

## Extending the XImtool Display Server Using Components

Michael Fitzpatrick and Doug Tody

*National Optical Astronomy Observatories*<sup>1</sup>, *IRAF*<sup>2</sup> Group

**Abstract.** The IRAF<sup>2</sup> display server, XImtool, has been enhanced to allow arbitrary custom components to be dynamically integrated, permitting the functionality of the main display server program to be extended in arbitrary ways with no change to the core program. Components may include both computational and GUI elements; new components are interfaced as named instances of the Widget Server *Client* class and are fully integrated into the base server using messaging. The compiled portion of a component executes as an external process, allowing components to be written in any language or to use any existing resources. To illustrate the use of this facility, an example is shown that provides real-time pixel value and WCS display using direct access to the actual disk image used to load the display.

### 1. Introduction

*XImtool* was originally designed using the IRAF Widget Server architecture (Tody, 1995). This provided a separation of the client application code and the user interface, either of which could be replaced or augmented with little or no change to the other. Using traditional methods, however, adding any major new functionality to the program would still likely involve substantial changes to both the client and GUI code. To solve this, XImtool has been enhanced to allow arbitrary custom components, executing as external processes, to be dynamically loaded at runtime.

New components are interfaced as named instances of the Widget Server *Client* class and are fully integrated into the base server using a simple text messaging scheme similar to that already used by client and GUI communications in the Widget Server. New GUI elements may be uploaded from the component and created when needed, and additional callback procedures can be defined that subscribe the component to events in the GUI (frame changes, cursor movement, etc.) to which it can react. Unlike the traditional *plug-in* model, the compiled portion of a component executes as an external process, allowing components

---

<sup>1</sup>National Optical Astronomy Observatories, operated by the Association of Universities for Research in Astronomy, Inc. (AURA) under cooperative agreement with the National Science Foundation

<sup>2</sup>Image Reduction and Analysis Facility, distributed by the National Optical Astronomy Observatories

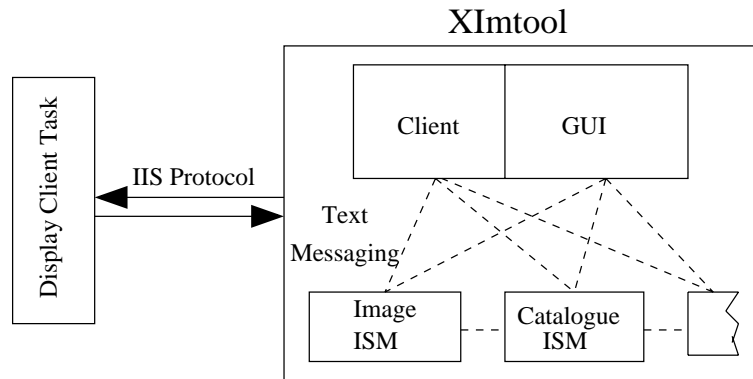


Figure 1. Architecture of the new system.

to be written in any language. The main server and components communicate over an IPC socket layer which is transparent to both. Components attach themselves by requesting a connection on a dedicated socket. The module and XIImtool then negotiate a private channel for communications, allowing other modules to later connect as needed.

*WCSPPIX*, the first of these Image Support Module (ISM) components to be implemented using this new facility, provides real-time pixel value and WCS display using direct access to the actual disk image used to load the display. The module is written as an IRAF task to allow easier access to all image formats supported by the IRAF interfaces (including Mosaics), something that could not be easily done using the old architecture. Other ISM components may now be easily written to provide new functionality such as vector graphics display, network catalog access, or other image analysis. With only a minimal understanding of the XIImtool GUI, users can write their own components using a standard API provided.

## 2. Architecture

The basic XIImtool architecture (Figure 1) is a single process in which text messages (e.g., to tell the application to change the frame, or the GUI to set the label of a button) are exchanged between the client and GUI to provide the interactivity and core functions of the task. New functionality is supplied by one or more ISM components running as a separate process and connected by text messaging over a socket layer between the component and the base server. ISM modules may send messages to the XIImtool client, its GUI or other ISM modules as needed.

