Presentation of Relay Fault, Settings and Waveform data as Human and Machine-Readable Web Pages

KEYWORDS: HTML, XML, XHTML, Web Server/Client, TCP/IP, Oscillographic Data, Relay Primary Settings, Data Neutralization, Self-Descriptive Data Structures, Comma-Delimited ASCII

AUTHOR: William T. Shaw, PhD – V.P. Qualitrol, Systems Division

SUBMITTED: For consideration by the 2003 Fault and Disturbance Analysis Conference

ABSTRACT:

Almost all modern, microprocessor-based protective relays contain a vast amount of information including real-time measurements and status, settings, statistical calculations, "power" calculations and event based data such as fault summary records, sequence of event data and oscillographic waveform captures. All relay manufacturers provide some form of proprietary software package for interrogating their relays (often several versions) and presenting this data to a user. Many relays offer partial or full access to this data via "standard" interfaces and protocols such as Modbus, DNP3.0 and IP (Ethernet) versions of the same. Unfortunately, this means that anyone needing to see this data needs a copy of the relevant (and often quite different) software packages from each vendor, and access to a dial-in phone circuit to the respective substations where the relays reside. Data uploaded to the user's PC may be lost from the relay and will often be stored in the PC in a proprietary format. This makes sharing of the data nearly impossible unless everyone also has copies of the vendor's software tools. Although most relays provide similar types of event information, the differences in data format, representation and quantity make it impossible to "share" data across vendor platforms without converting the data into a "generic" representation such as COMTRADE. But even that approach doesn't address all of the data available from the relays. For most of us, the one universal mechanism we have on our desktop, for seeking and viewing information, is the web browser. Therefore it is logical that making vital engineering data "universally" available (given necessary levels of access protection) can best be done by offering that data as web pages. In a similar vein, delivery of this data to 3rd-party applications and user-developed applications would make this data more useful. Unfortunately no good "standards" exist

.

for providing vendor-neutral data representations of such complex data. But, a new type of "web document" called an XML document, provides a simple way to deliver even complex data to 3rd-party applications. This paper discusses experiences with, and the challenges related to, extracting data from various types of relays via different interface mechanisms and turning this data into both conventional, human-readable (HTML) web pages and into XML web pages for delivery to 3rd-party applications.

.

Presentation of Relay Fault, Settings and Waveform data as Human and Machine-Readable Web Pages

William T. Shaw, PhD - V.P. Qualitrol/Systems Division
Hunt Valley, Maryland U.S.A.

INTRODUCTION - One of the difficulties faced by a typical protection engineer, and by the system operations personnel, is collecting, analyzing and managing the data generated and captured around the power system whenever a fault occurs. When you go into a typical substation these days you often find a number of IEDs that are located there totally, or partially, in order to "capture" event-related data. These IEDs may include Remote Terminal Units (RTU), sequence of event (SOE) recorders, digital fault recorders (DFR), power quality monitors and, most especially, protective relays. Fault events, depending upon their cause and nature, may be "seen" in multiple substations and by multiple IEDs. An RTU might record status input changes as part of its SOE monitoring. A DFR unit might note the event and "snapshot" oscillographic data, both pre and post event, as well as generating a summary of the key fault attributes. The relay(s) involved in clearing the fault will also contain a large amount of data: fault summary information, oscillographic waveforms, SOE data, etc. In addition, the specific, active "settings" of these relays represent additional data that may be relevant to the event, or at least to understanding the protective response to the fault. All told, a typical fault condition can create a lot of data in a lot of different IEDs, frequently in multiple substations. All of this data (not yet information) is relevant to understanding the actual fault (its cause, duration, etc.) and how the power system and protection scheme responded to the fault. The question is: how can all such data be collected, stored, managed and made available to the appropriate people and even made available for additional analysis and usage?

TOWER OF BABEL – Unfortunately, through the years, most (all?) vendors of the various types of IEDs have taken it upon themselves to develop proprietary approaches to the collection, representation and storage of data. (We are equally guilty in this regard.) In addition, there is also the problem of communicating with the various IEDs for the purpose of extracting this data. Often (although less often) vendors have also taken a proprietary approach in this aspect of their products. This might be a result of being the "first" vendor of such a product, or just because they thought they could do it "better" than the competition. For whatever the reason, the net result is that there has been little or no compatibility between the various IEDs, even of the same "class" (such as DFRs or relays). Today, more and more IEDs provide for some level of protocol compatibility, typically by offering a serial DNP3.0 or Modbus communications capability. But, this does not address the differences in how data is represented or what data is provided. In

addition, although some IEDs have the ability to package data in a "Standard" format (such as using COMTRADE format for oscillographic data) most older IEDs do not.

