Application Note – AN102

Computer Controlled Monitoring using CORBA

Ivan Ng
Software Project Manager, Pixelmetrix Corporation

Providing effective and efficient monitoring of today's broadcast networks requires not only manual control via local or remote interfaces, but often requires integration with other systems for the exchange of performance data. For example, quality management systems based on Oracle databases, subscriber management, and or conditional access systems.

*In addition to local and remote control interfaces, DVStation provides an open-standards interface based on CORBA which allows other **computer** systems to control and obtain performance information.*

Background

DVStation provides concurrent and multi-user control – both through a local interface and via remote control for monitoring digital television networks.

Remote control means that both configuration and results must be obtained over the internet or corporate LAN. Remote control can be via direct user control – for example using a web browser – or via direct control from a computer.

Since a remote application can be written in a variety of languages, such as C++ or Java, running on various operating systems, and different network types such as ethernet or token-ring can be used, using a *middleware* layer allows remote applications to retrieve monitoring results, as well as configure operating and alarm parameters — without having to worry about the specifics of the hardware, operating system, language, or network type used. CORBA is the answer.

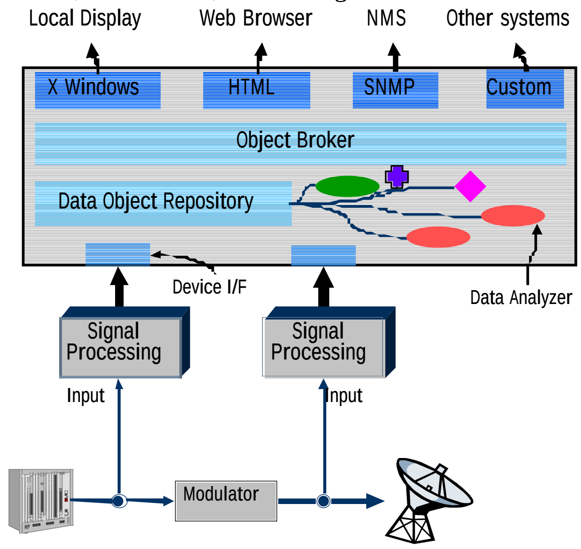
CORBA allows remote applications, through the help of an ORB (Object Request Broker) which handles communication, to access monitoring results as objects in the language that the application was written. If a remote application is written in C++, information will be packaged in a C++ object for the application to retrieve and display. If a remote application is written in Java, information will be packaged in a Java object for the application to retrieve and display.

This is possible by publishing IDL (Interface Definition Language) files, which describes the parameters available to the CORBA client for retrieval and configuration. Each IDL file describes an interface to a database object describing a logical grouping of information for a particular measurement or configuration. IDL is independent of any programming language and is supported by many different languages (eg C++, Java).

DVStation uses the TAO ORB, which comes with ACE (Adaptive Communication Environment), for its CORBA support.

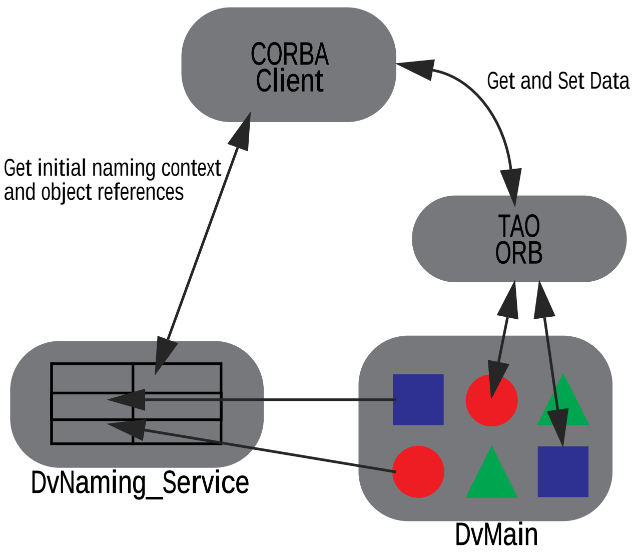
DVStation Software Architecture

DVStation employs multi-process architecture, dividing tasks between user/program interfacing and data collection, correlation, and storage.



Specifically, there are three processes involved in automating DVStation operations through CORBA:

- **DvMain**. The CORBA server and contains the measurement and configuration data objects. This process communicates with the card modules to obtain raw measurement data and results. Measurement results are written to database objects to be retrieved by various interface clients for display or processing. During the creation of the database object, DvMain also registers an object reference with the naming service using a unique key so that remote CORBA clients are able to retrieve the contents of the database object.
- **DVNaming_Service**. The naming service program which provides remote CORBA clients references to database objects within DvMain. Basically, it provides a look up service which allows CORBA clients to locate a particular database object using a unique key. A CORBA client can use an object reference to retrieve results within, or to configure parameters of, a database object.
- **The CORBA Client.** The client software runs either on DVStation itself or on a remotely connected computer.