Widget Server messages are normally of the form "**send** *object message*". As named *Client* objects, messages sent to an ISM require no special formatting and are simply directed to the channel that was negotiated when the ISM first connected. The syntax and content of the message sent to a component is arbitrary and determined by the needs and abilities of the ISM itself. ISM messages sent to XIImtool follow the same rule, however the message itself *does* require a special syntax. Namely, messages must be preceded with keyword



Figure 2. Integrated control panel showing the coordinate readout panel.

**source** if the message contains GUI Tcl code to be executed (e.g., callbacks needed by the ISM, new GUI objects, etc.), **alert** if the message is some text to be displayed on the XImtool alert panel, or **deliver** if the message text should be delivered to the ISM-specific callback that was previously uploaded in a *source* message.

More complex components (such as the WCSPIX module described here) may also require support code in the compiled portion of XImtool. To deal with this, a list of ISM callbacks is searched when a component first connects and any registered procedures are executed. In the case of the WCSPIX module, such callbacks are executed when it first connects (to initialize the ISM with a list of the currently displayed image regions) and when it disconnects (to clean up). A more general command callback for each ISM can also be registered so that new code specific to the ISM is isolated from the core system, minimizing the impact of an upgrade or a replacement of the component at a later time. Although not implemented in this release, such compiled code could be loaded dynamically as a shared object, further isolating it from the core system.

Formatting of the messages and the details of how they are used by the transport layer are hidden in an interface on each side of the connection. A standard API to connect the component and handle messages is available to support both IRAF and other compiled tasks using a simple IRAF library or an extension to the Client Display Library (Fitzpatrick, 1998).

### 3. The Image/WCS ISM

The Image/WCS ISM, **WCSPPIX**, is written as a general purpose interface to all types of images supported by IRAF and provides a set of generic methods to query an image object for pixel or header data, coordinate systems available for the image (logical, world, or derivatives such as sky coordinate transformations) and translations of coordinates. It is also responsible for caching all images currently displayed in the server. The module is written to allow image objects to be subclassed using new functions to provide the actual image interaction. This means support for new formats such as Mosaics, spectral data or even tables can be added without requiring a new ISM.

The client display task defines a WCS for each region of the display frame buffer, and this information is passed to WCSPPIX which initializes the object by opening the image and placing it in the cache. Cursor events in XImtool are translated to coordinates within each image region on the display, and these are then passed to the task to be translated and formatted to any of the WCS values requested by the user in the GUI. The XImtool GUI is responsible only for displaying the strings on the panel (Figure 2) and plays no direct part in the coordinate translation.

### 4. Other New Features

Along with the new component architecture providing real-time pixel and WCS readout, several other new features have also been added:

- The various panels for Control/Print/Load/Save have now been integrated into a single window selectable with a Tab widget.
- Keystroke commands will cause the cursor to centroid on an object and/or adjust the size of the centering box, providing greater accuracy for cursor commands.
- Frames can be registered with an offset and will maintain the registration of all frames while panning or zooming the image.
- All 16 frames permitted by the IIS protocol may now be used (more are possible). Blinking and registration are supported for all frames.
- A new panel was added to allow more control over the display of tiled images. All 16 frames may be displayed in any user-defined configuration.
- A *compass* indicator for the display orientation on the panner window is now available.
- Pixel table readout of an area around the cursor has also been added.

### References

- Fitzpatrick, M. 1998, in ASP Conf. Ser., Vol. 145, Astronomical Data Analysis Software and Systems VII, ed. R. Albrecht, R. N. Hook, & H. A. Bushouse (San Francisco: ASP), 200
- Tody, D. 1995, in ASP Conf. Ser., Vol. 77, Astronomical Data Analysis Software and Systems IV, ed. R. A. Shaw, H. E. Payne, & J. J. E. Hayes (San Francisco: ASP), 89