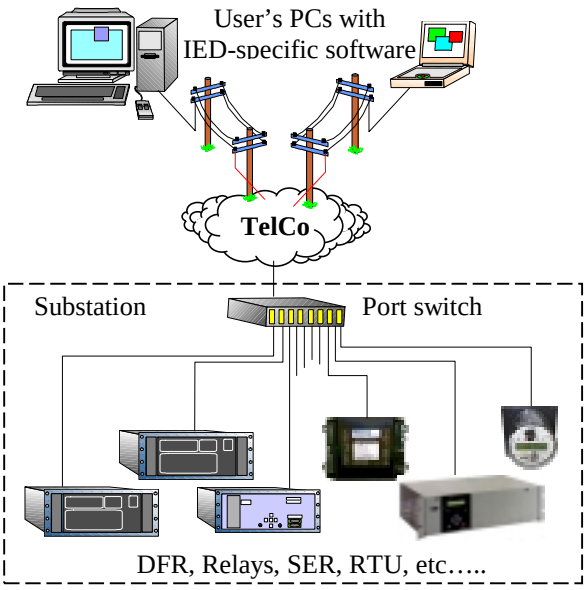

Figure 1.0 – Typical IED data access method

DATA COLLECTION – In many (most?) utilities, the gathering of the data that results from a fault, or other system event, is done in a less-than-optimal manner: engineering personnel generally have to "dial-in" (via the public telephone system) to effected substations and use vendor/IED-specific software packages to interrogate and extract the data. This may involve multiple dial-in sessions (one per IED) and the use of several different software packages (one per vendor/IED.) Either an engineer has to be familiar with multiple such packages, or multiple engineers may have to cooperate in the data collection process. The resulting data is generally captured and stored on the PC that was used to run the IED vendor's software, and all too often in a proprietary form and format. (Refer to Figure 1.0.)

Although the various IED vendor's software packages provide individual capabilities for display (and maybe even analysis) of the captured data from their respective IED, they don't normally offer any means for integrating and displaying the data from others. Nor do they typically provide a means for "exporting" the captured data in a form that would allow its use in other applications. Because of the inability to consolidate data from multiple sources, the process of system-wide data reconciliation (when done at all) is often done by hand with engineering personnel having to manually correlate data extracted from these multiple sources.

DATA EXTRACTION – Although the data collected in many of these IEDs is of a similar nature (e.g. time-tagged sequence-of-event data, time-sampled waveform data, etc.) The mechanisms provided for extracting this data can be very different. This is usually a result of the combination of the communications protocol used and the way data is "mapped" into these protocols. Obviously the software tools provided by the respective IED vendors are designed to be compatible with these mechanisms. Unfortunately this doesn't help us to consolidate data from multiple, incompatible IEDs. In order to extract the available data we have to be able to communicate with the IEDs. Although more and more of the latest IEDs support LAN-style connectivity (Ethernet)

.

traditionally the communications interfaces with IEDs have been serial (RS-232C or RS-485), low speed and based on one of the three following approaches:

1. "Dumb-terminal" ASCII
2. Proprietary communications protocol
3. "Standard" communications protocol

For some IED manufacturers, the approach has been to provide a "dumb terminal" ASCII mode of communications whereby data is output in the form of an ASCII "report". The IED actually "prints" a report, complete with carriage-return and line-feed characters and headings, in response to a simple ASCII "command". This is an inefficient means for extracting data and one potentially subject to communications errors (since no communications error detection and correction scheme is provided) but it does allow data to be human-readable. An application that wants to extract and use the data from such an IED will have to capture the stream of ASCII and then "parse" it looking for actual data. One of the dangers in using such an interface is that if the IED vendor changes, even slightly, the format of the "print out", the application that is doing the "parsing" of that output may no longer function properly. Computer programmers often refer to such a "report" parsing/dissecting application as a "screen scraper".

Prior to the emergence of "standard" communications protocols, such as DNP3.0 and Modbus, IED manufacturers provided communication in the form of a custom-designed, proprietary protocols. Unlike an ASCII interface, these protocols could provide much greater data efficiency (more data per bytes exchanged) and incorporated error detection and correction mechanisms. But, due to their proprietary nature, IED-specific "drivers" must be developed in order to communicate with these IEDs, presuming that the necessary protocol information is publicly available.

