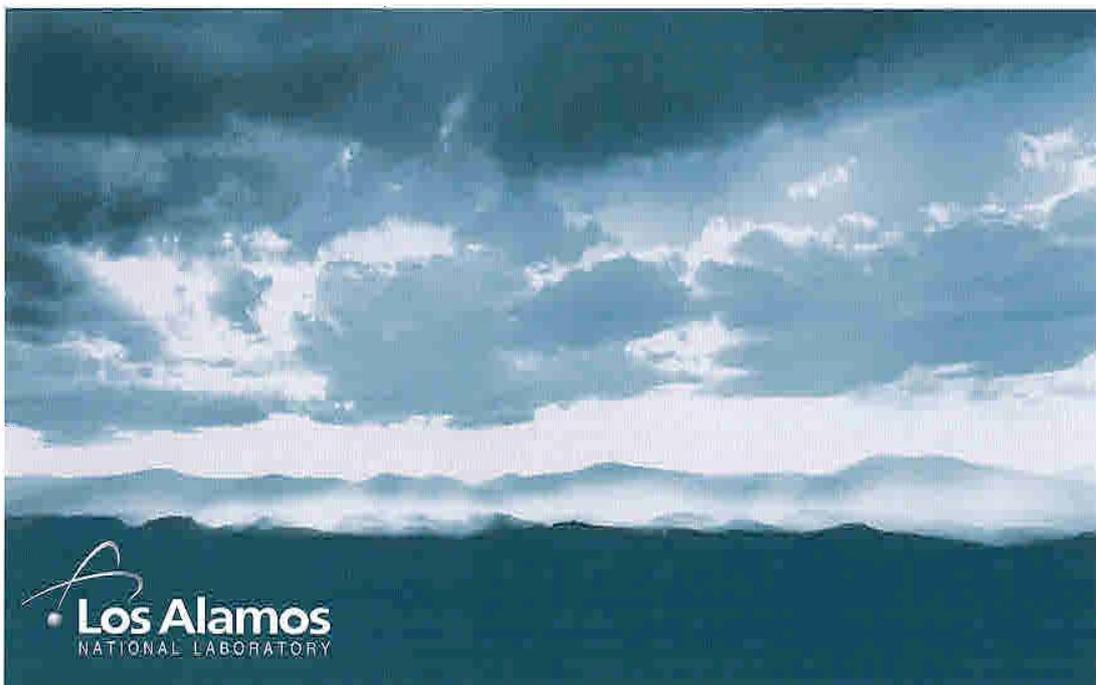


LA-UR-04-4360

An International Standardized Utility to Collect Data, MIC

David Pelowitz
Los Alamos National Laboratory
Los Alamos, NM 87545

*Presented at the
Institute of Nuclear Material Management
45th Annual Meeting
Orlando, Florida
July 18-22, 2004*



Chris J. Lindberg

An International Standardized Utility to Collect Data, MIC

David Pelowitz
Los Alamos National Laboratory
Los Alamos, NM 87545

Abstract:

Multi-Instrument Collect, MIC, is rapidly becoming the international standard for safeguard data collection. First implemented in 1998, it was adopted by the International Atomic Energy Agency the following year and fielded in numerous locations around the world. This application is capable of simultaneous collection from up to 100 independent instruments. It supports a variety of instrument types and a variety of communications media. Although created and currently being maintained by Los Alamos National Laboratory's Safeguards Science and Technology Group, N1, effort is underway (completion planned mid-year 2004) to open up the architecture promoting outside or third party development. This effort is nearly unique in the safeguards arena. It provides a standard user interface philosophy for a multiplicity of hardware vendors. User community advantages are: enhanced use-ability, decreased training, faster field implementation, decreased user data collection time, and ultimately lower implementation and maintenance costs. There are also significant advantages to the instrument developer. For example a variety of pre-existing communications objects may be used, pre-existing power management capabilities exist, and extensive data management capabilities. All of these may be leveraged into a vendors' development effort--significantly decreasing the amount of development effort and consequently decreasing cost. The presentation will include an overview of the existing MIC and MIC utilities from a developers and a users' point of view, the user interface philosophy, and will discuss the open architecture allowing third party development.

