Creating the DXLab Suite - a Free, Interoperating Array of Applications for the DXer

Dave Bernstein, AA6YQ

2002-09-12

If you're interested in DXing, and believe that better software would improve your performance and enjoyment, then I'd like to introduce you to the *DXLab Suite* of free, interoperating applications and the collaborative process that is driving it forward. While I've been designing digital hardware and software since the late 1960's, I didn't get my ham license until 1990. At that point, two things happened in very quick succession: I caught the DXing bug, and I began writing software in support of my radio activities. What started as a simple PacketCluster monitor grew over the years to include Icom transceiver control, logging, award tracking, QSL card printing, beam heading computation and rotator control, propagation forecasting, and QSL route searching.

This functional conglomeration, which I called *DXLab*, was a very effective DXing accomplice -- I could analyze the operating habits of DX stations I was chasing, correlate those habits to both primary and secondary band openings from my QTH, and QSY to a spotted DX station's frequency in one mouse-click, getting the needed QSO crucial seconds before the spot-chasing hoards arrived. There were two significant problems, however. While placing all of the functionality in one program made it easy to "integrate" these functions -- e.g. click on a DX spot plotted on the world map and QSY the transceiver to the appropriate frequency and mode -- the time required to build and test the program after adding a new feature was becoming problematic. Furthermore, the program was far too complicated for anyone else to install, much less understand or use. Visitors to my shack would marvel at DXLab’s capabilities, but evolving this monolith into something usable by anyone but its author seemed impossible.

Early in 1999, Tony, N2SS posted a message on an Icom reflector asking for help in getting his 781 transceiver to interoperate with his PW-1 amplifier. In Icom's design, the PW-1 determines the transmitter's frequency by monitoring commands sent on the CI-V bus; this bus was originally designed to permit multiple radios to transceive in a master-slave relationship, and then extended to support Personal Computer (PC) control. Tony's 781 and PW-1 just weren't communicating, and it wasn't obvious why. It occurred to me that I could quickly assemble a CI-V bus monitor by using "parts" from DXLab's transceiver control function. The notion of organizing software into components that can easily be re-used from one application to another is one of modern software engineering holy grails; this was a trivial example. The result was a standalone application I called *CI-V Explorer*. It enabled Tony to find the blown fuse in his 781 that prevented the PW-1 from responding to his transceiver's CI-V messages, and it taught me the basics of packaging an application for installation by amateurs.

I made CI-V Explorer available for downloading from my personal web site. Not many hams want to watch CI-V messages flow between their PC and transceiver, even if they have the choice of doing so in decimal or hexadecimal. None-the-less, the primitive search engines of that era made CI-V Explorer visible to a few diehards; their appreciation and encouragement was nearly as addictive as DXing. Not too long thereafter, the continuous "what's the QSL route for X?" messages on the various DXing reflectors and newsgroups led me to extract more code from DXLab and create a specialized web browser called *Pathfinder*. There were plenty of web sites with QSL information -- Buckmaster, QRZ, RW1QM, OZ1C, K4UTE, etc. While it was easy to add these to your browser's Favorites, navigating to each and then re-entering the callsign was painful and time-consuming. There were also an increasing number of online callbooks for various DXCC entities; in all, I located more than 100 different web-accessible sources of QSL information. Pathfinder lets you enter a callsign just once; it determines the DXCC entity for the callsign, and gives you a button to click that searches the appropriate online callbook for that entity. Twelve additional buttons can be associated with your favorite online QSL sources, allowing rapid searching of the web, and more recently the Radio Amateur's Callbook CDROM. Around this time, I was fortunate to meet Fab, IK4VYX, the author of DXTelnet. Fab helped refine my understanding of packaging software for web distribution, resulting in the beta release of Pathfinder 1.0 in August of 1999 to a group of 20 or so beta testers. I was also fortunate to discover the http://www.qsl.net site hosted by Al, K3TKJ, who graciously provided a web site from which Pathfinder could be distributed; without his support, neither Pathfinder nor any subsequent DXLab application would have ever seen the light of day.

