OTDR APIs Enable Customers to Build Their Own Systems

Torsten Born

Peter Thoma

In the past few years, OTDRs have evolved from being used only as standalone measurement instruments with limited functionality to become key instruments for servicing and characterizing global fiber-optic communication links. This trend has spurred the creation of standard file formats for OTDR data and standard software interfaces to control OTDRs remotely.

Optical Time Domain Reflectometers (OTDRs) have become key instruments for characterizing fiber-optic communication links. They are used to test the transmission performance of optical fibers by an optical pulse-response measurement method. During installation and maintenance of networks, OTDRs are primarily used to verify fiber and component parameters like optical loss and reflectance. The basic result from an OTDR measurement is the characteristic signature of the link called a *trace*, which is the amount of reflected light recorded versus time and distance. For an OTDR measurement, parameters such as pulse width, wavelength, range, and refractive index have to be defined. The trace can be postprocessed by a scantrace algorithm, which is used to detect faults, bad connections, and so on.

Today, optical network operators are driven by the need for high-quality service and high flexibility for quickly evolving the network according to market needs. The ability to access information about the fiber plant is essential for configuration, fiber network upgrade, and cost-effective maintenance.

Thus, many network operators archive installation data to be used as a reference for later comparisons during the lifetime of the network. OTDR measurement files form the core of this data collection. The total amount of data increases exponentially each year because of the worldwide deployment of optical fibers.

Torsten Born

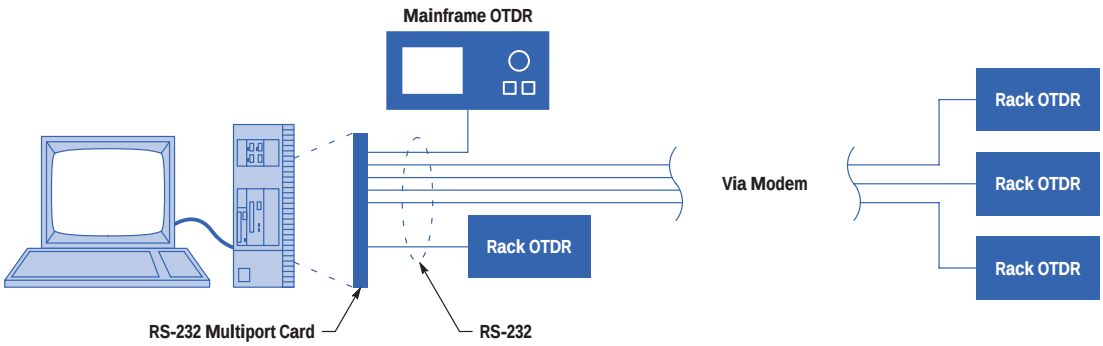
A software engineer at the HP Optical Communication Measurements Operation, Torsten Born is presently the technical supervisor for the OTDR toolkit and other projects and is the HP contact for the Bellcore SOR (standard OTDR record) file format. He joined HP after graduating from the University of Paderborn in 1994 with a Diplom Ingenieur in electrical engineering (focus on computer science). He was born in Hamburg, Germany, and his recreational interests include photography, music, traveling, volleyball, and inline skating.

Peter Thoma

Peter Thoma is an R&D project manager for the mini-OTDR HP E600A. He is currently investigating future optical network testing. He came to the HP Böblingen Instrument division in 1993. He received a Diplom Ingenieur in technical optics and control systems from the University of Stuttgart in 1984. Peter was born in Tuebingen, Germany. He is married and has three children.

Figure 1

Remote OTDR control.



In some networks, OTDRs are not only being used as operator-based instruments but are also being integrated as monitoring probes at the terminal and are controlled remotely (see **Figure 1**). With this approach the link quality of fibers and cables can be checked online. Restoration and repair can be optimized in case of a failure. Degradation can even be detected before the transmission quality suffers.

These trends call for a broader use of standards in OTDRs, such as standard file formats, standard remote interfaces, and application support with these standards.