Introduction:

Development of a new computer program began in 1997 when it was realized that having unique data collection programs supporting each type of commonly used Non-Destructive Assay, NDA, instrument presented a training and field support nightmare. At the inception of the project specific goals were established as developmental guidelines. Each of these goals would address identified field problems. Although mainly targeting the requirements of the International Atomic Energy Agency, IAEA, these goals would also address other venues. Multiple quantity of simultaneous instrument support, multiple type of simultaneous instrument support, high reliability, unattended data accumulation, rapid data retrieval, and support a quick and simple to understand user interface were some of the design objectives. Out of the objectives and derived requirements grew the program called Multi-Instrument Collect, MIC. MIC was designed to meet all of the identified goals—there was no compromise in any of the goals during its development. Currently, MIC supports the Jomar Shift Register (JSR), GRAND-III (Gamma Ray and Neutron Detector), MiniGRAND (Miniature GRAND), Advanced Multiplicity Shift Register (AMSR), Miniature 1k Analog to Digital Converter (MiniADC), and Ortec DSPEC+ Non-Destructive Assay (NDA) instruments. It also can control selected models of American Power Conversion's and Belkin's Un-Interruptible Power sources. It indirectly supports data collection from Global Positioning Systems (GPS), VACOS seals, and event information through the Intelligent Local Node (ILON) Event Instrument. MIC supports this wide variety of instruments using a simple to learn and instrument to instrument consistent user interface.

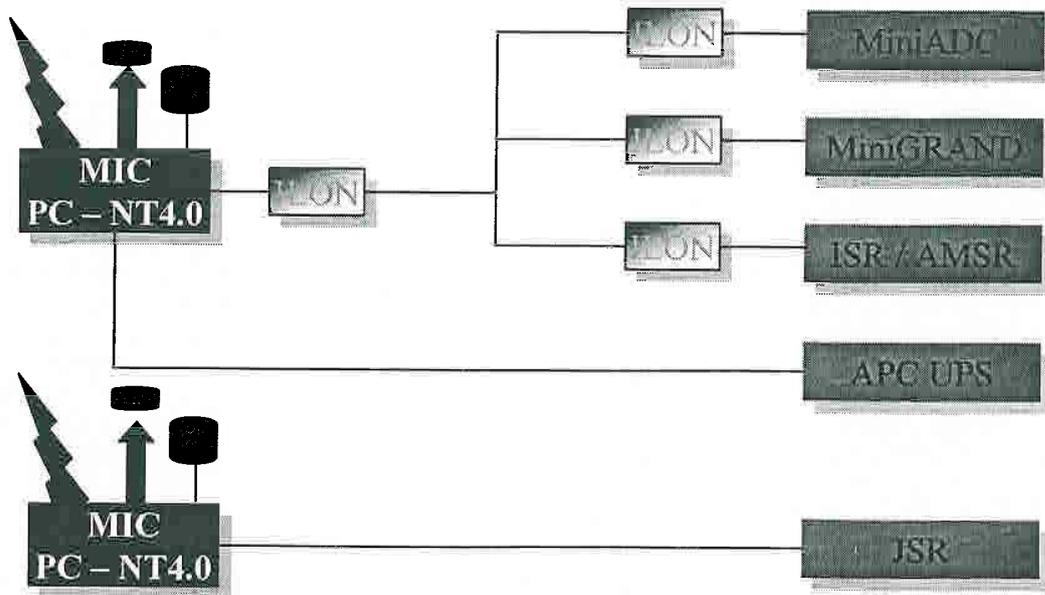


Figure 1. Two Typical MIC Configurations.

Over the seven years after MIC was developed new instrument support was added, support for existing instruments was improved and enhanced, and more adjunct support programs were added to the suite. During this time MIC became the, if not the major, standard data collection program approved for use by the IAEA. But of more significance was the emerging new requirement to support instruments developed by vendors other than LANL and for vendors to be able to add functionality to MIC to support their instruments. One of the early development derived requirements of MIC was extensibility. Toward that end MIC was written in C++ to take advantage of the encapsulation inherent in that language. As new instruments were developed at LANL it was discovered that adding MIC support would take between six and twelve weeks depending on the complexity of the instrument interface. Although the time to add new instrument support was acceptably short and all that advantages of functional encapsulation were employed, MIC was still a monolithic executable. What this meant was that only its developers could extend it.

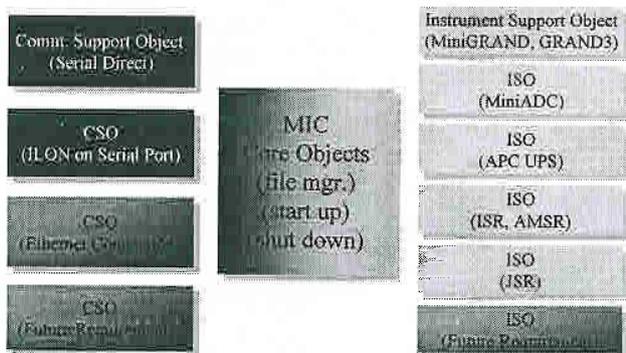


Figure 2. MIC Architecture.

Early in 2004 a project was started to modify the MIC application to meet this new requirement. The objective of this project was to leverage off the existing architecture (e.g. retain all existing functionality and not rewrite the application) in MIC and expose the existing interface between encapsulated objects. What this means is that a vendor, utilizing the exposed interface, could independently develop instrument specific support and connect that support into MIC.