Chasing down QSL information is a lot more interesting than watching CI-V messages go by, and lots of hams responded to my post seeking beta testers. No offense to the team, but I was not selective in assembling this group -- anyone who volunteered and had a reasonable PC running Windows 9X or NT was accepted. While this might seem foolhardy, it was extraordinarily effective -- the broad range of skills, backgrounds, and expectations forced me to optimize Pathfinder for ease-of-installation and ease-of-use. The term "intuitive user interface" graces the data sheets of most modern software applications, but I evolved a specific definition for this phrase: most hams should be able to use Pathfinder by simply running it, with little or no reference to any online documentation. While there are many aspects of this approach, two stand out: there are **no menus**, but there are **meaningful tooltips** for every control. Unless you're building a word processor, Microsoft's standard ***File Edit View Insert Format Tools Window Help*** menu structure is a poor match for most amateur radio applications. One must either shoehorn ham commands into this structure, or invent a new structure; either way, users are left to grope through the maze, hoping to remember how to change the RTTY baud rate to 50 before the pileup arrives. To avoid this, all capabilities are directly accessible via command buttons, check boxes, sliders, grids, and other visual controls Tooltips are Microsoft's name for a conveniently accessible documentation mechanism: let the mouse cursor hover momentarily over a control, and text describing the control's function pops up. Every control of every DXLab application should have a meaningful tooltip; yes, tooltips can be disabled en masse once you become familiar with an application. If the required controls can't fit in a window of reasonable size or would result in too complex a panel, tabbed dialog boxes let the user choose from a small, obvious set of activities, e.g. DXKeeper's **Log QSOs, QSL, Check Progress, my QTHs, Import**, and **Export** tabs.

After a rocky start, I established the infrastructure required to track incoming defect reports and enhancements; I created web pages for each of these, so that the user community could see what was going on and comment or critique as they deemed appropriate. These mechanisms are still in use -- you can visit Pathfinder's version history, defect log, and enhancement log online at http://www.dxlabsuite.com/pathfinder. Most interaction was accomplished with email messages, typically copying everyone who was working with the application.

Pathfinder proved that I could extract functionality from DXLab and make it broadly available, but the next advance came from an entirely different direction. Peter, G3PLX had developed the PSK31 protocol; he'd also developed PSK31SBW, a Windows application that together with a soundcard performed PSK31 modulation and demodulation. I was excited by the potential of this new mode, and began thinking about how to add support for it to DXLab, with appropriate interaction with the transceiver control and logging functions. Peter and I discussed the creation of a PSK engine with programmatic interfaces that could be used by many different applications; such interfaces would allow me to add PSK support to DXLab without re-inventing the DSP wheels required for modulation, demodulation, and soundcard interfacing. When I discovered that Moe, AE4JY had constructed such an engine -- PSKCORE -- I was off to the races. At first, my rationale for building a standalone PSK implementation using PSKCORE was simple -- the process of building it inside DXLab would be too slow and cumbersome; better to get it working separately first, and then integrate it. However, my experience with Pathfinder led me to believe that an easy-to-install, easy-to-use PSK application would be of interest to lots of hams. At the time, PSKCORE's ability to simultaneously decode multiple PSK QSOs was unexploited; I believed that PSK DXers would find this capability particularly useful.

The fly in the ointment was integration. If I were to construct a standalone PSK application, how would it perform transceiver control? How would it perform logging and award tracking? It might start out as a standalone application, but eventually, I'd have to add most of the functionality already present in DXLab. The only thing worse than one monster application is two of them! Fortunately, there was another approach: instead of one big program that does everything but becomes increasingly un-installable, un-usable and un-maintainable, design a system of individual, specialized applications that detect each other's presence and interoperate automatically.