Having IEDs that offer their data via a "standard" protocol interface is an improvement, in that the basic protocol specifications are known. Unfortunately, these specifications do not totally address the manner in which a vendor packs his data into the protocol "wrapper".

DATA REPRESENTATION – Numeric quantities, along with other types of data, can be represented in a wide range of equally valid forms: integer, fixed-point decimal, floating point, scientific notation, etc. Data can be optionally accompanied by additional information that "qualifies" the data: time and/or date tags, quality flags, validity flags, etc. Data can be "encoded" and represented in an alternate manner: value of '2' means "Lockout", value of '5' means "Tripped", etc. The representation of date and time values themselves can be made in a variety of ways. Unless a protocol specifically defines allowable data representations, the IED vendor is free to choose their preferred ways of representing such data. The popular Modbus protocol essentially defines no data representations beyond unsigned 16-bit integers. On the other hand the DNP3.0 protocol

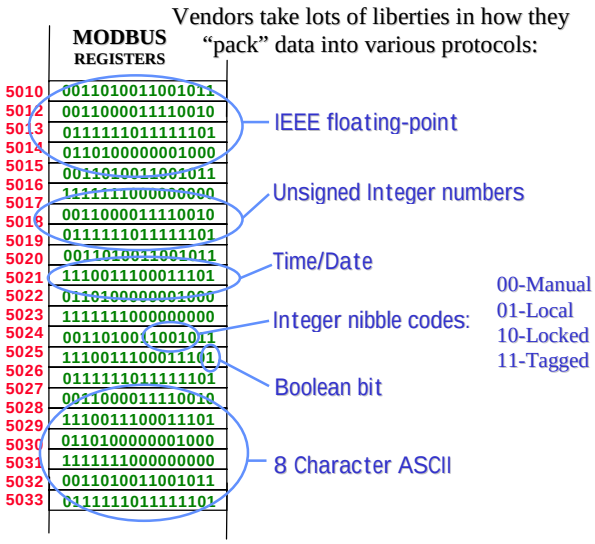

Figure 2.0 – Example of data "packing" in Modbus

defines multiple (too many?) ways of representing the same kind of data. In both cases the IED vendor is free to be creative within the confines of the protocol specification. Often this means that a "generic" Modbus or DNP3.0 driver cannot be used to extract IED data. What is required is an IED-specific version of these protocols that includes knowledge of how the data is "packed" and represented and how to manipulate the interface. (Refer to Figure 2.0.) A good example is the way some relay vendors use Modbus to deliver oscillographic data. There are not enough "registers" in the basic Modbus specification to hold all of the waveform sample data and so typically a "bank" of registers are used to "page" through the data, by writing the desired page number into another specific register. The driver needs to iterate through all of the available "pages" in order to extract the waveform data. Nothing in the Modbus protocol defines this usage or mechanism. It is specific to the vendor and IED, but does not violate the base dictates of the Modbus protocol specification. Similar schemes have been used with the DNP3.0 protocol as well because although DNP supports more complex data types/structures, it still has a fixed set that must be manipulated to transport many of the very complex and extensive data classes found in modern IEDs.

DATA STORAGE – Presuming that you have braved the rigors of communicating with all of the various IEDs in the substations, and have developed drivers that can extract the available data, where do you then put this data and in what format do you store it? One alternative would be to directly create web pages (XHTML) from the extracted data and be done with it. Since every IED, even those of the same type, have vastly different data representations, this would mean having an IED-specific web creation task for each IED. In addition, this would not make the data available for other purposes. A second, and preferable alternative, is to translate the data into a "vendor-neutral" format and store it into a relational database. There is certainly a LOT of data available from various IEDs and it needs to be stored in a manner that facilitates locating and associating related data. Fortunately, today we have low cost and high performance PCs that can be equipped with loads of RAM and hard disk space. It is not unreasonable to have 50 Gigabytes of hard disk space on a PC and 512 Mbytes of RAM. With the current crop of 1+ Gigahertz Pentium-class processors there is no reason why commercial relational database packages can't be used for data storage purposes, even without the expense of a multi-processor

based “server”. The data extracted from substation IEDs generally falls into three (3) categories:

1. real-time data (constantly updating)
2. event-based data (event generated and eventually over-written)
3. settings/parameter data (occasionally changed)

Relational databases allow users to define “records” (data structures) that will be used to hold data, as well as “key” information that uniquely identifies each set of such data. A record can contain lots and lots of data items (such as the oscillographic waveform data samples or the current primary/alternate settings of a relay) and combinations of different types of data (numbers, time/date, text strings, etc.) For real-time data storage, and storage of settings, a table is created to hold all such data and individual records are re-written (updated) with data as each data item is reported. For other types of data “blank” records of a given type are filled in with actual data and then these records appended to tables that hold records of this same type.