Hewlett Packard provides a range of products (for example, the HP E6090A OTDR toolkit and the HP 81700 Series remote fiber test system) that offer complete solutions for monitoring OTDR measurement data during the complete network life cycle. These solutions are based on the TMN (Telecommunications Management Network) approach and use standard databases and interfaces.[1]

Market Needs

The telecommunications environment has a number of requirements for customers and solution providers like Hewlett-Packard. For system software that works with OTDR measurement data, these requirements include:

- Data portability. Over the lifetime of an optical fiber (greater than 20 years), all acquired measurement data must be accessible. Thus, old data formats must be supported. This also implies compliance with industry standards such as the Bellcore OTDR file format.[2]
- Multivendor capability. Measurement data is typically generated by instruments from various vendors.
- Quality. The software volume covered by tests grows with every new product generation.
- Cost-effective development. Since the general principles of the software remain the same when developing a new product, it is very important to offer a large reuse potential to save development resources and allow cost-effective development.
- Knowledge concentration. Most customers do not have expertise in OTDR measurement data analysis.
- Integration into customer-specific systems. Besides integrating OTDR data into existing customer asset management systems, the software should also be able to:
  - □ Establish communication with an instrument
  - □ Provide instrument control
  - □ Decode and encode OTDR measurement data
  - □ Perform offline analysis of measurement data.
- Long-term consistency. To follow the technological evolution of operating systems or measurement capabilities, a solid platform (operating system) is required for future migrations.

To address these market needs, HP offers two libraries: OTDRAPI and OTDRCAPI, which are application programming interfaces that are based on HP's OTDR design and

application knowledge. These libraries allow customers to postprocess measurement results and control the OTDRs from their own software environment. These libraries are compiled as 16- or 32-bit Windows® DLLs (dynamic link libraries) and offer interfaces to standard C and Visual Basic. Both libraries are designed to work together and offer customers a transparent programming environment that can be integrated into any proprietary system software.

The rest of this article describes the features and implementation details of the OTDRAPI and OTDRCAPI libraries.

Solutions

Features offered in the OTDRAPI library include loading and storing measurement data in standard Bellcore format, performing analysis operations (scantrace, loss calculations), and creating configuration* files and templates. The OTDRCAPI provides complete encapsulation of the serial interface and the SCPI (Standard Commands for Programmable Instruments) command language. Both libraries provide access to all measurement data and parameters on the remote OTDR.

The hardware component is a faceless OTDR for system integration, the HP E605x rack OTDR, which offers maximum measurement accuracy at a minimum price. The OTDR APIs help customers to write their own system software around this measurement hardware without the need to be OTDR specialists or have intricate serial interface knowledge. By encapsulating the communications functions, it is possible to offer optimal connectivity and data transmission performance as well as transparent error handling.

Standards

The OTDR libraries support two industry standards: the Bellcore file format for optical data and the SCPI command language for communicating with the instrument.

Bellcore File Format. In the past, every OTDR manufacturer had at least one file format of its own for storing OTDR measurement results. This has been a problem for the customers who have to cooperate with companies that use different measurement equipment and data formats. Since several companies are involved in manufacturing, installing, monitoring, and servicing optical fibers, a common data format and documentation standard has become necessary. The Bellcore SOR (standard OTDR record) format is the first attempt to create a common, portable OTDR data format and overcome these problems.

Although the SOR format solves the data exchangeability problems, it generates some new tasks for those customers who need to decode the data because SOR:

- Is in binary format
- Is a "living" standard in that it will experience regular updates
- Represents a superset of OTDR data from different manufacturers, so that its structure is complicated, and it contains data that may not be relevant to all customers
- Stores data in uncommon formats for portability.

Why would a customer want to decode the SOR? The main reason for decoding a binary data file is to create documentation. Typically, customers have their own documentation rules (for example, how to organize the different OTDR data on a printed report). There is no standard rule about which data to use and how to interpret it.

SCPI Remote Language. SCPI is a standard that is being driven by Hewlett-Packard. Many HP instruments support SCPI for remote control via HP-IB (IEEE 488.1, IEC 625), RS-232, or LAN.

SCPI uses ASCII commands to offer a functional interface to the instrument. Therefore, long-term and multivendor compatibility are taken care of by design. The complete programming language is organized in a tree structure in which each root represents a different view of the instrument's measurement or configuration data. Branches allow for accessing the relevant data of any particular view. For example, changing the instrument's serial baud rate is achieved by using a command from the SYSTEM tree, which allows for accessing configuration data. For example:

```
SYSTEM:COMMUNICATE:SERIAL:BAUDRATE 57600
```

where COMMUNICATE indicates that the data is related to communication, SERIAL accesses data for the serial interface, and BAUDRATE indicates the actual value.