Microsoft Windows provides mechanisms that allow applications to communicate by sending messages to each other. There are actually several such mechanisms; I chose one called Dynamic Data Exchange (DDE). Using DDE, a PSK application can obtain the current transceiver frequency from the rig control application, or direct it to QSY the transceiver so that the current signal is centered in the receiver passband. The PSK application can send QSO information to the logging program, and send a received callsign to Pathfinder to find a QSL route. A constellation of applications can do everything that one big monolithic application does, but with several fundamental advantages:

- A user can start with whatever application suits his or her fancy, master it, and then add additional applications in whatever order seems appropriate
- development of individual applications can be more nimble and responsive to user feedback
- unique hardware devices -- e.g. the transceiver, the antenna rotator, the soundcard -- are accessible to multiple "client" applications simultaneously, including applications constructed by other developers

This revelation led me to a new mission: the construction of a system of interoperating applications that automate DXing activities as did the monolithic DXLab, with no reduction in integration among functions; this system is the *DXLab Suite*. The PSK application became *WinWarbler*, and was later extended with Mako, JE3HHT's *MMTTY* engine to support RTTY with a common user interface for both modes; with an outboard RTTY modem, WinWarbler can decode two RTTY signals simultaneously. *CI-V Explorer* became *Commander*, which provides a common user interface and DDE "server" for Kenwood, TenTec, and Yaesu as well as Icom radios. *DXView* was assembled to perform callsign lookups into a DXCC database, control the antenna rotator, and plot the resultant information on a world map along with beam headings and the real-time solar terminator position. As the original DXLab's logging functions were built on an earlier generation of database technology, I chose to build *DXKeeper* from scratch around Microsoft's Jet database engine, the same engine that underlies Microsoft Access; DXKeeper exploits this engine to provide powerful, yet easy-to-use **filtering** of one's log to show some specific set of QSOs, e.g. "all QSOs with VK9NS" or "any QSOs that started within an hour of 23-Jul-2001 @ 1410Z". DXKeeper retained its predecessor's ability to track award progress, identify the QSOs for which QSLs were required, and generate the necessary QSL cards or labels; it supports multiple logs, multiple operator callsigns, and operation from multiple QTHs. *SpotCollector* similarly exploits the Jet engine to create and maintain a local database containing merged DX spots and solar/geomagnetic parameters from a local PacketCluster, from up to four TelnetClusters, and from the DX Summit internet cluster; real-time analysis and filtering of this information provides the modern DXer with critical information in a form that facilitates rapid action - e.g. "show me all spots within 5 kHz of the transceiver's frequency". *PropView's* forecasting functionality was extracted from the original DXLab, but its ability to assess actual propagation by monitoring the NCDXF/IARU High Frequency Beacon Network -- directing Commander and DXView to QSY the transceiver and rotate the antenna to track a specified set of beacons -- is a recent addition that illustrates the power of interoperation among an array of applications like the DXLab Suite. Detailed descriptions and comprehensive online help for each of the above applications are available via www.dxlabsuite.com .

As each of these constituent applications came to life, Al, K3TKJ provided a web site to support its distribution, management, and online documentation. I recruited early adopters from the reflectors, accepting feedback and critique from anyone willing to offer it. To facilitate rapid response to this feedback, I employed iterative development, a style of software engineering characterized by frequent **development releases**. Every few months, a stable development release is used to create a **full release** containing all software and documentation components. To facilitate rapid evaluation, development releases are lightweight, containing only those components that have changed since the most recent full release. Users installing a DXLab application on a PC for the first time must first install the most recent full release. After installing a full release, one can obtain all subsequent defect repairs and feature additions by upgrading to the most recent development release. Instructions for downloading, installing and upgrading with a matrix of available full releases and development releases are always available via http://www.dxlabsuite.com/download.htm.

Since these applications interoperated, there was considerable overlap between their initial user communities. In a step that radically improved communication, Rich, W3ZJ created the DXLab email reflector at http://groups.yahoo.com/group/dxlab, providing a common forum that anyone can join. This reflector has become the primary means by which DXLab development moves forward -- suggestions are refined, alternatives are considered, and releases are critiqued; the result is a powerful flow of ideas whose implementation benefits all participants. Acknowledging the contributions made by reflector members to date would double the length of this article - and yet we've barely scratched the surface of what can be done with these technologies.