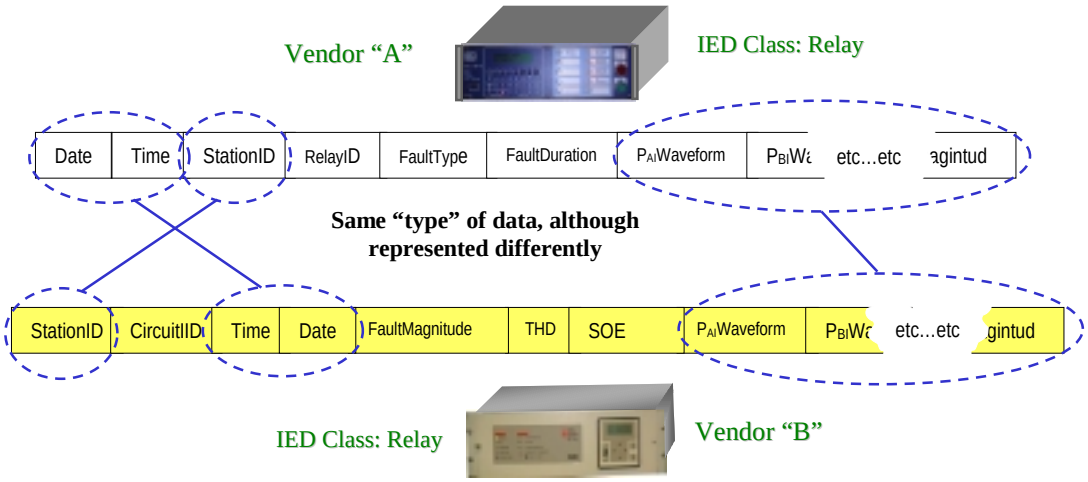

Figure 3.0 – Similar types of IEDs generate similar data, but often represented differently

Tables holding real-time data and relay settings data are normally of a fixed length whereas tables that hold event-based data “grow” as new event records are added over time. The placement of the individual records into these tables is based on the specified “key” information. For event data records the “key” information might be time/date of the event, feeder/circuit associated with the event, the IED associated with the data and even the name of the substation from which the data was extracted and an indication of the type of data held in the record. Modern relational databases can hold vast amounts of very complex data. In addition, the physical/internal representations of specific data types can be “promoted” into a common form: All numeric data can be stored as 32-bit floating point values, all strings can be “decoded” (if needed) and stored as 256 characters in

length, all times and dates can be converted to the "universal" millisecond representation format, etc. Thus, data of a given type will be the "same" (in the database tables), regardless of its native representation within the various IEDs.

To correlate data records from different IEDs it is necessary to establish a set of identical data fields such as time/date, station ID, circuit ID and anything else that will help to identify common and related data records. These data elements will allow us to locate related records. Of course the data from different IEDs may (will) be somewhat different in both content and format (Refer to Figure 3.0.) One of the ways in which the UCA2.0 standard and the GOMSFE data object definitions approach "standardization" between IEDs from different vendors is to define a subset of common data items, in a common representation form, that all such IEDs must provide (and to assign a standard "name" to such data items). Additional data is either ignored or must be dealt with as a special vendor-specific issue. With modern relational databases other approaches are possible: one method of dealing with data differences is to store all of the data from every type of IED into individual tables, specifically structured for the data generated by that IED (refer to Figure 4.0). If standard field names (such as "StationID" or "Date") are used for the same type of data from different IEDs, the relational database can manage these differences. If common keys are maintained, the database packages can "join" the same types of data from different tables using the key information. (E.g. If time/date keys are used for SOE events, the relational database can extract all records with matching keys from all of the various tables, and collect the specific data into a common table for manipulation. A second approach, for a given "class" of IED is to build a "super set" record definition that includes all of the common data, plus all of the IED-specific data, from all of the IEDs (refer to Figure 5.0.) Then the non-applicable fields are merely ignored when data from a given IED is written into a record. This still requires having separate tables for each IED "class", although not for each specific model of IED.