* The configuration file contains a subset of all instrument settings, including measurement parameters that are needed to configure a measurement. This allows all necessary parameters to be sent in just two commands (file transfer and load setting) in a very compact format to the OTDR.

Since commands can become quite large using such syntax, SCPI defines a short form that only uses the first three or four characters of each branch:

```
SYST:COMM:SER:BAUD 57600
```

If the instrument supports more than one serial interface, the command can be adjusted to address the correct port. For example,

```
SYST:COMM:SER2:BAUD 57600
```

addresses serial port 2.

Querying the instrument's configuration is achieved by the same command with a "?" instead of a value appended to the command string.

```
SYST:COMM:SER2:BAUD?
```

For more detailed information about SCPI see reference 3.

SCPI also defines the instrument's behavior for such things as syntax errors and parameter overflows and underflows. Higher-level interfaces have been defined on top of the SCPI language (for example, the VISA plug-and-play drivers that offer an interface to applications such as HP VEE or LabView® from National Instruments).

Although remote control of an OTDR instrument mainly serves the needs of a monitoring application, it also helps to reduce the need to maintain an engineering staff with a lot of knowledge about fiber-optic measurements. For example, one engineer can set up and control several measurements from the central office, while field technicians can focus on setting up the equipment at various measurement sites. This allows instrument control to be done by modem.

The OTDRCAPI library allows controlling an OTDR without even knowing the SCPI language because the library offers a high-level functional interface for the most common applications. For special applications, the OTDRCAPI library offers pass-through functions, which allow sending commands directly to the instrument without checking or processing. This provides complete transparency.

Software Architecture Considerations

When designing system software, several aspects have to be considered. First the operating system and hardware platform play an important role. Since the WINTEL combination (Microsoft Windows/95/NT® + Intel processor) is becoming more and more standard, we have focused on the Windows operating systems for our APIs. However, since most of the API libraries' implementation is done in standard C and C++, a port to a real-time or HP-UX operating system would be quite easy.

The interfaces to both API libraries are standard C, C++, and Visual Basic. Visual Basic enables customers to rapidly prototype applications, while standard C allows the creation of a very elaborate software system with good performance.

Implementation Details

Both libraries are subject to regular updates because the Bellcore and SCPI standards are living standards. Therefore, a major focus has been put into a generic definition for the API function calls because these interfaces will probably stay constant while some dynamic link libraries may change.

OTDRAPI. The core of the OTDRAPI library is a C++ library called OTDRLIB. This library is also an important element of all OTDR products such as the HP E4310A mainframe OTDR, the HP E6000 mini OTDR, the HP E6090 OTDR toolkit, the HP E605x rack OTDR, and the HP 81700 Series 200 remote fiber test system. This library provides all the mathematical and file access functions, which guarantees that all changes to the instruments' firmware can easily be updated in the OTDRAPI library.

Figure 2 shows HP OTDR products and their associated operating systems that are already covered by the OTDRLIB. The data storage shown in **Figure 2** contains the following OTDR measurement data:

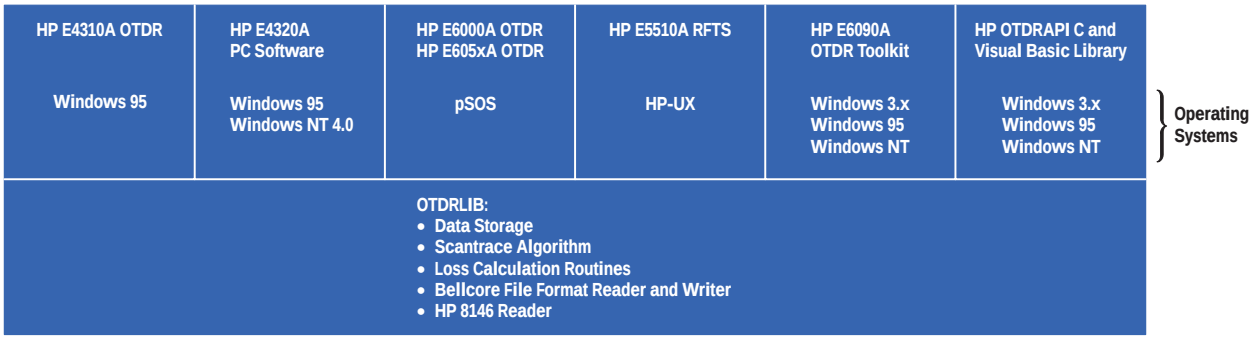
- Trace data points
- Measurement parameters
- Hardware-specific data
- Event and landmark tables
- Trace information such as cable and fiber identifiers.