A variety of techniques existed in the Windows environment to accomplish the connectivity. The Microsoft "Common Object Module", COM, was chosen. An interface control document was written. That document delineates the types of interfaces and the specific functions a vendor must use to tie support of their instrument into MIC.

Specifics:

Multi-Instrument Collect may be logically divided into three distinct groups: the core, the communications support objects, and the instrument support objects. The core portion of MIC can be considered the glue that holds each of the other components together. It instantiates (creates instances of) communications support objects and instrument support objects during startup and subsequently establishes the connection between them. The Communications Support Objects, CSOs, each provide communications services for MIC. See Figure 2.

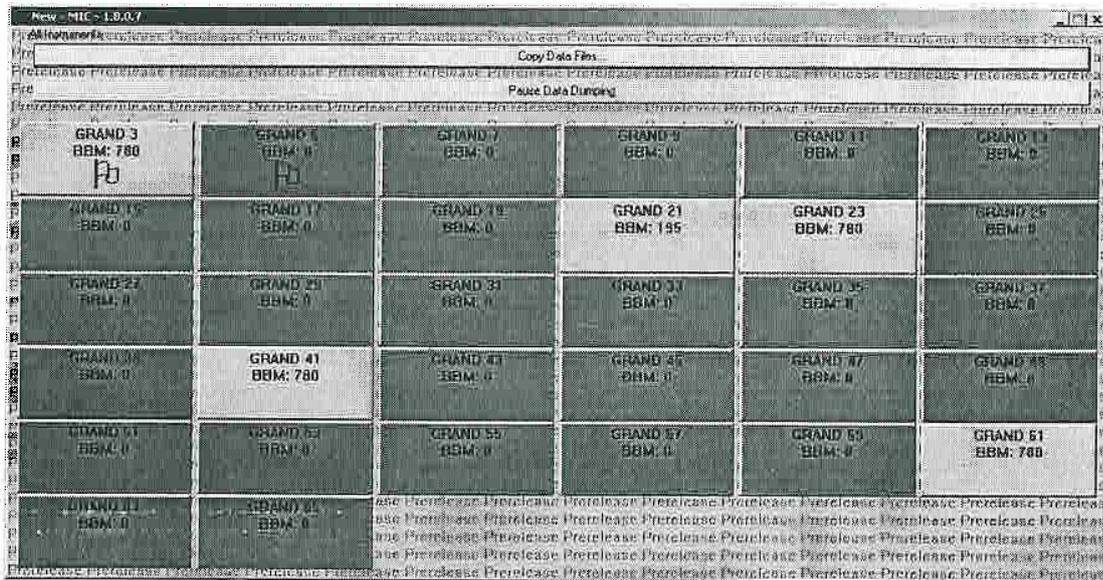


Figure 3. Typical MIC Display Showing Support for 32 MiniGrands Simultaneously.

Currently, there are three types of communications support objects provided with MIC: simple serial, ILON

serial, and IPX. Each of these CSOs adheres to the Communications Support Object definitions in the interface control document. That is to say they all have exactly the same externally accessible functions and from an external point of view behave in exactly the same manner. For example, the simple serial CSO provides communications through a serial port. An ISO connects to the CSO and sends commands out the serial port and receives responses back. The ILON CSO also provides communications to and from instruments but in its case via an ILON network. In either case an ISO does not know nor does it need to know whether it is talking to the instrument via an ILON network CSO or through a serial connection directly as is the case of the Serial CSO. Obviously, not all ISOs may work with all CSOs. An ISO is designed to support a specific instrument type. That type of instrument typically supports a particular type of communications, such as simple serial connection or UDP Ethernet. By disassociating the communications support object from the instrument support object and subsequently making the logical connection at run time MIC can be configured in any many instruments to one communication or one instrument to one communication configuration as needed. When the modified MIC is released each of the CSOs and ISOs will be in

modules external to MIC. They may be upgraded individually without the need to retest or recertify any of the rest of MIC, a considerable operational cost savings. If a vendor needs a specialized communications object it will be a simple task to create one. For example, to create an Ethernet communications support package could be created which could support not only that vendor's Ethernet based instrument but also any other instrument which has a similar Ethernet interface.

Project Status:

The project is rapidly nearing completion. Currently all three of the Communication Support Objects have been converted to external COM objects. Nearly all of the Instrument Support Objects have been converted with only two remaining. The core of the application is now supporting run time discovery of ISOs and CSOs. Along with the remaining ISOs the new MIC will need to be recertified by running the formal acceptance test. As the final part of the project an instrument developer/vendor will develop a non-LANL based Instrument Support Object and possibly an associated Communications Support Object.