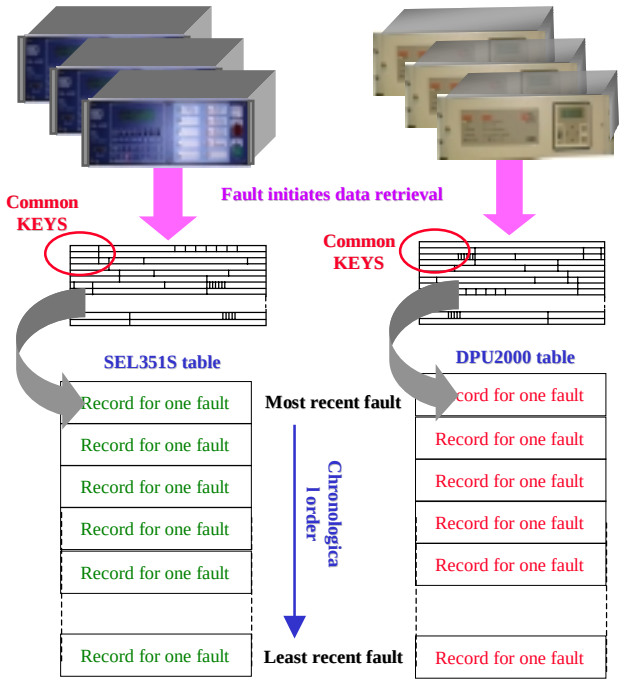

Figure 4.0 – Separate tables for each IED

DATA MANIPULATION – Once data has been neutralized (made non-device-specific) and stored into a relational database, there are many ways in which this data can be filtered, sorted, related and presented. All of the modern relational database packages provide graphical user interfaces of some form and offer interactive mechanisms for exploring the data in the tables. In addition, most also offer some form of data extraction and exporting mechanism so that data can be brought into other applications, such as spreadsheets and report generators, for additional processing and presentation. In addition, the automatic creation of web pages from this data is simplified since IED-specific issues are eliminated through the standard representation of specific data items.

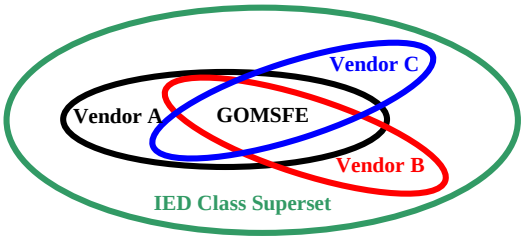

Figure 5.0 – IED data Subsets and Supersets

DATA CORRELLATION – Possibly one of the most powerful reasons for storing the various types of information, collected from the different IED sources, in relational database tables is the fact that this data can then be searched and correlated to find related data items. As was mentioned at the beginning of the paper, many types of fault/system events will cause multiple IEDs at various points around the power system, to capture and record data about the fault/event. Once stored into relational database tables it is a simple matter to use the "join" query capability of such databases (see the example given above) to isolate all data records that resulted from a given event. Data records of differing types (fault summary, S.O.E. recordings, waveform capture, etc…) may contain vastly different types of data, but the data is related based on being generated by a common event. These data records thus would all contain common "key" elements (such as date and time and substation, etc…) that link them to this common event and these keys would allow a user to extract all data related to a common event, regardless of the type/class of IED and the particulars of the data (refer to Figures 6.0 and 7.0.) Obviously examination and presentation of the data would be very data and IED-class specific.

Figure 6.0 –Cross-table "filter" selection by user-selected criteria

DATA PRESENTATION - Once we have captured and "normalized" data from a bunch of IEDs, and have the facility to search and sort this data, we are still left with the need to provide presentation mechanisms for this data. A lot of the data generated by IEDs is simple in nature: values, times, dates, strings, etc. But, a lot of it is in the form of time-ordered lists (SOE data) or time-ordered waveform samples (oscillographic data) that need to be presented graphically for best user comprehension. As previously mentioned, each IED vendor typically supplies some form of PC-based software package to display and manipulate the data. The problem for most utilities is that none of these packages are compatible and few can deal with data from other IEDs, unless the IED can present its data in a "neutral" form such as COMTRADE file format. In order to allow these vendor-specific packages to be used, the data extraction mechanism ought to provide for maintaining the data in a "native" format, as well as dissecting the data and placing it into

