Explorer, the following changes are made to support previews of .*fts*, .*fit* and .*fits* files:

1. Associate one or more file extensions with the custom preview handler  
   1. Locate the *HKEY\_CLASSES\_ROOT\.fts* key, if it doesn’t exist, it will be created.   
      Note that MaxIm DL adds support for explorer previews and thumbnails via the old-style *IExtractImage* interface. However, it only supports 32-bit systems – previews on 64-bit machines are not supported by MaxIm and do not function correctly
   2. If the key *shellex\{BB2E617C-0920-11d1-9A0B-00C04FC2D6C1}* is present, this denotes the GUID for an old-style *IExtractImage* interface – this will need to be “disabled” or it will conflict with the new custom preview handler and thumbnail provider
   3. Create *HKEY\_CLASSES\_ROOT\.fts\shellex\{8895b1c6-b41f-4c1c-a562-0d564250836f}*  
      This GUID denotes the *IPreviewImage* interface.  
      Set the *Default* value (as a *REG\_SZ*) for the key to be the CLSID for the custom preview handler (e.g*. {1672d48c-c812-4d4d-a852-b779764373c7}*)
   4. Repeat a. through c. to add support for the .*fit* and .*fits* file extensions
2. Create a class ID (*CLSID*) related to the custom preview handler  
   1. A sub-key, which is the CLSID (*{1672d48c-c812-4d4d-a852-b779764373c7}*) for our preview handler, is added under the follow key for the custom preview handler:  
        
      32-bit systems: *HKEY\_CLASSES\_ROOT\CLSID*64-bit systems: *HKEY\_CLASSES\_ROOT\Wow6432Node\CLSID*The CLSID entry holds details of our preview handler, including *CLSID*, *AppID*, *DisplayName*. It also points to the assembly for our preview handler (*FITSPreviewHandler.dll*), which is installed in the global assembly cache (GAC)
   2. Note that a *ProgID* entry is also created at *HKEY\_CLASSES\_ROOT\RussellArcher.FITSPreviewHandler*This has a single sub-key (*CLSID*) which holds the GUID for our preview handler
3. Add the custom preview handler to the list of all preview handlers  
   1. A list of all the system’s preview handlers is maintained at   
      *HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\PreviewHandlers*



Note that for Windows 7 systems, if FITS Explorer is installed for all users, the *HKEY\_LOCAL\_MACHINE* key is used. If it is installed only for a particular user, the *HKEY\_CURRENT\_USER* key is used.

### Adding a Custom Thumbnail Provider

Adding a shell extension for a custom thumbnail provider is very similar to adding a preview handler:

1. Associate one or more file extensions with the custom thumbnail provider  
   1. Create *HKEY\_CLASSES\_ROOT\.fit\shellex\{E357FCCD-A995-4576-B01F-234630154E96}*  
      This GUID denotes the *IThumbnailProvider* interface.   
      Set the *Default* value (as a *REG\_SZ*) for the key to be the CLSID for the custom thumbnail handler (e.g*. {8071E9C3-B199-4ED4-ACBC-E8FDD399A0E4}*)
   2. Repeat a. to add support for the .*fit* and .*fits* file extensions
2. Create a class ID (*CLSID*) related to the custom thumbnail provider  
   1. A sub-key, which is the CLSID (*{8071E9C3-B199-4ED4-ACBC-E8FDD399A0E4}*) for our thumbnail provider, is added under the follow key:  
        
      32-bit systems: *HKEY\_CLASSES\_ROOT\CLSID*64-bit systems: *HKEY\_CLASSES\_ROOT\Wow6432Node\CLSID}*

The CLSID entry will hold two keys: *Default* value will be the path to *FITSThumbnailProvider.dll*, and *ThreadingModel* value will be set to *Apartment*

### Installing the Preview Handler and Thumbnail Provider

The preview handler (*FITSPreviewHandler.dll*) is a managed code assembly which is installed in the GAC using:

gacutil -i FITSPreviewHandler.dll  
regasm /codebase FITSPreviewHandler.dll

The thumbnail provider (*FITSThumbnailProvider.dll*) is an unmanaged code COM component which is registered using the regsvr32 tool:

regsvr32 FITSThumbnailProvider.dll

### Removing the Preview Handler and Thumbnail Provider

The preview handler and thumbnail provider are removed as follows:

gacutil -u FITSPreviewHandler.dll  
regasm /unregister FITSPreviewHandler.dll  
regsvr32 /u FITSThumbnailProvider.dll

## References

* *How to Register a Preview Handler*:  
  <http://msdn.microsoft.com/en-us/library/windows/desktop/cc144144(v=vs.85).aspx>
* *View Data Your Way With Our Managed Preview Handler Framework*:  
  <http://msdn.microsoft.com/en-us/magazine/cc163487.aspx>
* *IThumbnailProvider Re-Visited*:  
  <http://www.codemonkeycodes.com/2010/01/11/ithumbnailprovider-re-visited>
* *Thumbnail Handlers*:  
  <http://msdn.microsoft.com/en-us/library/windows/desktop/cc144118(v=vs.85).aspx>
* *Creating Shell Extension Handlers*:  
  <http://msdn.microsoft.com/en-us/library/windows/desktop/cc144067(v=vs.85).aspx>