How to write a CORBA Client for DvStation

The following section addresses some of the issue in writing a CORBA client with the assumption the developer has sufficient familiarity in CORBA and basic programming techniques.

First of all, current versions of the IDL files which match the version of correct version of DVStation system software are required. These files are contained within the file system of each DVStation and are automatically updated with each software revision. To get more information on how to retrieve these files or about DVStation software in general please contact support@pixelmetrix.com.

What information does a Client need to get the initial naming context of the naming service?

First of all, a CORBA client must obtain the initial naming context of the naming service. It needs to determine the endpoint of the naming service, which is made up of 2 parts: The IP address and the port number. By default, DVStation is shipped with an IP address of 192.168.15.150 – this address can be changed via the Network settings panel in the configuration menu of DVStation. The port number for the naming service is fixed at 2740.

How to obtain the object reference of a database object?

Provided the CORBA client has already obtained the initial naming context of the naming service, the client would then need to know the key of the database object which was used in its registration with the naming service.

A key used in the naming service is made up of one or more parts. The first part of the key is always the name of the database object. System level database objects not tied to a port or slot usually only have one part to the key, which is the name. The second part of the key usually indicates the port or slot number.

In rare cases, more complicated database objects might have multiple parts in the naming service key.

Tips for Working with Object References

Instead of querying the object reference to a database object each time it is required, store it somewhere so it can be used again later. If the object reference becomes invalid (eg. DvMain being restarted), then query the object reference of the database object from the naming service again. If the initial naming context of the naming service becomes invalid (eg Naming Service restarted), then query the initial naming context of the naming service before querying the database object reference.

Note that if a reference becomes invalid, a CORBA exception will be thrown when attempting to call a API with the invalid object reference. The CORBA client application will be notified of the exception.

Structure of DVStation IDL Files

Each IDL file contains only one interface. IDL files are being constantly updated to match the evolution of the DVStation System Software, an overview of each IDL file, the name of the interface, and a short description of the purpose of the object is included below. Naturally, exist for the other modules in the system, i.e. COFDM, QPSK, etc.

Additionally, the details of the various attributes/ structures in the IDL file is documented in the IDL file itself as well as the naming service key which is also documented in the IDL file.

System Level Interfaces

Interface	Description
DvCommon	Common typedef, structure and constant
DvCommonDef	Common typedef and enum
DvAlarmActionList	List of alarm actions supported
DvCardName	Names of the various card modules
DvDateTime	Current date/time and timezone information
DvHWMetrics	Hardware metrics parameter
DvLogIntervalConfigSetBC	Interval between logging of the following: • Bandwidth • Packet interval • PCR measurement • QMM • QAM • QPSK • COFDM
DvLogParameters	Maximum number of log files and log file size to configure
DvLogMessage	The 40 most recent log messages
DvPortConfig	The configuration for each port
DvPortStatus	The status of each port
DvProfileList	The list of system and stream level profiles
DvRackConfig	The system level parameters in DVStation
DvScheduleFileList	The schedule for loading profiles

Transport Stream Level Interfaces

Interface	Description
DvBWAlarmConfigSettingBC	The alarm and threshold setting for the bandwidth of each PID.
DvBWChannelService	ID and name of each service being monitored. Used with DvBandWidthBC.
DvBWMeasureSet	The bandwidth measurement set
DvBandWidthBC	The bandwidth of the PIDs being monitored. (Requires the use of DvPIDPerChannel to get the PID of each channel.) The bandwidth of all services being monitored. (Requires the use of DvBWChannelService to get the service name of each channel.)
DvDecodedTSPacket	Decoded transport stream packet (PID)
DvDecodedTSPacketSet	Information about a set of TS packets
DvETR290ConfigSetBC	ETR-290 Alarm Configuration: • CAT error • CRC error • Continuity count error • RST error • Sync byte error • Sync loss error • Transport error
DvETR290Error	The status of each ETR290 parameter
DvJitterAlarmSet	Alarm configuration set for PCR Jitter
DvJitterMeasureSet	PCR Jitter measurement set
DvPCRConfigSet	ETR290 PCR Jitter Alarm configuration
DvPIDPerChannel	The PID for each bandwidth channel in DvBandwidthBC (Bandwidth)
DvPSIServices	PSI service structure of current transport stream: service names for each PID, list of unreferenced PIDs, and list of PCR PIDs.
DvPktIntAlarmSet	Packet Interval Alarm configuration
DvPktIntHistogram	Packet Interval Histogram
DvPktIntMeasureSet	PCR Jitter measurement set
DvServiceStructure	Service structure template information
DvSSMAlarmSet	Service structure alarm configuration objects
DvSIRepetitionConfigSet	Alarm configuration setting for SI Repetition of ETR290
DvTsCapture	Parameters for transport stream capture. Used to start or abort a transport stream capture
DvTSPSlotConfig	The TSP slot configuration
DvTotalPIDBW	The number of TS packets collected for each PID over 1 PID statistical report.
DvUnreferencedPIDConfigSet	Alarm configuration setting for unreferenced PID of ETR290. Set of unreferenced PIDs to mask off.