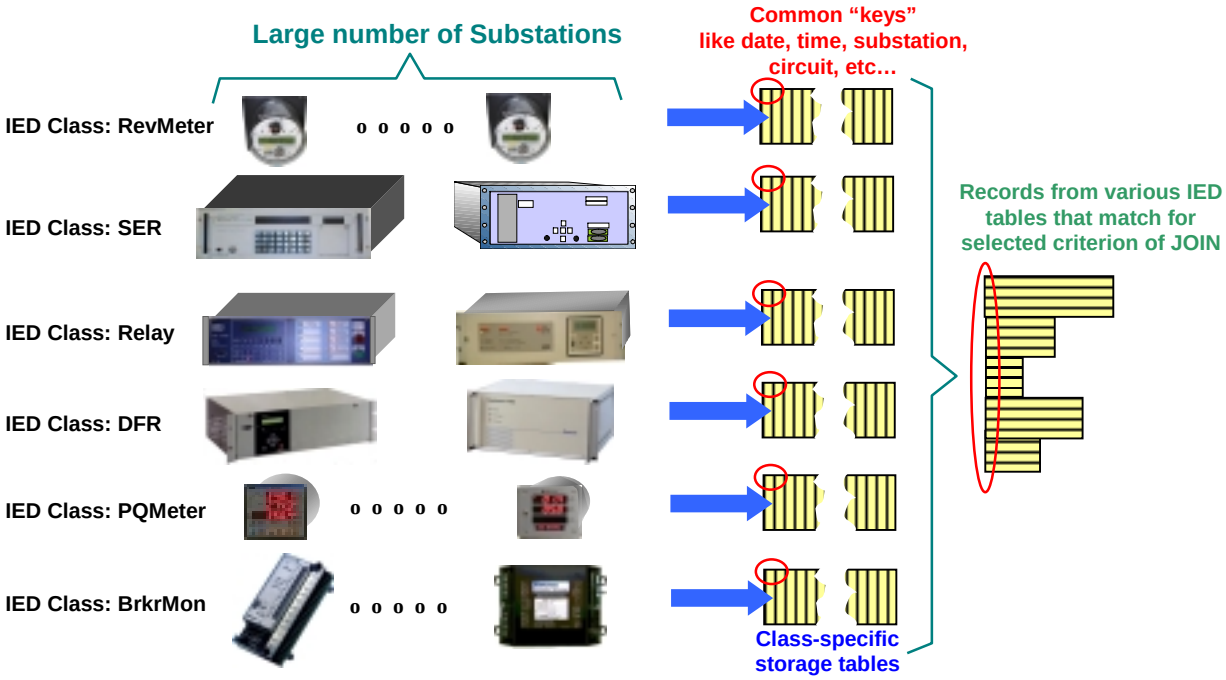

Figure 7.0 –Record correlation across multiple IED-class relational database tables

relational tables. Thus, the IED vendor's proprietary applications can still be used to display and manipulate the data. But, to make this data more widely accessible (presuming that to be desirable) other possible approaches would be to:

1) use the data presentation capabilities of the relational database packages, or
2) export the data to other applications such as spreadsheets, or
3) develop viewers (web based?) that can work off of the tables.

The majority of the commercial relational database packages have simple visualization tools that provide for extracting data and presenting it in tabular "report" formats, even

including simple plotting and charting features similar to those of spreadsheets. Unfortunately these tools also normally require that database "client" software be present on the user's PC. Paying for, and installing, software "client" licenses for every PC misses the point of making data more universally available.