All OTDRLIB internal definitions (such as structures and constants) are encapsulated by the OTDRAPI library, so that even major changes to internal data handling will not affect the library's external interfaces such as function calls and data types.

We decided not to offer an object-oriented C++ interface because standard C is already a subset. We chose to give

Figure 2

HP OTDR products and their associated operating systems covered by OTDRLIB.

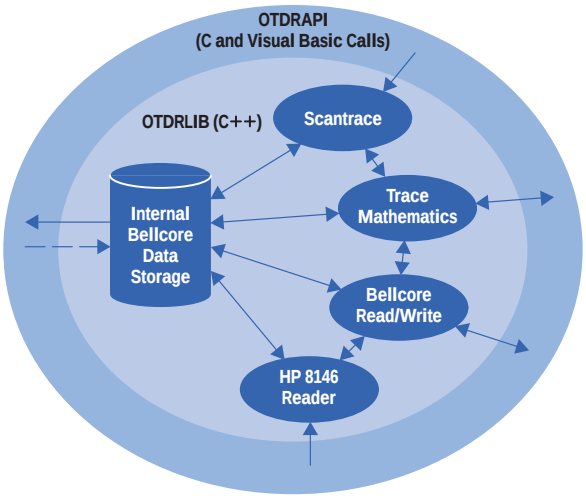


customers the flexibility of defining their own objects depending on the available software structure.

Figure 3 shows the internal architecture for the OTDRAPI library. The OTDRAPI library allows read access to all data in the internal Bellcore data storage. However, it is not possible to modify core measurement data such as trace data, measurement parameters, and hardware-specific data. This guarantees measurement data integrity under all circumstances. It is possible to modify all documentation data such as the event table or the trace comments.

Figure 3

The internal architecture for the OTDRAPI library.



The library can also be used to define a set of measurement parameters for a new measurement that can then be stored in a configuration file.

Because of the large amount of data handled internally, MS-DOS® is not supported. The size of one Bellcore trace normally exceeds 32K bytes, and we prefer to use flat memory internally. For 16-bit Windows, we were able to minimize the restrictions by using a large memory model, which is unfortunately not possible with standard MS-DOS.

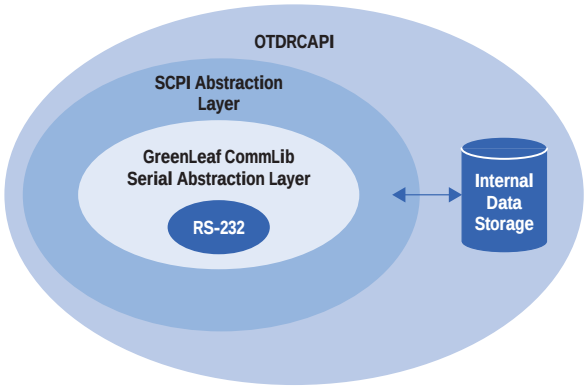
OTDRCAPI. One major problem that we had to overcome with the OTDRCAPI library is the cumbersome implementation of serial communications under different operating systems. Even the different Windows operating systems do not have a consistent interface. To have a stable level, we decided to use CommLib from GreenLeaf as a serial interface abstraction layer. The CommLib offers a consistent function layer for initializing, opening, closing, and communicating over a serial interface. However, the customer will not see any of the GreenLeaf functions because the OTDRCAPI library encapsulates the instrument's SCPI remote language (see **Figure 4**).