Example IDL File

The example below shows the IDL file describing the port status data object. The port status object contains the status of each layer of the protocol stack: Physical, MPEG-2 Transport Layer, PSI information, and Program Content.

```
////////////////////////////////////////////////////////////////
//    Name : DvPortStatus.idl
//    Description : IDL file for DvPortStatus interface
//    The port status in DVStation
//    Key used in naming service :
//    1) "PortStatus"
//    2) Port number
//    Author : Ivan Ng
////////////////////////////////////////////////////////////////
#ifndef __DVPORTSTATUS_IDL__
#define __DVPORTSTATUS_IDL__
#include "DvCommon.idl"
module sys
{
    // The structure containing the port status
    struct DvPortStatusStruct
    {
        // The status of the physical layer
        unsigned short phyLayer;
        // The status of the transport layer
        unsigned short transportLayer;
        // The status of the elementary layer
        unsigned short elementaryLayer;
        // The status of the video quality layer
        unsigned short videoqualityLayer;
        // The state of the port
        unsigned short portState;
        // The percentage of bandwidth by non-stuffing packets
        unsigned short bwPercent;
        // Whether the bandwidth threshold has exceeded
        boolean thresholdExceeded;
    };

    interface DvPortStatus
    {
        // Get the port status in DVStation
        DvPortStatusStruct getPortStatusStruct();
    };
};
#endif
```

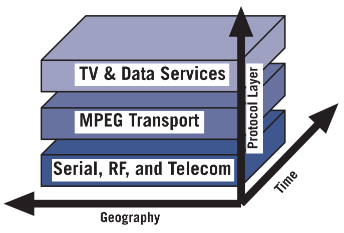
References

"Measurement Guidelines for DVB Systems", Draft TR 101 290, DVB.

Additionally, the following websites:
http://www.infosys.tuwien.ac.at/Research/Corba/docu.html
http://developer.netscape.com/docs/manuals/corba.html
http://www.cs.wustl.edu/docs/manuals/corba.html
http://www.isg.sfu.ca/~hak/corba/
http://www.cbbrowne.com/info/corba.html

About DVStation

Pixelmetrix has focused on creating a single self-contained monitoring station that can analyze thousands of parameters within hundreds of digital television signals. Through the use of plug-in modules and parallel processing, we monitor all these parameters in real time, simultaneously and continuously. We've targeted our development efforts to insure the quality of the signal, the integrity of the program service and the delivery of essential technical information to the right people in a timely and meaningful manner.



Our engineers began with a simple premise: Effective monitoring of digital television networks – just as with telecom networks – requires the use of real-time, continuous and simultaneous evaluation of hundreds of points along the transmission chain. To receive this necessary network intelligence, adequate data collection, analysis and correlation is needed on three axis – time, layer and geography. Monitoring of all layers – physical, transport, coding, and quality – is essential for a complete maintenance picture.

Plug-in modules allow flexibility and accommodate changes in a fast evolving technical infrastructure. So far, we've focused on three categories of plug-in modules: physical line interfaces (ASI, SPI, RF, ATM etc.); a transport stream processor (TSP); and picture quality processors.

In our design, a line interface module extracts the MPEG-2 transport stream from the native RF or telecom signals and passes that data to a TSP – Transport Stream Processor. Line interface modules provide monitoring capability on the physical layer. For RF interfaces (QPSK, QAM, COFDM, 8VSB, etc.) monitoring means to check carrier level, C/N (carrier-to-noise ratio), bit error rate and EVM (Error Vector Magnitude), or other parameters that may be applicable. Additionally, a simple constellation diagram indicates overall modulation health.

Our ATM interface connects to a 155 Mb/s optical fiber and extracts MPEG transport streams from several VP/VCs (virtual path/virtual circuit). In addition to this basic functionality, the interface detects physical layer errors and parameters with the optical and Sonet/SDH signals.

For More Information

To learn more about the DVStation, request a demo, or learn how Pixelmetrix might help you optimize video network integrity, contact us today!

On the Internet:	sales@pixelmetrix.com www.pixelmetrix.com
North America:	1-877-71-PIXEL
Europe:	+41-79742-7454
Asia Pacific:	+65-547-4935

About the Author

Ivan Ng is a Software Project Manager with Pixelmetrix Corporation, manufacturer of the DVStation, a preventative monitoring solution for digital broadcast networks.