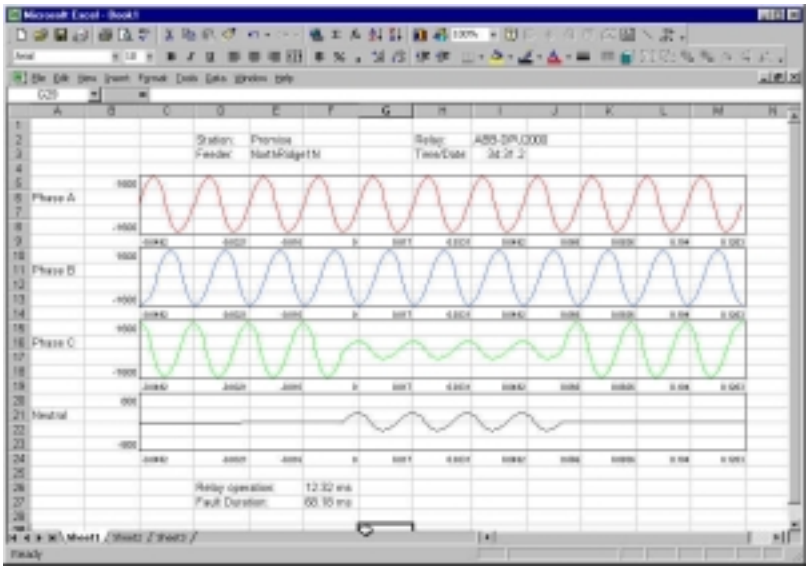

Figure 8.0 – Using spreadsheets to display exported IED data

Most database packages have the ability to process batch SQL commands and generate a text file output from the results. Such as text file can usually be imported into standard PC based spreadsheet packages where the data can be manipulated, plotted and formatted into displays. Since nearly every PC comes with spreadsheet software, this is potentially a slightly better approach towards providing universal data access. Specific spreadsheets can be developed (including the necessary data file import commands) and distributed to everyone who needs data access (refer to Figure 8.0.) These could also be placed on a common server so that anyone who needs them can obtain a copy.

Possibly the best approach for making data universally available, without the need to distribute data files or software, is to make the data available in the form of web pages accessible using standard web browser software. If there is any standard software that is universally provided in every PC sold today, it is web browser software.

Commercial relational database packages may also offer the ability to create static web pages using table-based data. For dynamic, interactive web pages it is necessary to use something more powerful and flexible, such as Java Applets to

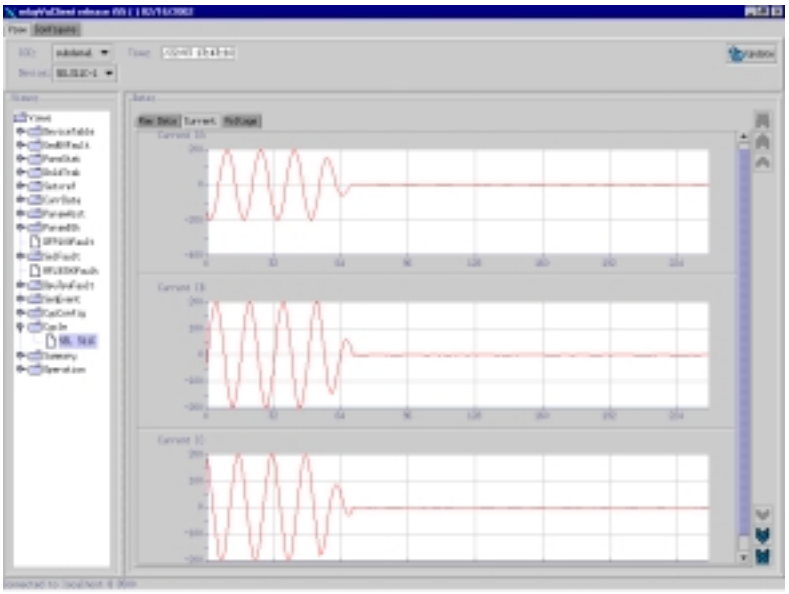

Figure 9.1 –Java Applet based web display of IED data

produce web pages that support interactive access and display of table-based data. An Applet is able to interact with the relational database tables via a Java Servlet that resides and runs in the database server computer. Such Applets can be customized for the specific data requirements of the class of IED for which they are designed. Thus a separate Applet can be developed for reviewing a relay's fault summary data or waveforms or settings, with each Applet designed to handle the type of data produced by that IED class (refer to Figures 9.1 and 9.2.) With Web/Applet based data access no special software (other that a web browser) is needed on a user's PC. Applets can be developed to run off of the "neutralized" (genericized ?) IED-class-based data in the relational database tables. Therefore one Applet per IED class is all that is necessary, rather than having to develop an Applet per specific IED. Applets can incorporate a wide range of HMI capabilities and functions and be quite sophisticated in their nature and operations.