For example, opening a communication with an OTDR instrument using the OTDRCAPI library involves:

- Configuring the baud rate, handshake, and parity
- Opening the serial interface
- Reading the instrument's identification
- Configuring some of the OTDRCAPI functions regarding the detected instrument type

Figure 4

The OTDRCAPI library encapsulates the SCPI calls.

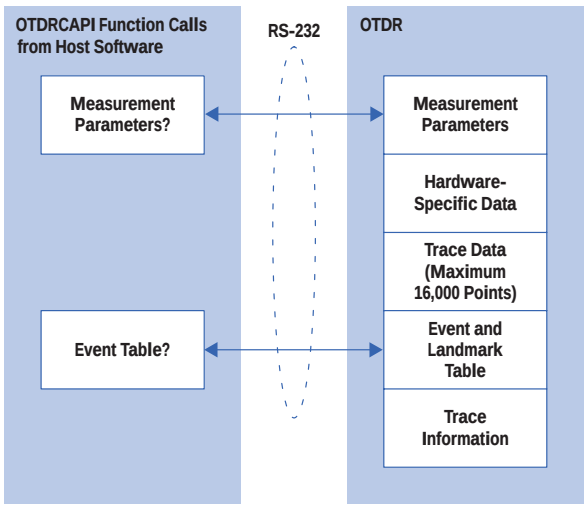


- Reading the instrument's status (for example, measurement running)
- Switching to a higher transfer baud rate if necessary
- Returning an error when one of the above operations fails.

The same error handling applies to all other functions such as accessing trace data, measurement parameters, and event tables (see **Figure 5**). Using a higher transfer baud rate is useful if the instrument always boots at a default baud rate (19.2 Kbits/s) because transferring large amounts of measurement data requires a higher transfer rate (115.2 Kbits/s). The OTDRCAPI library takes care of all synchronization, necessary delays, error handling, and returning to the default baud rate when the port is closed.

Figure 5

OTDR function calls from the host accessing information from the OTDR instrument.



Unlike the internal data storage for the OTDRAPI library, the OTDRCAPI library holds only a part of the Bellcore file. The OTDRCAPI library only provides a window to the Bellcore data stored on the instrument. Therefore, because all data structures are automatically handled between the OTDRCAPI and OTDRAPI libraries, the programmer receives the same data structures when reading from a remote instrument as when reading data from a Bellcore file. Communication between the two libraries can be handled by either using the same data structures for items such as parameters and trace data or passing a storage pointer from the OTDRAPI library to the OTDRCAPI library.

By combining both APIs, it is possible to read a complete Bellcore trace from a remote instrument into the OTDRAPI's internal Bellcore storage and then perform all data and mathematical operations on this data locally using the functions in the OTDRAPI library. This capability allows distribution of the steps of an OTDR measurement between the OTDR and the controlling host software. The following steps illustrate this concept:

- Perform the measurement on the OTDR without scantrace
- Upload the data to the host
- Perform the scantrace on the host (this will help to improve performance)
- Postprocess the event table on the host
- Edit trace comments, event comments, and landmarks on the host
- Download the data back to the OTDR
- Store the measurement on the host and OTDR.

Figure 6 shows a programming example that demonstrates how simple the implementation of the steps listed above can be using the two APIs. All calls with the prefix OTDRCOM are from the OTDRCAPI library, and all calls with the prefix OTDR are from the OTDRAPI library.