Figure 9.2 –Java Applet based web display of IED data

DATA EXCHANGE – Although some work has been done in the area of cross-vendor/platform data exchange (such as using the COMTRADE file format for fault recorder waveform data) there is still a long way to go towards usable standards. Once data has been extracted from an IED, "neutralized" and stored in a relational database table, the data is available for "export" out of the tables as a file. As previously mentioned, some databases offer the ability to create comma-delimited text files which can be imported into other desk-top applications, such as spreadsheets. Another database export facility is the ability to create XML documents/files from the table's data (refer to Figure 10.0.) Unlike mere comma-delimited text, an XML document contains both the actual data as well as "tags" that can describe the data, its representation, its structure and its usage. XML "documents" have an additional advantage: standardized web browser technology can be used to display these documents. XML documents are wasteful of storage since they often contain a lot more self-describing information (as "tags") than they do actual data. But with today's computers and huge disk drives, storage is no longer a concern. Industry groups are looking at using the XML approach for inter-system, inter-application data transfers and as an actual mechanism for data "neutralizing". In the future, an increasing number of advanced applications will accept XML as data input and more IEDs will produce data output directly as XML documents. IED manufacturers are already building embedded web

server software into their IEDs, but this is normally for human-viewable purposes. Adding XML web pages will permit generic applications to "view" the IED and manipulate the available data. This might also be a viable mechanism for merging the UCA2.0 GOMSFE efforts into the reality of already-available computer/Web technology.

The biggest challenge for using XML as a delivery mechanism for settings, fault data and other such complex data, is the definition of "standard" tags for the particular data, including sub-tags that can defined any allowable variations in the main data type. For example, any capture of relay fault data ought to include both TIME and DATE information. We could agree on tags such as <TimeOfFault> and <DateOfFault> to denote these particular data items. But, since both can have several different representations (e.g. date can be represented as "January 3, 2002" or "1/3/02" or several other representations) we may need sub-tags called <DateFormat> and <TimeFormat> that indicate the actual representation of those data items. There is a new working group forming within the IEEE to actually take on the task of defining standardized tags for use in XML "pages" that will be used to deliver electric power information.

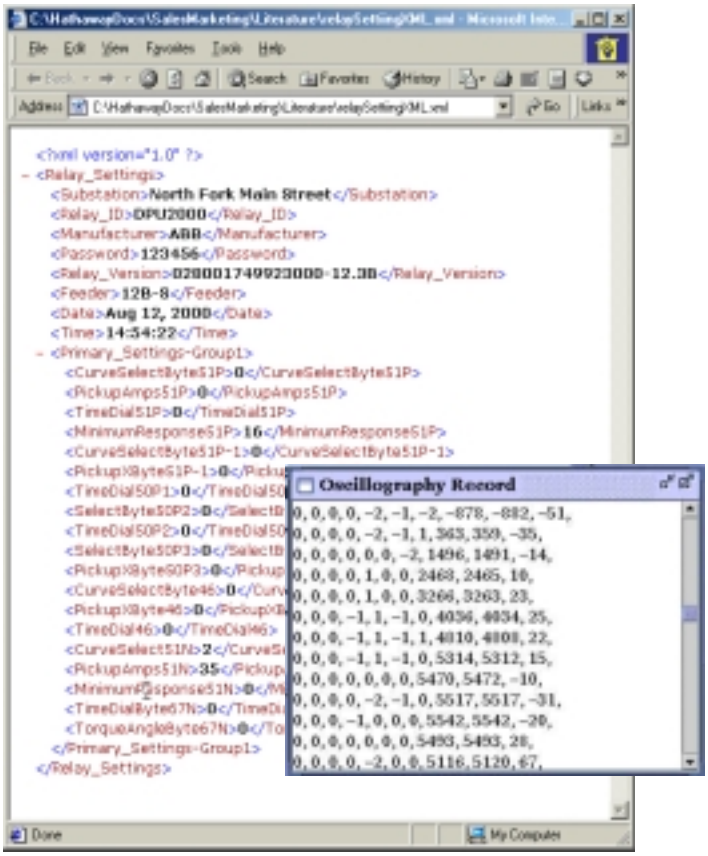

Figure 10.0 –IED data as XML web document

CONCLUSION – It has been said that knowledge is power. Today, information is power and the challenge to most utilities is collecting, managing and disbursing information to those that need it, in a timely and useable manner. There is a huge amount of raw information (or just plain data) contained within the various IEDs found in most substations. Very few utilities have the means or mechanisms that permit them to take advantage of this information and to use it to their best advantage. Relational database technology combined with various "Internet" technologies, such as XHTML and XML web pages, offer a possible approach to taming, harnessing and dispensing this information.