Figure 6

Example of code using OTDRAPI and OTDRCAPI library functions.

```
// declarations ...
COMDEV pComdev;   // the communications
                  // device pointer
OTDRP pOTDRapi;   // the OTDRAPI pointer
short status;     // holds the status of
                  // function calls

// initialize OTDRCAPI ...
if(OTDRCOMInitLib( NULL ) )
   {
   printf("Unable to initialize
   OTDRCAPI!\n";
   exit(0);
   }

// initialize port ...
pComdev = OTDRCOMInitPort(&status, "COM1");
if( status )
   {
   printf("Error: %s\n",
   OTDRCOMStatusMessage(pComdev));
   exit(0);
   }

// ... code for setting up the port goes
// here ...

// open the port ...
if( OTDRCOMOpenPort( pComdev ) )
   {
   printf("Error opening the serial
   port!\n");
   exit(0);
   }

// initialize the OTDRAPI; further error
// checking not listed!
pOTDRapi = OTDRInitLib ( &status );

// start a remote measurement and wait
// for 30 sec...
OTDRCOMStartMeasurement(pComdev);
Sleep(30)
```

```
// stop the measurement ...
OTDRCOMStopMeasurement(pComdev);

// check if the measurement is stopped ...
BOOL bRunning=TRUE;
OTDRCOMIsMeasurementInProgress(pComdev,
&bRunning);
while (bRunning)
   {
   Sleep(500);
   OTDRCOMIsMeasurementInProgress(pComdev,
   &bRunning);
   }

// load the measurement data into the
// OTDRAPI ...
OTDRCOMGetFileToAPI(pComdev, pOTDRapi,
NULL);

// perform a scantrace locally; maybe set
// parameters before ...
OTDRScanTrace(pOTDRapi, NULL);

// send the data back to the OTDR ...
OTDRCOMSendFileFromAPI(pComdev, pOTDRapi,
NULL);

// store locally ...
OTDRSaveTrace(pOTDRapi, "LOCAL.SOR");

// store remotely ...
OTDRCOMSaveCurrentTrace(pComdev,
"REMOTE.SOR");

// clean up all pointers ...
OTDRCOMClosePort(pComdev);
OTDRCOMFreePort(pComdev);
OTDRFreeLib(pOTDRapi);
```

Outlook

Both libraries will be subject to regular updates, as the Bellcore and the SCPI standards develop further. Bellcore plans to release a new revision of the SOR format before middle of 1998, which must be updated in the OTDRAPI library. Also the remote function set of the HP E60xx and HP E4310 OTDRs is growing.

Another area of development is the number of supported platforms. Currently, all Windows platforms are supported (16-bit only with a large memory model). There might emerge a need for UNIX® versions or even real-time operating systems like pSOS+.

The OTDRAPI library uses no Windows-specific functions, so a port to other 32-bit platforms will be quite easy. For the OTDRCAPI library, the RS-232 abstraction layer will need to be replaced when porting to another operating system. For example, the GreenLeaf CommLib approach might not work on a UNIX operating system.

Other elements such as the scantrace algorithm or the loss calculation routines will experience regular improvements that will also be updated in the OTDRAPI library.

Conclusion

The OTDRAPI and OTDRCAPI libraries offer a cost-effective, high-quality solution for fiber-optic system integrators. This was achieved by reuse of the software modules developed for our OTDR products.

This approach provides a fast development time for customers and guarantees that they will benefit from future enhancements of the remote control facilities and the data handling (Bellcore file format and scantrace algorithm) of Hewlett-Packard's OTDRs.

By offering these APIs, we enable the customers to maximize the benefit obtained from their OTDRs.

Acknowledgments

The authors would like to thank the OTDR software team, especially Jürgen Sang, Alf Clement, Joachim Winkler, Jonathan McEwan, and Oliver Berger for their contributions. We also want to thank John Peters at Bellcore.

References

1. J. Nemeth-Johannes, "A Standardized Instrument Programming Language Based on IEEE Standard 488.2," *Autotestcon '89*.

URL: hpswtsvr.lvld.hp.com/goodstuff/standards/scpi/index.html

2. *Generic Requirements for Optical Time Domain Reflectometers*, GR-196, no. 1, Bellcore, 1995 (Rev. 1, December 1997).

URL: www.bellcore.com

3. P. Ghadayammuri, "A Platform for Building Integrated Telecommunications Network Management Applications," *Hewlett-Packard Journal*, Vol. 47, no. 5, October 1996.

Bibliography

1. T. Born, "OTDR Measurements Harmonize With Bellcore Standard", *Test & Measurement World*, August, 1997, pp. 37-38, *Test & Measurement Europe*, November, 1997, pp. 21-24.

Online Information

Additional information about the OTDR is available at:

- http://www.tmo.hp.com/tmo/datasheets/English/HPE6090A.html
- http://www.tmo.hp.com/tmo/datasheets/English/HPE6000A.html
- http://www.tmo.hp.com/tmo/datasheets/English/HP8147.html
- http://www.tmo.hp.com/tmo/datasheets/English/HP81700_Series_200.html